

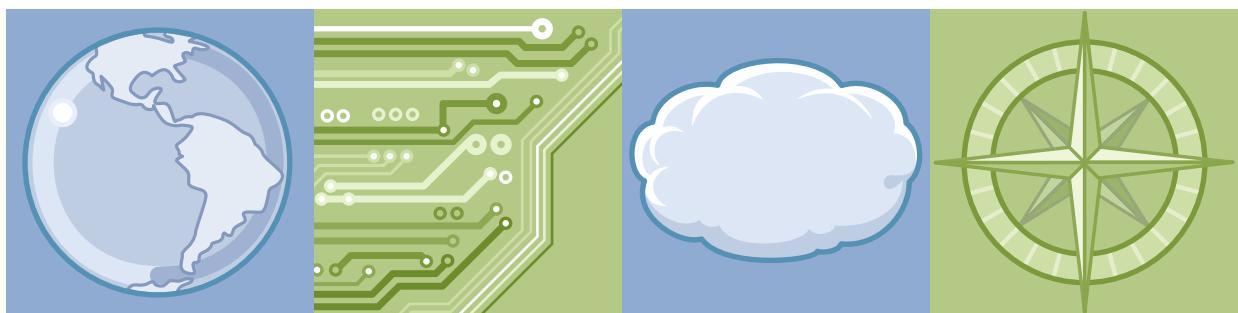


IBM Training

Student Notebook

Accelerate, Secure and Integrate with IBM DataPower V6

Course code WE601 / ZE601 ERC 1.1



WebSphere Education

Trademarks

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

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

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

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

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

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

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

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

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

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

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

June 2014 edition

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

Contents

Trademarks	xv
Course description	xix
Agenda	xxi
Unit 1. DataPower administration overview.....	1-1
Unit objectives	1-2
Developing on the appliance	1-3
DataPower SOA appliance administration	1-4
WebGUI web administration application	1-5
Administration by using the web browser	1-6
Navigation bar categories	1-8
System control features (1 of 3)	1-9
System control features (2 of 3)	1-11
System control features (3 of 3)	1-12
File management	1-14
File directories for configuration	1-15
File directories for security	1-16
File directories for logging	1-17
More file directories	1-18
Administrative access control	1-19
Create an application domain	1-20
Application domain: Configuration tab	1-22
Configuration Checkpoints	1-23
View application domain status	1-24
Create a user account and a user group	1-25
Manage user group details	1-26
Manage user account details	1-28
Export the system configuration	1-29
Import a system configuration	1-30
Saving configuration changes	1-31
Administration by using the command-line interface	1-32
First boot after unpacking	1-33
CLI login screen	1-34
Quick initial configuration procedure	1-35
Retrieve system information by using the CLI	1-36
Administration by using SOAP: SOMA and AMP	1-37
SOMA: Create a user account	1-38
SOMA: Account creation response	1-39
AMP: Appliance information request	1-40
AMP: Appliance information response	1-41
Management interface summary	1-42
Globalization: Displaying other languages in WebGUI and the log	1-43
Enabling languages	1-44
Getting the WebGUI to display an alternative language	1-45
Getting an alternative language for the log and messages	1-46
Unit summary	1-47

Checkpoint questions	1-48
Checkpoint answers	1-49
Exercise 1	1-50
Exercise objectives	1-51
Unit 2. DataPower services overview	2-1
Unit objectives	2-2
Services in a DataPower appliance	2-3
Front sides and back sides, and sideways	2-4
Services available on the DataPower appliance	2-5
XSL proxy service	2-6
XML firewall service	2-7
Multi-protocol gateway service	2-8
Web service proxy service	2-9
Web application firewall service	2-10
Other services	2-11
DataPower services feature hierarchy	2-12
Unit summary	2-14
Checkpoint questions	2-15
Checkpoint answers	2-16
Unit 3. Structure of a service	3-1
Unit objectives	3-2
Object-based configuration	3-3
Approach to configuring objects	3-4
Basic architectural model	3-5
Front side access	3-6
Connection to the back side	3-7
Message processing phases	3-8
Configuring a service policy (processing policy)	3-10
Policy editor	3-11
Configuring rules within a service policy	3-12
Processing rules	3-13
Match action	3-14
Processing actions	3-15
Processing actions	3-16
More processing actions	3-18
Multi-step processing rules	3-19
Pre-defined context variables	3-20
Validate action for XML	3-21
Validate action for JSON	3-22
Transform action for XML	3-23
Transform action that uses XQuery (JSON and XML)	3-24
Transform action for binary transformations	3-25
Filter action	3-26
Filter action: Replay attack	3-27
Content-based routing	3-28
Route action configuration	3-29
Style sheet programming with dynamic routing	3-30
Results action	3-32
Results asynchronous and multi-way results mode	3-33

Service settings	3-35
Service types	3-36
URL rewriting	3-37
XML Manager	3-39
Default XML Manager configuration	3-40
XML parser limits	3-41
JSON document limits within the XML manager	3-43
Exporting a service configuration	3-45
Cloning an XML firewall configuration	3-46
Troubleshooting a service configuration	3-47
Unit summary	3-48
Checkpoint questions	3-49
Checkpoint answers	3-50
Exercise 2	3-51
Exercise objectives	3-52
Exercise overview	3-53
Unit 4. Multi-protocol gateway service.....	4-1
Unit objectives	4-2
What is a multi-protocol gateway?	4-3
Protocol handlers at a glance (1 of 2)	4-4
Protocol handlers at a glance (2 of 2)	4-5
Front side protocol handlers	4-6
Static back-end gateway	4-7
Dynamic back-end gateway	4-8
Multi-protocol gateway and XML firewall compared	4-9
Multi-protocol gateway editor	4-10
Scenario 1: Provide HTTP and HTTPS access	4-12
Step 1: Configure the back side transport	4-13
Step 2: Create a document processing rule	4-14
Step 3: Create the front side handlers	4-15
Step 4: Configure the front side handler	4-16
Step 5: Configure the SSL proxy profile	4-17
Scenario 2: Dynamic back-end service	4-18
Step 1: Configure the back-end transport	4-19
Sample service that targets a style sheet	4-20
Scenario 3: Provide WebSphere MQ access	4-21
Scenario 4: Provide WebSphere JMS access	4-22
Scenario 5: Provide IMS Connect access	4-23
Scenario 6: Provide a RESTful interface	4-24
REST support in DataPower	4-25
Supported IMS features	4-26
IMS Connect support	4-27
Comparing services	4-28
Unit summary	4-29
Checkpoint questions	4-30
Checkpoint answers	4-31
Unit 5. Problem determination tools.....	5-1
Unit objectives	5-2
Common problem determination tools	5-3

Appliance status information	5-4
Troubleshooting	5-5
Troubleshooting: Networking	5-6
Troubleshooting: Packet capture	5-7
Troubleshooting: Logging	5-8
Troubleshooting: System log	5-9
Filtering system log	5-11
Troubleshooting: Generate Log Event	5-12
Troubleshooting: Reporting	5-13
Troubleshooting: Advanced	5-14
Troubleshooting: XML File Capture	5-15
Troubleshooting: Send a test message	5-16
Troubleshooting: Multi-step probe	5-17
Troubleshooting: Enabling the multi-step probe	5-18
Multi-step probe window	5-19
Multi-step probe content	5-20
Problem determination with cURL	5-21
Communicating with DataPower support	5-22
Logging basics	5-23
Log targets	5-24
Available log levels	5-25
Log target configuration	5-26
Ten-log target types	5-27
Event filters	5-28
Object filters	5-29
Event subscriptions	5-30
Log action	5-31
Unit summary	5-32
Checkpoint questions	5-33
Checkpoint answers	5-34
Exercise 3	5-35
Exercise objectives	5-36
Exercise overview	5-37
 Unit 6. Handling errors in a service policy.....	 6-1
Unit objectives	6-2
Error handling constructs	6-3
Configure an On Error action	6-4
Creating an error rule	6-5
Configure Transform action in error rule	6-6
Style sheet programming that use error variables	6-7
Example custom error style sheet	6-8
Error rule versus On Error action	6-9
Error Policy	6-10
Typical Error Policy use cases	6-12
How the Error Policy works (1 of 3)	6-13
How the Error Policy works (2 of 3)	6-14
How the Error Policy works (3 of 3)	6-15
Error Policy Configuration	6-16
Error Policy Configuration - Multi-Protocol Gateway	6-17
Additional Error Policy Processing (1 of 2)	6-19

Additional Error Policy Processing (2 of 2)	6-20
Unit summary	6-21
Checkpoint questions	6-22
Checkpoint answers	6-23
Exercise 4	6-24
Exercise objectives	6-25
Exercise overview	6-26
Unit 7. DataPower cryptographic tools and SSL setup	7-1
Unit objectives	7-2
DataPower crypto tools	7-3
Generating crypto (asymmetric) keys on-board (1 of 2)	7-4
Generating crypto (asymmetric) keys on-board (2 of 2)	7-5
Download keys from temporary storage	7-6
Key and certificate objects point to files	7-7
Crypto shared secret (symmetric) key	7-8
Crypto (asymmetric) key	7-9
Crypto certificate	7-10
Crypto identification credential	7-11
Crypto validation credential	7-12
Import and export crypto objects	7-14
Convert Crypto Key Objects / Certificate Objects	7-15
Uploading keys	7-17
Certificates can expire or get revoked	7-18
Certificate revocation list (CRL) retrieval	7-19
Crypto certificate monitor	7-20
Hardware Security Module (HSM)	7-21
DataPower support for SSL	7-22
A crypto profile specifies details of the SSL connection	7-23
Crypto profile	7-24
Securing connections from client to appliance	7-25
Step 1: Appliance supplies cryptographic certificate	7-26
Step 2: Configuring SSL server crypto profile	7-27
Create an SSL server crypto profile	7-28
Securing connection from appliance to external application server	7-29
Step 1: Appliance validates presented certificate	7-30
Step 2: Configuring an SSL client crypto profile	7-31
The SSL proxy profile	7-32
SSL proxy profile when the appliance is the SSL server	7-33
SSL proxy profile when the appliance is the SSL client	7-34
SSL Proxy Profile list	7-35
User agent	7-36
User Agent defined	7-37
Configuring a user agent	7-38
Create a user agent configuration	7-39
Unit summary	7-40
Checkpoint questions	7-41
Checkpoint answers	7-42
Exercise 5	7-43
Exercise objectives	7-44
Exercise overview (1 of 2)	7-45

Exercise overview (2 of 2)	7-46
Unit 8. Web service proxy service	8-1
Unit objectives	8-2
Web service proxy overview	8-3
Web service virtualization	8-4
Web service proxy benefits	8-5
Web service proxy configuration tabs (1 of 2)	8-6
Web service proxy configuration tabs (2 of 2)	8-7
Web service proxy basic configuration steps	8-8
Step 1: Obtain WSDL document	8-9
WSDL structure	8-10
Step 2: Creating a web service proxy	8-11
An alternative: Web service proxy object editor	8-12
Step 3: Add WSDL document to web service proxy	8-13
Step 4: Configure WSDL endpoint	8-14
Step 5: Configure local endpoint handler	8-16
Step 6: Add the WSDL to the service	8-17
Initial WSDL completed	8-18
Other ways to get a WSDL	8-19
View WSDL services	8-20
Retrieve the client WSDL from the service	8-21
Modifying the location in the client WSDL	8-22
Step 7: Configuring web service proxy policy (optional)	8-23
Configure web service proxy policy rule	8-24
Adding a rule	8-25
Default validation (user policies)	8-26
Create reusable rule	8-27
Advanced web service proxy configuration	8-28
WS-Policy	8-29
Conformance policy	8-30
Conformance policy object	8-31
Service priority	8-32
Proxy settings (1 of 4)	8-33
Proxy settings (2 of 4)	8-34
Proxy settings (3 of 4)	8-36
Proxy settings (4 of 4)	8-37
Web service proxy: SLM Policy tab	8-38
WSDL cache policy	8-39
Troubleshooting a web service proxy	8-40
Unit summary	8-41
Checkpoint questions	8-42
Checkpoint answers	8-43
Exercise 6	8-44
Exercise objectives	8-45
Exercise overview	8-46
Unit 9. Service level monitoring	9-1
Unit objectives	9-2
Service level monitoring (SLM) in DataPower	9-3
The pieces of SLM (1 of 2)	9-4

Approaches to define SLM policies	9-5
Approach 1: Add an SLM action to a request rule	9-6
The pieces of SLM (2 of 2)	9-7
The SLM credential class	9-8
The SLM resource class	9-10
SLM resource class example	9-11
The SLM schedule	9-12
The SLM action	9-13
SLM statement (1 of 2)	9-14
SLM statement (2 of 2)	9-15
SLM policy: Main tab	9-17
SLM policy: Statements tab	9-18
Getting SLM statements into the Statement list	9-19
Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2)	9-20
Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2)	9-21
SLM Policy tab: Graphs	9-22
SLM action granularity	9-23
Unit summary	9-24
Checkpoint questions	9-25
Checkpoint answers	9-26
Exercise 7	9-27
Exercise objectives	9-28
Unit 10. XML and web services security overview	10-1
Unit objectives	10-2
Review of basic security terminology	10-3
Web services security	10-5
Components of WS-Security	10-6
Specifying security in SOAP messages	10-7
Scenario 1: Ensure confidentiality with XML encryption	10-8
XML encryption and WS-Security	10-9
DataPower support for XML encryption	10-10
Encrypt action	10-11
Decrypt action	10-13
Field-level encryption and decryption	10-14
XPath tool	10-15
Sample encrypted SOAP message	10-16
Scenario 2: Ensure integrity with XML signatures	10-17
DataPower support for XML signature	10-19
Sign action	10-20
Verify action	10-22
Verify action: Advanced tab	10-23
Field-level message signature and verification	10-24
Sample signed SOAP message	10-25
Summary of security and keys	10-26
Unit summary	10-27
Checkpoint questions	10-28
Checkpoint answers	10-29
Exercise 8	10-30
Exercise objectives	10-31
Exercise overview	10-32

Unit 11. Authentication, authorization, and auditing (AAA)	11-1
Unit objectives	11-2
Authentication, authorization, and auditing	11-3
Authentication and authorization framework	11-4
AAA action and access control policy	11-6
How to define an access control policy (1 of 2)	11-7
How to define an access control policy (2 of 2)	11-8
Access control policy processing	11-9
Scenario 1: Authorize authenticated clients	11-10
Scenario 1: Sample SOAP request message	11-11
Scenario 1: Identify the client	11-12
Scenario 1: Authorize access to resources	11-14
Scenario 2: Security token conversion	11-16
Scenario 2: Sample HTTP request message	11-17
Scenario 2: Identify the client	11-18
Scenario 2: Authorize access to resources	11-19
Scenario 3: Multiple identity extraction methods	11-20
Scenario 3: Identify the client	11-21
Scenario 3: Authorize access to resources	11-22
Internal access control resources	11-23
AAA XML file	11-24
Example AAA XML file	11-25
Lightweight Third Party Authentication	11-26
External access control resource	11-27
Lightweight Directory Access Protocol	11-28
Security Assertion Markup Language	11-29
Types of SAML assertions	11-30
Scenario 4: Authorize valid SAML assertions	11-31
Scenario 4: SAML authentication statement (1 of 2)	11-32
Scenario 4: SAML authentication statement (2 of 2)	11-33
Scenario 4: SAML attribute statement	11-34
Scenario 4: Identify the client	11-35
Scenario 4: Authorize access to resources	11-36
Scenario 4: Match SAML attributes	11-37
Access control policy by SAML information	11-38
Unit summary	11-39
Checkpoint questions	11-40
Checkpoint answers	11-41
Exercise 11	11-42
Exercise objectives	11-43
Exercise overview	11-44
 Unit 12. OAuth overview and DataPower implementation	 12-1
Unit objectives	12-2
What is OAuth?	12-3
Delegated authorization example	12-4
Example: Allow third-party access to social account	12-5
Example: Third-party access to online photo album	12-6
Before OAuth - sharing user passwords	12-7
OAuth - Three-legged scenario	12-8
OAuth 2.0 - Authorization code grant and access token	12-10

OAuth 2.0 roles	12-11
OAuth 2.0 roles in the DataPower world (1 of 2)	12-12
OAuth 2.0 roles in the DataPower world (2 of 2)	12-13
Sample three-legged scenario in DataPower (1 of 4)	12-14
Sample three-legged scenario in DataPower (2 of 4)	12-15
Sample three-legged scenario in DataPower (3 of 4)	12-16
Sample three-legged scenario in DataPower (4 of 4)	12-17
Authorization request	12-18
Authorization response	12-19
Access token request	12-20
Access token response	12-21
Resource request	12-22
OAuth client and the OAuth Client Profile object	12-23
DataPower OAuth objects: OAuth Client (1 of 4)	12-24
DataPower OAuth objects: OAuth Client (2 of 4)	12-26
DataPower OAuth objects: OAuth Client (3 of 4)	12-27
DataPower OAuth objects: OAuth Client (4 of 4)	12-28
DataPower OAuth objects: OAuth Client Group	12-30
DataPower OAuth objects: Web Token Service	12-31
AAA policy: Key to OAuth behavior	12-32
AAA policy for the web token service	12-33
AAA policy for the resource server (1 of 2)	12-34
AAA policy for the resource server (2 of 2)	12-35
Unit summary	12-36
Checkpoint questions	12-37
Checkpoint answers	12-38
Exercise 10	12-39
Exercise objectives	12-40
Exercise overview (1 of 3)	12-41
Exercise overview (2 of 3)	12-42
Exercise overview (3 of 3)	12-43
Unit 13. Patterns for service configuration	13-1
Unit objectives	13-2
What is a pattern?	13-3
Creating a pattern	13-4
Deploying a pattern	13-5
The pattern console	13-6
Getting to the pattern console	13-7
The pattern console	13-8
The pattern console menu bar	13-9
The pattern console "Getting Started" tab	13-10
The pattern console "Endpoints" tab	13-11
The pattern console "Patterns" tab	13-12
The pattern toolbar	13-13
Steps to generate a service from a pattern	13-14
Deployment: selecting a pattern	13-15
Deployment: filling out the wizard	13-16
Points of variability	13-17
Unit summary	13-18
Checkpoint questions	13-19

Checkpoint answers	13-20
Exercise 11	13-21
Exercise objectives	13-22
Exercise overview	13-23
 Unit 14. Integration with WebSphere MQ	 14-1
Unit objectives	14-2
WebSphere MQ fundamentals	14-3
WebSphere MQ message	14-4
Transactions	14-5
DataPower support for WebSphere MQ	14-6
Provide WebSphere MQ access	14-7
Step 1: Create a WebSphere MQ queue manager (1 of 2)	14-8
Step 1: Create a WebSphere MQ queue manager (2 of 2)	14-9
Step 1: Use SSL in mutual authentication mode	14-10
Step 2: Add a WebSphere MQ front side handler	14-11
Step 3: Configure a WebSphere MQ back-end transport	14-12
Publish/subscribe: WebSphere MQ front side handler support	14-13
Publish/subscribe: WebSphere MQ back-end transport support	14-14
Message properties	14-15
Ordered processing of WebSphere MQ messages	14-16
Controlling backout of WebSphere MQ messages	14-18
Decision tree for the backout settings	14-19
WebSphere MQ header action in service policy	14-20
Typical uses of a WebSphere MQ header action	14-21
Transactions and WebSphere MQ	14-22
WebSphere MQ front side transactions	14-23
WebSphere MQ back side transactions	14-24
WebSphere MQ DataPower URL	14-25
WebSphere MQ queue manager Group object	14-26
Unit summary	14-27
Checkpoint questions	14-28
Checkpoint answers	14-29
 Unit 15. DataPower and WebSphere JMS	 15-1
Unit objectives	15-2
Messaging middleware	15-3
DataPower support of messaging middleware	15-4
Java Message Service	15-5
JMS models	15-6
WebSphere service integration bus (SIBus)	15-7
JMS queue resources on SIBus	15-8
JMS topic resources on SIBus	15-9
WebSphere JMS support	15-10
DataPower and WebSphere JMS interaction	15-11
WebSphere JMS object: Main (1 of 2): Messaging bus	15-12
WebSphere JMS object: Main (2 of 2): Optional settings	15-13
WebSphere JMS object: WebSphere JMS Endpoint	15-15
Communicating to WebSphere JMS	15-16
WebSphere JMS Front Side Handler	15-17
WebSphere JMS back-end URL	15-18

TIBCO EMS support	15-19
Ordered processing of JMS messages	15-20
Unit summary	15-22
Checkpoint questions	15-23
Checkpoint answers	15-24
Exercise 12	15-25
Exercise objectives	15-26
Exercise overview	15-27
Unit 16. REST and JSON support for Web 2.0 and Mobile applications	16-1
Unit objectives	16-2
Alternatives to SOAP-based web services	16-3
Web SOA (Web 2.0) versus Enterprise SOA	16-4
Web SOA protocols and standards	16-5
Growth of mobile clients	16-6
DataPower as the reverse proxy for Web 2.0 / Mobile clients	16-7
Introduction to REST	16-8
REST style services	16-9
Example - Employee processing	16-10
Employee REST interface	16-11
Example: REST interaction	16-12
Example: Add employee REST request explained	16-13
Example: Add employee REST response explained	16-14
Common DataPower REST patterns: Façade	16-15
Common DataPower REST patterns: Bridge	16-16
Common DataPower REST patterns: REST enrichment	16-17
Tooling to support REST: service or protocol handler related	16-18
Front side handler support of HTTP method selection	16-19
JSON request and response types	16-20
Process bodyless messages	16-21
JSON threat protection	16-22
Tooling to support REST: service policy related	16-23
Matching Rule on HTTP methods	16-24
Changing the HTTP method in the processing rule	16-25
Convert Query Params to XML action	16-26
Convert Query Params to XML action example	16-27
Programmatic access to the HTTP method	16-28
Programmatic access to the HTTP status code	16-29
JSON	16-30
JSON data types	16-31
JSON version of XML	16-32
JSONNx	16-33
JSONNx version of JSON data structure	16-34
Handling JSON in the request body	16-35
Converting XML to JSON	16-36
Validate action for JSON	16-37
Example JSON and a JSON schema	16-38
Transform action that uses XQuery (JSON and XML)	16-39
XQuery/JSONiq example	16-40
Sample service policy: bridging REST and SOAP	16-41
Unit summary	16-42

Checkpoint questions	16-43
Checkpoint answers	16-44
Exercise 13	16-45
Exercise objectives	16-46
Exercise overview	16-47
Unit 17. Course summary	17-1
Unit objectives	17-2
Course learning objectives	17-3
Course review (1 of 3)	17-4
Course review (2 of 3)	17-5
Course review (3 of 3)	17-6
Lab exercise solutions	17-7
To learn more on the subject	17-8
More DataPower education opportunities	17-9
Unit summary	17-10
Appendix A. List of abbreviations.....	A-1
Appendix B. Resource guide.....	B-1

Trademarks

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

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

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

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

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

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

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

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

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

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

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

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

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

Course description

Accelerate, Secure and Integrate with IBM DataPower V6

Duration: 5 days

Purpose

In this 5-day course, you learn the fundamental skills that are required to implement IBM WebSphere DataPower SOA Appliances with firmware version 6.0.0.

The IBM WebSphere DataPower SOA Appliances allow an enterprise to simplify, accelerate, and enhance the security capabilities of its Extensible Markup Language (XML) and web services deployments, and extend the capabilities of its service-oriented architecture (SOA) infrastructure. The appliances also extend these capabilities into the REST and JSON application areas.

Through a combination of instructor-led lectures and hands-on lab exercises, you learn how to implement the key use cases for the DataPower appliances. These include web service virtualization, web services security, integrating with IBM WebSphere MQ and WebSphere Java Message Service (JMS), mobile and REST support, integrating with OAuth 2.0, and authentication, authorization, and auditing (AAA). You also learn how to use various problem determination tools such as logs, monitors, and probes, and also techniques for testing DataPower services and handling errors.

Hands-on exercises give you experience working directly with an IBM WebSphere DataPower SOA Appliance. The exercises focus on skills such as creating multi-protocol gateways and web service proxies, working with encryption and cryptographic objects, configuring service level monitoring, troubleshooting services, and handling errors.

Audience

This course is designed for integration developers who configure service policies on IBM WebSphere DataPower SOA Appliances.

Prerequisites

Before taking this course, you should successfully complete course VW600, Technical Introduction to WebSphere DataPower V6. You should also be familiar with:

- Security-based concepts and protocols
- XML-related technologies such as XML schema, XPath, and XSLT

- Web service fundamentals and the web services security specifications
- REST-based services

Objectives

After completing this course, students should be able to:

- Describe how WebSphere DataPower SOA Appliances are configured
- Create a web service proxy to virtualize web service applications
- Implement web services security
- Create and configure cryptographic objects
- Configure Secure Sockets Layer (SSL) to and from WebSphere DataPower SOA Appliances
- Configure a multi-protocol gateway (MPGW) to handle multiple protocols for a single service
- Configure a service level monitoring (SLM) policy to control message traffic
- Configure support for IBM WebSphere MQ and WebSphere Java Message Service (JMS)
- Use logs and probes to troubleshoot services
- Configure the DataPower resources that are needed to support OAuth 2.0
- Use patterns to define and deploy new services
- Use DataPower resources and options to support REST and JSON-based services
- Handle errors in service policies

Agenda

Day 1

- Course introduction
- Unit 1: DataPower administration overview
- Exercise 1: Exercise setup
- Unit 2: DataPower services overview
- Unit 3: Structure of a service
- Exercise 2: Creating a simple XML firewall
- Unit 4: Multi-protocol gateway service

Day 2

- Unit 5: Problem determination tools
- Exercise 3: Creating an advanced multi-protocol gateway
- Unit 6: Handling errors in a service policy
- Exercise 4: Adding error handling to a service policy
- Unit 7: DataPower cryptographic tools and SSL setup
- Exercise 5: Creating cryptographic objects and configuring SSL

Day 3

- Unit 8: Web service proxy service
- Exercise 6: Configuring a web service proxy
- Unit 9: Service level monitoring
- Exercise 7: Implementing an SLM monitor in a web service proxy
- Unit 10: XML and web services security overview
- Exercise 8: Web service encryption and digital signatures

Day 4

- Unit 11: Authentication, authorization, and auditing (AAA)
- Exercise 9: Web service authentication and authorization
- Unit 12: OAuth overview and DataPower implementation
- Exercise 10: Defining a three-legged OAuth scenario that uses DataPower services
- Unit 13: Patterns for service configuration
- Exercise 11: Using a DataPower pattern

Day 5

- Unit 14: Integration with WebSphere MQ
- Unit 15: DataPower and WebSphere JMS
- Exercise 12: Configuring a multi-protocol gateway service with WebSphere MQ
- Unit 16: REST and JSON support for Web 2.0 and Mobile applications
- Exercise 13: Using DataPower to implement REST services
- Unit 17: Course summary

Unit 1. DataPower administration overview

What this unit is about

This unit introduces three management interfaces for the DataPower appliance: the WebGUI web application, the command-line interface (CLI), and the XML Management Interface. It also presents the basic resources that are needed by a developer: user account, user group, and domain.

What you should be able to do

After completing this unit, you should be able to:

- List the methods that can be used to administer WebSphere DataPower Appliances
- Manage user accounts and domains on the appliance

How you will check your progress

- Checkpoint



Unit objectives

After completing this unit, you should be able to:

- List the methods that can be used to administer WebSphere DataPower Appliances
- Manage user accounts and domains on the appliance

© Copyright IBM Corporation 2014

Figure 1-1. Unit objectives

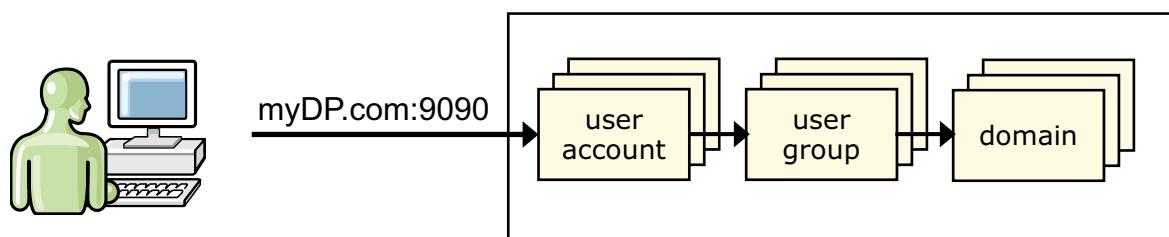
WE601 / ZE6011.1

Notes:

Developing on the appliance

To develop services on the appliance, you need

- IP address and port of the web management interface (WebGUI)
 - The WebGUI is the interface that is used for development
- User account and password
- User group: definition of permissions
- Domain: “sandbox” in which you develop services



“Administration” defines all of these resources

© Copyright IBM Corporation 2014

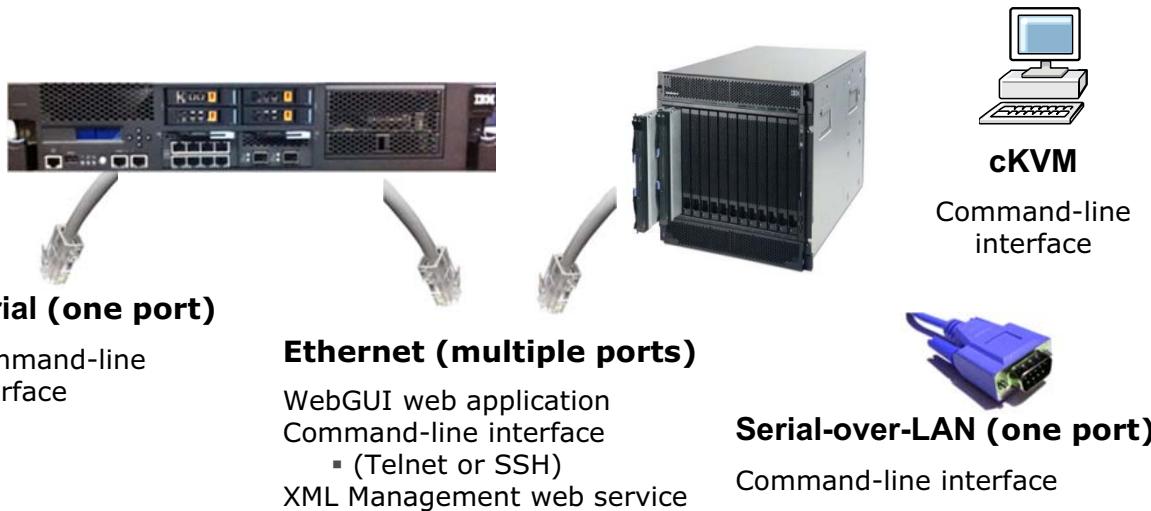
Figure 1-2. Developing on the appliance

WE601 / ZE6011.1

Notes:

DataPower SOA appliance administration

- Use one of the following interfaces to perform administration tasks on the DataPower SOA appliance:
 - WebGUI web application
 - Command-line interface (CLI)
 - SOAP-based XML Management API (SOMA, AMP, others)



© Copyright IBM Corporation 2014

Figure 1-3. DataPower SOA appliance administration

WE601 / ZE6011.1

Notes:

Because the XI50B does not have a physical serial port, the XI50B uses the serial-over-LAN (SOL) mechanism to emulate a CLI session that comes into the serial port. There is also a concurrent keyboard-video-mouse (cKVM) instance available that provides a CLI interface that uses a teletype-style interface (glass terminal). Both SOL and cKVM are part of the typical support in the BladeCenter.

Without configuration, only the serial connection is active for use. The administrator must enable the other three administration interfaces by using the command-line interface.

You can enable the WebGUI application, a CLI over Telnet or Secure Shell (SSH), the XML Management web service over one of the four Ethernet interfaces, or all Ethernet interfaces. Typically, the administration services are only available over an internal network connection while external traffic flows through the remaining Ethernet ports.

The number, types, and speeds of the Ethernet interfaces depends on the appliance model.

The serial port on the XG45, XI52, and XB62 uses an RJ-45 connector. It is the only port active when the appliance is first unpacked.



WebGUI web administration application

- The WebGUI web application allows administrators to configure and troubleshoot the DataPower SOA appliance
 - The WebGUI application must be activated through the command-line interface before its first use
 - Role-based management restricts access to predefined administrators
 - Same web interface that is used for service development

- Navigate to the network address and port that are assigned to the WebGUI application
 - The default port for the WebGUI application is 9090
 - Requires HTTPS for both administrative tasks and development work



© Copyright IBM Corporation 2014

Figure 1-4. WebGUI web administration application

WE601 / ZE6011.1

Notes:

Remember to enter the `https` protocol in front of the network host name or address for your DataPower appliance. The default value that is provided in the documentation is port 9090 for the WebGUI application. However, you are free to assign any port number in range for this administration interface.

The officially supported browsers for firmware 6.0.0 are:

- Microsoft Internet Explorer, Version 9 in compatibility mode is supported.
- Microsoft Internet Explorer Version 8 is supported.
- Mozilla Firefox, Version 17 Extended Support Release

Other versions and other browsers, such as Chrome, can be used, but there is no support for problems.



Administration by using the web browser

Intensive Level of Logging is enabled, which impacts performance. [Change Troubleshooting settings.](#)

Firmware: XI52.6.0.0.0
Build: 231528
IBM WebSphere DataPower
Copyright IBM Corporation 1999-2013
[View License Agreement](#)

© Copyright IBM Corporation 2014

Figure 1-5. Administration by using the web browser

WE601 / ZE6011.1

Notes:

The navigation bar provides access to configuration or management options.

The Control Panel allows quick access to common administration functions. The services section allows you to create or modify the primary DataPower services. The Pattern Console is used to create, view, edit, delete, and deploy patterns. This console is covered in the Patterns unit.

1. The navigation bar provides access to configuration or management options. The Control Panel icon and the Pattern Controls icon allow you to switch between displays. The Control Panel provides quick access to common administration functions.
2. The Services section allows you to create or modify the primary DataPower services.
3. The Monitoring and troubleshooting section provides a view of the DataPower SOA Appliance status, traffic, and load.
4. The Files and Administration section manages the configuration files, access levels, and cryptographic keys and certificates on the appliance.

All links on the Control Panel are also available through the navigation bar.

The Search field above the navigation bar is used for searching within the categories in the navigation bar.



Navigation bar categories



The screenshot shows the WebSphere Control Panel interface. On the left, there's a sidebar with 'Control Panel' and 'Pattern Console' links, a search bar, and a tree view with nodes for Status, Services, Network, Administration, and Objects.

Category	Description
Status	Provides access to real-time operational data maintained by the appliance's management system
Services	Configures services that accelerate, secure, and integrate XML-based applications
Network	Configures network services and interfaces and retrieve information about network connectivity
Administration	Provides access to troubleshooting, logging, access control, and file and configuration administration
Objects	Provides direct access to the <i>object store</i> that represents the configuration for the entire appliance

© Copyright IBM Corporation 2014

Figure 1-6. Navigation bar categories

WE601 / ZE6011.1

Notes:

The following set of slides focus on the administration features found in the WebGUI.



System control features (1 of 3)

The screenshot shows the 'System Control' page with the following sections and steps:

- Set Time and Date (Step 3):** Includes fields for Date (2012-07-05) and Time (23:14:10), and a 'Set Time and Date' button.
- Boot Image (Step 4):** Includes a checkbox for accepting license terms, a 'Firmware File' dropdown set to '(none)', and buttons for 'Upload...', 'Fetch...', 'Edit...', and 'View...'.
- Firmware Roll-Back (Step 5):** Includes a 'Firmware Roll-Back' button.
- Select Configuration (Step 6):** Includes a 'Configuration File' dropdown set to 'config:/// (none)', and buttons for 'Upload...', 'Fetch...', 'Edit...', and 'View...'.

© Copyright IBM Corporation 2014

Figure 1-7. System control features (1 of 3)

WE601 / ZE6011.1

Notes:

The **System Control** page groups several system-wide updates that affect the firmware, clock, and system certificate. Certain options are only available from the default domain.

System Control can be accessed by using the following steps:

1. Click **Control Panel > Administration > Main > System Control**.
2. Click the **System Control** icon on the Control Panel.
3. Use the time and date features to set the current time in your locale. To modify the time zone, click **Administration > Device > Time Settings** from the navigation bar. The DataPower appliance sets the clock in Coordinated Universal Time (UTC). You can also define a reference to the Network Time Protocol (NTP) server instead.
4. The Boot Image feature allows you to upgrade the system to a newer firmware level. Use the Upload function to copy a new firmware image onto the DataPower appliance. After the new firmware is loaded onto the appliance, click **Boot Image** to restart the DataPower appliance with the new firmware.

5. If you encounter problems when using a new firmware level, click **Firmware Roll-Back** to revert the DataPower appliance to the previous firmware level. The firmware rollback reverts to the previous firmware; it is not possible to revert to multiple firmware versions.
6. The **Select Configuration** section determines which configuration file is used on the next system restart.



System control features (2 of 3)

The screenshot shows the 'System Control' section of a web-based management interface. It includes four main sections: 'Secure Backup', 'Secure Restore', 'Shutdown', and 'Change User Password'. A legend on the left maps colors to actions:

- Yellow box:** Reload firmware
- Green box:** Reboot system
- Pink box:** Halt system

Callouts numbered 1 through 4 point to specific fields or buttons:

- Secure Backup:** Points to the 'Crypto certificate' dropdown.
- Secure Restore:** Points to the 'Source' field.
- Shutdown:** Points to the 'Mode' dropdown set to 'Reboot System'.
- Change User Password:** Points to the 'Old Password' field.

© Copyright IBM Corporation 2014

Figure 1-8. System control features (2 of 3)

WE601 / ZE6011.1

Notes:

This slide continues examining the system control page.

1. Use the **Secure Backup** option to create an encrypted backup of all files on the appliance, including keys and certificates.
2. Use the **Secure Restore** option to reload files onto an appliance from the encrypted backup.
3. Use the **Shutdown** option to restart the DataPower appliance in one of three modes:
 - **Reload firmware** restarts the device without rebooting the DataPower SOA appliance. Temporary files and applied, but unsaved, changes are kept intact.
 - **Reboot system** restarts the DataPower appliance. All temporary files and unsaved configuration changes are lost.
 - **Halt system** shuts down the DataPower appliance.
4. The **Change User Password** section allows you to assign a new password to the user who is logged in. If you are logged in as `admin`, this operation changes both the WebGUI and CLI passwords for the user.



System control features (3 of 3)

The screenshot shows a web-based interface for managing system control features. The features are listed vertically with numbered callouts:

- 1** Restart Domain
- 2** Reset Domain
- 3** Generate Device Certificate
 - Common Name (CN) input field
 - Generate Self-Signed Certificate radio buttons (on selected)
 - Generate Device Certificate button
- 4** Control Locate LED
 - Blue Locator LED radio buttons (off selected)
 - Control Locate LED button
- 5** Quiesce
 - Timeout input field (seconds)
 - Quiesce button
- 6** Unquiesce
 - Unquiesce button

© Copyright IBM Corporation 2014

Figure 1-9. System control features (3 of 3)

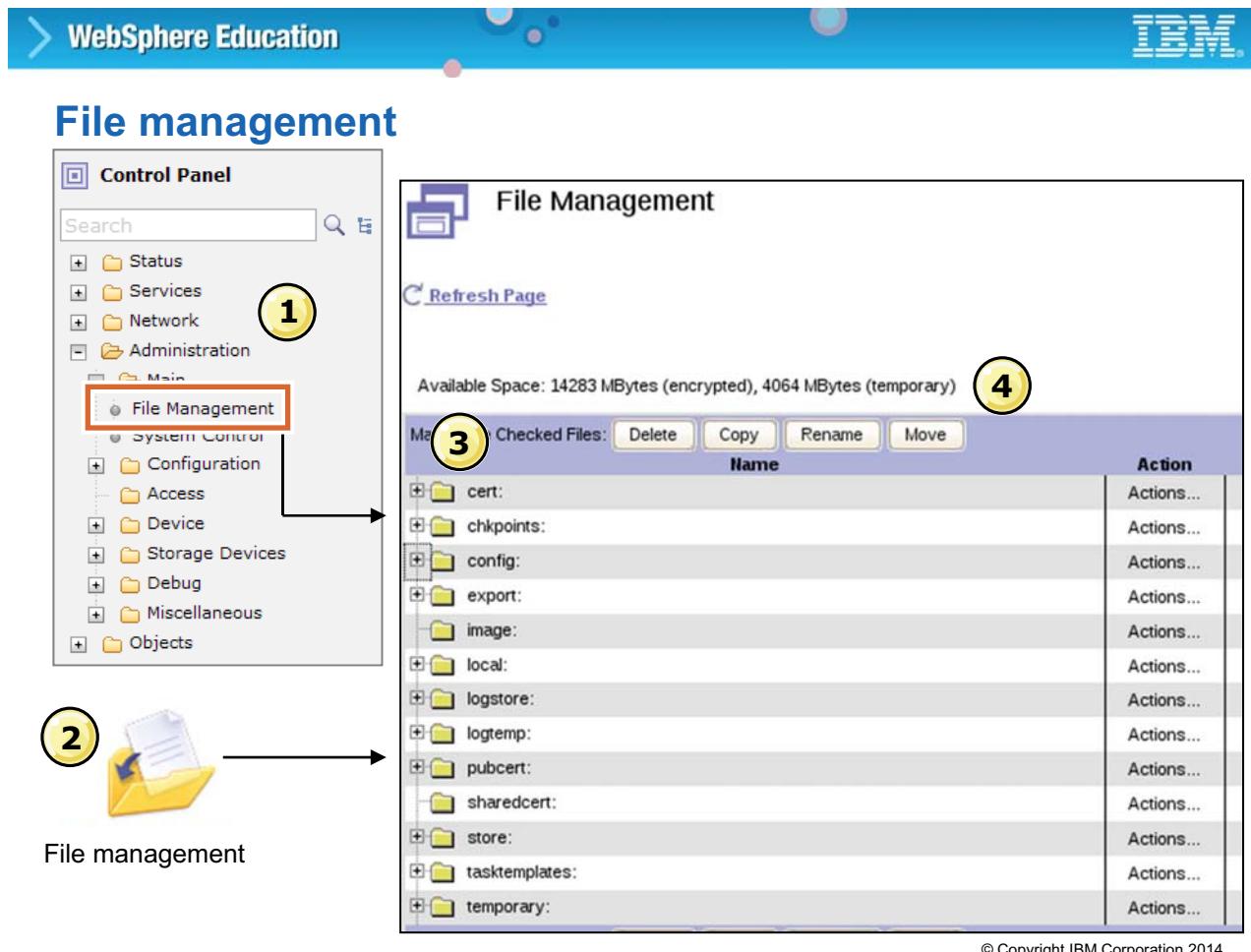
WE601 / ZE6011.1

Notes:

This slide finishes examining the system control page.

1. **Restart Domain** reloads the configuration for the current application domain. Any unsaved configuration changes are lost.
2. **Reset Domain** erases any configured objects in the domain. Be careful when using this feature. It does not delete any files in the file system.
3. **Generate Device Certificate** creates a digital certificate that represents the current DataPower appliance. The common name must be mapped to a valid IP address.
4. **Control Locate LED** controls a blue "locate" LED on the front panel of the physical appliance, or on the XI50B Blade and the chassis.
5. **Quiesce** blocks new transactions to the appliance, while allowing existing transactions to complete. This operation changes protocol handlers to the down operational state, changes services to the down operational state, and changes domains to the down operational state. When the default domain is quiesced, it does not change to the down operational state. Quiescing proceeds in a child-parent relationship, where protocol handlers are quiesced first, then the services, and then the domains.

6. **Unquiesce** returns the handler, service, domain, or appliance to an **up** state after a previous quiesce.



© Copyright IBM Corporation 2014

Figure 1-10. File management

WE601 / ZE6011.1

Notes:

1. From the navigation bar **Administration** section, click **Main > File Management**.
2. Alternatively, you can open the File Management page through the icon of the same name in the Control Panel.
3. The file stores are divided into different directories. Most directories are specific to one application domain, except for the `store`, `pubcert`, and `sharedcert` directories.
4. The available space statistics display the amount of nonvolatile memory available for all encrypted data and all temporary data in the system.

File directories for configuration

Store	Scope	Usage
config:	Per application domain; not shared	Stores configuration files for the current application domain
export:	Per application domain; not shared	Holds any exported configuration that is created with the Export Configuration operation
local:	Per application domain; possibly visible to other domains	Storage space for files that are used by local services, including XML style sheets, XML schemas, and WSDL documents <ul style="list-style-type: none"> Use the visible domains setting to view the local file store of other application domains
store:	System-wide; shared	Sample and default style sheets that are used by DataPower services <ul style="list-style-type: none"> A common practice is to copy these style sheets into your local directory before you make any changes
temporary:	Per application domain; not shared	Temporary disk space that is used by document processing rules and actions, and is cleared on an appliance restart

© Copyright IBM Corporation 2014

Figure 1-11. File directories for configuration

WE601 / ZE6011.1

Notes:

When auxiliary storage is enabled, it is accessible as a subdirectory of the **local:** and the **logstore:** directories.

File directories for security

Store	Scope	Usage
cert:	Per application domain; not shared	Location to store private keys and digital certificates <ul style="list-style-type: none"> • System automatically encrypts all files in this store • After being added, files cannot be copied or modified • You can delete digital certificates and private keys
sharedcert:	System-wide; shared between application domains	Stores digital certificates to be shared with business partners <ul style="list-style-type: none"> • System automatically encrypts all files in this store
pubcert:	System-wide; shared between application domains	Provides security certificates for root certificate authorities, such as the ones used by web browsers <ul style="list-style-type: none"> • System automatically encrypts all files in this store • Files cannot be modified, but they can be copied

© Copyright IBM Corporation 2014

Figure 1-12. File directories for security

WE601 / ZE6011.1

Notes:

If you specify Disaster Recovery mode on the first initialization of an appliance or blade, there are certain situations in which you can export the keys.

File directories for logging

Store	Scope	Usage
logtemp:	Per application domain; not shared	Default location of log files, such as the system-wide default log <ul style="list-style-type: none"> • The file store size is fixed at 13 Mb
logstore:	Per application domain; not shared	Long-term storage space for log files

© Copyright IBM Corporation 2014

Figure 1-13. File directories for logging

WE601 / ZE6011.1

Notes:

When auxiliary storage is enabled, it is accessible as a subdirectory of the **local:** and the **logstore:** directories.

More file directories

Store	Scope	Usage
audit:	Default domain	Stores the audit log Available from the CLI in the default domain only
checkpoints:	Per application domain; not shared	Contains the checkpoint configuration files
dpcert:	Default domain	Encrypted directory that contains files that are used by the appliance Available from CLI in the default domain only
image:	Default domain	Contains the primary and rollback firmware
tasktemplates:	Default domain	XSL files that are used by the WebGUI

© Copyright IBM Corporation 2014

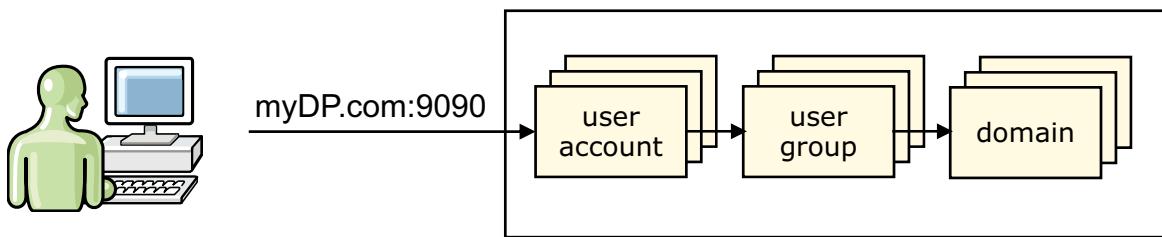
Figure 1-14. More file directories

WE601 / ZE6011.1

Notes:

Administrative access control

- **Application domains** provide a virtualized, enclosed environment for services
 - Only the **default** domain allows administrators to perform system level tasks, such as configuring an Ethernet interface
- **User groups** apply a specific access policy to a set of user accounts
 - **Privileged** access allows users to perform system-level tasks
 - **User** access provides read-only guest access
 - **Group-defined** relies on a user-defined, fine-grained access policy for each resource
- **User accounts** provide users with access to the WebGUI



© Copyright IBM Corporation 2014

Figure 1-15. Administrative access control

WE601 / ZE6011.1

Notes:

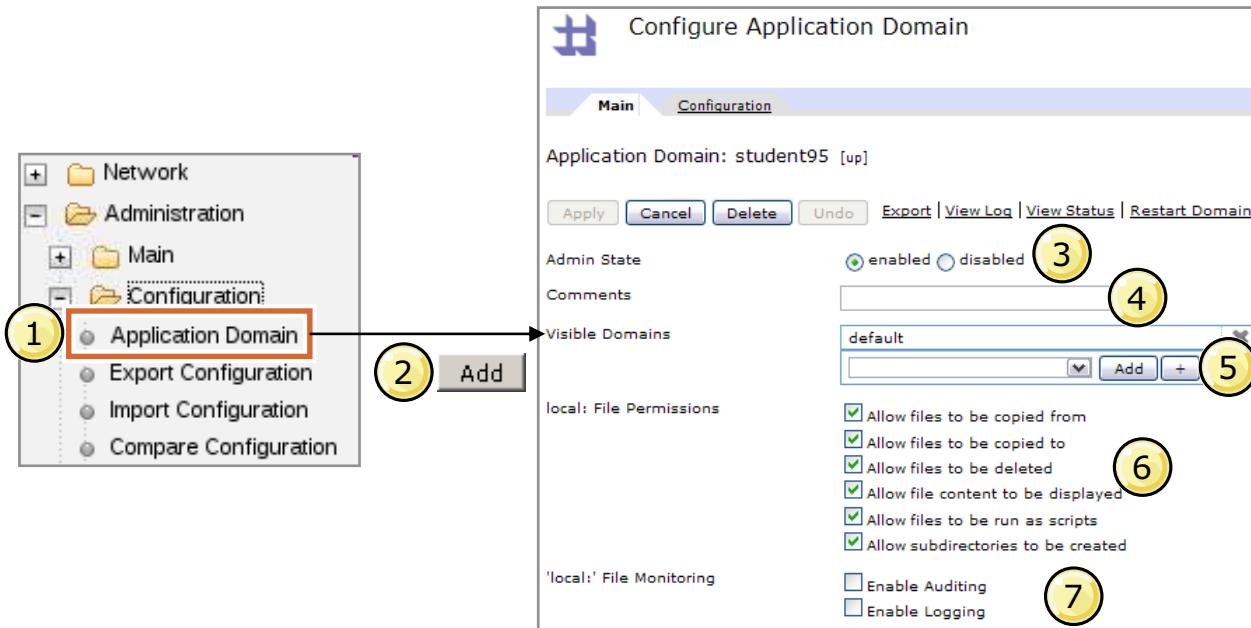
Users can also access more than one application domain by using the visible domain setting for application domains.

Privileged and user access levels represent the highest and lowest access levels on the DataPower SOA appliance. The group-defined setting allows an administrator to fine-tune the access level within either end of the spectrum.

User accounts that are created through the WebGUI interface apply to the CLI and XML Management Interface as well.

WebSphere Education

Create an application domain



© Copyright IBM Corporation 2014

Figure 1-16. Create an application domain

WE601 / ZE6011.1

Notes:

1. From the WebGUI navigation bar, expand the **Administration** section and click **Configuration > Application Domain**.
2. In the listing of available application domains (not shown), click **Add**. Provide a name for the new application domain; this field is mandatory.
3. Leave the **Admin State** at **enabled**. This administration state setting determines whether a particular DataPower object is available for use.
4. Enter any appropriate comments.
5. The visible domains setting determines whether this domain can access files in the **local:** file store of another application domain. In the figure, the **student95** domain can access the files in the **local** file store of the **default** domain.
6. Local file permissions determine the access rights to files stored in the local file store of the current domain.
7. When enabled, changes to files in the local file store generate auditing or logging events.

You use the **Configuration** tab to specify whether the configuration is stored locally or imported from a specified URL every time the configuration is saved or the system is restarted.



Application domain: Configuration tab

- The **Configuration** tab specifies the location to retrieve the domain configuration
 - Local** indicates that the configuration files are on the local appliance file system
 - Import** indicates that the configuration files are on a remote server
- You can set the limit on the number of Configuration Checkpoints that can be saved at one time

Configure Application Domain

Main Configuration

Application Domain:student15_domain [up]

Apply Cancel Delete Undo

Configuration Checkpoint Limit: 3

Configuration Mode: Import (selected)

Import URL

Import Format: ZIP

Deployment Policy: (none)

Deployment Policy Variables: (none) +

Local IP Rewrite: on (radio button selected)

© Copyright IBM Corporation 2014

Figure 1-17. Application domain: Configuration tab

WE601 / ZE6011.1

Notes:

Remote configuration allows you to have multiple appliances retrieve the same version of the domain configuration.



Configuration Checkpoints

- A Configuration Checkpoint contains configuration data for an application domain at a specific point in time
 - Saves the current state of the application domain without persisting it
 - An alternative to Save Config
 - Can be used for continuing work between sessions
- Saving Configuration Checkpoints
 - Click **Administration > Configuration > Configuration Checkpoints**
 - Enter a name and click **Save Checkpoint**

The screenshot shows a web-based administrative interface titled "Configuration Checkpoints". At the top left is a small icon of two overlapping windows. Below the title is a "Refresh List" button. The main area has a table with three columns: "Name", "Time", and "Actions". One row in the table is visible, showing "Checkpoint1" and the timestamp "2012-10-03 22:12:35 GMT" in the "Time" column. In the "Actions" column, there are three buttons: "Rollback", "Remove", and "Compare". Below the table is a section titled "Create a new Configuration Checkpoint". It contains a text input field labeled "Checkpoint Name:" and a "Save Checkpoint" button.

© Copyright IBM Corporation 2014

Figure 1-18. Configuration Checkpoints

WE601 / ZE6011.1

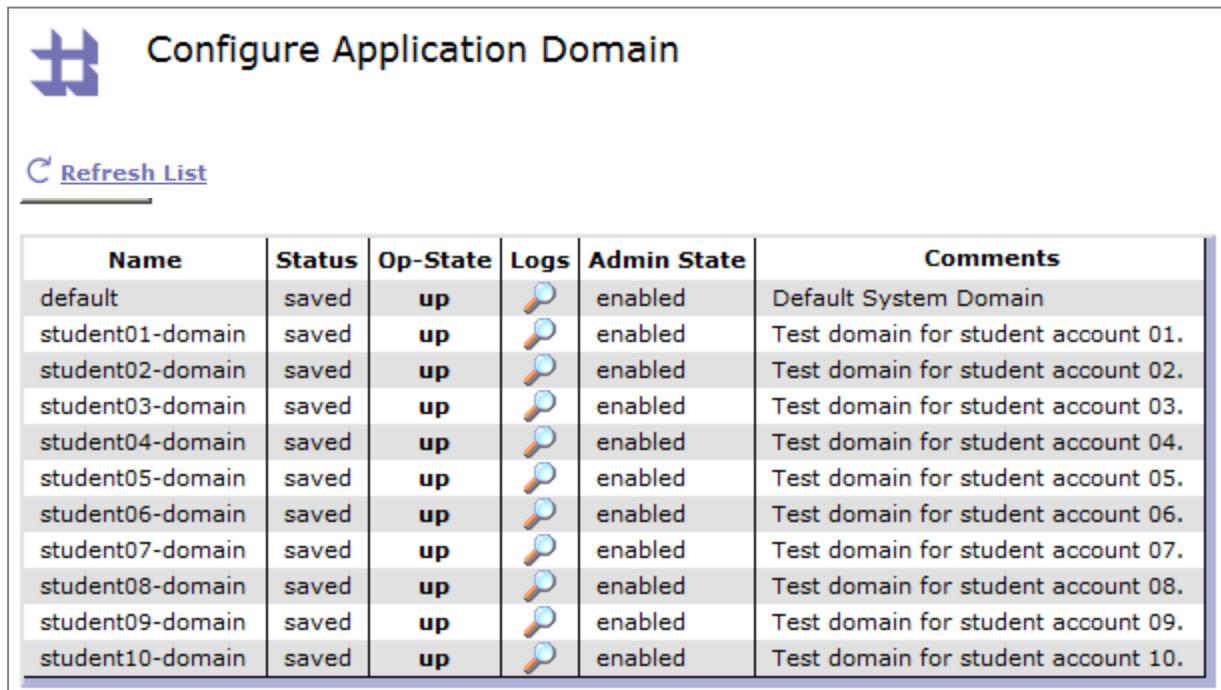
Notes:

Configuration checkpoints can also be used as a form of a rollback for a single domain.

Existing checkpoints can be removed, compared, or rolled back (that is, redefine the domain configuration).

The logo for WebSphere Education, featuring the IBM globe icon and the text "WebSphere Education".

View application domain status

A screenshot of a web-based application titled "Configure Application Domain". It features a "Refresh List" button and a table listing ten application domains. The columns are: Name, Status, Op-State, Logs, Admin State, and Comments. The domains listed are: default, student01-domain, student02-domain, student03-domain, student04-domain, student05-domain, student06-domain, student07-domain, student08-domain, student09-domain, and student10-domain. All domains are in a "saved" status and "up" op-state, with "enabled" admin state and logs. The comments column provides a brief description for each domain.

Name	Status	Op-State	Logs	Admin State	Comments
default	saved	up	🔍	enabled	Default System Domain
student01-domain	saved	up	🔍	enabled	Test domain for student account 01.
student02-domain	saved	up	🔍	enabled	Test domain for student account 02.
student03-domain	saved	up	🔍	enabled	Test domain for student account 03.
student04-domain	saved	up	🔍	enabled	Test domain for student account 04.
student05-domain	saved	up	🔍	enabled	Test domain for student account 05.
student06-domain	saved	up	🔍	enabled	Test domain for student account 06.
student07-domain	saved	up	🔍	enabled	Test domain for student account 07.
student08-domain	saved	up	🔍	enabled	Test domain for student account 08.
student09-domain	saved	up	🔍	enabled	Test domain for student account 09.
student10-domain	saved	up	🔍	enabled	Test domain for student account 10.

© Copyright IBM Corporation 2014

Figure 1-19. View application domain status

WE601 / ZE6011.1

Notes:

The main application domain page lists all configured domains on the DataPower appliance. This page is only visible from the **default** domain.

The screenshot shows three sequential steps in a wizard:

- Step 1:** "Create a user account" (highlighted with a yellow circle containing the number 1). It asks if the user should be restricted to a domain (deprecated). A note says selecting 'Yes' restricts the user to a domain, while 'No' allows login to all domains. Buttons: Yes, No, Cancel.
- Step 2:** "Create a user account" (highlighted with a yellow circle containing the number 2). It asks to which domain the user should be restricted (deprecated). It includes a dropdown for "User Domain" with "(none)" selected, and buttons: Back, Next, Cancel.
- Step 3:** "Create a user account" (highlighted with a yellow circle containing the number 3). It asks what kind of user account to create. It lists "Domain Account Type" options: Developer (configuring services in a domain) (selected), Backup User (domain backup), Guest (read-only in domain), and User-Defined Group. Buttons: Back, Next, Cancel.

© Copyright IBM Corporation 2014

Figure 1-20. Create a user account and a user group

WE601 / ZE6011.1

Notes:

The New User Account wizard allows you to create a user account, a domain, and a user group at the same time. To access this wizard, click **Administration > Access > New User Account** from the Navigation bar.

1. The first page asks whether you would like to restrict access for this user to a specific domain. If so, the wizard creates a user group to restrict access permissions within the application domain.
2. With a restricted user domain, you have the option of either selecting one of the existing user domains or creating an application domain.
3. With an application domain selected, you can choose from one of three preconfigured domain account types. For greater flexibility, either select an existing user group or create your own group.

The remaining wizard pages finalize the settings for the user account and the user group.

The **user** account type is deprecated in version 6.0. Although deprecated, the **user** account type provides the user with access to view configuration details to most, but not all, data.

Configure User Group

User Group: developer_student95 [up] 1

Main CLI Command Groups

Admin State: enabled 2

Comments: Developer in student95

Access Profile: */student95/*?Access=rwadx 2

Permissions:

- Read
- Write
- Add
- Delete
- Execute

Access Profile property syntax:
`address/domain/resource?Access=permissions& [field=value]`

© Copyright IBM Corporation 2014

Figure 1-21. Manage user group details

WE601 / ZE6011.1

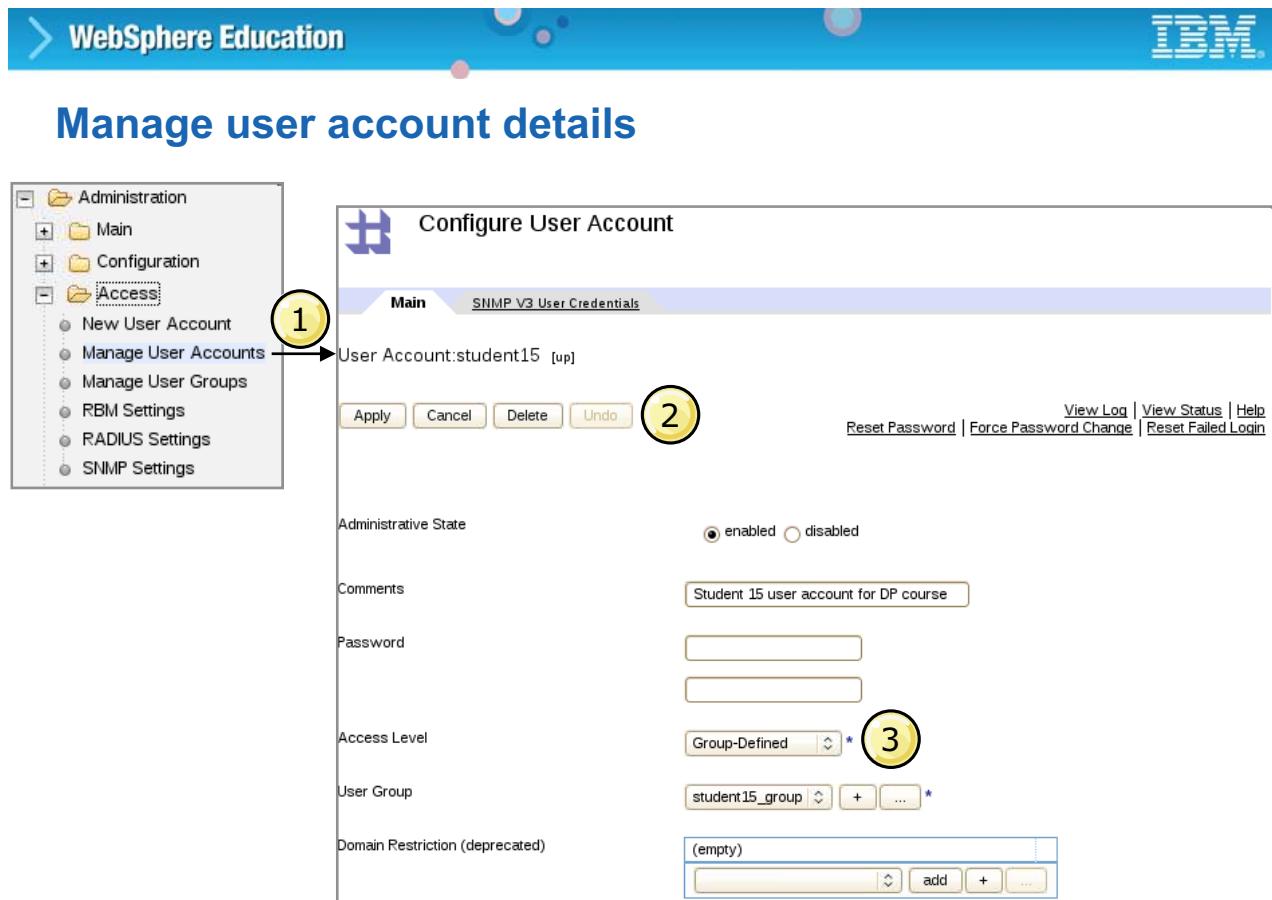
Notes:

1. User groups provide a convenient way for applying one or more access profiles to a set of user accounts. The access profile policy syntax restricts the access permission of any user to which the user group is applied. If two access profile policies affect the same resource, the most specific policy is applied.
- **address** refers to the DataPower appliance host name, IP address, or local host alias.
- **domain** specifies the name of one particular application domain.
- **resource** represents one type of DataPower object within the configuration.
- **permissions** are **r** (read), **w** (write), **a** (add), **d** (delete), or **x** (execute).
- **field** and **value** allow you to specify a particular object, such as the name of a web service proxy.

In the figure that is shown, users in the group have read, write, add, delete, and execute permissions for all resources in the student95 domain.

2. Click **Build** to use a graphical form to build the access profile policy.

The **CLI Command Groups** tab allows you to specify which sets of command-line interface commands are available to users within the user group.



© Copyright IBM Corporation 2014

Figure 1-22. Manage user account details

WE601 / ZE6011.1

Notes:

From the navigation bar **Administration** section, click **Access > Manage User Accounts**.

1. Clicking this selection presents a list of current user accounts. Select the appropriate account.
2. The Configure User Account page allows you to modify the comments and the password for a specified user in the application domain.
3. Select one of three access level settings: privileged, user, or group-defined.

The **Domain Restriction** list limits which domains a user account can access. An existing access policy, or an RBM access policy can supersede this list. Notice that this setting is deprecated.

The **SNMPv3 User Credentials** tab allows you to associate SNMP users with the current user account. SNMP users are granted access to the local management information base (MIB) for monitoring and configuring the DataPower appliance.



Export the system configuration

The screenshot shows the WebSphere Control Panel interface. On the left, there's a sidebar with links like 'Control Panel', 'Pattern Console', and a search bar. The main area is titled 'Export Configuration'. It contains a section for 'Export' with four options: 'Copy or move configuration and files between domains', 'Create a backup of one or more application domains', 'Create a backup of the entire system', and 'Export configuration and files from the current domain'. The fourth option is selected and highlighted with a red box. At the bottom are 'Next' and 'Cancel' buttons.

- The **Export Configuration** feature exports the definition of objects, services, application domains, user groups, and user accounts
 - Use the administrator account to export the system configuration
- Export configurations at a particular scope:
 - Entire system
 - One or more application domains
 - Specific configured objects and files in the current domain

© Copyright IBM Corporation 2014

Figure 1-23. Export the system configuration

WE601 / ZE6011.1

Notes:

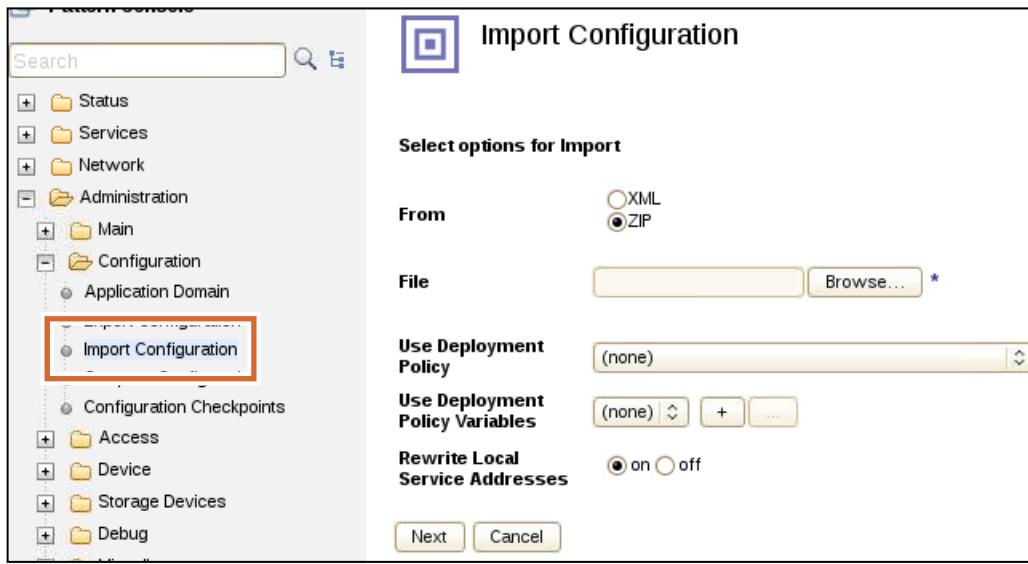
Use the export configuration command to back up the current configuration or to duplicate services and settings in other application domains. The export configuration command writes a series of XML files that follow the DataPower XML Management schema. In the last step of the Export Configuration page, you have an opportunity to download the .zip file that contains the XML configuration files. Alternatively, you can retrieve the configuration files from the `export:` file store that is associated with the current domain.

Since certain certificates, keys, and objects are visible only to the administrator, log in to the administrator account before performing an export operation.



Import a system configuration

- The **Import Configuration** feature updates the domain configuration with a previously saved version
 - Useful for duplicating configured services from one application domain to another
 - Administrators and developers must confirm changes that overwrite already configured services and interfaces



© Copyright IBM Corporation 2014

Figure 1-24. Import a system configuration

WE601 / ZE6011.1

Notes:

The import configuration feature accepts only DataPower XML Management documents as an XML file or as a .zip file.

A deployment policy allows an imported configuration to be preprocessed, and certain properties to be modified.

Deployment policy variables allow the externalization of the substitution values within the deployment policy.

Rewriting local service addresses updates the local service bindings to the equivalent interfaces in the imported configuration.

The screenshot shows the WebSphere Education interface. At the top, there's a blue header bar with the IBM logo on the right. Below it, the main title "Saving configuration changes" is displayed. In the center, there's a dialog box with three buttons: "Apply" (highlighted with a red box), "Cancel", and "Delete". The dialog content reads: "Configuration changes have not been saved! Please choose one of the following options:" followed by four options: "Save changes and switch domain.", "Review changes before continuing.", "Switch domain without saving.", and "Cancel (stay in this domain)".

Below the dialog, the main interface shows a "Domain: student01-domain" dropdown and two buttons: "Save Config" (highlighted with a red box) and "Logout".

- Configuration changes take effect after you click **Apply**
 - Remember to click **Apply** on each web page
 - A warning window appears if you attempt to switch application domains or log out of the WebGUI without saving applied changes
- Click **Save Config** on the upper-right corner of the web page to commit changes to the file system

© Copyright IBM Corporation 2014

Figure 1-25. Saving configuration changes

WE601 / ZE6011.1

Notes:

The **Apply** button submits configuration changes that are made in the current WebGUI page. However, such changes are stored in temporary memory. You must click **Save Config** on the upper-right corner of the WebGUI interface to commit changes to permanent storage. If you attempt to switch application domains without committing your changes, a warning dialog box is shown, allowing you to switch domains without saving any changes, or to save the changes immediately.

Administration by using the command-line interface

- The command-line interface (CLI) provides a text terminal for administering the DataPower appliance
 - For security purposes, the CLI is not a complete command shell with the ability to execute arbitrary programs
 - The CLI allows you to configure every service and interface available in the DataPower appliance
 - In the initial setup, you must enable the WebGUI application and Ethernet ports with the CLI through a serial connection
 - Administrators have the option of enabling the CLI over a Telnet or Secure Shell (SSH) connection

© Copyright IBM Corporation 2014

Figure 1-26. Administration by using the command-line interface

WE601 / ZE6011.1

Notes:

For security purposes, the CLI was not designed to be a generic command shell environment. Its function is strictly limited to the configuration and administration of the DataPower appliance. Nonetheless, it is a powerful interface that has access to all of the services and interfaces on the appliance itself.

By default, the DataPower SOA appliance is disabled and included with all Ethernet interfaces. To activate the ports, you must enable the interfaces within the CLI over a serial port connection (the XI50B uses cKVM or SOL). The WebGUI administration web application must also be enabled in this way before it is used.

After the DataPower appliance is properly configured, you can allow Telnet or Secure Shell connections to the CLI.



First boot after unpacking

- The CLI requires you to create a new password (Note: **Do not forget it**)
- License needs to be reviewed and accepted
- Several operating mode questions are presented, depending on model and series:
 - Disaster recovery (secure backup allowed)
 - Common Criteria EAL4 (Evaluation Assurance Level) required
 - Not configurable after setting (requires a reinitialization to change settings)
- Installation wizard is displayed
 - Network information
 - Name servers and gateways
 - WebGUI and remote CLI access
 - RAID auxiliary storage setup

© Copyright IBM Corporation 2014

Figure 1-27. First boot after unpacking

WE601 / ZE6011.1

Notes:

The installer must go through a series of steps during the first boot of the appliance (or blade) after it is removed from the shipping container.

All of the specifications asked for in the installation wizard are also configurable by using regular CLI commands.

WebSphere Education

IBM

CLI login screen

```

login: admin 1
Password: *****
Domain (? for all): default

Welcome to DataPower XI52 console configuration.
Copyright IBM Corporation 1999-2013

Version: XI52.6.0.0.3 build 236646 on 2013/11/11 11:04:50 2
Serial number: 6XXXXXXX

xi52# show system 3

description: DataPower XI52
serial number: 6XXXXXXX
entitlement id: 6XXXXXXX
product id: 719942X [Rev 0001]
OID: 1.3.6.1.4.1.14685.1.3
uptime: 6 days 22:30:52
contact: Jim Brown/Los Angeles

```

© Copyright IBM Corporation 2014

Figure 1-28. CLI login screen

WE601 / ZE6011.1

Notes:

You must provide a valid user login and password to access the command-line interface. After reviewing and accepting the license agreement (which is not shown), you must provide a new password for the admin account.

The initial welcome message displays the firmware build level and date, and the serial number of the appliance.

Use the `show` command to display system information about the interfaces, objects, and the appliance itself.

Quick initial configuration procedure

- Enable the WebGUI application over the management interface in the global configuration mode

```

xi52# configure terminal
Global configuration mode
xi52(config)# interface mgt0
Interface configuration mode (mgt0)
xi52(config-if[mgt0])# ip address 10.0.1.1/8
xi52(config-if[mgt0])# exit
xi52(config)# web-mgmt 10.0.1.1 9090
Web management: successfully started
xi52(config)# ssh 10.0.1.1 22
%
      Pending

SSH service listener enabled
xi52(config)# exit
xi52#

```

- 1
- 2
- 3
- 4
- 5

© Copyright IBM Corporation 2014

Figure 1-29. Quick initial configuration procedure

WE601 / ZE6011.1

Notes:

After logging on to the DataPower appliance for the first time over a serial connection, perform these steps to enable the WebGUI web application over the management port (mgt0).

1. While logged in as the administrator, enter the global configuration mode.
2. Configure the management Ethernet interface (mgt0).
3. Assign a static IP address and a subnet mask for the management Ethernet port.
4. In the global configuration mode, create an HTTP server with the WebGUI web application (web-mgmt).
5. For convenience, enable CLI access over SSH on the designated port.

These steps are performed as part of the installation wizard.

Retrieve system information by using the CLI

- **show version**
 - Returns the serial number, firmware level and build date, XML accelerator version, and more libraries
- **show services**
 - Returns a list of all active DataPower services and their respective ports
- **show users**
 - Lists all users who are currently logged in to the appliance
- **show log**
 - Returns the default log file
- **show startup-config**
 - Displays the configuration, in CLI commands, that the device used when it was last booted or restarted
- **show route**
 - Displays the device routing table

© Copyright IBM Corporation 2014

Figure 1-30. Retrieve system information by using the CLI

WE601 / ZE6011.1

Notes:

Administration by using SOAP: SOMA and AMP

- SOAP Configuration Management (SOMA) supports administration commands through a web service
 - The web service provides only one generic operation: **request**
 - The **request** operation takes one parameter, which maps to an administration category
 - Within each category, your client can issue multiple administration actions
 - The response from the web service call provides the results of each administration action call
- Appliance Management Protocol (AMP) is focused on appliance management
 - Uses a SOAP format
 - Has capabilities not available in SOMA
 - Used by tools such as WebSphere Appliance Management Toolkit and WebSphere Appliance Management Center
- Some overlap between functions, but different format for the commands

© Copyright IBM Corporation 2014

Figure 1-31. Administration by using SOAP: SOMA and AMP

WE601 / ZE6011.1

Notes:

Many different endpoints are defined for the XML Management Interface. The most popular are the SOMA and AMP endpoints. The other endpoints are:

- SLM Endpoint (service level management)
- WS-Management Endpoint
- WSDM Endpoint (web services distributed management)
- UDDI Subscription (Universal Description, Discovery, and Integration registry)
- WSRR subscription (WebSphere Service Registry and Repository)

SOMA: Create a user account

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <dp:request
      xmlns:dp="http://www.datapower.com/schemas/management">
      <dp:set-config>
        <User name="student05">
          <Password>student05</Password>
          <GroupName>developer_student05_domain</GroupName>
          <AccessLevel>group-defined</AccessLevel>
          <UserSummary>Developer account</UserSummary>
        </User>>
      </dp:set-config>
    </dp:request>
  </env:Body>
</env:Envelope>
```

© Copyright IBM Corporation 2014

Figure 1-32. SOMA: Create a user account

WE601 / ZE6011.1

Notes:

When you export an application domain configuration through the WebGUI, the XML file structure matches the elements within the **dp:request** element. That is, the SOAP interface to the XML Management system uses the same XML schema as the XML configuration files.



SOMA: Account creation response

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>

    <dp:response
      xmlns:dp="http://www.datapower.com/schemas/management">
      <dp:timestamp>
        2013-12-22T15:22:04-05:00
      </dp:timestamp>
      <dp:result>
        OK
      </dp:result>
    </dp:response>

  </env:Body>
</env:Envelope>
```

© Copyright IBM Corporation 2014

Figure 1-33. SOMA: Account creation response

WE601 / ZE6011.1

Notes:

Each set configuration call returns a result value. If the administration operation fails, an error message and error code might appear in the result field.



AMP: Appliance information request

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

    <soapenv:Body>
        <dp:GetDeviceInfoRequest xmlns:dp=
            "http://www.datapower.com/schemas/appliance/management/3.0"/>
    </soapenv:Body>

</soapenv:Envelope>
```

© Copyright IBM Corporation 2014

Figure 1-34. AMP: Appliance information request

WE601 / ZE6011.1

Notes:

The AMP WSDL defines multiple operations, unlike SOMA.

AMP: Appliance information response

```

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Body>
  <amp:DeviceInfoResponse xmlns:amp=
    "http://www.datapower.com/schemas/appliance/management/3.0">
    <amp:DeviceName>DP #15</amp:DeviceName>
    <amp:DeviceSerialNo>XXXXXXX</amp:DeviceSerialNo>
    <amp:DeviceID>719942X</amp:DeviceID>
    <amp:DeviceType>XI52</amp:DeviceType>
    <amp:FirmwareVersion>XI52.6.0.0.3</amp:FirmwareVersion>
    <amp:FailureDetected>false</amp:FailureDetected>
    <amp:CurrentAMPVersion>3.0</amp:CurrentAMPVersion>
    <amp:ManagementInterface type="web-mgmt">
      9090</amp:ManagementInterface>
    <amp:DeviceFeature>MQ</amp:DeviceFeature>
    <amp:DeviceFeature>TAM</amp:DeviceFeature>
    . . .
  </amp:DeviceInfoResponse>
</env:Body></env:Envelope>

```

© Copyright IBM Corporation 2014

Figure 1-35. AMP: Appliance information response

WE601 / ZE6011.1

Notes:

The complete list of device features that are returned by the command are not listed due to space restrictions.



Management interface summary

- WebGUI web application
 - Easy-to-use interface accessible over a web browser
 - Built-in online help for fields and operations
 - Separate apply and save steps allow administrators to discard changes after testing
- CLI
 - The only interface available as is, without modification
 - Simple environment, like UNIX
- XML Management Interface
 - Allows configuration of an application domain or the entire appliance through a batch of SOMA and AMP commands
 - Structured XML request and response messages allow user-created applications to parse through results
 - Web service (SOAP) interface allows external applications to manage the DataPower appliance

© Copyright IBM Corporation 2014

Figure 1-36. Management interface summary

WE601 / ZE6011.1

Notes:

The **WebGUI** web application is the simplest management interface to use. On most pages, a help link provides online help through a pop-up browser window. Most fields also provide inline help when selected. Finally, the two-step process for committing configuration changes provides an opportunity to discard changes.

The **CLI** provides a simple but powerful management interface. Its syntax is familiar to terminal users on a UNIX like environment. Unlike the WebGUI, configuration changes are immediately committed. An undo command allows administrators to revert to a previous configuration. All administrators need to be familiar with basic CLI commands, as this management interface is the only one available on first use. You must enable one of the other management interfaces by using the configure terminal command.

The **XML Management Interface** provides a structured language for sending a batch of configuration commands. This interface allows for a quick and automated configuration of new application domains or entire DataPower appliances. The SOMA and AMP interfaces extends its function to third-party web service clients.

The SNMP interface is not mentioned on this list. It allows the monitoring and configuration of the DataPower appliance through an industry-standard API.



Globalization: Displaying other languages in WebGUI and the log

- Supported languages:
 - English
 - German
 - French
 - German
 - Spanish
 - Brazilian Portuguese
 - Japanese
 - Chinese: Mainland
 - Chinese: Taiwan
- Language files are contained within the firmware
 - No language packs required

© Copyright IBM Corporation 2014

Figure 1-37. Globalization: Displaying other languages in WebGUI and the log

WE601 / ZE6011.1

Notes:



Enabling languages

- For any language other than English, that language must be enabled before it can be used
 - If incorrect settings are made, English is the language
- Click **Administration > Device > Language**

Configure Language

[Refresh List](#)

Name	Status	Op-State	Logs	Administrative State	Comments
de	modified	up		enabled	German
en	saved	up		enabled	English
es	saved	down		disabled	Spanish
fr	saved	down		disabled	French
it	saved	down		disabled	Italian
ja	saved	down		disabled	Japanese
ko	saved	down		disabled	Korean
pt_BR	saved	down		disabled	Portuguese
zh_CN	saved	down		disabled	Simplified Chinese
zh_TW	saved	down		disabled	Traditional Chinese

© Copyright IBM Corporation 2014

Figure 1-38. Enabling languages

WE601 / ZE6011.1

Notes:



Getting the WebGUI to display an alternative language

- Set the alternative language as the **primary** language in the browser
 - Location of language option is browser-dependent
- German as the primary language in the browser

A screenshot of a web browser showing the 'WebSphere DataPower-Anmeldung' (Login) page. The page is in German. At the top, there is an error message: '*Unvollständige Anmeldung. Bitte versuchen Sie es erneut.' Below it, the title 'WebSphere DataPower-Anmeldung' and subtitle 'XI52-Konsole auf DP99' are displayed. There are three input fields: 'Benutzername:' (Username), 'Kennwort:' (Password), and 'Domäne:' (Domain). The 'Domäne:' dropdown menu is set to 'default'. At the bottom of the form are two buttons: 'Anmeldung' (Login) and 'Abbrechen' (Cancel). A copyright notice at the bottom of the page reads: 'Licensed Materials - Property of IBM Corp. © IBM Corporation and other(s) 1999-2013. IBM ist eine eingetragene Marke der IBM Corporation in den USA und/oder anderen Ländern.'

© Copyright IBM Corporation 2014

Figure 1-39. Getting the WebGUI to display an alternative language

WE601 / ZE6011.1

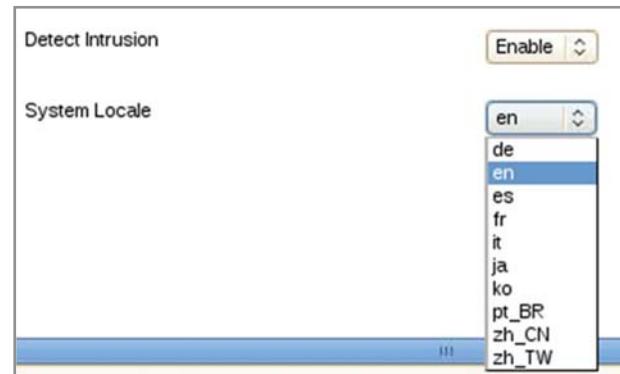
Notes:

The language must be enabled in DataPower first.

WebSphere Education 

Getting an alternative language for the log and messages

- Click **Administration > Device > System Settings > System Locale**
- Reboot the appliance after changing the language preferences
- The alternative language must also be enabled
- Logs and messages are in the alternative language, or English if not translated



direction	client	msgid	message
Show last 50 100 all			
response	172.16.78.230	0x80e0039f	xmlfirewall (web-mgmt): url-open: Syntaxanalyse der Antwort aus http://127.0.0.1:63503/ abgeschlossen
response	172.16.78.230	0x80e0039e	xmlfirewall (web-mgmt): url-open: Antwortcode 200
request		0x80c00004	xmlfirewall (map): Von der Protokollsicht wurde kein Inhaltstyp (content-type) angegeben
request		0x80c00004	xmlfirewall (map): Von der Protokollsicht wurde kein Inhaltstyp (content-type) angegeben

© Copyright IBM Corporation 2014

Figure 1-40. Getting an alternative language for the log and messages

WE601 / ZE6011.1

Notes:



Unit summary

Having completed this unit, you should be able to:

- List the methods that can be used to administer WebSphere DataPower Appliances
- Manage user accounts and domains on the appliance

© Copyright IBM Corporation 2014

Figure 1-41. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: One way to restrict access to an application domain is to define user groups to restrict user account access to a particular domain.
2. Which user account and application domain do you need to perform an upgrade?
 - A. Any domain
 - B. default
 - C. Linux root
 - D. admin
3. Match the advantages in performing administration tasks through the diverse DataPower interfaces:

Description	Definition
1. The WebGUI web application	A. Creation of scripts, less bandwidth
2. The CLI	B. Easier to use
3. The XML Management Interface	C. Programmatic

© Copyright IBM Corporation 2014

Figure 1-42. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here.

- 1.
- 2.
- 3.



Checkpoint answers

- 1. True.**
- 2. B and D.**
- 3. 1 - B, 2 - A, and 3 - C.**

© Copyright IBM Corporation 2014

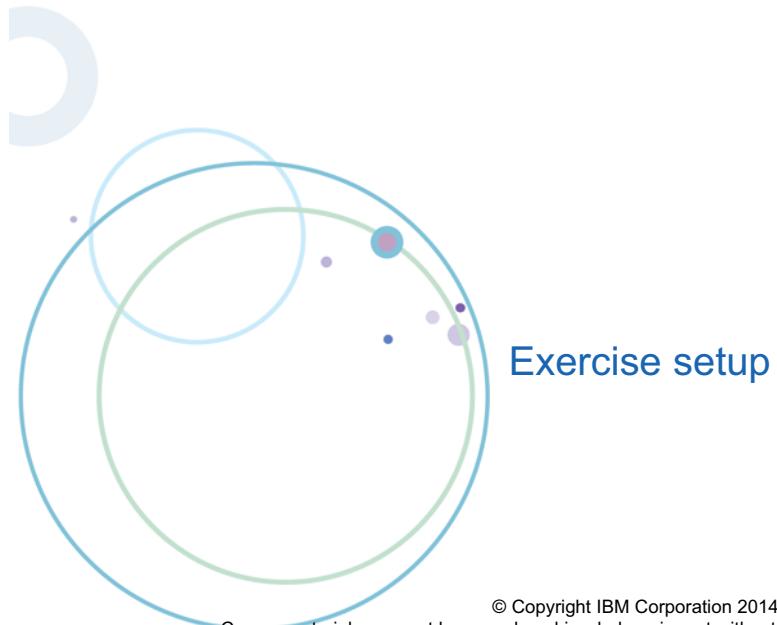
Figure 1-43. Checkpoint answers

WE601 / ZE6011.1

Notes:



Exercise 1



© Copyright IBM Corporation 2014

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

9.0

Figure 1-44. Exercise 1

WE601 / ZE6011.1

Notes:

This first exercise prepares your system for the following DataPower exercises.



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2014

Figure 1-45. Exercise objectives

WE601 / ZE6011.1

Notes:

Unit 2. DataPower services overview

What this unit is about

In this unit, you learn about the various service types that are supported on the DataPower appliance. You begin by examining, at a high level, what a service is, and what it can communicate with. Then, the characteristics of each of the service types are reviewed. Finally, a graphic is shown that displays the relationships between the XML-based services.

What you should be able to do

After completing this unit, you should be able to:

- Define what a DataPower service is
- List the supported services on the WebSphere DataPower SOA Appliance
- Describe the similarities and differences in the features that each WebSphere DataPower service supports

How you will check your progress

- Checkpoint



Unit objectives

After completing this unit, you should be able to:

- Define what a DataPower service is
- List the supported services on the WebSphere DataPower SOA Appliance
- Describe the similarities and differences in the features that each WebSphere DataPower service supports

© Copyright IBM Corporation 2014

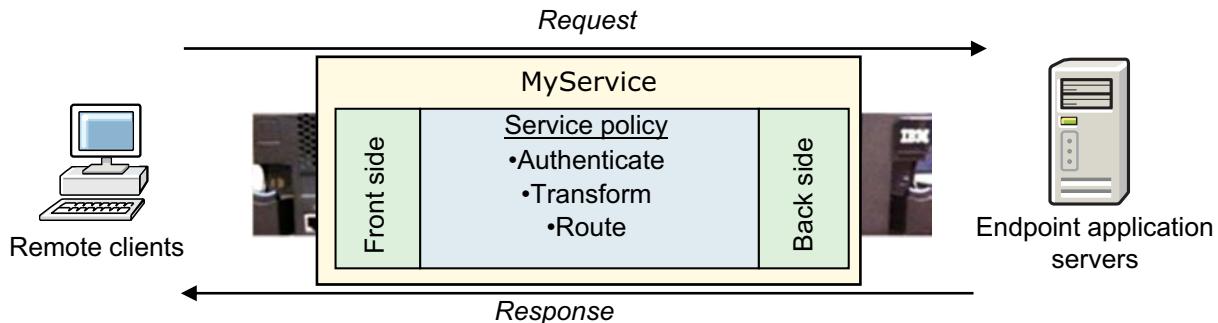
Figure 2-1. Unit objectives

WE601 / ZE6011.1

Notes:

Services in a DataPower appliance

- A service on the appliance is required to deliver DataPower functions
- An appliance supports one or more configured services



- A service is of a specific type. The type that is selected depends on:
 - Processing needs
 - Communication protocol
 - Type of endpoint application servers
- “Front side” defines the client interface to the DataPower service
- Most services have a **service policy**, which you configure to deliver the DataPower functions that the service needs
- “Back side” defines the DataPower outbound service interface to the endpoint application servers

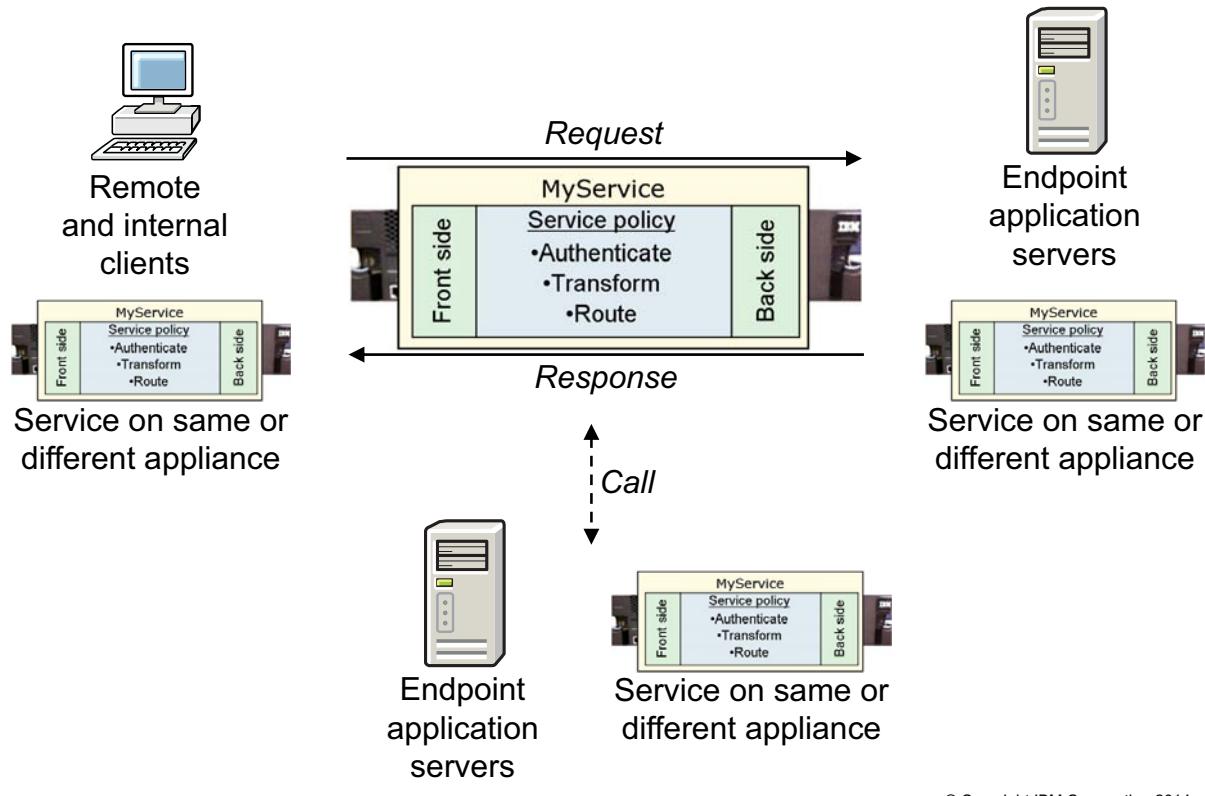
© Copyright IBM Corporation 2014

Figure 2-2. Services in a DataPower appliance

WE601 / ZE6011.1

Notes:

Front sides and back sides, and sideways



© Copyright IBM Corporation 2014

Figure 2-3. Front sides and back sides, and sideways

WE601 / ZE6011.1

Notes:

The front side of a service can receive requests from a remote client, an internal client, or from another service on the appliance.

While executing a service policy, the service can call to other services on the appliance or to other application servers.

The back side of the service calls the target application server, or perhaps another service on the appliance.

Services available on the DataPower appliance

- XSL proxy
 - Accelerates XML processing, such as schema validation and XSL transformations
- XML firewall
 - Secures and offloads XML processing from back-end XML-based applications
 - Supports XML encryption, XML signatures, and AAA
- Multi-protocol gateway (MPGW)
 - Receives messages from clients that use multiple protocols and sends messages to back-end services over many protocols
 - Supports XML encryption, XML signatures, and AAA
- Web service proxy (WS-Proxy)
 - Virtualizes and secures back-end web service applications
 - Supports XML encryption, XML signature, and AAA
- Web application firewall (WAFW)
 - Secures and offloads processing from web-based applications
 - Threat mediation, AAA, and web-based validation



© Copyright IBM Corporation 2014

Figure 2-4. Services available on the DataPower appliance

WE601 / ZE6011.1

Notes:

AAA: authentication, authorization, and auditing

The primary DataPower services are multi-protocol gateway and the web service proxy.

The web service proxy configuration is WSDL-based. It is the only service that **requires** a WSDL file.

All services support monitors and logging.

XSL proxy service

- Use the XSL proxy to accelerate XML processing
 - Use XML schema files to validate XML messages
 - Perform XSL transformations
 - Communicate with client and back-end servers by using SSL
 - Monitor messages that are passing through the appliance
 - Monitor and log activity, deliver log information to external managers
- Use cases
 - Continuing support for existing customer implementations
 - Superseded by more powerful service types
- Available on the XG45, XI52, XI50 blades



© Copyright IBM Corporation 2014

Figure 2-5. XSL proxy service

WE601 / ZE6011.1

Notes:

The XSL proxy service supports XML validation and transformation at wire speed.

The term "wire speed" is often used to describe the XML processing performance of a WebSphere DataPower SOA appliance. That is, the average XML processing rate is almost as high as the network connection transmission rate. Runtime variables, such as the complexity of XML messages and the XSL transform, affect processing speed.

Companies that provide XML applications or web services often skip the XML schema validation step due to performance cost. With the XSL proxy service, these companies can validate XML messages against an existing schema without significant degradation in performance. This solution also requires no modification to the existing back-end service.

XML firewall service

- Secure and offload processing from back-end XML-based applications with the XML firewall service
 - Ensures document legitimacy by providing tamper protection that uses XML signatures
 - Protects against XML-based attacks
 - Uses XML encryption to secure messages
 - Provides dynamic routing of XML documents to the appropriate back-end service
 - Access control is based on user credentials in the message
- Supports all the features of the XSL proxy
- Available on the XG45, XI52, XI50 blades



© Copyright IBM Corporation 2014

Figure 2-6. XML firewall service

WE601 / ZE6011.1

Notes:

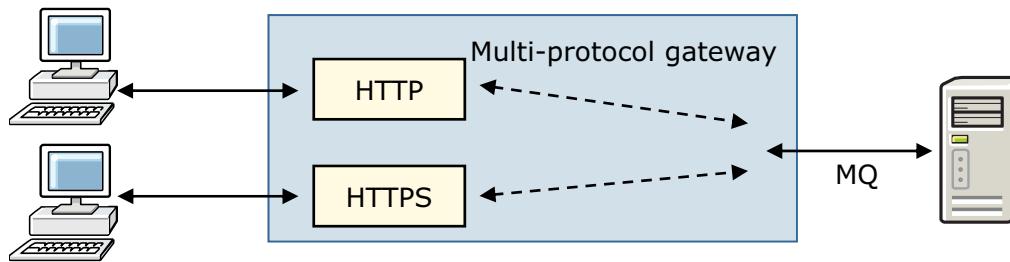
The features that are listed for the XML firewall are not exhaustive. The XML firewall also supports the same features that are mentioned previously for the XSL proxy.

An XML firewall uses a document processing policy to enforce the features that are mentioned in the slide. For example, a firewall policy can require that messages be decrypted and then schema-validated. Other features, such as XML signatures, access control, and dynamic routing, have associated actions that are used in a firewall policy.

XML threat protection and SSL communication are configured at the service level instead of the policy level.

Multi-protocol gateway service

- A multi-protocol gateway (MPGW) connects client requests that are sent over one or more transport protocols to a back-end service that uses the same or a different protocol
 - Single policy that is applied to multiple messages over many protocols
 - Uses static or dynamic back-end protocol and URL
- Features are a **superset** of the XML firewall
- **Preferred** choice for non-WSDL-based services
- Available on the XG45, XI52, XI50 blades



© Copyright IBM Corporation 2014

Figure 2-7. Multi-protocol gateway service

WE601 / ZE6011.1

Notes:

The multi-protocol gateway does not support the loopback proxy mode as supported by the XML firewall and XSL proxy, but the same effect can be specified by using a DataPower variable within the service.

The protocol that is used on the client-side of the gateway does not need to be the same as the one on the back-end.

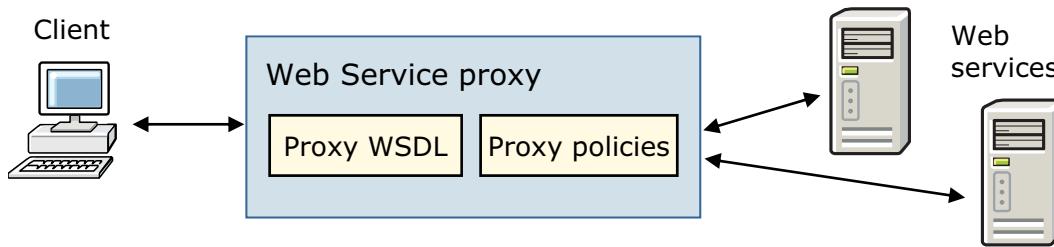
The supported protocols are HTTP(S), FTP(S), SFTP, NFS, raw XML, WebSphere MQ, WebSphere MQ File Transfer Edition (MQFTE), TIBCO EMS, WebSphere JMS, and IMS Connect.

The gateway can use GET and PUT queues to communicate by using WebSphere MQ messages.

Raw XML is an implementation that allows messages to flow from the client to the back-end server and back again by using persistent TCP connections.

Web service proxy service

- The web service proxy (WS-Proxy) is used to secure and virtualize multiple back-end web service applications
 - WSDL-based configuration
 - Policies, monitoring, and logging can be done at various levels of the WSDL file
 - WSDL and governance policy can be updated dynamically
- Features are a **superset** of the XML firewall
- Preferred** choice for WSDL-based services
- Available on the XG45, XI52, XI50 blades



© Copyright IBM Corporation 2014

Figure 2-8. Web service proxy service

WE601 / ZE6011.1

Notes:

An XML firewall or multi-protocol gateway can be created from a WSDL file as well. However, the web service proxy is simpler to configure with the WSDL file because it includes built-in support for creating rules at different levels of the WSDL, and service virtualization.

Multiple WSDL files can be associated with the web service proxy, producing a single virtual WSDL that the client sees.

The web service proxy does not support the loopback proxy mode as supported by the XML firewall and XSL proxy, but the same effect can be specified by using a DataPower variable within the service.

You can receive requests over various transports (front side handlers). It is the same list that a multi-protocol gateway supports for the front side.

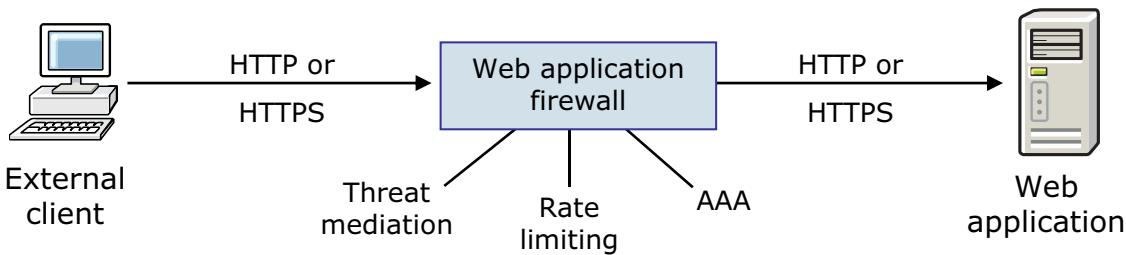
WSDLs can be retrieved from a Universal Description, Discovery, and Integration (UDDI) registry and from WebSphere Service Registry and Repository (WSRR).

Governance policy, such as WS-SecurityPolicy and WS-MediationPolicy, can also be retrieved from a UDDI registry and WebSphere Service Registry and Repository.

WebSphere Service Registry and Repository can "push" changes to a web service proxy.

Web application firewall service

- A web application firewall (WAFW) is used to secure and offload processing from web-based applications
 - Proxies back-end web applications by listening for requests on multiple Ethernet interfaces and TCP ports
 - Provides threat mediation, AAA, and SSL
 - Limits the number of requests or simultaneous connections to back-end web applications
 - Configuration steps are different from XML-focused services
- Customized firewall for HTTP-based traffic
- Available on the XG45, XI52, XI50 blades



© Copyright IBM Corporation 2014

Figure 2-9. Web application firewall service

WE601 / ZE6011.1

Notes:

The web application firewall service contains functionality that is required for securing, load balancing, and accelerating web-based applications. This service is unlike the other services, which focus on XML-based applications.

Threat mediation is provided by checking for malicious JavaScript within HTTP messages.

The concept of the web application firewall is similar to other services, except that it applies to HTTP traffic.

The web application firewall provides features specific to web applications such as session management, web-based validation, and cookie handling.

The configuration is based on request and response maps, rather than a service policy.



Other services

- Web Token Service
 - Loopback service to support OAuth token services
- Interoperability Test Service
 - Development tool that simplifies the testing of style sheets and schemas
- XSL coprocessor service
 - Loopback service that accepts JAXP-based requests
 - **Deprecated**
- Three secondary services are available for handling message traffic without executing a service policy
 - HTTP service: serves documents from a device directory
 - TCP proxy service: forwards TCP traffic to another address and port
 - SSL proxy service: used by log targets to securely connect to remote log systems

© Copyright IBM Corporation 2014

Figure 2-10. Other services

WE601 / ZE6011.1

Notes:

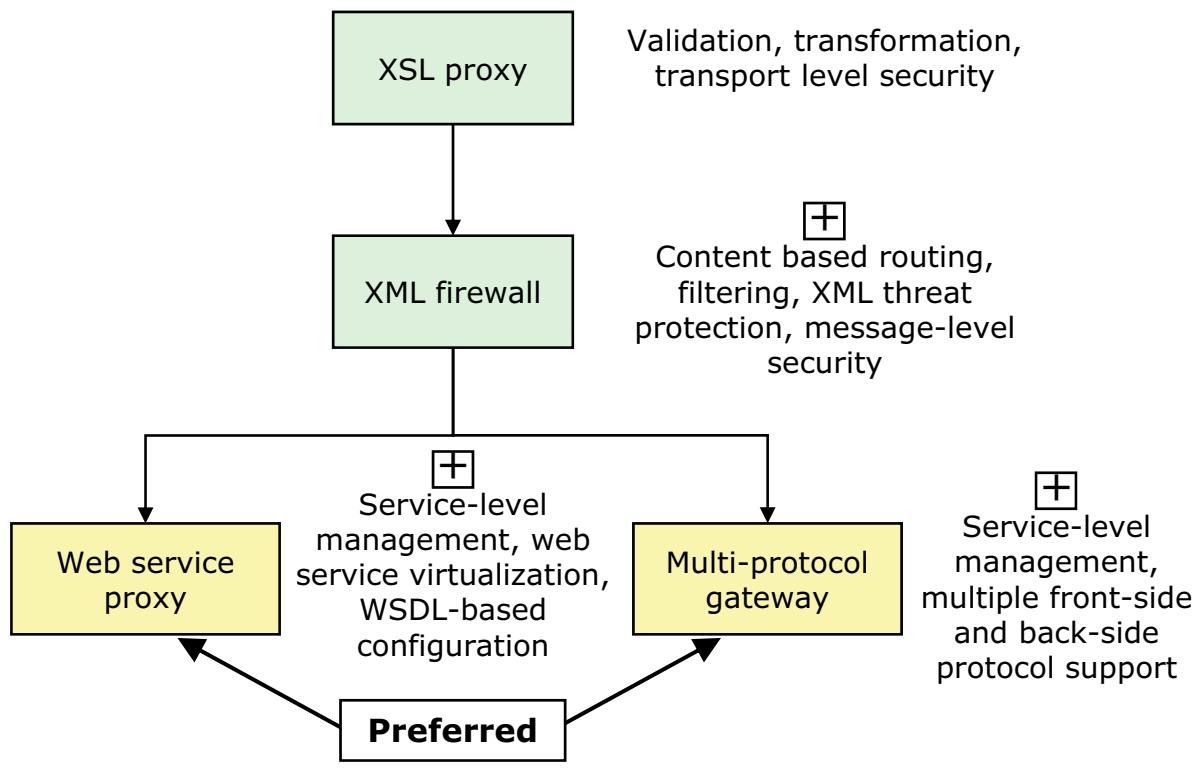
Web token service and interoperability test service are discussed in other units.

XSL coprocessor service is a variant of the XSL proxy service. It is deprecated, and should not be used. In the past, this service was commonly used to test style sheets. This capability is now available in the interoperability test service. Although this service supported JAXP-based requests, there is no Java running in the firmware. It conforms to the JAXP interface.

By default, the appliance does not create an HTTP service on port 80. It must be explicitly created. This service is meant for low-volume or testing purposes; there is not much room for the disk requirements of a typical web server.

The TCP and SSL Proxy services listen for requests on the specified port number and forward the requests to a remote host address and port.

DataPower services feature hierarchy



© Copyright IBM Corporation 2014

Figure 2-11. DataPower services feature hierarchy

WE601 / ZE6011.1

Notes:

This diagram illustrates the object relationship between the different services that are covered in this course.

The **XSL proxy** provides XML schema validation, XML transformation, and support for transport level security (SSL connections).

The **XML firewall** provides security features for XML applications, at the message header and payload level.

The **web service proxy** inherits all of the abilities of the XML firewall and adds features specific to web services. Web service virtualization allows a web service proxy to support many back-end web service applications. In addition, the WSDL-based configuration feature allows developers to set processing rules at a service, portType (interface), or operation level. Although this level of granularity is possible when using an XML firewall, it is up to the developer to apply a processing policy to an element of a web service by using custom XPath expressions.

Finally, the **multi-protocol gateway** allows any-to-any mapping of connections, by using a set of front-end protocol handlers and back-end protocol handlers.

Both the web service proxy and multi-protocol gateway services support service level management policies.

The **web application firewall**, which is not shown on this diagram, is a service that has a feature set similar to the XML firewall, but it is designed for non-XML traffic.



Unit summary

Having completed this unit, you should be able to:

- Define what a DataPower service is
- List the supported services on the WebSphere DataPower SOA Appliance
- Describe the similarities and differences in the features that each WebSphere DataPower service supports

© Copyright IBM Corporation 2014

Figure 2-12. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: The web service proxy is the only service that requires a WSDL.
2. True or False: While executing a service policy, the service can invoke only other services on the appliance.
3. Which service type should be selected for this requirement? A service needs to schema-validate and transform a message before it is placed on a WebSphere MQ queue for mainframe processing. Input comes over HTTPS from external clients, and over HTTP from internal clients.
 - A. XML firewall
 - B. Multi-protocol gateway
 - C. Web service proxy
4. Which service type should be selected for this requirement? An enterprise has operations within several existing web services that it wants to expose to external clients as a single web service.
 - A. XML firewall
 - B. Multi-protocol gateway
 - C. Web service proxy

© Copyright IBM Corporation 2014

Figure 2-13. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here.

- 1.
- 2.
- 3.
- 4.



Checkpoint answers

- 1. True.**
- 2. False.** While executing a service policy, the service can invoke other application servers and other services on the appliance.
- 3. B. Multi-protocol gateway.** It can support both an HTTP and an HTTPS front side handler, and can communicate with a WebSphere MQ queue on the back side.
- 4. C. Web service proxy.** This service type can present a single virtual web service to the client that is composed of specific operations from several web services.

© Copyright IBM Corporation 2014

Figure 2-14. Checkpoint answers

WE601 / ZE6011.1

Notes:

Unit 3. Structure of a service

What this unit is about

Customers purchase WebSphere DataPower Appliances to provide application-related solutions. The key component that DataPower developers configure is a DataPower service. In this unit, you learn about various components that comprise a DataPower service, and the relationships between them. To configure a service, you learn about how the front side access and how the back side connection to the application server is described. Some of the service-wide settings are presented. You also learn how to construct the service policy, which controls the processing within the service.

What you should be able to do

After completing this unit, you should be able to:

- List the basic structural components of a service and describe their relationships
- List the ways a service configures its front-side access and back-side connections
- Use the policy editor to configure a service policy
- Create a service policy with actions that process the client request or server response
- List some of the processing actions and describe their function
- Configure service-wide settings such as:
 - Service type: static back-end, dynamic back-end, and loopback proxy
 - XML Manager
 - URL rewriting

How you will check your progress

- Checkpoint
- Exercise 2: Create a simple service

Unit objectives

After completing this unit, you should be able to:

- List the basic structural components of a service and describe their relationships
- List the ways a service configures its front-side access and back-side connections
- Use the policy editor to configure a service policy
- Create a service policy with actions that process the client request or server response
- List some of the processing actions and describe their function
- Configure service-wide settings such as:
 - Service type: static back-end, dynamic back-end, and loopback proxy
 - XML Manager
 - URL rewriting

© Copyright IBM Corporation 2014

Figure 3-1. Unit objectives

WE601 / ZE6011.1

Notes:

Object-based configuration

- Configuration is object-based
 - A service (an object itself) is composed of many lower-level objects
 - These objects are “data” objects, not the traditional object-oriented entities with custom-coded methods (behavior)
- In the vertical navigation bar, expand **Status > Main > Object Status**
 - List of lower-level objects that compose a service

<input type="checkbox"/> SecureTransformMPGW [Multi-Protocol Gateway]	Saved	up
<input type="checkbox"/> MySSLFSH [HTTPS (SSL) Front Side Handler]	Saved	up
<input type="checkbox"/> MySSLProxy [SSL Proxy Profile]	Saved	up
<input type="checkbox"/> MyCryptoProfile [Crypto Profile]	Saved	up
<input type="checkbox"/> AliceIDCred [Crypto Identification Credentials]	Saved	up
Alice [Crypto Key]	Saved	up
Alice [Crypto Certificate]	Saved	up
<input type="checkbox"/> default [XML Manager]	Saved	up
default [User Agent]	Saved	up
<input type="checkbox"/> MyMPGWPOLICY [Processing Policy]	Saved	up
MatchAll [Matching Rule]	Saved	up
<input type="checkbox"/> MyMPGWPOLICY_rule_0 [Processing Rule]	Saved	up
MyMPGWPOLICY_rule_0_xform_0 [Processing Action]	Saved	up
<input type="checkbox"/> MyMPGWPOLICY_rule_1 [Processing Rule]	Saved	up
MyMPGWPOLICY_rule_1_sign_0 [Processing Action]	Saved	up
MyMPGWPOLICY_rule_1_encrypt_0 [Processing Action]	Saved	up

© Copyright IBM Corporation 2014

Figure 3-2. Object-based configuration

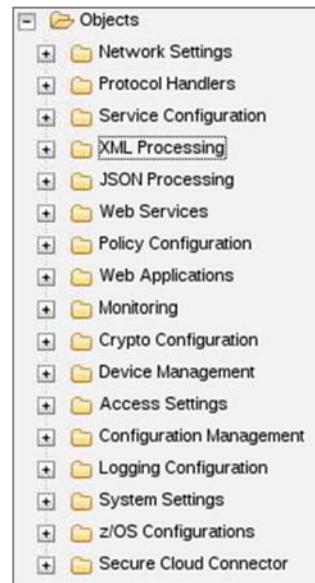
WE601 / ZE6011.1

Notes:



Approach to configuring objects

- Objects can be configured individually
 - Expand **Objects** in the navigation bar
- Most times, lower-level objects are configured from higher-level objects
 - Front-side handlers and service policy are created and configured during the configuration of a service
 - Processing rules and actions are created and configured during the configuration of a service policy
- Some objects (XML firewall and others) have wizards



The screenshot shows a configuration interface titled 'Configure XML Firewall'. It displays a table with the following data:

XML Firewall Name	Op-State	Logs	Req-Type	Local Address	Port
LoopbackXMLFW	up		unprocessed	dp_public_ip	597

At the bottom of the interface are two buttons: 'Add Wizard' and 'Add Advanced'.

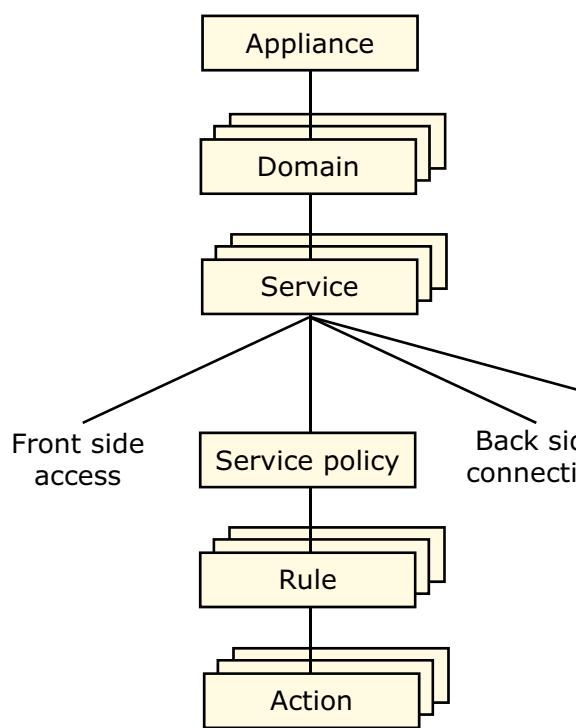
© Copyright IBM Corporation 2014

Figure 3-3. Approach to configuring objects

WE601 / ZE6011.1

Notes:

Basic architectural model



- One appliance has multiple domains
- A domain has multiple services
- Each service has one service policy
 - XML-based services
- A policy references multiple rules
 - Types of rules: request, response, both, error
- Each rule contains multiple actions
 - Some standard actions are **Validate**, **Transform**, **Results**, and more
 - Custom XSLT always available using the **Transform** action

© Copyright IBM Corporation 2014

Figure 3-4. Basic architectural model

WE601 / ZE6011.1

Notes:

From the **service** level on down, the focus is on the XML-based services that are covered in this course: XML firewall, multi-protocol gateway, and web service proxy.

For a service, the configuration is divided between the front side access, connection to the back side, general service settings, and the service policy.



Front side access

- Specifies how a service accepts requests
- Part of the service definition
 - XML firewall
- Protocol handler object
 - Multi-protocol gateway: front side protocol handler
 - Web service proxy: endpoint handler
 - Specification depends on protocol

Front End
Local IP Address 172.16.78.250
Port Number 5971 *
Reverse (Server) Crypto Profile (none)

Front side settings
Front Side Protocol MySSLFSH (HTTPS (SSL) Front Side Handler)

Local
Local Endpoint Handler
AddressSearchFSH

Create a New:
FTP Poller Front Side Handler NFS Poller Front Side Handler SFTP Poller Front Side Handler FTP Server Front Side Handler HTTP Front Side Handler HTTPS (SSL) Front Side Handler IMS Connect Handler TIBCO EMS Front Side Handler WebSphere JMS Front Side Handler MQFTE Front Side Handler MQ Front Side Handler SFTP Server Front Side Handler Stateless Raw XML Handler Stateful Raw XML Handler

© Copyright IBM Corporation 2014

Figure 3-5. Front side access

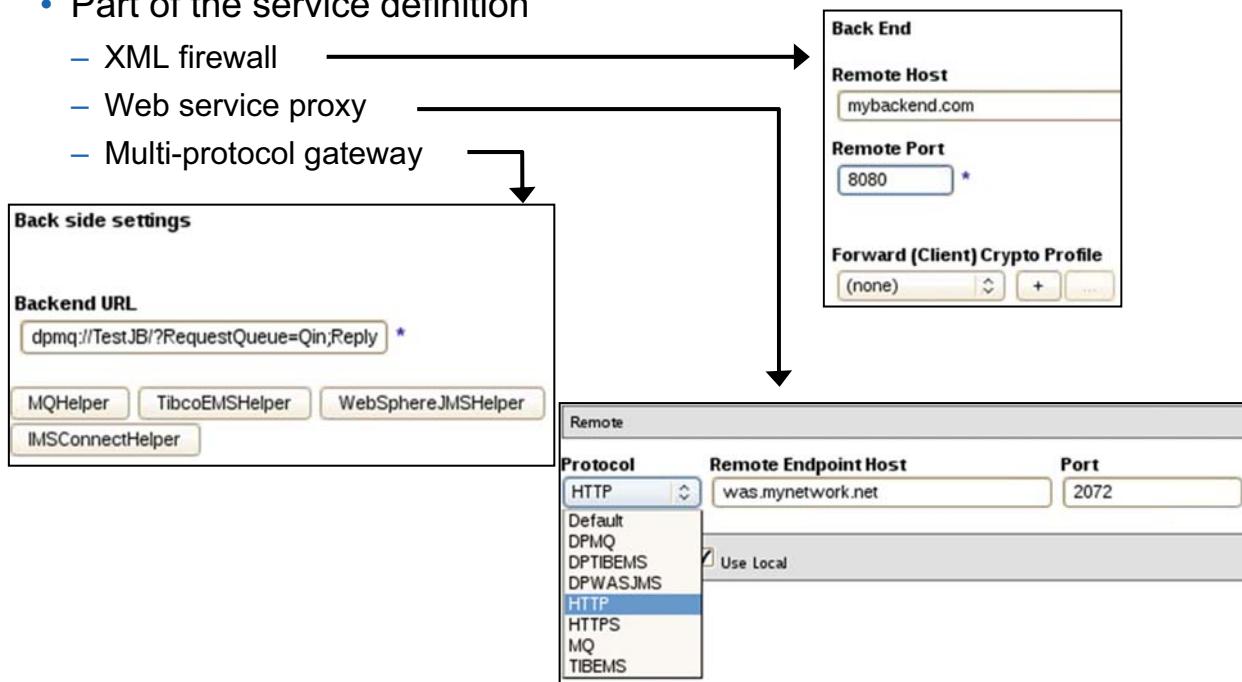
WE601 / ZE6011.1

Notes:



Connection to the back side

- Specify how a service connects to the back side application
- Part of the service definition
 - XML firewall
 - Web service proxy
 - Multi-protocol gateway



© Copyright IBM Corporation 2014

Figure 3-6. Connection to the back side

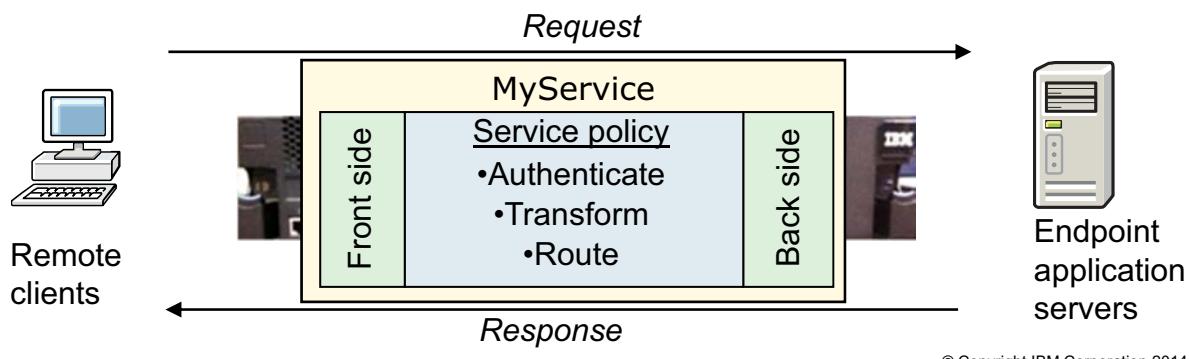
WE601 / ZE6011.1

Notes:

The back side connection can be dynamically determined, rather than hardcoded. In this case, the connection is made from a style sheet within the service policy.

Message processing phases

- Each message passes through three phases:
 1. Front side, client side
 - System throttle, listen for IP address or port, ACL, SSL, attachment processing, URL rewrite, HTTP header injection and suppression, and monitors
 2. Service policy
 - Service traffic type (SOAP, XML, preprocessed, unprocessed), XML Manager, SOAP validation
 3. Back side, server side
 - Streaming, URI propagation, user agent, SSL, load balancer, HTTP options
- Service settings control client-side and server-side behavior



© Copyright IBM Corporation 2014

Figure 3-7. Message processing phases

WE601 / ZE6011.1

Notes:

Response messages from the server then pass through these phases in reverse. Response processing is the same as request processing except that the server must deal with errors from the back-end service.

During client-side processing, the URL submitted by the client might be rewritten, the HTTP headers are altered, and the format of the message validated (SOAP or XML).

During service policy processing, the message might be transformed in any number of ways, and filtered, encrypted, decrypted, signed, verified, or duplicated, and sent to a third-party resource for handling.

During server-side processing, the message might be routed, TCP and HTTP options set, or SSL connections negotiated.

URI propagation refers to the part of the URL after the host-port combination.

A user agent can be configured with an SSL proxy profile to communicate securely to the back-end service.

A load balancer object is used to provide redundancy for multiple back-end servers. The service sends the message to the load balancer group instead of the back-end server. The load balancer group chooses the back-end server.

Multi-step scope refers to the sequence of actions that are executed on the request and response. Variables can be set to pass information between the actions.



Configuring a service policy (processing policy)

- A service policy is configured within a policy editor
 - Create the different types of rules
 - Drag action icons within a rule
- For a multi-protocol gateway and XML firewall, the policy editor opens in its own window
 - You configure **all** rules within the service policy in this window
 - All of the rules are visible in the window
- For the web service proxy, the policy editor displays as a section on the **Policy** tab
 - Only the rules that relate to the currently selected level of the WSDL (proxy, wsdl, service, port, operation) are configured
 - In the web service proxy, the policy editor does not show all the rules that apply to the service at the same time

© Copyright IBM Corporation 2014

Figure 3-8. Configuring a service policy (processing policy)

WE601 / ZE6011.1

Notes:

Policy:

Policy Name: MyMPGWPOLICY

Policy area

Rule:

Rule Name: MyMPGWPOLICY_rule_0 Rule Direction: Client to Server

New Rule Delete Rule

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

Action icons: Filter, Sign, Verify, Validate, Encrypt, Decrypt, Transform, Route, AAA, Results, Advanced.

Rule configuration area

CLIENT → ↗ ↘ → ↙ ↘ → ORIGIN SERVER

Create Reusable Rule

Configured Rules

Order	Rule Name	Direction	Actions
↑ ↓	MyMPGWPOLICY_rule_0	Client to Server	edit rule delete rule
↑ ↓	MyMPGWPOLICY_rule_1	Server to Client	edit rule delete rule

Configured rules area

© Copyright IBM Corporation 2014

Figure 3-9. Policy editor

WE601 / ZE6011.1

Notes:

The policy area is where the service policy is named, and the configuration applied. In the web service proxy, there is no policy area in the policy editor.

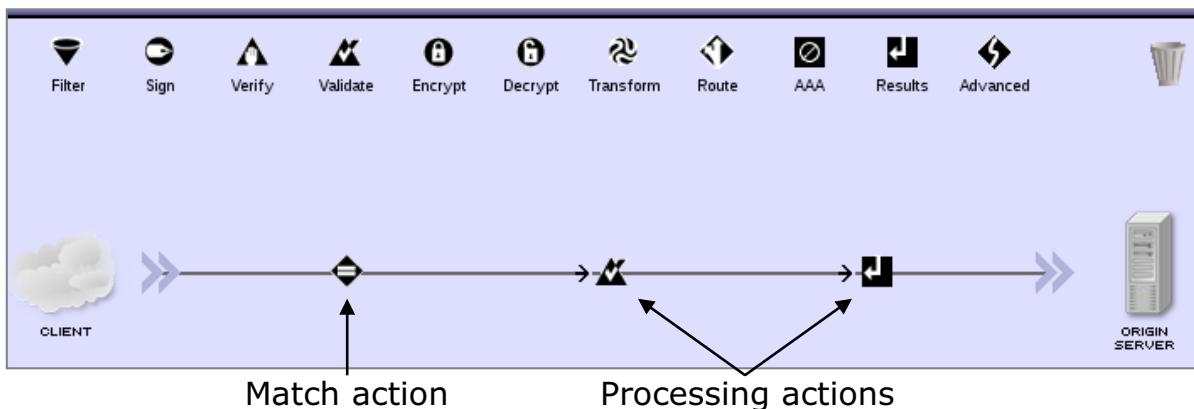
The rule configuration area is where rules are created, named, deleted, and configured. Action icons are dragged onto the rule configuration path.

The configured rules area lists the current rules that are part of the service policy.



Configuring rules within a service policy

- Each rule contains:
 - **Match action:** defines criteria to determine whether this rule processes the incoming message
 - **Processing actions:** a rule defines one or more actions that are taken on the submitted message
- Actions are dragged onto the path or line
 - Execution is from left to right
 - Background graphic adjusts to match rule direction



If the request matches the conditions that are set in the **Match action**, then the **actions** are executed.

© Copyright IBM Corporation 2014

Figure 3-10. Configuring rules within a service policy

WE601 / ZE6011.1

Notes:

This example defines a rule with a Match action and two actions (Validate and Results).

A rule can be configured to apply to:

- Server to client (server response)
- Both directions (client request and server response)
- Client to server (client request)
- Error (errors during message processing)



Processing rules

- Rules have the following directions:
 - Server to client (response)
 - Client to server (request)
 - Both directions (request and response)
 - Error: executes when errors occur during processing in the request and response rules
- Rules have priority and are ordered
 - Multiple rules might match on the same URL; order is critical to selection
 - Specific rules must have higher priority than catch-all rules
- Other capabilities
 - Programmatic actions such as loops are available; otherwise actions are performed in sequential order
 - The asynchronous option allows the next action to start without waiting for the current action to complete

The screenshot shows a 'Rule' configuration interface. At the top, there's a 'Rule Name' field containing 'LDAPTest_request' and a 'Rule Direction' dropdown menu with options: Client to Server, Both Directions, Server to Client, and Client to Server. Below these are 'New Rule' and 'Delete Rule' buttons. A message at the bottom says 'Create rule: Click New, drag action icons onto line.' and 'Edit rule: (Error)'. The main area displays a table titled 'Configured Rules' with columns: Order, Rule Name, Direction, and Actions. The table contains three rows:

Order	Rule Name	Direction	Actions
	LDAPTest_request	Client to Server	
	LDAPTest_Rule_1	Server to Client	
	LDAPTest_Rule_2	Error	

© Copyright IBM Corporation 2014

Figure 3-11. Processing rules

WE601 / ZE6011.1

Notes:

A specific matching rule can match on the URL `*/test`. A catch-all rule can match on all URLs by using the asterisk (`*`).

Processing in rules occurs sequentially in the order that the actions appear. Actions that allow for programmatic processing, such as looping and if-then-else statements, are available.

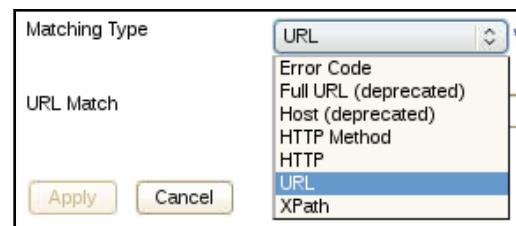


Match action

- A **Match** action allows you to provide different processing that is based on matching conditions



- Match criteria can be based on:
 - Error code value
 - Fully qualified URL (deprecated)
 - Host (deprecated)
 - HTTP method
 - HTTP header value
 - URL
 - XPath expression



© Copyright IBM Corporation 2014

Figure 3-12. Match action

WE601 / ZE6011.1

Notes:

A Match action allows you to define criteria that is matched against the incoming traffic to determine whether the actions configured in the rule are applicable.

Each rule is configured with a Match action.

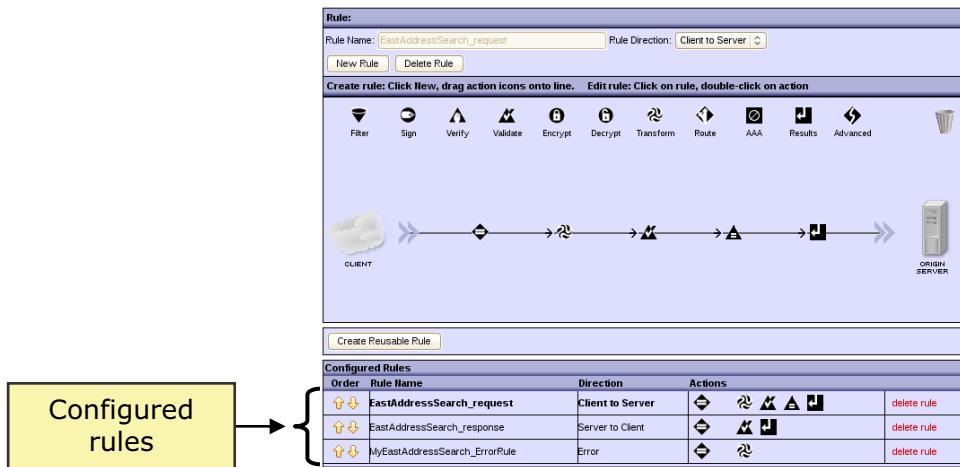
The error code is not an HTTP error code, but a DataPower internal error code value.

-
-
-
-



Processing actions

- A rule consists of multiple processing actions with scope
 - Actions such as **Transform** or **Validate** execute during the request or response rule (if there are any)
 - Contexts or defined variables within the scope are used to pass information between actions
 - Asynchronous options allow the following action to start before the current action completes
 - Programmatic actions allow for looping and if-then-else logic in rules



The *contexts* and *variables* that are set during the request processing are available to the actions used in the response processing because of a shared scope.

© Copyright IBM Corporation 2014

Figure 3-13. Processing actions

WE601 / ZE6011.1

Notes:

Variables can be set by using a **Set Variable** action (**Advanced > Set Variable**).

Contexts are temporary variables that contain XML data, binary non-XML data, user, or system variables.

The **Log** action is a good example of asynchronous processing. You might want to log asynchronously so that subsequent processing can continue without delay while logging is being completed. If you want to wait until later and continue after your previous asynchronous actions complete, you can add an **Event Sink** action. In this action, you can list previous asynchronous actions that you wait on.

The **Conditional** action implements if-then-else processing based on XPath expression values.

The **For-each** action implements a loop on designated actions that are based on XPath expression values.

 WebSphere Education 

Processing actions

Action	Description	
Filter	Performs an accept or reject on incoming documents	 Filter
Sign	Attaches a digital signature to a document	 Sign
Verify	Verifies the digital signature that is contained in an incoming document	 Verify
Validate	Performs schema-based validation of XML documents	 Validate
Encrypt	Performs complete and field-level document encryption	 Encrypt
Decrypt	Performs complete and field-level document decryption	 Decrypt
Transform	Uses a specified style sheet to perform XSLT processing on XML or non-XML documents	 Transform
Route	Implements dynamic style sheet-based routing or XPath-based routing	 Route
AAA	Invokes a AAA policy	 AAA
Results	Sends a message in specific context to an external destination	 Results
Advanced	A grouping of lesser-used actions	 Advanced

© Copyright IBM Corporation 2014

Figure 3-14. Processing actions

WE601 / ZE6011.1

Notes:

The **Encrypt** and **Decrypt** actions are used for XML encryption. The **Sign** and **Verify** actions are used in XML signatures. These actions are discussed in the web services security unit.

The **AAA** action is discussed in the AAA lecture.

The advanced actions are:

- **Anti-Virus:** This action scans a message for viruses by using an external ICAP server.
- **Call Processing Rule:** This action invokes a named rule; processing resumes on the next step.
- **Conditional:** This action selects an action for processing based on an XPath expression.
- **Convert Query Params to XML:** This action converts non-XML CGI-encoded input (an HTTP POST of HTML form or URI parameters) into an equivalent XML message.
- **Crypto Binary:** This action performs a cryptographic operation (sign, verify, encrypt, decrypt) on binary data.
- **Event-sink:** This action forces a wait for asynchronous actions before continuing.

- **Extract Using XPath:** This action applies an XPath expression to a context and stores the result in another context or a variable.
- **Fetch:** This action retrieves an identified external resource and places the result in the specified context.
- **For-each:** This action defines looping based on a count or expression.
- **Header Rewrite:** This action rewrites HTTP headers or URLs.
- **Log:** This action sends the content of the specified input context as a log message to the destination URL identified here.
- **Method Rewrite:** This action rewrites the HTTP method for the output message.
- **MQ Header:** This action manipulates WebSphere MQ headers.
- **On Error:** This action sets a named rule as the error handler; it is invoked if subsequent processing encounters errors.
- **Results Asynchronous:** This action asynchronously sends a message in a specified context to a URL or to the special output context.
- **Route (using Variable):** This action routes the document according to the contents of a variable.
- **Set Variable:** This action sets the value of a variable for use in subsequent processing.
- **SLM:** This action invokes an SLM (service level monitor) policy.
- **SQL:** This action sends SQL statements to a database.
- **Strip Attachments:** This action removes either all or specific MIME or DIME attachments.
- **Transform binary:** This action performs a specified transform on a non-XML message, such as binary or flat text.
- **Transform with processing control file:** This action transforms by using XQuery on an input document (XML or JSON) with a processing control file.
- **Transform (using processing instruction):** This action transforms by using XSLT that is specified by processing instructions within the XML document; the parameters might be passed.

More processing actions

Action	Description	
For-each	Loops through each defined action; either an XPath expression triggers it, or it iterates a predetermined number of times	
Conditional	Implements programmatic if-then-else processing	
Event-sink	Causes processing to wait until specific asynchronous actions complete	
Antivirus	Invokes a named, reusable rule that sends messages to a virus scanning server defined as host, port, or URI	

© Copyright IBM Corporation 2014

Figure 3-15. More processing actions

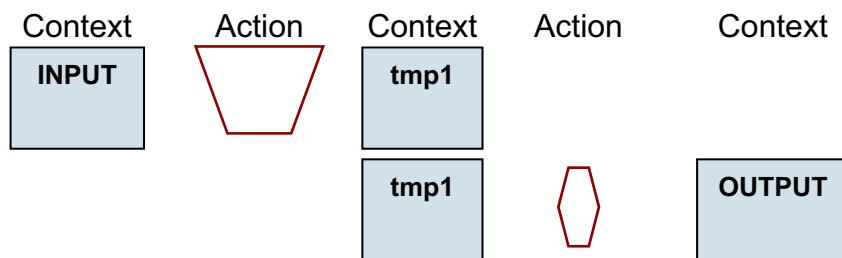
WE601 / ZE6011.1

Notes:

Many actions have an asynchronous option. **Event-sink** is used in processing rules to wait for certain asynchronous actions to complete before processing continues.

Multi-step processing rules

- A multi-step processing rule contains a scope of contexts, actions, and variables
- A context is a DataPower or user-created, action-specific operational workspace
 - Contains an XML tree or binary (non-XML) data
 - Variables are copied from original context to newly created context
 - Contexts can be chained during multi-step processing



© Copyright IBM Corporation 2014

Figure 3-16. Multi-step processing rules

WE601 / ZE6011.1

Notes:

Each action has an input and output. The appliance can explicitly define or generate it.

The **tmp1** context variables are temporary variables that are used to pass information between the actions.

The **INPUT** and **OUTPUT** context variables are predefined by the appliance to represent the input and output messages.

A multi-step processing rule refers to a rule with at least one processing action.



Pre-defined context variables

There are four special system context variables:

- INPUT
 - Data entering the processing rule
 - Example: the data that is contained within an incoming client request (the POST body, in typical HTML), or the data that is contained within a server response
- OUTPUT
 - Data exiting the processing rule
 - Example: the data is passed to a transport protocol, such as HTTP or WebSphere MQ, for transmission to a target client or server device
- PIPE
 - Identifies a context whose output is used as the input of the next action
 - Every action that outputs to PIPE must be followed with an action that inputs from PIPE
- NULL
 - When used in output context, silently discards any data that the action generates
 - When used in input context, passes no message to the action
 - Empty input can be useful when executing a style sheet that does not require input

© Copyright IBM Corporation 2014

Figure 3-17. Pre-defined context variables

WE601 / ZE6011.1

Notes:

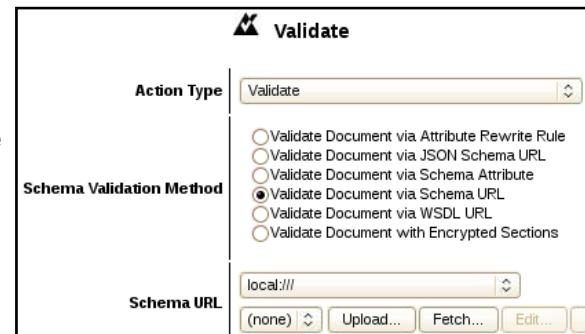
It is not always necessary to specify a context within an action. The WebGUI provides default input and output contexts that can be used.

PIPE can improve processing efficiency and reduce latency by eliminating the need for temporary storage of processed documents. This feature is used for streaming documents through the appliance.

Validate action for XML

Perform schema-based validation of XML documents:

- Validate Document via Attribute Rewrite Rule
 - Scans the document for `xsi:schemaLocation` attribute, applies a URL rewrite policy, and uses the result to find schemas to apply to the document
- Validate Document via Schema URL
 - Specifies a schema URL of an XML schema file
- Validate Document via Schema Attribute (default)
 - Documents are validated by using an `xsi:schemaLocation` attribute to locate an XML schema document
- Validate Document with Encrypted Sections
 - Uses a schema exception map object to validate a document with encrypted parts
- Validate Document via WSDL URL
 - Uses an XML schema that is contained in a WSDL document



© Copyright IBM Corporation 2014

Figure 3-18. Validate action for XML

WE601 / ZE6011.1

Notes:

The **Validate** action is used to validate the schema of XML documents. The schema URL can reference either a local or remote file.

A schema exception map object uses an XPath expression to specify the encrypted and unencrypted parts of an XML document. It allows for encrypted XML documents to be validated by using XML schemas that do not support XML encryption.

The **Fetch** button can be used to download a style sheet from a URL and store it on the appliance.

The **Validate Document via Attribute Rewrite Rule** option searches for an `xsi:schemaLocation` attribute and rewrites this attribute value by using a URL rewrite policy. The validation is then performed against the rewritten schema reference.



Validate action for JSON

Perform schema-based validation of JSON documents:

- Validate Document via JSON Schema URL
 - Specifies a schema URL of a JSON schema file

Validates the input document against the specified JSON schema

- Similar to validating an XML document against an XSD or WSDL
- Supports Draft 3 of the IETF specification:

<http://tools.ietf.org/html/draft-zyp-json-schema-03>

The screenshot shows a 'Validate' dialog box. The 'Action Type' dropdown is set to 'Validate'. Under 'Schema Validation Method', the radio button for 'Validate Document via JSON Schema URL' is selected. The 'JSON Schema URL' field contains 'local:///'. Below the URL field are buttons for '(none)', 'Upload...', 'Fetch...', and 'Edit...'. To the right of the dialog box, the text '© Copyright IBM Corporation 2014' is visible.

© Copyright IBM Corporation 2014

Figure 3-19. Validate action for JSON

WE601 / ZE6011.1

Notes:

The **Validate** action is also used to validate the schema of JSON structures. The JSON schema URL can reference either a local or remote file.

The expected file type for a JSON schema is JSV or JSON.

DataPower version 6.0.0 supports draft 3 of the IETF JSON Schema specification. There are a few options in the specification that DataPower version 6.0.0 does not support. For more information about the options, see the product documentation.



Transform action for XML

Use XSLT to manipulate documents

- Transform with XSLT style sheet
 - Identifies the XSL style sheet that is referenced in the **Transform File** field
- Transform with embedded processing instructions, if available
 - Incoming XML document contains a processing instruction that identifies the XSL style sheet to use in transformation

© Copyright IBM Corporation 2014

Figure 3-20. Transform action for XML

WE601 / ZE6011.1

Notes:

The **Transform action** is also used for supporting custom XSLT actions.

The style sheet can be either referenced from the appliance or uploaded from a remote site.

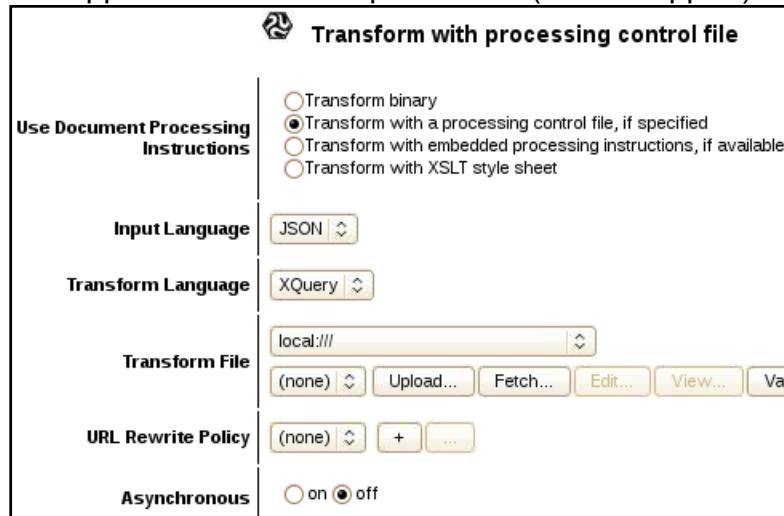
The **URL Rewrite Policy** rewrites external references that are contained within the input document.

 WebSphere Education 

Transform action that uses XQuery (JSON and XML)

Use XQuery expressions to manipulate JSON and XML documents

- Transform with a processing control file, if specified
 - Identifies the XQuery transform file that is referenced in the Transform File field
- XQuery is a query language for XML data (like SQL for relational data)
 - DataPower v6.0.0 adds the support for the JSONiq extension (JSON support)
- Input Language:
 - JSON
 - XML
 - XSD
- Transform Language:
 - XQuery



Transform with processing control file	
Use Document Processing Instructions	<input type="radio"/> Transform binary <input checked="" type="radio"/> Transform with a processing control file, if specified <input type="radio"/> Transform with embedded processing instructions, if available <input type="radio"/> Transform with XSLT style sheet
Input Language	JSON
Transform Language	XQuery
Transform File	local:/// (none) <input type="button"/> Upload... <input type="button"/> Fetch... <input type="button"/> Edit... <input type="button"/> View... <input type="button"/> Validate
URL Rewrite Policy	(none) <input type="button"/> + <input type="button"/> ...
Asynchronous	<input type="radio"/> on <input checked="" type="radio"/> off

© Copyright IBM Corporation 2014

Figure 3-21. Transform action that uses XQuery (JSON and XML)

WE601 / ZE6011.1

Notes:

This option for the **Transform action** supports XQuery as the transformation language, rather than XSLT.

XQuery is a language that is designed to query XML data, much like SQL is used to query relational data. DataPower v6.0.0 includes the JSONiq extension to XQuery. This extension adds support for JSON to XQuery.

DataPower version 6.0.0 supports XQuery 1.0, and its related specifications. The JSONiq extension support is for 0.4.42.

The **Input Language** indicates whether the input document is JSON or XML. The third option of XSD indicates that the input document is XML, but it also displays another entry field that accepts an XML schema file location. This schema is used to type the data (for example, integer, number, text) for the XQuery processing, but it does **not** perform any validation against the schema. To perform validation, you must use a Validate action.

The **Transform Language** indicates the language of the transformation file. The only valid option now is XQuery.

The **URL Rewrite Policy** rewrites external references that are contained within the input document.



Transform action for binary transformations

- Use a WebSphere Transformation Extender (WTX) mapping to transform binary-to-XML, XML-to-binary, or binary-to-binary
 - Typically used to mediate between XML-based clients and COBOL-based mainframe applications
- Transform binary
 - A WTX mapping is used to transform the input document to the output document structure
- WTX Map file
 - File that contains the WTX transformation instructions
- WTX Map Mode
 - Format of the WTX mapping file

The screenshot shows the 'Transform binary' configuration dialog box. It has four main sections:

- Use Document Processing Instructions:** Contains radio buttons for "Transform binary" (selected), "Transform with a processing control file, if specified", "Transform with embedded processing instructions, if available", and "Transform with XSLT style sheet".
- Transform File:** Contains an input field "local:/// (none)" with "Upload..." and "Fetch..." buttons, and a "Var Builder" button.
- WTX Map file:** Contains an input field "local:/// (none)" with "Upload...", "Fetch...", "Edit...", "View...", and "Var" buttons.
- WTX Map Mode:** Contains an input field "DPA" with a dropdown arrow.

© Copyright IBM Corporation 2014

Figure 3-22. Transform action for binary transformations

WE601 / ZE6011.1

Notes:

This version of the **Transform action** uses a WebSphere Transformation Extension (WTX) mapping file to control the transformation.



Filter action

- A **Filter** action accepts or rejects an incoming message
 - Identifies an XSL style sheet that is used for message filtering
 - Does not perform an XSL transformation
- The XSL style sheet uses the `<dp:reject>` and `<dp:accept>` tags to filter messages
- The **Filter** action can be used to prevent replay attacks



© Copyright IBM Corporation 2014

Figure 3-23. Filter action

WE601 / ZE6011.1

Notes:

A standard filter employs the selected XSLT style sheet to either accept or reject the submitted document.



Filter action: Replay attack

The screenshot shows the 'Advanced' tab of the 'Filter' configuration. Under the 'Input' section, 'Input' is set to '(auto)'. In the 'Options' section, the 'Action Type' is set to 'Filter' and 'Filter Method' is set to 'Replay Filter'. Under 'Transform File', the 'store:///' dropdown is selected, and the 'replay-filter.xsl' file is chosen. A note below says 'Stylesheet Summary: Check for replay attacks'. The 'Asynchronous' setting is 'on'. Under 'Replay Filter Type', it is set to 'WS-Addressing Message ID' with the 'Save' checkbox checked. The 'Replay duration' is set to '600 sec' with the 'Save' checkbox checked.

- Protect against replay attacks by using the **Filter Advanced** tab
 - Values from messages are cached and checked on subsequent requests
- Three types are supported:
 - WS-Addressing message ID
 - WS-Security user name token and password digest nonce
 - Custom XPath
- **The Replay duration** value is the duration of time to check for potential replays

© Copyright IBM Corporation 2014

Figure 3-24. Filter action: Replay attack

WE601 / ZE6011.1

Notes:

A replay attack protects against hackers that send a valid message multiple times. This attack occurs when the intruder intercepts a valid message and sends that message on behalf of someone else. To protect against replay attacks, messages pass unique values in each message. The unique values that the replay filter supports are WS-Addressing messages that contain a message ID, a WS-Security username token with a nonce value, or a custom XPath. A nonce is a bit string that is generated to produce a unique string. It is used in authentication and security situations to create a unique ID.

The replay attack filter uses a standard style sheet, `replay-filter.xsl`, to check whether messages are executing replay attacks.

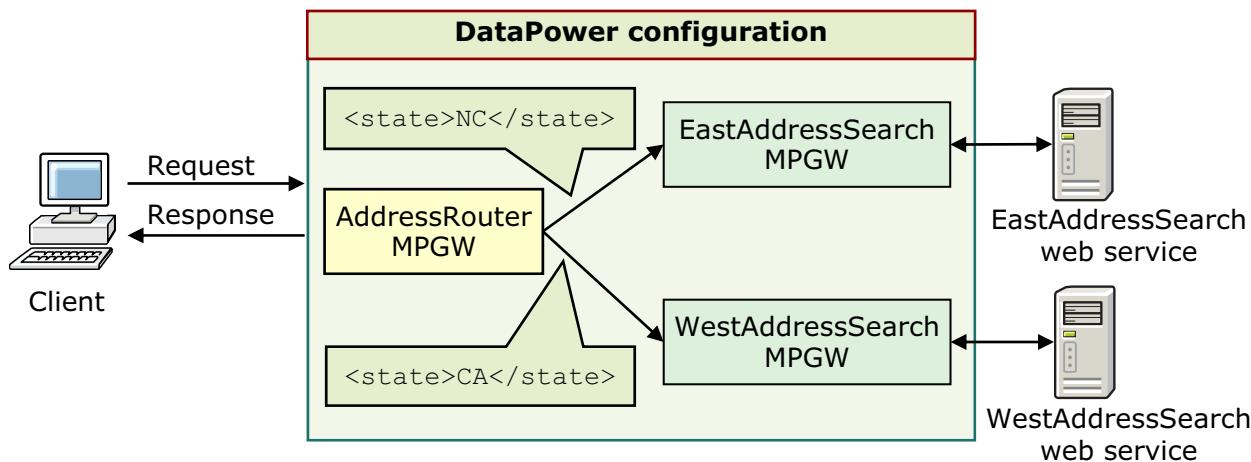
The WS-Addressing message ID is a unique message identifier.

The WS-Security username token can contain a password digest, which is a hashed value of the password. Optionally, it can contain a nonce value, which is a unique base 64-bit encoded value.

Custom XPath uses content from the XML message to detect replay attacks.

Content-based routing

- With content-based routing, the service can use a back-end service at run time that is based on incoming message content
 - The service type must be **dynamic back-end**
- Example:
 - Route requests to different servers based on `<state>` value



© Copyright IBM Corporation 2014

Figure 3-25. Content-based routing

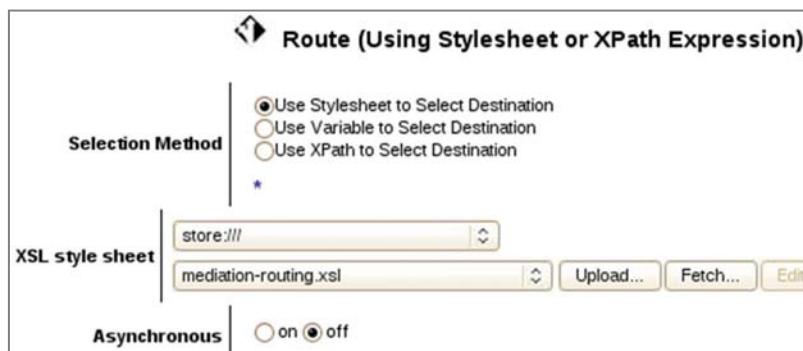
WE601 / ZE6011.1

Notes:

The content-based routing example that is shown in this slide routes the message to separate web services based on the value of the `<state>` field in the message. The **AddressRouter** multi-protocol gateway (MPGW) uses an XPath expression to extract the state value. If the value is "NC" (North Carolina), an eastern state in the United States, the message is forwarded to the **EastAddressSearch** multi-protocol gateway, which sends the message to the EastAddressSearch web service. If the value is "CA" (California), a western state in the United States, the message is forwarded to the **WestAddressSearch** multi-protocol gateway, which forwards the message to the WestAddressSearch web service.

Route action configuration

- The **Route** action dynamically routes XML messages by using:
 - Style sheet (default): routes by using a style sheet
 - XPath: routes by using an XPath expression
 - Variable: routes to a specified destination specified in a variable
- Dynamically specify the endpoint host address and port number



© Copyright IBM Corporation 2014

Figure 3-26. Route action configuration

WE601 / ZE6011.1

Notes:

The XPath Routing Map allows you to specify static destinations that are based on the evaluation of an XPath expression.

The XSL style sheet that is used in a **Route** action can use the DataPower extension function `<dp:set-target>` to set the endpoint.

Style sheet programming with dynamic routing

- <dp:set-target(*host, port, isSSL, sslProxyProfile*) />
 - Specify the back-end host, port, and optionally, SSL
 - Cannot specify the protocol
- <dp:xset-target(*XPath, XPath, XPath, sslProxyProfile*) />
 - Extended version of <dp:set-target> that evaluates attributes as XPath expressions
- <dp:url-open(...)/>
 - Opens a URL connection and places the response in the output that is named in the OUTPUT context

```
<dp:url-open
    target="http://example.com:2064/echo" response="xml">
    <xsl:copy-of select=". . ."/>
</dp:url-open>
```

- dp:soap-call(*url, msg, sslProxyProfile, flags, soapAction, httpHeaders*) />
 - Sends a SOAP message and obtains a response from the call

© Copyright IBM Corporation 2014

Figure 3-27. Style sheet programming with dynamic routing

WE601 / ZE6011.1

Notes:

This example uses dp:soap-call in an XSL style sheet.

Set up a variable `call` to contain the XML message.

Use `dp:call-soap()` to send the message and save the response in a variable, `result`.

```
<xsl:variable name="result"
select="dp:call-soap('http://fn.com/test', $call)"/>
```

Use the `dp:soap-fault` extension function to generate a custom SOAP fault message.

The `dp:http-request-header(headerFieldName)` is a common extension function that is used to extract an HTTP header from a message.

Example:

```
<xsl:variable name="SOAPAction"
select="dp:http-request-header('SOAPAction')"/>
```

The **SOAPAction** parameter needs single quotation marks ('') because the function expects an XPath expression.

The equivalent usage of the `<dp:set-target>(. . .)` can also be accomplished by using DataPower service variables. For example, to set the back-end URI in a style sheet, use the following code:

```
<dp:set-variable name=" var://service/routing-url'"'
value=" http://1.2.3.1:2068'"'>
<dp:set-variable name=" var://service/URI'"'
value=" /SomeBank/services/checking'"'>
```

The **sslProxyProfile** parameter is the name of a DataPower **sslProxyProfile** object.



Results action

- The **Results** action sends the document in the input context to:
 - Destination URL, can be a list
 - Output context, if no destination URL is specified
- If the **Results** action is the last action in a rule, it is usually writing to the OUTPUT predefined context
- Use the **Results** action in the middle of the rule to send results asynchronously
 - Enable **Asynchronous** to send results to destination and continue processing in the rule
 - Can use a subsequent Event-sink action to wait on Results completion



© Copyright IBM Corporation 2014

Figure 3-28. Results action

WE601 / ZE6011.1

Notes:

The **Results** action is typically the last action in a rule, since it is used to return a response at the end of the service policy. Make sure that the input context contains the variable with the document to return to the client.

An alternative is to have the last action itself write to the OUPUT context.

The default **Results** action copies the input context to the output context.

Results asynchronous and multi-way results mode

- The **Results Asynchronous** action is similar to the **Results** action except that it:
 - Requires a destination URL
 - Does not wait for a response from the remote server
- When a **Results/Results Asynchronous** action specifies a list of remote server destinations, it is considered a **multi-way Results** action
 - Three options are given for the list: **Attempt All**, **First Available**, **Require All**
 - These options are in the **Advanced** tab

Results Asynchronous

Destination	<input type="text" value="http://"/>
Number of Retries	<input type="text" value="0"/>
Retry Interval	<input type="text" value="1000"/> msec

Advanced

Destination	<input type="text" value="http://"/>
Output Type	<input type="button" value="Default"/>
Asynchronous	<input checked="" type="radio"/> on <input type="radio"/> off
Multi-Way Results Mode	<input type="button" value="First Available"/>
Number of Retries	<input type="button" value="First Available"/> <input type="button" value="Attempt All"/> <input type="button" value="Require All"/>

© Copyright IBM Corporation 2014

Figure 3-29. Results asynchronous and multi-way results mode

WE601 / ZE6011.1

Notes:

A regular **Results** action can be set to asynchronous mode, which can be used with an **Event Sink** action to wait for the remote server response.

A **Results Asynchronous** action cannot have an output context.

If a **Results** or a **Results Asynchronous** action needs to specify multiple locations as destinations, you must use a variable to represent the destination. An example of the format of that variable is (from the product documentation):

```
<results mode="require-all" multiple-outputs="true">
<url input="context1">http://127.0.0.1:22223</url>
<url input="context2">http://127.0.0.1:22224</url>
<url input="context3">http://127.0.0.1:22225</url>
<url input="context4">http://127.0.0.1:22226</url>
</results>
```

Attempt All sends the results in the input context to all destinations and succeeds even if all the remote servers fail.

First Available attempts each destination in order and stops with success after successfully sending the input to at least one remote server.

Require All sends the input context to all destinations and fails if any of the remote servers fail.



Service settings

- Specifications on how the service operates
 - TCP connection parameters
 - HTTP versions
 - Connection timeout values
 - XML manager
 - Traffic monitors
 - XML threat protection
 - HTTP header injection and suppression
 - More...
- Varies by service type

© Copyright IBM Corporation 2014

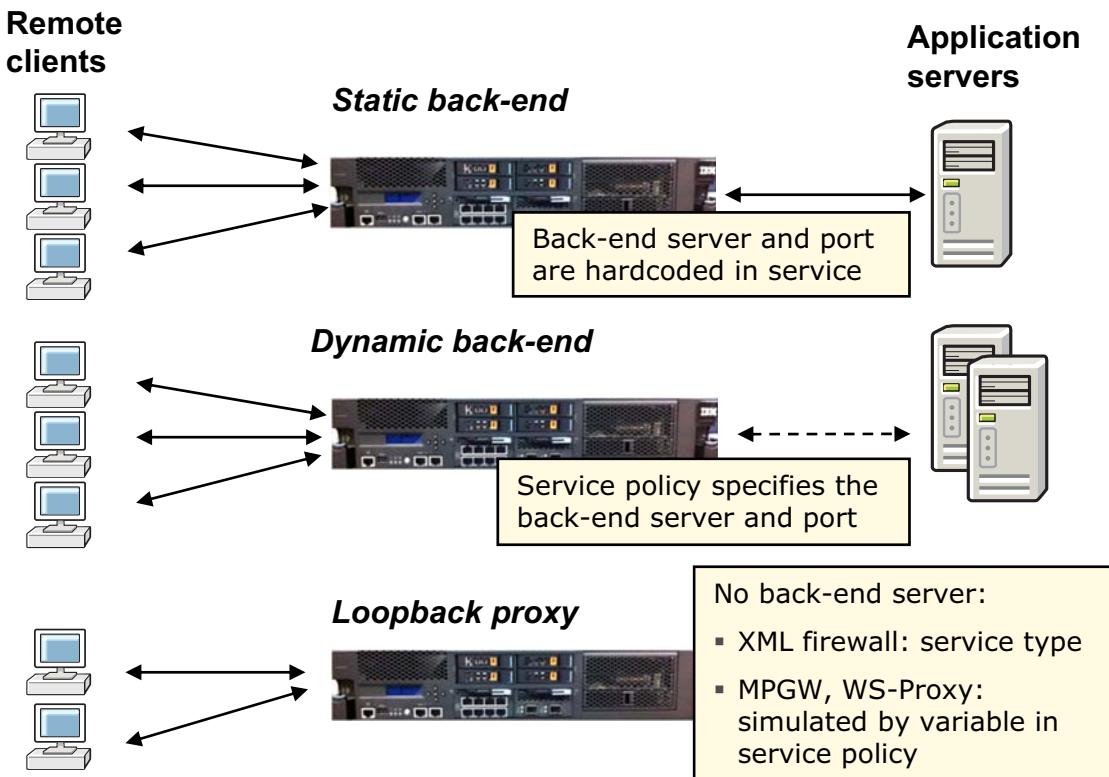
Figure 3-30. Service settings

WE601 / ZE6011.1

Notes:

Traffic monitors are covered in another unit.

Service types



© Copyright IBM Corporation 2014

Figure 3-31. Service types

WE601 / ZE6011.1

Notes:

The static back-end forwards traffic to a statically defined endpoint.

The dynamic back-end forwards traffic that is based upon the execution of a policy that specifies the back-end host address and port.

A loopback proxy does not forward the message to a back-end service after processing is complete. This service type is often useful for validation and transformation services.

A multi-protocol gateway (MPGW) and a web service proxy (WS-Proxy) can use a Set Variable action to set var://service/mpgw/skip-backside to "1". This setting makes these services act like a loopback proxy. Although you can use this variable in a web service proxy, it is unlikely.



URL rewriting

- Create a URL rewrite policy to rewrite some or all of a client URL

`http://www.example.com/myservice` `URL rewrite policy` `http://10.44.31.123/order`

- Create a URL rewrite rule
 - Specify expression to match URL
 - Define replacement expression

Adding new URL Rewrite Rule property
of **URL Rewrite Policy**

URL Rewrite Type	<input type="text" value="absolute-rewrite"/> *
Match Expression(PCRE)	<input type="text"/>
Input Replace Expression	<input type="text"/>
Stylesheet Replace Expression	<input type="text"/>
Input URL Unescape	<input type="radio"/> on <input checked="" type="radio"/> off
Stylesheet URL Unescape	<input checked="" type="radio"/> on <input type="radio"/> off
URL Normalization	<input type="radio"/> on <input checked="" type="radio"/> off

Save **Cancel**

© Copyright IBM Corporation 2014

Figure 3-32. URL rewriting

WE601 / ZE6011.1

Notes:

The URL rewrite policy executes at the service level and before the service policy.

Rewriting the URL at the service level affects the matching rule of the service policy. If you rewrite the URL, make sure that it still matches one of the matching rules.

A URL rewrite policy can also be executed within a service policy by adding a **Header Rewrite** action to the policy header and referencing a URL rewrite policy.

PCRE refers to Perl-compatible regular expression. The match expression must be entered in this syntax.

The five options available under **URL Rewrite Type** are:

- absolute-rewrite**: Rewrites the entire body of the URL
- content-type**: Rewrites the contents of the content-type header field
- header rewrite**: Rewrites the contents of a specific HTTP header field
- post-body**: Rewrites the data that is transmitted in the HTTP post body

The **Stylesheet Replace Expression** is used to specify a style sheet that transforms or filters a document that is identified by a rewritten URL.

The **Input URL Unescape** is used to specify whether URL-encoded characters (that is, %2F) are rewritten to literal character equivalents.

The **Stylesheet URL Unescape** is used to specify whether the style sheet identified in **Stylesheet Replace Expression** is subject to literal character replacement of URL-encoded characters.

The **URL Normalization** field is used to enable normalization of URL strings (for example, ..).

Optionally, if the **URL Rewrite Type** is **header-rewrite**, then a **Header Name** field is available to specify a target HTTP header field.

A URL rewrite policy can also be specified at the action level for transform, validate, and header rewrite actions.



XML Manager

- The XML Manager obtains and manages XML documents, style sheets, and other resources on behalf of one or more services
 - All services use the **default** XML Manager object
 - Accessed from the navigation bar by clicking **Objects > XML Processing > XML Manager**
- An XML Manager does the following functions:
 - Set manager-associated limits on the parsing of XML and JSON documents
 - Enable document caching
 - Perform extension function mapping
 - Enable XML-manager-based schema validation
 - Schedule an XML-manager-initiated processing rule

© Copyright IBM Corporation 2014

Figure 3-33. XML Manager

WE601 / ZE6011.1

Notes:

Click **Objects > XML Processing > XML Manager** to display the XML Manager objects list, which provides the list of XML Managers that are currently configured, along with their configuration details.

WebSphere Education



Default XML Manager configuration

Configure XML Manager

Main XML Parser Document Cache Extension Functions

XML Manager : default [up]

Apply Cancel Undo Export | View Log | View Status

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	Default XML-Manager
URL Refresh Policy	(none) <input type="button" value="..."/>
Compile Options Policy	(none) <input type="button" value="..."/>
XSL Cache Size	256 stylesheets
SHA1 Caching	<input checked="" type="radio"/> on <input type="radio"/> off
Static Document Call	<input checked="" type="radio"/> on <input type="radio"/> off
XSLT Expression Optimization	<input type="radio"/> on <input checked="" type="radio"/> off
Load Balance Groups	(empty) <input type="button" value="Add"/> <input type="button" value="..."/>
User Agent Configuration	default <input type="button" value="..."/> *

- The **default XML Manager** can be used and edited as any other user-created manager
- Creating an XML Manager requires the **name** field only
 - Modify basic default values or implement optional, enhanced functionality
- The URL refresh policy is used to schedule periodic updates of cached style sheets
- User agent is used to specify policies when invoking remote services

© Copyright IBM Corporation 2014

Figure 3-34. Default XML Manager configuration

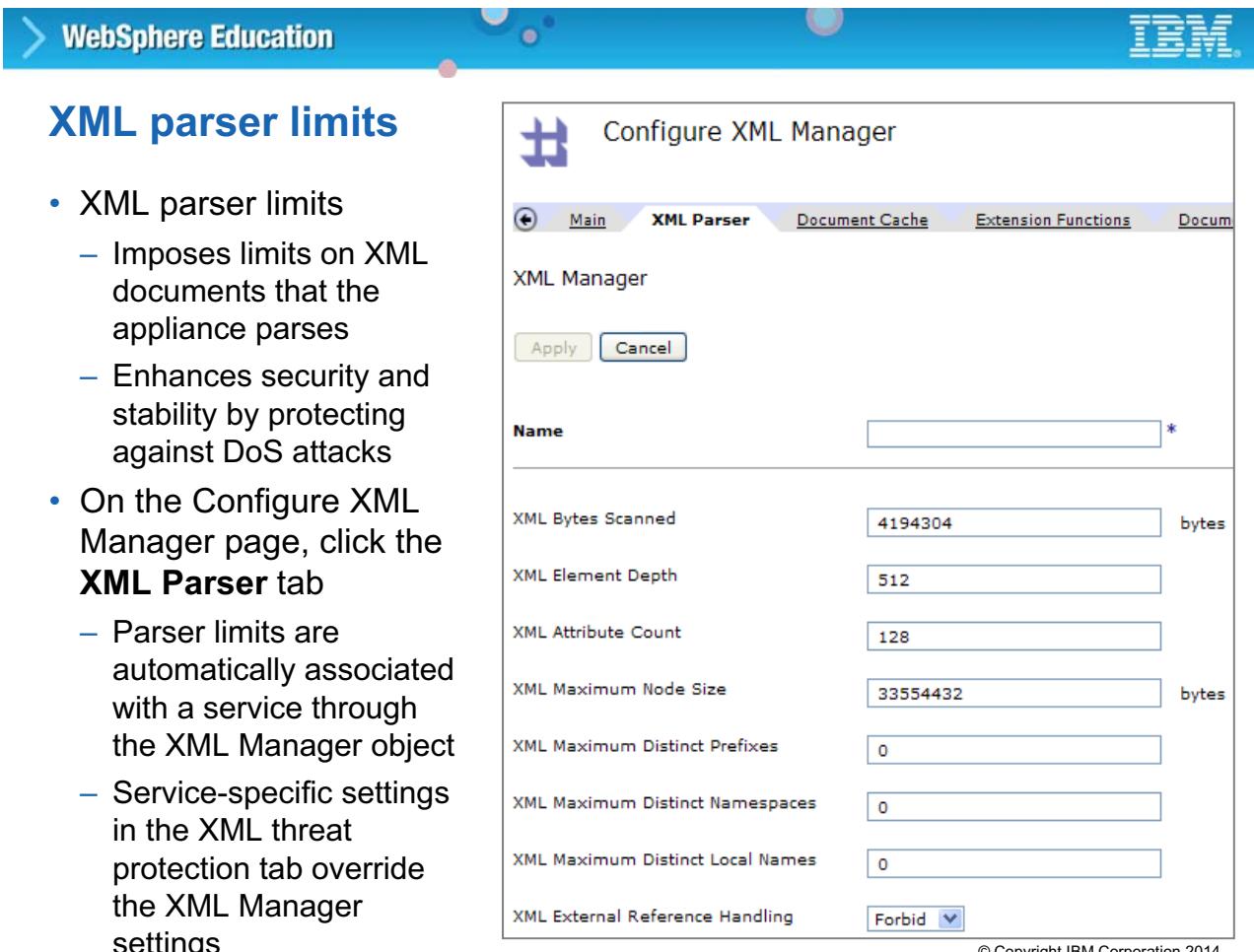
WE601 / ZE6011.1

Notes:

Each XML Manager maintains a cache of compiled style sheets to facilitate wire speed XML processing.

A load balancer group, or server pool, provides redundancy among back-end resources.

A User agent uses URL mappings to specify many options: proxy policies, SSL proxies, FTP client options, and more.



The screenshot shows the 'Configure XML Manager' page in the WebSphere Education interface. The 'XML Parser' tab is selected. The page includes fields for 'Name' (with a required asterisk), and various XML parser limit settings:

Setting	Value	Unit
XML Bytes Scanned	4194304	bytes
XML Element Depth	512	
XML Attribute Count	128	
XML Maximum Node Size	33554432	bytes
XML Maximum Distinct Prefixes	0	
XML Maximum Distinct Namespaces	0	
XML Maximum Distinct Local Names	0	
XML External Reference Handling	Forbid	dropdown

At the bottom right of the page, there is a copyright notice: © Copyright IBM Corporation 2014.

Figure 3-35. XML parser limits

WE601 / ZE6011.1

Notes:

The XSL proxy service does not have an XML threat protection tab.

"DoS" is "denial of service."

XML Parser limits:

- XML Bytes Scanned:** The maximum number of bytes scanned in one message by the XML parser. "0" indicates no restriction.
- XML Element Depth:** The maximum depth of element nesting.
- XML Attribute Count:** The maximum number of attributes that are allowed within an XML element.
- XML Maximum Node Size:** The maximum size of an individual XML node in bytes.
- XML External Reference Handling:** To allow references in DTD to URLs outside the appliance.
- XML Maximum Distinct Prefixes:** Defines the maximum number of distinct XML namespace prefixes in a document.

- **XML Maximum Distinct Namespaces:** Defines the maximum number of distinct XML namespace URIs in a document.
- **XML Maximum Distinct Local Names:** Defines the maximum number of distinct XML local names in a document.



JSON document limits within the XML manager

- JSON Settings
 - Imposes limits on JSON documents that the appliance parses
 - Enhances security and stability by protecting against DoS attacks
- On the Configure XML Manager page, click the **Main** tab
 - Parser limits are automatically associated with a service through the XML Manager object
 - A JSON Settings object is associated with the XML Manager object

The screenshot shows the 'JSON Settings' configuration dialog box. At the top are 'Apply' and 'Cancel' buttons. Below is a 'Name' field with a required asterisk (*). The 'General' section includes an 'Administrative State' field with radio buttons for 'enabled' (selected) and 'disabled'. The 'Label-Value pairs' section contains three fields: 'Maximum label length' (256 characters), 'Maximum value length for strings' (8192 characters), and 'Maximum value length for numbers' (128 characters). The 'Threat Protection' section contains two fields: 'Maximum nesting depth' (64 levels) and 'Maximum document size' (4194304 bytes).

© Copyright IBM Corporation 2014

Figure 3-36. JSON document limits within the XML manager

WE601 / ZE6011.1

Notes:

A JSON Settings object is selected to be attached to an XML manager. The JSON Settings choice is on the **Main** tab of the XML manager page.

JSON Parser limits:

- **Maximum label length:** The maximum label length limits the number of characters in the label portion of the JSON label-value pair. The length includes any white space that is contained between quotation marks. Enter a value in the range 256 - 8192. The default value is 256.
- **Maximum value length for strings:** The maximum value length limits the number of characters in the value portion of a label-value pair when the value is a string. The length includes any white space that is contained between quotation marks. Enter a value in the range 8192 - 2097152. The default value is 8192.
- **Maximum value length for numbers:** The maximum number length limits the number of characters in the value portion of a label-value pair when the value is a number. The number must be a contiguous string of characters that contain no white space. The number can include a minus sign and a positive or negative exponent. Enter a value in the range 128 - 256. The default value is 128.

- **Maximum nesting depth:** The maximum nesting depth provides threat protection by limiting the number of nested label-value pairs that are allowed in the JSON message. Enter a value in the range 64 - 256. The default value is 64.
- **Maximum document size:** The maximum document size provides threat protection by limiting the number of bytes in the body of the JSON message. If the message is converted to JSONx, the maximum document size specifies the size before conversion to JSONx. Note that the document size of the JSON message and the size of the JSONx equivalent might differ. Enter a value in the range 4194304 - 134217728. The default value is 4194304.

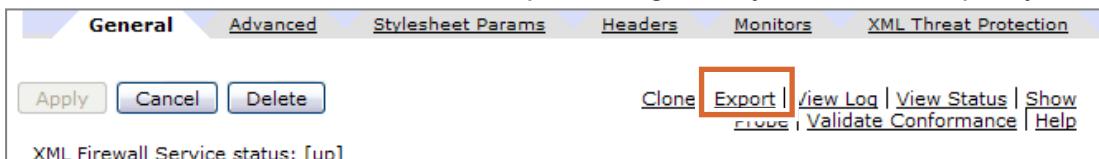
If no JSON Settings object is associated with a service's XML manager, the default values are in effect.

Because the XML parser is used in addition to the JSON parser when parsing a JSON document, the more restrictive parser limits (JSON or XML) apply.



Exporting a service configuration

- Export a .zip file of the service configuration
 - The saved configuration can be imported on another device
 - Allows for a more productive way to manage multiple configurations
 - Available in an XML firewall, multi-protocol gateway, web service proxy



- An object can be exported from **Administration > Configuration > Export Configuration**



© Copyright IBM Corporation 2014

Figure 3-37. Exporting a service configuration

WE601 / ZE6011.1

Notes:

Click **Export** to download a .zip file of the XML firewall configuration. The .zip file contains only the configuration data and files of the selected XML firewall service.

Click **Administration > Configuration > Export Configuration** to have more control over the objects and files that are exported.

Notice that there is an Import Configuration as well.



Cloning an XML firewall configuration

- Cloning
 - Creates a “near-copy” of an existing XML firewall
 - Referenced objects such as a service policy are referenced but are not copied
 - Allows for an existing configuration to be duplicated and configured with minor changes

The screenshot shows the 'Configure XML Firewall' interface. At the top, there are tabs: General, Advanced, Stylesheet Params, Headers, Monitors, and XML Threat Pr. Below the tabs are buttons for Apply, Cancel, and Delete. To the right of these buttons is a 'Clone' button, which is highlighted with a red box. Other buttons include Export, View Log, and View Status. Below the buttons, a message says 'XML Firewall Service status: [up]'. The main area is titled 'General Configuration' and contains fields for 'Firewall Name' (LoopbackXMLFW), 'Comments', and 'Firewall Type' (Loopback). On the right side, there are sections for 'XML Manager' (default), 'Processing Policy' (LoopbackXMLFW), and 'URL Rewrite Policy' (none). At the bottom right of the interface, it says '© Copyright IBM Corporation 2014'.

Figure 3-38. Cloning an XML firewall configuration

WE601 / ZE6011.1

Notes:

Use **Clone** to initiate the cloning process.

Since the XML firewall is a top-level object (no other objects depend upon it), you can delete a firewall at any time. Deleting the XML firewall does not delete any of the objects that the firewall uses (such as the service policy).

Make sure to change the port number of the cloned XML firewall.

Cloning is not available for web service proxies or multi-protocol gateways.

Troubleshooting a service configuration

- The system log is the first place to start your problem determination exercise
 - Click the “magnifying glass” icon to open the system log for entries on the selected service (XML firewall, in this example)

XML Firewall Name	Op-State	Logs	Req-Type	Local Address	Port	Resp-Type	Remote Address	Port
AddressRouter	up		soap	0.0.0.0	2050	soap		

- Logs are arranged in reverse chronological order
 - Latest information is at the top

System Log for XML Firewall Service "AddressRouter"

Help

Refresh Log Target: default-log Filter: (none) (none)

current time: 20:59:50 on 2011-07-18

time	category	level	tid	direction	client	msgid	message	Show last 50
Fri Jul 01 2011								
01:14:38	mgmt	notice	95			0x00350014	xmlfirewall (AddressRouter): Operational state up	
01:14:38	mgmt	notice	95			0x00350016	xmlfirewall (AddressRouter): Service installed on port	
01:14:38	mgmt	notice	95			0x00350015	xmlfirewall (AddressRouter): Operational state down	
01:14:38	mgmt	warn	95			0x00340017	xmlfirewall (AddressRouter): Service removed from port	

- Details on troubleshooting are covered in another unit

© Copyright IBM Corporation 2014

Figure 3-39. Troubleshooting a service configuration

WE601 / ZE6011.1

Notes:

The system log displayed by the XML firewall is a filtered version of the main system log, and it shows only the events that your XML firewall generates.



Unit summary

Having completed this unit, you should be able to:

- List the basic structural components of a service and describe their relationships
- List the ways a service configures its front-side access and back-side connections
- Use the policy editor to configure a service policy
- Create a service policy with actions that process the client request or server response
- List some of the processing actions and describe their function
- Configure service-wide settings such as:
 - Service type: static back-end, dynamic back-end, and loopback proxy
 - XML Manager
 - URL rewriting

© Copyright IBM Corporation 2014

Figure 3-40. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: A service has a single policy with many rules, and each rule has many actions.
2. True or False: PIPE improves the processing efficiency by eliminating the need for temporary storage of processed documents. This technique is used for streaming documents through the appliance.
3. True or False: All services support the loopback proxy mode.
4. What is the impact of using a URL rewrite policy on a service policy?
 - A. The URL rewrite policy rewrites the user's cookies
 - B. The URL rewrite policy might rewrite the message URL, so the **Match** actions in the service policy rules need to account for the rewrite
 - C. The URL rewrite policy might rewrite the service policy to another service

© Copyright IBM Corporation 2014

Figure 3-41. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.



Checkpoint answers

- 1. True.**
- 2. True.**
- 3. False.** Of the primary services that are presented, only the XML firewall supports the loopback proxy mode. The loopback can be simulated in the multi-protocol gateway and the web service proxy by using a DataPower variable within the service policy.
- 4. B.**

© Copyright IBM Corporation 2014

Figure 3-42. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 2



Creating a simple XML firewall

© Copyright IBM Corporation 2014
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

9.0

Figure 3-43. Exercise 2

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

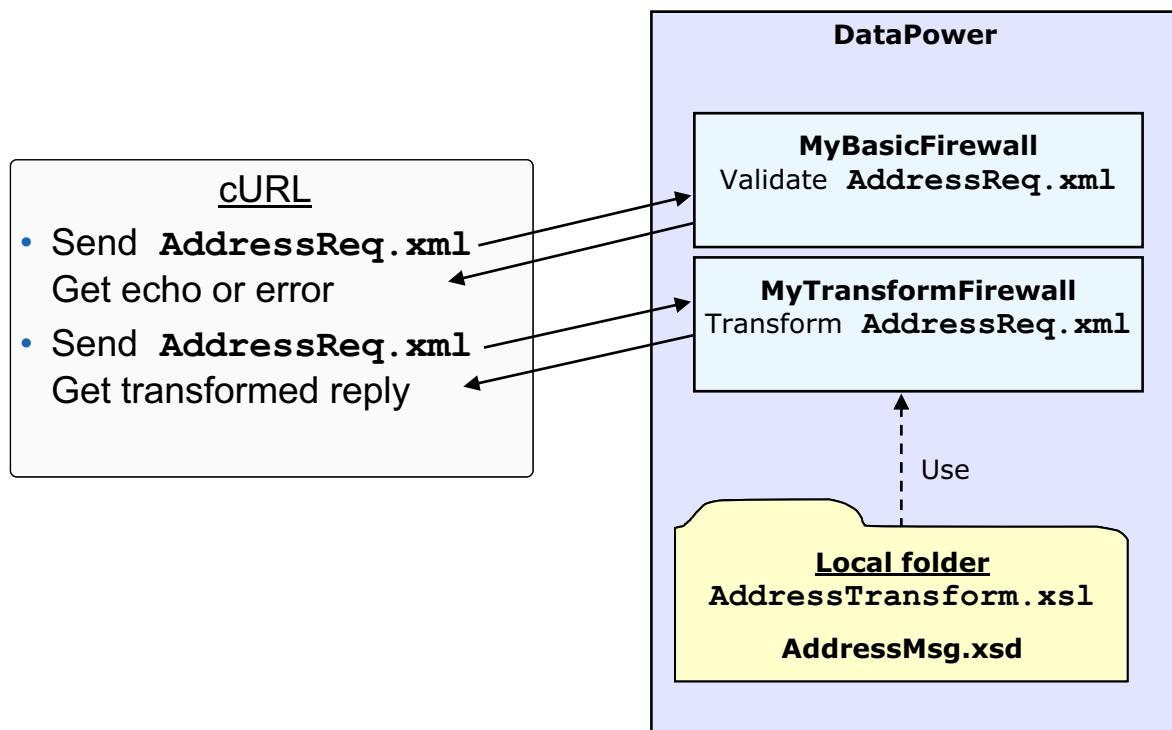
© Copyright IBM Corporation 2014

Figure 3-44. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview



© Copyright IBM Corporation 2014

Figure 3-45. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 4. Multi-protocol gateway service

What this unit is about

This unit describes the features of the multi-protocol gateway in the IBM WebSphere DataPower SOA Appliance. The gateway allows a many-to-many service mapping: multiple transport protocols can access a list of operations, and more than one back-end service can provide the implementation for these operations.

What you should be able to do

After completing this unit, you should be able to:

- Configure a multi-protocol gateway to provide a service over a set of different protocols
- Configure a connection to a static back-end service
- Configure a processing rule to select a back-end service at run time

How you will check your progress

- Checkpoint
- Exercise 4: Configure a multi-protocol gateway service

References

[http://www.ibmsystemsmag.com/mainframe/trends/
IBM-Announcements/datapower_ims_appliance/?page=1](http://www.ibmsystemsmag.com/mainframe/trends/IBM-Announcements/datapower_ims_appliance/?page=1)



Unit objectives

After completing this unit, you should be able to:

- Configure a multi-protocol gateway to provide a service over a set of different protocols
- Configure a connection to a static back-end service
- Configure a processing rule to select a back-end service at run time

© Copyright IBM Corporation 2014

Figure 4-1. Unit objectives

WE601 / ZE6011.1

Notes:

What is a multi-protocol gateway?



A multi-protocol gateway (MPG) connects client requests that are sent over one or more transport protocols to a back-end service with the same or a different protocol

- **Front side protocol handlers** accept requests from the client over a specific protocol
- **Rules** within a document processing policy inspect, modify, and route messages from the client to the back-end service
- **Back-end transports** forward the processed request to the back-end service
 - **Static back ends** route the request to a specific destination over a specific transport
 - **Dynamic back ends** rely on processing rules to determine to which endpoint and over which transport to deliver the request

© Copyright IBM Corporation 2014

Figure 4-2. What is a multi-protocol gateway?

WE601 / ZE6011.1

Notes:

Protocol handlers at a glance (1 of 2)

Handlers	Description
HTTP	Supports GET and POST operations The POST operations payload might contain XML, SOAP, DIME, SOAP with attachments, or MTOM
HTTPS	Supports the same features as the HTTP protocol, which is secured over Transport Layer Security (TLS)
Stateful raw XML	A stateful implementation that allows messages to flow between the client and the server with persistent connections
Stateless raw XML	Supports the same features as the stateful raw XML protocol, with a stateless implementation
IBM WebSphere MQ	Places and retrieves messages on GET and PUT queues from an IBM WebSphere MQ system
TIBCO EMS	Supports the TIBCO Enterprise Message Service product

© Copyright IBM Corporation 2014

Figure 4-3. Protocol handlers at a glance (1 of 2)

WE601 / ZE6011.1

Notes:

MTOM: Message Transmission Optimization Mechanism. This W3C recommendation is for vendor-neutral and platform-neutral attachments in the SOAP environment.

Raw XML messages begin and end with the root XML node over a TCP/IP connection; no headers are included, as with HTTP.

For an overview on IBM WebSphere MQ messaging, review the WebSphere MQ white paper, "The continuing benefits of commercial messaging." This paper might be found by entering the title for a search at <http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>

The TIBCO Enterprise Message Service product website provides a summary of its features at <http://www.tibco.com>

Protocol handlers at a glance (2 of 2)

Handlers	Description
FTP poller	Polls a remote FTP server for input
FTP server	Accepts connections from FTP clients
SFTP poller	Polls a remote SFTP server for input
SFTP server	Accepts connections from SFTP clients
NFS poller	Polls an NFS server for input
WebSphere JMS	Processes JMS messages received from WebSphere Application Server
IMS Connect / Callout	Accepts incoming IMS protocol requests and can initiate IMS connections on the back side

© Copyright IBM Corporation 2014

Figure 4-4. Protocol handlers at a glance (2 of 2)

WE601 / ZE6011.1

Notes:

The FTP poller front side handler object polls inside the director for files from an FTP server. The FTP server URL is specified as `ftp://user:password@host:port/path/path/`

A regular expression can be used to restrict the files within the directory that are polled.

The FTP server front side handler object acts as a virtual FTP server. There is a limited amount of storage on the DataPower appliance, hence, you should be careful when you are using this object.

The NFS poller is configured in a way that is similar to an FTP poller, except that it polls an NFS server for input.

The IMS Connect handler enables communication between the appliance and an IMS Connect server.



Front side protocol handlers

- Protocol handlers provide protocol-specific connection points to clients that request services from a server
- The following transport protocols are supported:
 - HTTP, HTTPS
 - SFTP, FTP, NFS
 - IBM WebSphere MQ
 - IBM WebSphere JMS, TIBCO Enterprise Messaging System (EMS)
 - Stateless and stateful raw XML
 - IMS Connect
- Each instance of an HTTP, HTTPS, SFTP, FTP, and raw XML protocol handler listens to a specific pair of IP address and port number
- Each WebSphere MQ protocol handler connects to a WebSphere MQ queue manager and the associated PUT and GET queues
- Each WebSphere JMS and TIBCO EMS handler connects to a JMS server and the associated GET and PUT queues

© Copyright IBM Corporation 2014

Figure 4-5. Front side protocol handlers

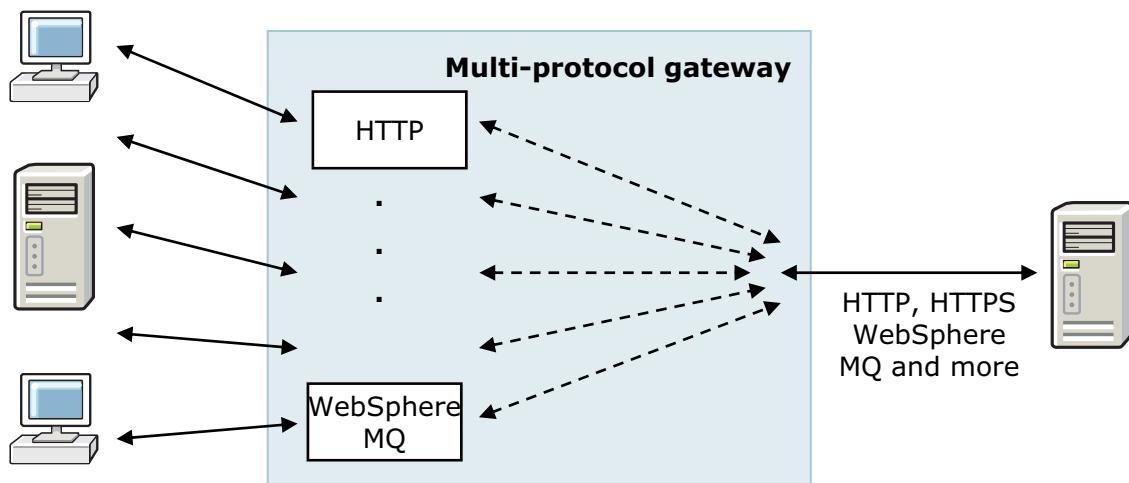
WE601 / ZE6011.1

Notes:

Some protocol handlers appear only when you have the appropriate license on the appliance: for example, WebSphere MQ, WebSphere JMS, and TIBCO EMS.

Static back-end gateway

- With a static back-end system, the multi-protocol gateway accepts requests with any of the defined protocol handlers
 - A static URL determines the destination for all traffic
 - The connection to the back-end system can employ any of the protocols shown (HTTP, HTTPS, WebSphere MQ, or Tibco EMS)



© Copyright IBM Corporation 2014

Figure 4-6. Static back-end gateway

WE601 / ZE6011.1

Notes:

As the name suggests, a static back-end gateway maps exactly one back-end resource for all requests that pass through the gateway. WebSphere MQ, WebSphere JMS, and TIBCO EMS resources require more information to describe the back-end resource. The IBM WebSphere DataPower SOA Appliance uses a custom syntax for these resources.

Dynamic back-end gateway

- A dynamic back-end gateway selects the back-end service and its respective protocol at execution
 - Messages that are sent over a stateful raw XML or an IMS Connect front side protocol handler are forwarded to a similar back side handler

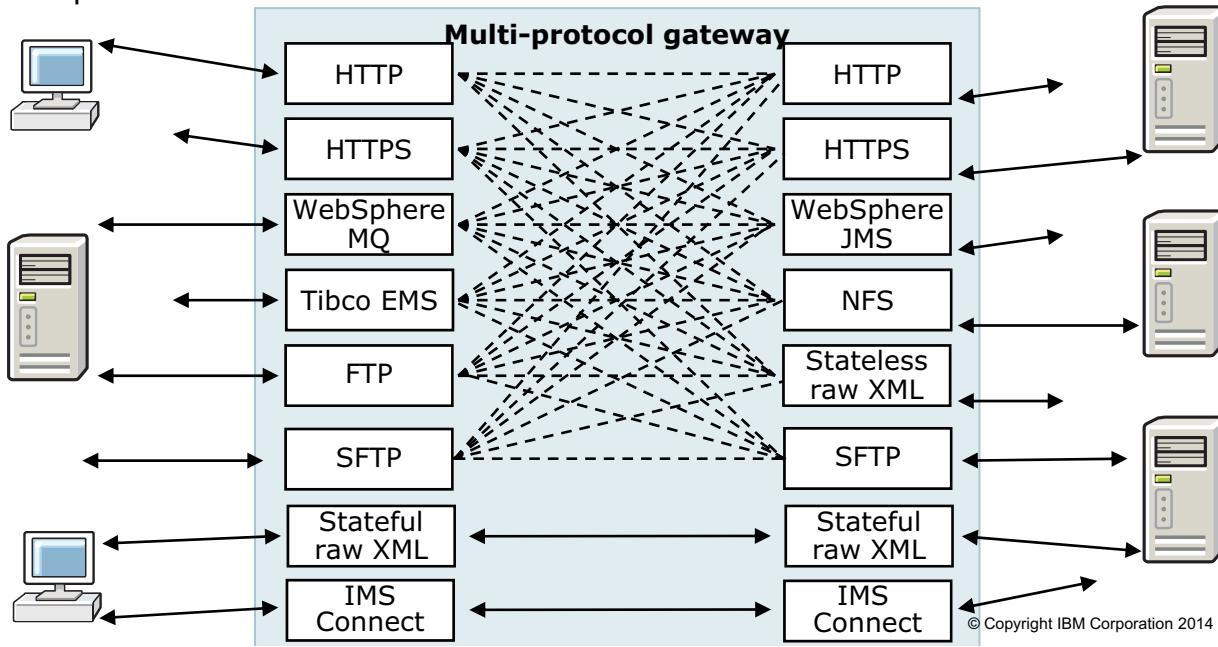


Figure 4-7. Dynamic back-end gateway

WE601 / ZE6011.1

Notes:

The route action or a url-open within a style sheet specify dynamic back-ends.

With the stateful raw XML handler, the client sends a message by a stateful communication protocol, such as an HTTP session. The handler preserves the session from the client to the back-end service. For this reason, the stateful raw XML front side handler can be matched only with the stateful raw XML back-end handler.

Multi-protocol gateway and XML firewall compared

- The multi-protocol gateway offers all of the same message processing capabilities as an XML firewall, regardless of the transport protocol chosen:
 - Encrypts and decrypts the entire message or individual token
 - Signs and verifies the entire message or individual token
 - Validates XML messages
 - Applies a custom XML transform style sheet
 - Authenticates clients and authorizes access to resources
- The service level management (SLM) policy action tracks and shapes message traffic
- Unlike the XML firewall, the gateway does not loop the results from a document processing rule back to the client
 - Use a separate XML firewall to configure a loopback proxy
- In general, a multi-protocol gateway is selected because it provides more capability for future enhancements

© Copyright IBM Corporation 2014

Figure 4-8. Multi-protocol gateway and XML firewall compared

WE601 / ZE6011.1

Notes:

The multi-protocol gateway inherits most of the features from the XML firewall object. In a sense, the gateway provides multiple front side and back-end handlers to the XML firewall. The only exception is the loopback proxy feature.

Use the **Advanced** action to enforce a service level management (SLM) policy in a processing rule.

In the previous exercise, you used XML firewalls because they are easier to learn. If this scenario was a more realistic environment, the services would have been implemented as multi-protocol gateways.

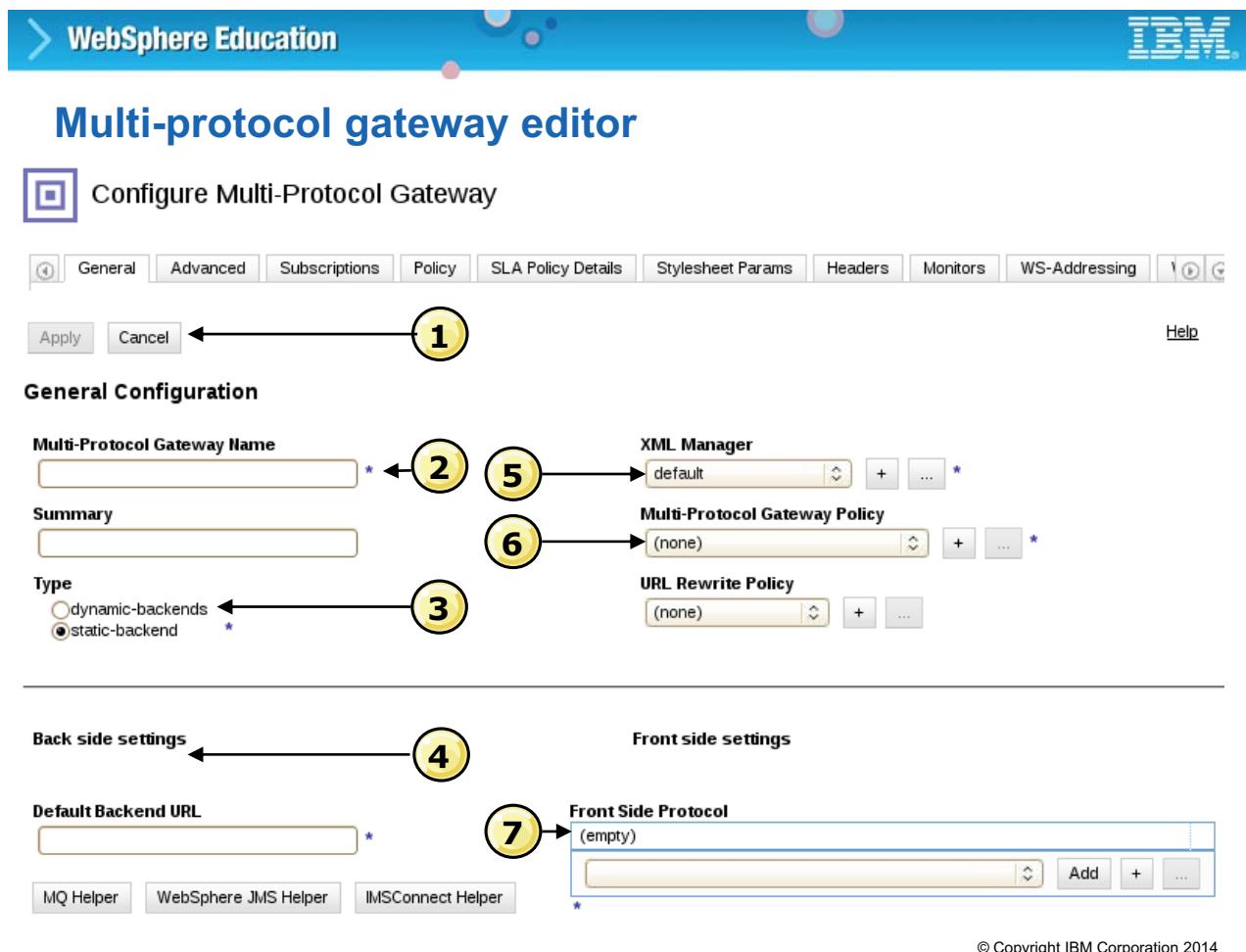


Figure 4-9. Multi-protocol gateway editor

WE601 / ZE6011.1

Notes:

The multi-protocol gateway inherits most of the XML firewall features. The following list explains some new or modified settings that are specific to the multi-protocol gateway. For an explanation on the remaining settings in the editor, refer to the XML firewall presentation.

Remember to click **Apply** to commit changes that are made in the editor.

Specify a name and a description for the multi-protocol gateway.

Specify whether the back-end service URL is defined at design time (static back-end) or defined at execution (dynamic back-end). Keep in mind that the left side of the editor covers **Gateway to back-end** settings, while the right side covers **Client to gateway** settings.

For a static back-end, enter the endpoint address for the back-end service.

The **XML Manager** handles style sheet and document processing options. This setting is the same as a regular XML firewall. In fact, the gateway can reuse an XML Manager that was created for an XML firewall.

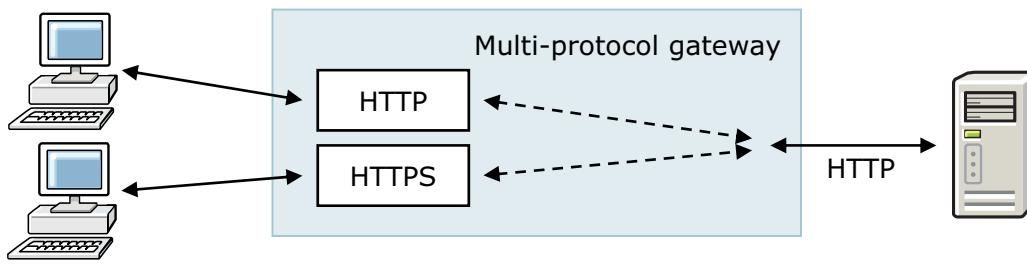
The **Multi-Protocol Gateway Policy** defines the rules in a document processing policy. The processing rule actions are the same as the ones that are available to the XML firewall, with the addition of the SLM policy action.

The **Front Side Protocol** section lists one or more front side handlers that are configured for the gateway. You can either add an existing front side protocol handler or create a protocol handler for the gateway.

The **Propagate URI** choice must be set to **off** for non-HTTP back-end protocols.

Scenario 1: Provide HTTP and HTTPS access

- Create a multi-protocol gateway to accept web service requests from either a secured or an unsecured HTTP connection
 - All requests are sent to the back-end web service over an HTTP connection
 - Validates web service request messages that pass through the gateway



© Copyright IBM Corporation 2014

Figure 4-10. Scenario 1: Provide HTTP and HTTPS access

WE601 / ZE6011.1

Notes:

In this scenario, the client can access the back-end service over a regular HTTP connection or a secure HTTPS connection. The IBM WebSphere DataPower SOA Appliance sits on the edge of the network: that is, the connection between the gateway and the back-end service exists in the intranet. The connection to the back-end service is made by an unsecured HTTP connection. For this scenario, assume that communication between the IBM DataPower SOA appliance and the back-end service is secure in a corporate intranet.



Step 1: Configure the back side transport

General Configuration

Multi-Protocol Gateway Name: EastAddressSearch * 1

Summary: East Address Search MPG

Type: static-backend 2

Back side settings

Backend URL: http://WSserver:9080/EastAddress/service * 3

MQHelper TibcoEMSHelper WebSphereJMSHelper
IMSSConnectHelper

User Agent settings 4

Match Property
Note: To edit the User Agent, please access via the XML Manager above.

SSL Client Crypto Profile: (none) 5

1. Provide a name and a summary for the newly created multi-protocol gateway
2. Select **static-backend** for a back-end service that is set at design time
3. Provide the HTTP address for the back-end service
4. Review the **User Agent settings**, if defined in the XML manager
 - **User Agent settings** match the identity of the sender of an HTTP message
5. For back-end services that use HTTPS, configure an **SSL Client Crypto Profile** for the connections

© Copyright IBM Corporation 2014

Figure 4-11. Step 1: Configure the back side transport

WE601 / ZE6011.1

Notes:

To create a multi-protocol gateway, click **New Multi-Protocol Gateway** from the WebGUI Control Panel.

The figure on this slide covers the left side of the main multi-protocol gateway editor.



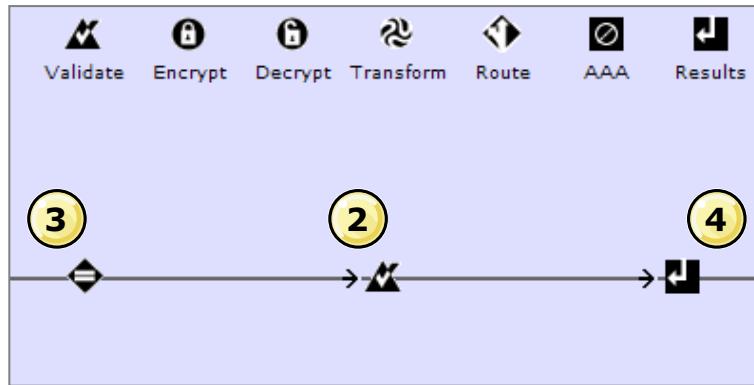
Step 2: Create a document processing rule

1. Create a multi-protocol gateway policy

- Define one or more processing rules for all messages that pass through the gateway

Policy:	1
Policy Name: AddressSearchMPGPolicy	
Apply Policy Cancel	

2. Add a **Validate** action to validate the document according to the back-end service WSDL file



3. Configure the **Match** action to accept any requests with a URL matching /EastAddressSearch

4. Add a **Results** action to output the processing rule results

5. Set the direction to handle messages inbound, outbound, or both

Rule Direction: Both Directions
--

© Copyright IBM Corporation 2014

Figure 4-12. Step 2: Create a document processing rule

WE601 / ZE6011.1

Notes:

After you define a processing rule in the policy, click **Apply** to save the changes that are made in the processing rule.

The **Match** action accepts calls with a particular URI path. The gateway automatically rejects any request if it does not match any of the defined rules.

The **Match action** must be the first action on any processing rule. The **Validate action** appears after the match rule.

The **Results action** directs the gateway to connect and send the message to the back-end service or the original client.

Step 3: Create the front side handlers

1. Create an **HTTP Front Side Protocol Handler**
 - See the next slide for a discussion on the HTTP Front Side Handler editor
2. Add the newly created front side handler to the gateway
3. Create an **HTTPS (SSL) Front Side Handler** and add it to the gateway

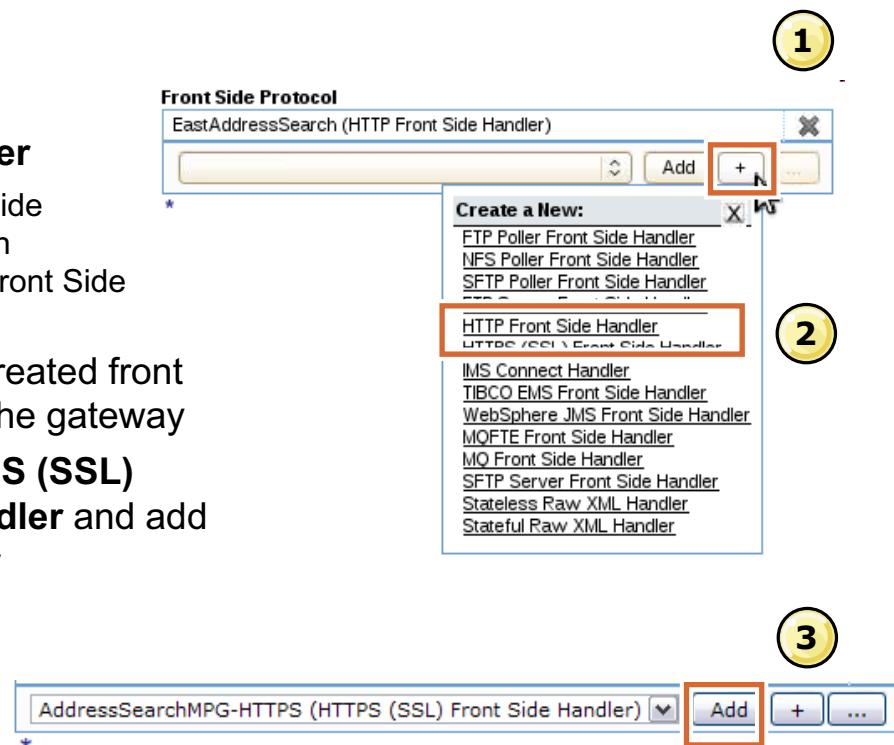


Figure 4-13. Step 3: Create the front side handlers

WE601 / ZE6011.1

Notes:

You can reuse front side protocol handlers that you created. However, you can associate the handler with only one service (XML firewall, Web Service Proxy, multi-protocol gateway, and other services) at a time.

Usually, a new handler is automatically added to the protocol list after you configure the handler.



Step 4: Configure the front side handler

1. Activate the handler by setting the **Admin State** to **enabled**
2. Set the **Local IP Address** to **dp_public_ip** to listen to request on all external ports
3. Specify a unique port number that the handler monitors
4. Select the HTTP version reported to the client
5. Choose which HTTP version and method to permit
 - Web service clients use the HTTP POST method to send SOAP request messages
6. For **HTTPS Handlers** only, specify the **SSL Proxy** profile

HTTP Front Side Handler: EastAddressSearch [up]

Apply Cancel Undo

Administrative State	1 <input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Local IP Address	2 dp_public_ip
Port Number	3 6957
HTTP Version to Client	4 HTTP 1.1
Allowed Methods and Versions	5 <input checked="" type="checkbox"/> HTTP 1.0 <input checked="" type="checkbox"/> HTTP 1.1 <input checked="" type="checkbox"/> POST method

SSL Proxy	6 Address SSL <input type="button"/> + ... *
Access Control List	(none) <input type="button"/> + ...

© Copyright IBM Corporation 2014

Figure 4-14. Step 4: Configure the front side handler

WE601 / ZE6011.1

Notes:

The **SSL Proxy** setting is unique to the HTTPS Front Side Handler editor. It does not appear in the HTTP Front Side Handler editor. All other options appear in both the HTTP and HTTPS front side handler.

The DataPower SOA appliance includes multiple Ethernet interfaces. Services can be mapped to one or more interfaces on the appliance. For a list of all available Ethernet interfaces, click **Network > Interface > Ethernet Interface** from the WebGUI.

Step 5: Configure the SSL proxy profile

1. Activate the SSL proxy by setting the **Admin State** to **enabled**
2. Configure the SSL proxy to receive SSL connections by setting the direction to **reverse**
3. Add or create a new **Reverse (Server) Crypto Profile** with the certificate-key pair with to secure the connection
4. Determine the settings for caching SSL sessions to clients

Name	<input type="text" value="AliceSSLProxy"/> *
Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
SSL Direction	Reverse <input type="button" value="▼"/> *
Reverse (Server) Crypto Profile	AliceCryptoProfile <input type="button" value="▼"/> <input type="button" value="+"/> <input type="button" value="..."/> *
Server-side Session Caching	<input checked="" type="radio"/> on <input type="radio"/> off
Server-side Session Cache Timeout	<input type="text" value="300"/> seconds
Server-side Session Cache Size	<input type="text" value="20"/> entries (x 1024)
Client Authentication Is	<input type="radio"/> on <input checked="" type="radio"/> off

© Copyright IBM Corporation 2014

Figure 4-15. Step 5: Configure the SSL proxy profile

WE601 / ZE6011.1

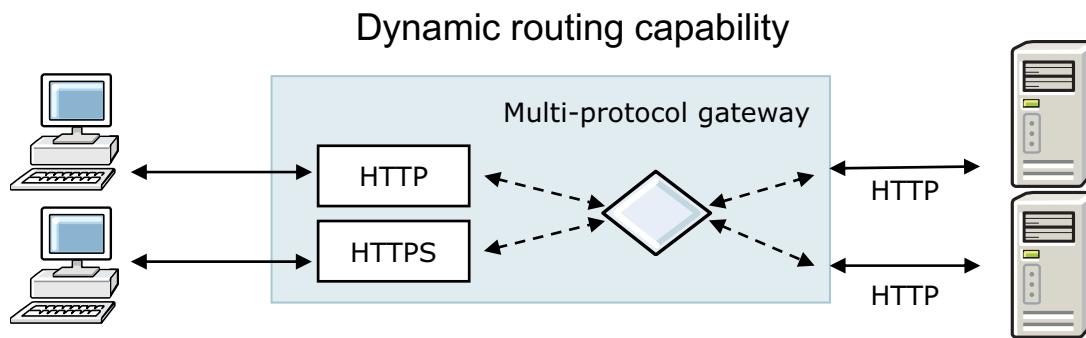
Notes:

The SSL Proxy profile defines a set of keys and certificates that the gateway uses to build an SSL connection. The **Forward (Client) Crypto Profile** defines the keys and certificates that are used in an SSL connection between the gateway and the back-end service. The **Reverse (Server) Crypto Profile** defines a set of keys and certificates that are used to establish an SSL connection from the client to the gateway.

Using the same crypto profile for the **forward** and **reverse** connections does not imply that the service uses the **same** SSL connection in both connections. Only the keys and certificates are shared; two distinct SSL connections are used for each side of the gateway.

Scenario 2: Dynamic back-end service

- Create a multi-protocol gateway with access to two back-end services, which are selected at execution
 - Accepts web service requests from either a secured or unsecured HTTP connection
 - All requests are sent to the back-end web service over an HTTP connection
 - Validates web service request messages that pass through the gateway



© Copyright IBM Corporation 2014

Figure 4-16. Scenario 2: Dynamic back-end service

WE601 / ZE6011.1

Notes:

The dynamic back-end service allows one endpoint on the DataPower SOA appliance to represent a single service, which is composed of different operations from different back-end services.

The diamond in the middle of the multi-protocol gateway diagram represents a decision point. One or more processing rules define the actual back-end service for each incoming request. The decision itself to choose one endpoint over another occurs at execution.

Step 1: Configure the back-end transport

1. Open the multi-protocol gateway for the previous scenario
2. Set the back-end transport type to **dynamic-backend**
 - A processing rule must set the back-end address
3. Add a processing rule to the multi-protocol gateway policy
4. Add a **Transform** action in a request rule
5. Specify a custom style sheet that targets the back-end service
6. Use a **URL Rewrite Policy** to change the URL path, if needed

The screenshot shows the configuration interface for a dynamic-backend gateway. It includes:

- Step 2:** A panel titled "Type" with a radio button selected for "dynamic-backend".
- Step 4:** A diagram showing a flow from a source node to a target node (represented by a swirl icon) with a yellow circle containing the number 4.
- Step 5:** A "Control File" section with a dropdown menu set to "local:///".
- Step 6:** A "URL Rewrite Policy" section with a dropdown menu set to "(none)".

© Copyright IBM Corporation 2014

Figure 4-17. Step 1: Configure the back-end transport

WE601 / ZE6011.1

Notes:

The following steps assume the multi-protocol gateway was created according to the first scenario. The actual back-end service is defined by a custom style sheet in a processing rule.

Sample service that targets a style sheet

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dp="http://www.datapower.com/extensions"
    xmlns:dpconfig="http://www.datapower.com/param/config"
    extension-element-prefixes="dp"
    exclude-result-prefixes="dp dpconfig" >

    <xsl:output method="xml"/>

    <xsl:template match="/">
        <xsl:copy-of select=". "/>
        <dp:set-target>
            <host>address.training.ibm.com</host>
            <port>9080</port>
        </dp:set-target>
    </xsl:template>

</xsl:stylesheet>
```

© Copyright IBM Corporation 2014

Figure 4-18. Sample service that targets a style sheet

WE601 / ZE6011.1

Notes:

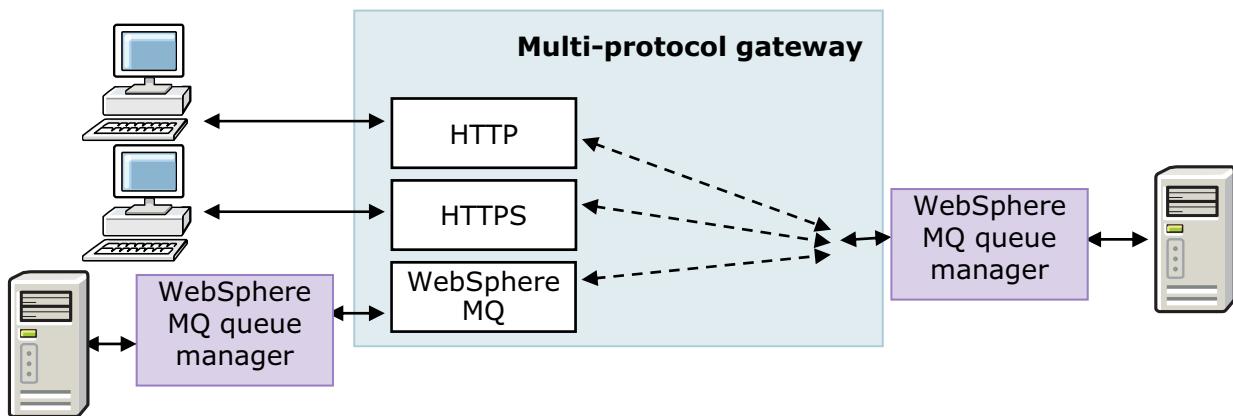
The `<dp:set-target>` built-in element defines the IP address (or host name) and the port for a particular back-end server. Additional attributes are available to set up an SSL connection to the back-end service.

This style sheet example matches any incoming message to one particular endpoint. In a real world scenario, different template match rules would trigger different `<dp:set-target>` settings.

You can also use a `url-open` element in a style sheet to communicate to a specific back-end service.

Scenario 3: Provide WebSphere MQ access

- Modify the multi-protocol gateway to accept requests from an IBM WebSphere MQ system
 - Request and response messages reside in queues on a WebSphere MQ queue manager
 - All requests are sent to the back-end web service over another set of WebSphere MQ queues
 - Validates web service request messages that pass through the gateway



© Copyright IBM Corporation 2014

Figure 4-19. Scenario 3: Provide WebSphere MQ access

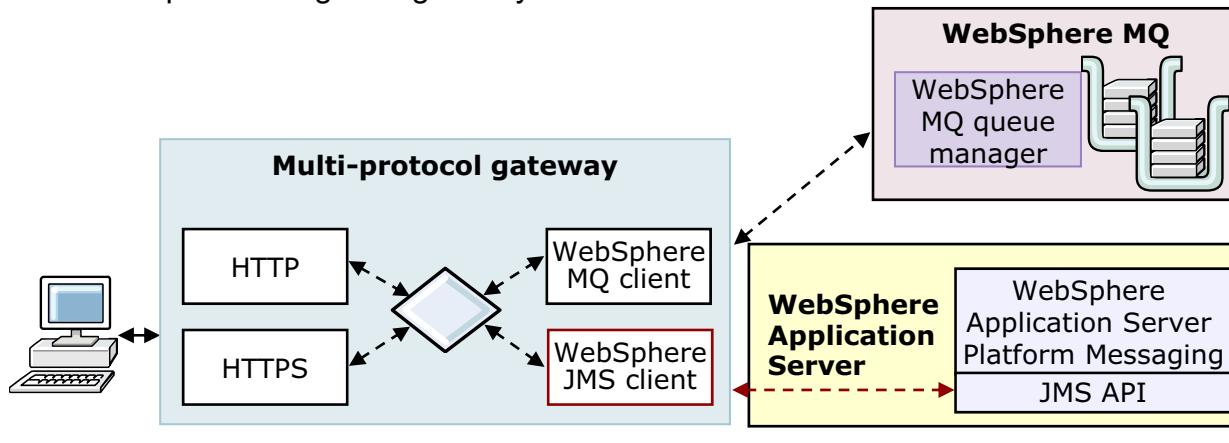
WE601 / ZE6011.1

Notes:

There is no requirement to use WebSphere MQ in both the front side and back side. The multi-protocol gateway can act as an HTTP-to-WebSphere MQ message converter, and the reverse.

Scenario 4: Provide WebSphere JMS access

- Modify the multi-protocol gateway so that it:
 - Also uses the JMS API to forward requests to IBM WebSphere Application Server platform messaging system
 - Request and response messages reside in queues
 - Back-end J2EE web services poll queues to obtain messages
 - Validates web service request messages that pass through the gateway



© Copyright IBM Corporation 2014

Figure 4-20. Scenario 4: Provide WebSphere JMS access

WE601 / ZE6011.1

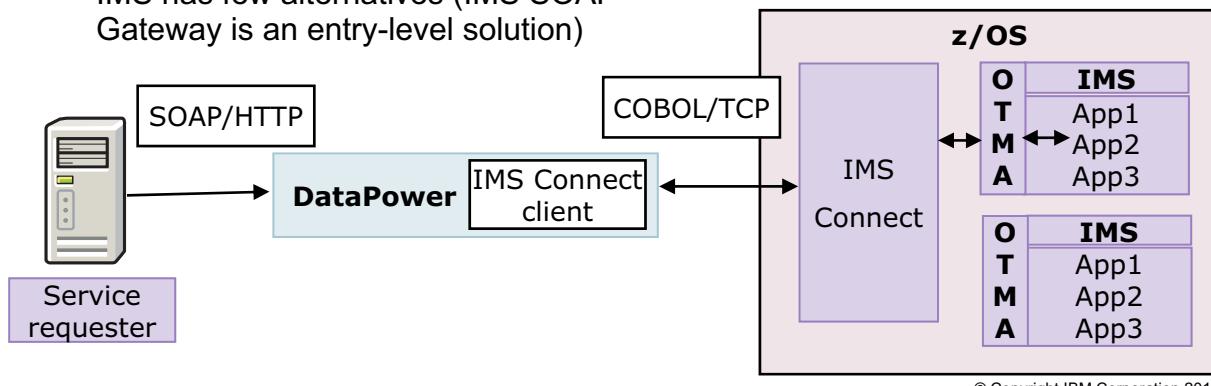
Notes:

WebSphere MQ and WebSphere Application Server are separate products that both support asynchronous messaging.

The WebSphere Application Server platform messaging engine maintains a set of queues that process asynchronous messages.

Scenario 5: Provide IMS Connect access

- Implement an “IMS Connect Client” on DataPower that natively connects to IMS Connect by using its custom request/response protocol with a well-defined header structure
 - Highly consumable for the common case
 - Highly extensible and integrates well with DataPower model
 - Accepts output from a mapping mediation (for example, SOAP-to-COBOL copybook)
- Removes the WebSphere MQ requirement of WS-enablement of IMS
 - IMS has few alternatives (IMS SOAP Gateway is an entry-level solution)



© Copyright IBM Corporation 2014

Figure 4-21. Scenario 5: Provide IMS Connect access

WE601 / ZE6011.1

Notes:

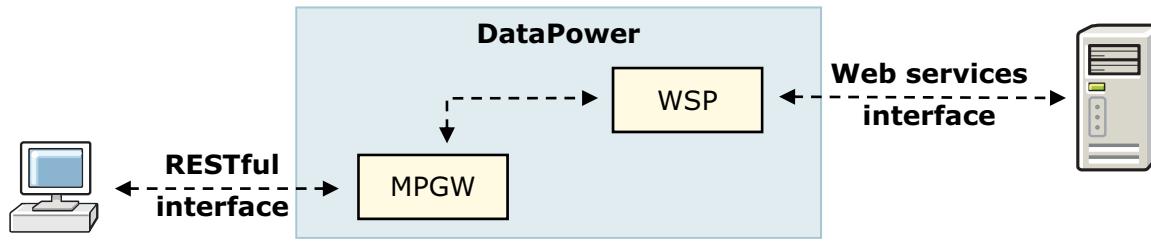
OTMA: Open Transaction Management Access.

The IMS OTMA facility is a transaction-based connectionless client/server protocol that runs on IMS Version 5.1 or later. It functions as an interface for host-based communications servers that access IMS TM applications through the z/OS Cross Systems Coupling Facility (XCF).

IMS Connect communicates to OTMA by XCF.

Scenario 6: Provide a RESTful interface

- Use a multi-protocol gateway to convert RESTful requests to a SOAP-based web service request
 - MPGW
 - RESTful request is converted to a SOAP request
 - Style sheet uses `dp:url-open` or `dp:soap-call` to send reformatted SOAP request to web service proxy (WSP)
 - SOAP response is converted to RESTful response
 - WSP
 - WSP is unaware of original RESTful style
 - Allows normal web service processing like AAA, transformation, monitoring



© Copyright IBM Corporation 2014

Figure 4-22. Scenario 6: Provide a RESTful interface

WE601 / ZE6011.1

Notes:

For a discussion of REST and DataPower, see: *Implementing REST services with WebSphere DataPower SOA Appliances* at

http://www.ibm.com/developerworks/websphere/techjournal/0903_peterson/0903_peterson.html

Check developerWorks for more DataPower articles, and search the DataPower Information Center for REST support.



REST support in DataPower

- **Process Messages whose body is empty** option in multi-protocol gateway and XML firewall services
- New matching rule type **HTTP Method** in Match action
- **HTTP Method** on dp:url-open
- **Method Rewrite** Advanced Processing action
- **Method Rewrite** that uses a Set Variable action
- **JSON Encoding** in Convert HTTP action
- **JSON Choice** for request/response types

© Copyright IBM Corporation 2014

Figure 4-23. REST support in DataPower

WE601 / ZE6011.1

Notes:

Process Messages whose body is empty option: useful for RESTful message patterns in which some message flows might not incorporate a body but multi-step rules still need to run. It bypasses the built-in "One Way Exchange Pattern" in multi-step.

Matching Rule type **HTTP Method**: supports processing rule that matches on the HTTP method: HEAD, DELETE, PUT, POST, and GET.

HTTP Method on dp:url-open: allows control of the HTTP method on a dp:url-open.

Method Rewrite Advanced Processing action: rewrites HTTP method requests to the back-end.

Method Rewrite by Set Variable action: another way to rewrite the HTTP method.

JSON encoding in Convert HTTP action: automates the conversion of JSON passed in a RESTful request to an XML representation called JSONX. There is also a style sheet to convert JSONX into JSON.

JSON Choice for request/response types: An additional request and response type of JSON is added.

Supported IMS features

- IMS support includes:
 - 1) IMS Connect, 2) IMS Synchronous Callout, and 3) IMS Database (DB)
- IMS DB support enables direct connection to an IMS DB through the IMS Universal JDBC driver. With it, applications can issue dynamic SQL calls, such as basic CRUD operations, against any IMS DB.

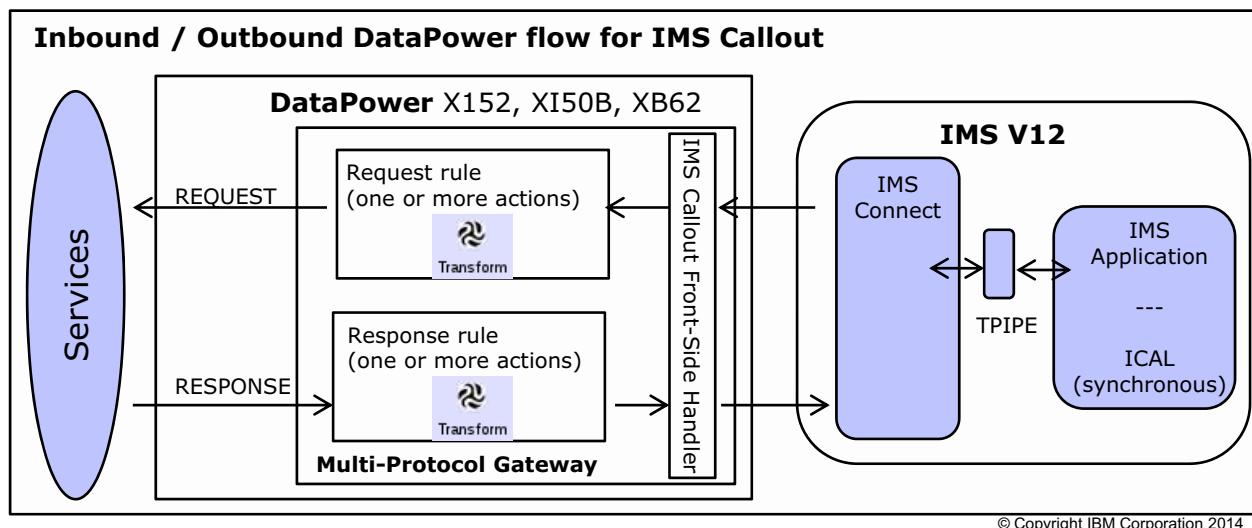


Figure 4-24. Supported IMS features

WE601 / ZE6011.1

Notes:

IMS Synchronous Callout support is a feature for allowing IMS to consume an external service through DataPower. By defining an IMS Callout Front Side Handler to DataPower MPG, an IMS application can initiate synchronous calls to an external service through DataPower following the IMS Call (ICAL) protocol. The ICAL protocol enables an application program running in an IMS technology-dependent region to synchronously send outbound messages to request services or data, and receive responses.

For synchronous callout requests, an IMS application program issues a DL/I ICAL call and waits in the dependent region to process the response. DataPower retrieves the callout request, processes it based on the rules and actions that are defined in the MPG policy, and sends it out to the back-end service. In a similar manner, the response is flown back and processed through the MPG. The figure here illustrates the callout inbound and outbound flow through DataPower.



IMS Connect support

- IMS Connect support (also known as IMS Connect Helper or IMS Provider) enables distributed services to drive an IMS transaction through DataPower.
- DataPower Multi-Protocol Gateway (MPGW) services can be configured with an IMS Connect back-side handler to receive a request from a client, process it, and send it to IMS Connect. A response will be sent back to the client after IMS processes the message.

DataPower Support Matrix

Support Type	DataPower Models Supporting V6.x Firmware
IMS Synchronous Callout (IMS V12 or beyond)	XI52, XI50B, XB62, XI52 VE
IMS Connect	XI52, XI50B, XI50Z, XB62, XI52 VE
IMS DB (IMS V12 or beyond)	XG45, XI52, XI50B, XB62, XI52 VE

© Copyright IBM Corporation 2014

Figure 4-25. IMS Connect support

WE601 / ZE6011.1

Notes:

Typical uses include:

An IMS Connect proxy to IMS Connect clients: Existing IMS Connect clients can use this feature to make in-flight modifications to headers and payloads without changing the client or IMS.

Web service facade to IMS Connect transactions: Organizations can use the web service features in DataPower to quickly enable web service support for IMS Connect.



Comparing services

- Select a Web Service Proxy when working with WSDLs
 - Web service virtualization, service policy definition by operation, and service level management by operation are easier to define by using this service type
- Select a multi-protocol gateway when working with other services
 - Has more capabilities than XML firewall to handle new requirements for the service
- XML firewall can be used for testing or loopback needs
 - Web service proxy and multi-protocol gateway services do not support loopback directly

© Copyright IBM Corporation 2014

Figure 4-26. Comparing services

WE601 / ZE6011.1

Notes:



Unit summary

Having completed this unit, you should be able to:

- Configure a multi-protocol gateway to provide a service over a set of different protocols
- Configure a connection to a static back-end service
- Configure a processing rule to select a back-end service at run time

© Copyright IBM Corporation 2014

Figure 4-27. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: With a dynamic back-end, the multi-protocol gateway relies on a custom style sheet action within a processing rule to configure the back-end destination. It is up to the developer to create the custom style sheet.
2. True or False: A Multi-Protocol Gateway (MPG) services can be configured with an IMS Connect back-side handler to receive a request from a client, process it, and send it to IMS Connect.
3. Which scenarios are better suited for a multi-protocol gateway as opposed to a web service proxy?

Description	Definition
<ol style="list-style-type: none"> 1. Multi-protocol gateway 2. Web service proxy 	<ol style="list-style-type: none"> A. WSDL B. WSRR concepts C. Multiple front side handlers D. Easy service level rule configuration E. WebSphere MQ integration

© Copyright IBM Corporation 2014

Figure 4-28. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. True.
2. True.
3. 1 – C and E
2 – A, B, and D

© Copyright IBM Corporation 2014

Figure 4-29. Checkpoint answers

WE601 / ZE6011.1

Notes:

Unit 5. Problem determination tools

What this unit is about

This unit describes the troubleshooting tools available for debugging problems on the DataPower appliance. Several tools are available for use that depend on the nature of the problem, ranging from low-level networking tools to probes that aid in debugging service policies. DataPower objects generate information that the logging utilities capture.

What you should be able to do

After completing this unit, you should be able to:

- Capture information by using system logs for messages that pass through the WebSphere DataPower SOA Appliance
- Configure a multistep probe to examine detailed information about actions within rules
- List the problem determination tools available on the WebSphere DataPower SOA Appliance

How you will check your progress

- Checkpoint
- Problem determination steps in Exercise 4: Create an advanced Multi-Protocol Gateway

Unit objectives

After completing this unit, you should be able to:

- Capture information by using system logs for messages that pass through the WebSphere DataPower SOA Appliance
- Configure a multistep probe to examine detailed information about actions within rules
- List the problem determination tools that are available on the WebSphere DataPower SOA Appliance

© Copyright IBM Corporation 2014

Figure 5-1. Unit objectives

WE601 / ZE6011.1

Notes:



Common problem determination tools

- Default system log
 - Displays system-wide log messages
 - Log messages can be filtered according to object and priority
- Audit log
 - Displays changes to the configuration of the appliance and files that are stored on the appliance
 - **Status > View Logs > Audit Log**
- Multi-step probe
 - Displays actions, messages, variable values as processing rule executes
 - Information is captured after processing rule executes
- Object status
 - Displays current operational status of all objects in the domain
 - **Status > Main > Object Status**
- Ping remote
 - Pings a remote host address to establish connectivity
- TCP connection test
 - Creates a TCP connection to remote destination to test connectivity
- Send test message
 - Builds and sends a SOAP request for testing
 - **Administration > Debug > Send a Test Message**

© Copyright IBM Corporation 2014

Figure 5-2. Common problem determination tools

WE601 / ZE6011.1

Notes:

Appliance status information

- File system information
 - Displays available encrypted, unencrypted, and temporary space for file storage
 - **Status > System > Filesystem Information**

- CPU usage
 - Displays percentage of CPU usage
 - **Status > System > CPU Usage**

- System usage
 - Displays load and work queue status
 - **Status > System > System Usage**

Free Encrypted Space	8	Mbytes
Total Encrypted Space	233	Mbytes
Free Temporary Space	223	Mbytes
Total Temporary Space	242	Mbytes
Free Internal Space	241	Mbytes
Total Internal Space	242	Mbytes

10 sec	4	%
1 min	28	%
10 min	28	%
1 hour	28	%
1 day	28	%

Task ID	Task Name	Load (%)	Work List	CPU (%)	Memory (%)	File Count
1	main	1	0	2	1	258

© Copyright IBM Corporation 2014

Figure 5-3. Appliance status information

WE601 / ZE6011.1

Notes:

It is a recommended practice to check the appliance file system memory for available space. The logging system can fill up the available file storage space, which can prevent the system from writing log entries. This situation prevents the system from processing messages.

Temporary Space is used for processing, logging, and debugging.

Internal Space is used for import, export, firmware upgrades, and debug data.

System Usage indicates the current load on the server and the length of the work queue. If the server suddenly slows down or becomes unresponsive, the cause might be system usage. If the system has a throttle in place, the high memory usage (load) might be causing the throttle to refuse connections.

Troubleshooting

The Troubleshooting page contains the following tools:

- Ping Remote
 - Pings a remote host address
- TCP Connection Test
 - Creates a TCP connection to remote endpoint
- Packet Capture (default domain only)
 - Captures network packets to and from the appliance
- View System Log and generate log messages
 - Specifies log level of messages to record
 - Generates log messages for testing log targets
- Error Report
 - Includes the running configuration and relevant system log entries for errors
 - Emails error report to an email address
- XML File Capture (default domain only)
 - Captures inbound XML files that are submitted to the appliance
- Probe
 - Enables or disables probes on services



© Copyright IBM Corporation 2014

Figure 5-4. Troubleshooting

WE601 / ZE6011.1

Notes:

The best tool to use first when a problem occurs often depends on how the appliance is being used at the time.

During the development phase, the default system log is often the best place to start, followed by use of the multi-step probe.

During the testing phase, generating an error report (which contains the running configuration of the appliance and the relevant log entries) is an excellent first step, followed by use of the multi-step probe.

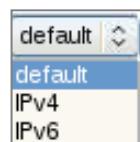
During the production phase, first check the system usage for load and work lists; then object status for objects that have changed to the down state, and finally the default system log.

Include a generated error report to DataPower support.



Troubleshooting: Networking

- Use the **Ping Remote** tool to test connectivity to a remote host
 - Enter IP address or host name and click **Ping Remote**
 - Optionally, enter the IP version to use
 - The default is IPv4
- Use the **TCP Connection Test** to test connectivity to a remote destination
 - Enter IP address or host name
 - Enter the port number
 - Click **TCP Connection Test**



The screenshot shows the DataPower interface under the Networking section. It contains two main panels:

- Ping Remote** panel: Contains fields for "Remote Host" (with an asterisk) and "Use IP version" (set to "default"). A "Ping Remote" button is present.
- TCP Connection Test** panel: Contains fields for "Remote Host" (with an asterisk), "Remote Port" (with an asterisk), and "Use IP version" (set to "default"). A "TCP Connection Test" button is present.

© Copyright IBM Corporation 2014

Figure 5-5. Troubleshooting: Networking

WE601 / ZE6011.1

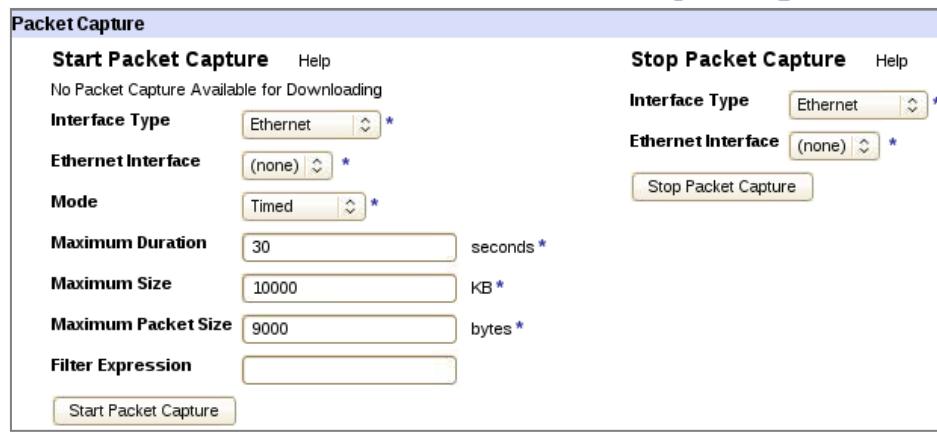
Notes:

Ping Remote allows DataPower to ping a host system. Using ping confirms that there is network connectivity to the host / IP address the DataPower appliance is attempting to reach.

The TCP Connection Test confirms that DataPower can reach the IP address and the port. This step is useful to confirm if a service is running remotely. For example, you can use TCP Connection Test with the IP address of WebSphere Application Server and port 9080 to confirm that the server is up and running on the remote host.

Troubleshooting: Packet capture

- Available in default domain only
- Captures the IP packets sent to and from the appliance
 - Captures full network-level exchange between the appliance and other endpoints
 - Captured in **pcap** format
 - Tools such as **WireShark** can be used to view the traffic in detail
- Useful when troubleshooting network connectivity, TCP sequencing, or other network-level problems
- The packet capture file is available from the **temporary:** directory



© Copyright IBM Corporation 2014

Figure 5-6. Troubleshooting: Packet capture

WE601 / ZE6011.1

Notes:

In the Troubleshooting web page, scroll down to the packet capture section. Click the **Packet Capture** icon to begin the capture. A dialog box confirms the action. When the capture is complete, a **Download Packet Capture** icon appears on the Troubleshooting page.

You can control the network interface to monitor the duration of monitoring and the number of KB that can be captured.

DataPower support expects the pcap format when a PMR is opened.

Before installing a packet capture tool, such as Wireshark (formerly called Ethereal), make sure that you have the necessary permission from your network staff.

Restarting the device automatically turns off packet capture.



Troubleshooting: Logging

- Use **Set Log Level** to set the log level for the current domain
- Use **Generate Log Event** to verify that log targets are active and able to capture events

Logging

Set Log Level Help

View System Logs

Log Level *

Enable Internal Logging on off

Enable RBM Debug logging on off

Global IP Address Log Filter

Set Log Level

Generate Log Event Help

Log Category *

Log Level *

Log Message

Event Code

Generate Log Event

© Copyright IBM Corporation 2014

Figure 5-7. Troubleshooting: Logging

WE601 / ZE6011.1

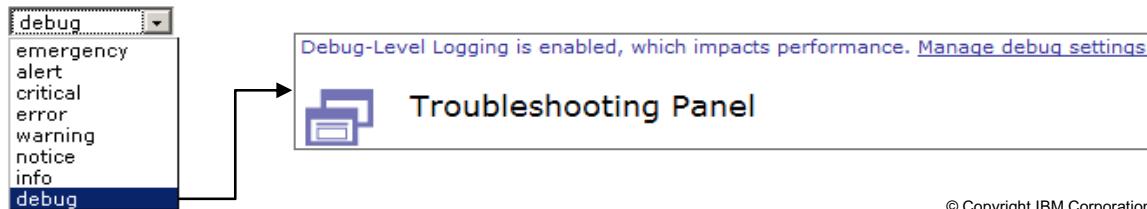
Notes:

Setting the log level to DEBUG is helpful during development but it impacts processing. Therefore, DEBUG mode should not be used in production.

Generate Log Event is used to test out a log event and log target that is configured.

Troubleshooting: System log

- Displays system-wide log messages that the appliance generates
 - Click the **View Logs** icon in the Control Panel
 - In the Troubleshooting pane, scroll down to the Logging section
 - Click **View System Logs**
- By default, log messages are captured only with severity of notice or greater
 - Log levels are hierarchical
 - Highest severity (emergency) is at the top of the list
 - Each level captures messages at or above the current level
 - To enhance troubleshooting, set the log level to debug
 - Lowest severity (debug) captures the most information



© Copyright IBM Corporation 2014

Figure 5-8. Troubleshooting: System log

WE601 / ZE6011.1

Notes:

The highest priority is **emergency** and the lowest priority is **debug**.

The target captures messages only at or above the configured level. For example, the error level captures messages at the error, critical, alert, and emergency levels. To capture all messages, set the log level to **debug**.

Setting the level to either **info** or **debug** causes a blue **Troubleshooting Enabled** notice to appear on all WebGUI pages.

Log level of the default system log.

- **emergency**: An emergency-level message. The system is unusable.
- **alert**: An alert-level message. Immediate action must be taken.
- **critical**: A critical message. Immediate action must be taken.
- **error**: An error message. Processing might continue, but action should be taken.
- **warning**: A warning message. Processing should continue, but action should be taken.
- **notice**: A notice message. Processing continues, but action might need to be taken.

- **information:** An information message. No action is required.
- **debug:** A debug message for processing information to help during troubleshooting.

Filtering system log

- In the default domain, the system log shows all log entries
 - In non-default domains, log entries are shown only for the objects in that domain
- Filter the system log by:
 - Log target
 - Domain (shown only in the default domain)
 - DataPower objects (xmlfirewall, ws-proxy, and more)
 - Log level type (debug, info, and more)

The screenshot shows the 'System Log' interface. At the top, there are two dropdown menus: 'Category' and 'Log level', both currently set to '(none)'. Below these are buttons for 'Refresh Log' and 'Target: default-log'. A status message indicates the 'current time: 13:33:32 on 2012-08-28'. The main area displays log entries from 'Tue Aug 28 2012' in a table format. The columns are: time, category, level, tid, direction, client, msgid, and message. The log entries are:

time	category	level	tid	direction	client	msgid	message
13:32:27	memory-report	debug	25568737		172.16.80.11	0x80e00690	mpgw (EastAddressSearch): Response Finished: memory used 616424
13:32:27	mpgw	info	25568737	error	172.16.80.11	0x80e000b6	mpgw (EastAddressSearch): No match from processing policy 'EastAddressSearch' for code '0x00230001'
13:32:27	mpgw	notice	25568737		172.16.80.11	0x80c0007b	stylepolicy (EastAddressSearch): No error rule is matched.
13:32:27	mpgw	error	25568737	error	172.16.80.11	0x00230001	mpgw (EastAddressSearch): Dynamic Execution Error

© Copyright IBM Corporation 2014

Figure 5-9. Filtering system log

WE601 / ZE6011.1

Notes:

The system log is defined as a log target. A log target receives log entries from objects to post. Each domain always has a log target that is called **default-log** to represent the default system log. Additional log targets can be defined and customized with the log entries from objects to post.

The most recent log entries are shown at the top of the system log.

The logs can be sorted by the categories that are listed at the top.



Troubleshooting: Generate Log Event

- Use the **Generate Log Event** tool to test whether:
 - Log messages are generated in appropriate log target on the appliance (default system log captures all log messages)
 - Log messages are sent to remote host when off-box logging is used
- Configure log messages with:
 - Log Type: object class or category
 - Log Level: debug, info, and more
 - Log Message: string inside log message
 - Event Code: for generating an event code-based message



© Copyright IBM Corporation 2014

Figure 5-10. Troubleshooting: Generate Log Event

WE601 / ZE6011.1

Notes:

The Generate Log Event tool is used to test the configuration of a newly created log event and log target.



Troubleshooting: Reporting

- Generate Error Report
 - Error report is required when engaging with IBM DataPower support
 - Error report file is created in the **temporary:** directory
- Error Report contains:
 - Current configuration
 - Current contents of the system log
 - Contents of CLI log
- Send Error Report:
 - DataPower uses an external mail server (SMTP) to email the error report to a specific email recipient

The screenshot displays a user interface titled "Reporting". On the left, under "Generate Error Report", there is a message "No Error Report Available for Viewing" and a button labeled "Generate Error Report". On the right, under "Send Error Report", there are four input fields: "SMTP Server", "Location", "E-mail Address", and "Email Sender Address", each with an asterisk (*) indicating it is a required field. Below these fields is a "Send Error Report" button.

© Copyright IBM Corporation 2014

Figure 5-11. Troubleshooting: Reporting

WE601 / ZE6011.1

Notes:

Click **Generate Error Report**. A dialog box asks for confirmation and indicates the location of the resulting file.

If an error report is available, an icon appears that allows immediate access to the file.



Troubleshooting: Advanced

- Use XML File Capture to allow the configuration of system-wide file-capture mode
 - The file capture facilitates the visibility of erroneous XML and XSLT content
- Use **View Running Config** to view the configuration of all the objects that are currently in memory



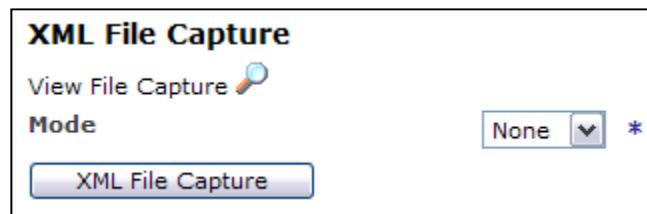
Figure 5-12. Troubleshooting: Advanced

WE601 / ZE6011.1

Notes:

Troubleshooting: XML File Capture

- Captures XML messages from any service
 - XML messages that services cannot parse can also be captured
- File capture can fill the available storage space
 - Files are cycled FIFO
 - Maximum of 5000 files or 200 MB can be captured
 - Stored in compressed format
 - Supported by using RAM-Disk
- XML File Capture must be enabled only in test environments
 - Significant performance penalties are incurred when mode is set to always or errors
- Default domain only



© Copyright IBM Corporation 2014

Figure 5-13. Troubleshooting: XML File Capture

WE601 / ZE6011.1

Notes:

The XML File Capture is sets the configuration of system-wide file-capture mode. The file capture facilitates the visibility of erroneous XML and XSLT content.

Troubleshooting: Send a test message

WebSphere. DataPower XI52

Send a Test Message

Request

URL: **1**

Request Headers:

Header Name	Value
<input type="text"/>	<input type="text"/> 2
<input type="text"/>	<input type="text"/>

Request Body: **3**

Response

Response Code:
Response Headers:
Response Body: **4**

- **Control Panel > Administration > Debug > Send a Test Message**
- Builds a SOAP request with a customized header, content, and body that is used for testing
 1. A URL can be generated by using the different helpers
 2. Request headers can be added
 3. A request body can be typed or pasted here
 4. The response is displayed here

© Copyright IBM Corporation 2014

Figure 5-14. Troubleshooting: Send a test message

WE601 / ZE6011.1

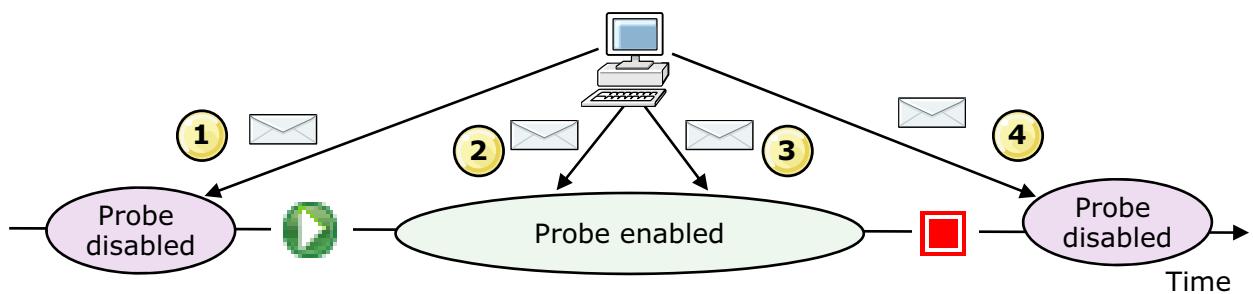
Notes:

Using the **Send a test message** tool versus **cURL**:

The test message tool is a quick and useful tool for creating SOAP requests, and it can be used in place of open source tools like cURL. However, when using the test message tool, you cannot upload a file to the DataPower box to send; you need to copy and paste text. You also cannot persist the test message after it is created. The advantage of using tools like cURL is that it can send files directly from the file system.

Troubleshooting: Multi-step probe

- Displays the lifecycle of the message as it executes in a processing rule
 - Information is captured after processing rule executes
- Aids in debugging processing rules
 - Step-by-step debugging to view message content after execution of each action in the processing rule
 - Enable only in test environment because it impacts appliance performance



© Copyright IBM Corporation 2014

Figure 5-15. Troubleshooting: Multi-step probe

WE601 / ZE6011.1

Notes:

In the diagram on the slide, four messages are sent to the probe. Only message 2 and message 3 are captured. The probe functions like a recorder. When the probe is enabled, it starts recording messages that enter the appliance. When the probe is disabled, recording is stopped and the probe stops capturing messages.

The multi-step probe can be used to view:

- Action execution trace
- Message content
- Header values
- Attachments
- Variable values (local, global, service)



Troubleshooting: Enabling the multi-step probe

Two ways to enable a probe for a service:

- Click the **Debug Probe** tab in the Troubleshooting pane

- Click **Add Probe** to add a multi-step probe for that service

- On the service configuration page, click the **Show Probe** button to open the multi-step probe window
 - Enable the probe inside the multi-step probe window

© Copyright IBM Corporation 2014

Figure 5-16. Troubleshooting: Enabling the multi-step probe

WE601 / ZE6011.1

Notes:

Probes are enabled for the following services:

- XSL proxy and XSL coprocessor
- XML firewall
- Multi-protocol gateway
- Web service proxy
- WebSphere MQ proxy and WebSphere MQ host



Multi-step probe window

- Enable the probe in the multi-step probe window
 - Start sending messages to the service
 - Click **Refresh** in the multi-step probe window
 - Examine the captured request and response rule processing results

View probe data

view	trans#	type	inbound-url	outbound-url	rule
[+]	5506257	request	http://172.16.78.44:6957/EastAddressSearch	http://WSserver:9080/EastAddressSearch	EastAddressRequest
[+]	5510545	request	http://172.16.78.44:6957/EastAddressSearch	http://WSserver:9080/EastAddressSearch	EastAddressRequest
[+]	5510545	response	http://172.16.78.44:6957/EastAddressSearch	http://WSserver:9080/EastAddressSearch	EastAddressResponse

© Copyright IBM Corporation 2014

Figure 5-17. Multi-step probe window

WE601 / ZE6011.1

Notes:

The multi-step probe window opens with the probe disabled when you enable the probe from the service configuration page.

Rules that generate an error while executing are displayed in red text inside the multi-step probe window.

The **Flush** button clears the requests inside the multi-step probe window.

Restarting the appliance disables all probes.

Input Context 'INPUT' of Step 1

Step 1: Transform Action: Input=INPUT, Transform=local:///Address-EastRenameNamespace.xsl, Output=tempvar1, OutputType=default, Transactional=off, SOAPValidation=body, SQLSourceType=static

Content **Headers** **Attachments** **Local Variables** **Global Variables** **Service Variables**

1 Content of context 'INPUT':

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

Select to show unformatted content

2 **3** View the message content as it traverses each action

- Each action has an input and output message that can be viewed by clicking the magnifying glass
 - Message content
 - Protocol headers and message attachments
 - Local, global, and service variables
 - Actions that the processing rule executes

© Copyright IBM Corporation 2014

Figure 5-18. Multi-step probe content

WE601 / ZE6011.1

Notes:

The magnifying glass to the left of the action represents the input message. The magnifying glass to the right of the action is the result of executing that action.

Click the **Next** and **Previous** buttons to view the message step-by-step as it is executed by the processing rule.

The local, context, global, and service variables are DataPower variables that are generated by the appliance.

Problem determination with cURL

- Use cURL with **-v** option to output more information to trace client-side errors
 - This option is independent of the DataPower appliance troubleshooting tool
- Use the **--trace** or **--trace-ascii** options with a file name to write the logging data
 - Provides more details on the client/server interaction
- Sample tracing with cURL:

```
curl --trace-ascii trace1.txt
      -D headers1.txt
      -H "Content-Type:text/xml"
      -d @AddressReq.xml
      http://dpedu1:2064
```

© Copyright IBM Corporation 2014

Figure 5-19. Problem determination with cURL

WE601 / ZE6011.1

Notes:

The **-v** verbose flag produces much information output. It allows the user to see all of the client/server interaction.



Communicating with DataPower support

- DataPower support information links are at the bottom of the Control Panel page
- Generally, use the Troubleshooting pane to supply DataPower support with the following files:
 - Generate an error report
 - Save the running configuration to a file

© Copyright IBM Corporation 2014

Figure 5-20. Communicating with DataPower support

WE601 / ZE6011.1

Notes:

Logging basics

- Logging system is based on the publish/subscribe model
 - Objects *publish* events
 - Subscribers *subscribe* to events of interest
- The DataPower logging system uses **log targets** as *subscribers* and **log events** (generated by objects) as *publishers*
- Logs can be written on-device or off-device
 - On-device logs can be moved off-device (SFTP, SCP, HTTP, SHTTP)
 - Off-device support for syslog, syslog-NG, SNMP
- Log targets do not capture the actual message
 - Add a **Log** action in a processing rule to capture the entire message

© Copyright IBM Corporation 2014

Figure 5-21. Logging basics

WE601 / ZE6011.1

Notes:

Log files can be encrypted or signed for more security.

Objects that generate log messages have different priorities. These messages range from verbose debugging to infrequent critical or emergency level message.

WebSphere Education

Log targets

List of log levels for the system log:

- **Emergency**: system is unusable
- **Alert**: take immediate action
- **Critical**: critical condition
- **Error**: an error occurred
 - The error code is included
- **Warning**: a warning condition occurred
 - Nothing might be wrong, but conditions indicate that a problem might occur soon if nothing changes
- **Notice**: a normal but significant condition applies
- **Information**: an informational message only
- **Debug**: debug-level messages
 - This level generates many messages

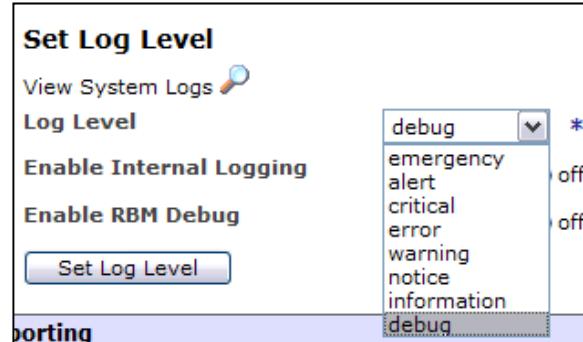


Figure 5-22. Log targets

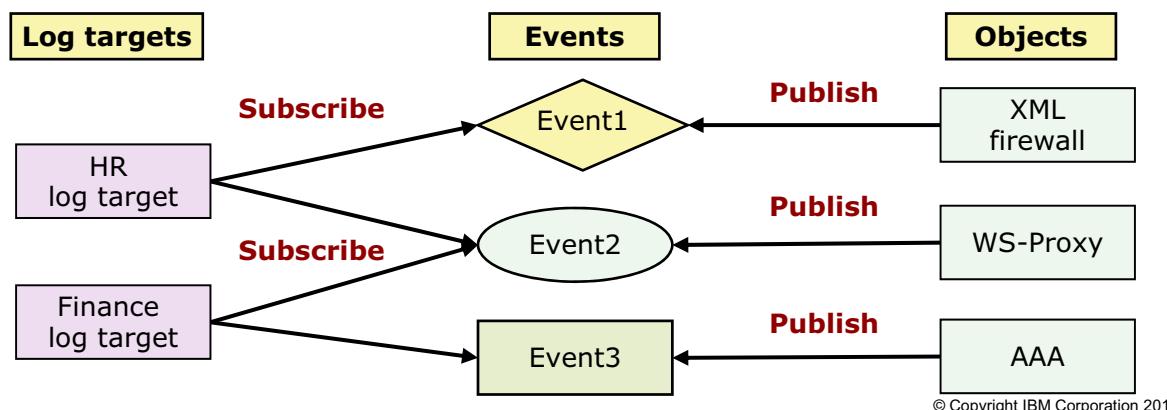
© Copyright IBM Corporation 2014

WE601 / ZE6011.1

Notes:

Available log levels

- Log targets subscribe to log messages posted by the various running objects
 - Create a log target by clicking **Administration > Miscellaneous > Manage Log Targets**
- Log target subscription can be restricted to:
 - Object filters: events specific to an instance of an object
 - Event category: any object that generates events
 - Event priority: objects of a specific class and message priority that generates events



© Copyright IBM Corporation 2014

Figure 5-23. Available log levels

WE601 / ZE6011.1

Notes:

The diagram in the slide shows two log targets: HR and Finance log targets. These log targets subscribe to certain types of events that are generated or published by objects on the DataPower appliance.

Use the Generate Log Event tool in the Troubleshooting pane to test whether log messages are captured by log targets.

Log target configuration

Configure Log Target

Main Event Filters Object Filters IP Address Filters Event Triggers

Log Target

Name: myLogTarget

General Configuration

Administrative State: enabled disabled

Comments:

Target Type: Cache

Log Format: XML

Timestamp Format: syslog

Feedback Detection: on off

Identical Event Detection: on off

Apply Cancel

Configuring log target tabs

- Main
 - Target type
- Event filters
 - Can restrict messages by event code
- Object filters
 - Can restrict messages that appear in a target by object
- Event subscriptions
 - Subscribed to event categories or object class
 - Predefined event categories are: auth, mgmt, xsslt, and more
 - Categories have a priority level
- Log targets are required to subscribe to at least one event category

© Copyright IBM Corporation 2014

Figure 5-24. Log target configuration

WE601 / ZE6011.1

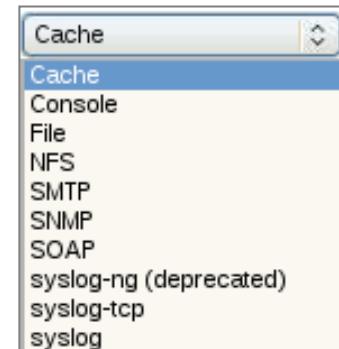
Notes:

Log targets capture messages that are posted by the various objects and services that are running on the appliance. Target types enable additional capabilities that include rotating files, encrypting and signing files or messages, and sending files to remote servers.

Ten-log target types

A **Target Type** field of a log target supports the following values

- **Cache**: writes log entries to system memory
- **Console**: writes log entries to a Telnet, SSH, or CLI screen
- **File**: writes log entries to a file on the device flash
- **NFS**: writes log entries to a file on a remote NFS server
- **SMTP**: forwards log entries as an email to configured addresses
- **SNMP**: forwards log entries as SNMP traps
- **SOAP**: forwards log entries as SOAP messages
- **syslog-*ng* (deprecated)**: use `syslog-tcp`
- **syslog-*tcp***: forwards log entries by using TCP to a remote syslog daemon
 - The local address, remote address, remote port, syslog facility can be set
 - An SSL connection to the syslog host can be created
 - The processing rate can be limited
- **syslog**: forwards log entries to a remote syslog daemon



© Copyright IBM Corporation 2014

Figure 5-25. Ten-log target types

WE601 / ZE6011.1

Notes:

The log entries that are stored on a **local** or **NFS** file can be rotated, emailed, or uploaded to other locations. The entire file can also be encrypted and signed.

SNMP is a network protocol that allows for the exchange of management information between network devices. This protocol is included in the TCP/IP protocol suite.

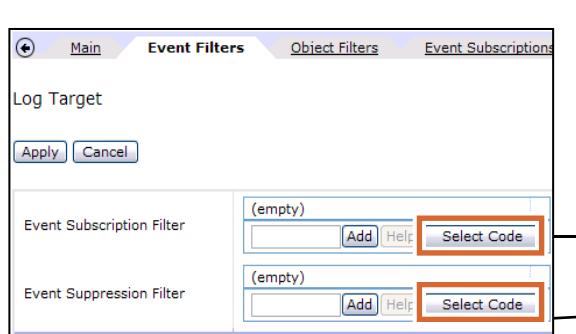
Syslog is the format and protocol that is used to send messages over TCP or UDP to a Syslog daemon (`syslogd`). It allows for log messages to be collected from many applications.

Syslog-*NG* (New Generation) is being deprecated. Use `syslog-tcp` in place of `syslog-ng`.

Event filters

- On the Configure Log Target page, click the **Event Filters** tab
- Event filters create filters for a log target that is based on **event codes**
 - Use the **Event Subscription Filter** to subscribe to specific event codes
 - Use the **Event Suppression Filter** to exclude certain event codes from being written to the log target
 - Click the **Select Codes** button to add event codes to **Event Code** value list

Event Code	Category	Severity	Message
0x01530001	clock	error	Time zone config mismatch.
0x01b10001	crypto	alert	Crypto accelerator not supported by this
0x01b20002	crypto	critical	HSM is uninitialized
0x01b20003	crypto	critical	HSM PED login timed out
0x01b20004	crypto	critical	HSM PED login failed
0x01b10005	crypto	alert	Microcode file not found
0x01b10006	crypto	alert	Microcode load failed
0x01b10007	crypto	alert	HSM credentials not found
0x01b20008	crypto	critical	HSM password login failed



© Copyright IBM Corporation 2014

Figure 5-26. Event filters

WE601 / ZE6011.1

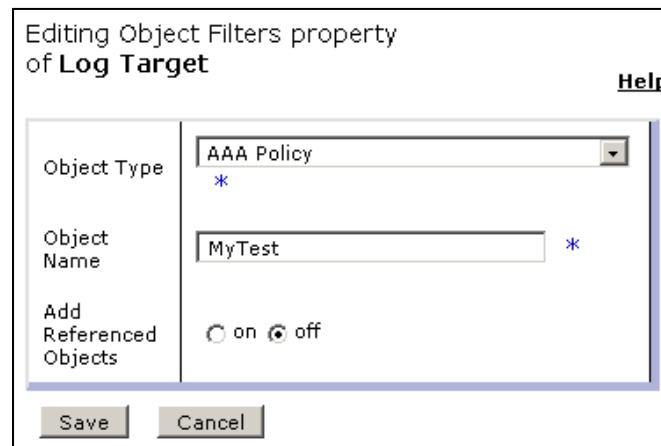
Notes:

You can subscribe the current log target to particular event code categories. Example event codes include out of memory, failed to install on local port, and more.

These event codes are DataPower specific event conditions.

Object filters

- On the Configure Log Target page, click the **Object Filters** tab
- Object filters allow only those messages that are generated by selected objects to be written to a log target
- It is possible to create a log target that collects log messages for a particular class of objects
 - Example: AAA policy object called MyTest



© Copyright IBM Corporation 2014

Figure 5-27. Object filters

WE601 / ZE6011.1

Notes:

The object filter is more specific than the object class name. This filter collects log messages of an instance of a class.

For example, a log target would collect messages from an XML firewall that is named **MyFirewall** and not all XML firewall instances.



Event subscriptions

- On the Configure Log Target page, click the **Event Subscriptions** tab
- Log targets subscribe to particular event categories
- Example event categories:
 - xmlfirewall**: for XML firewall objects
 - auth**: authorization
 - mgmt**: for configuration management events
- A priority level can be specified for each event category that is chosen
 - Additional level of filtering

Adding new Event Subscriptions property
of **Log Target**

Event Category	<input type="text" value="xmlfirewall"/> <input type="button" value="+"/> <input type="button" value="..."/> *
Minimum Event Priority	<input type="text" value="debug"/> <input type="button" value="..."/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

© Copyright IBM Corporation 2014

Figure 5-28. Event subscriptions

WE601 / ZE6011.1

Notes:

Event categories is the same term that is used to describe an object class name.

At least one event category must be defined for a log target to capture messages.

The screenshot shows the 'Configure Log Action' dialog box from the WebSphere Education interface. The dialog is divided into sections: 'Input', 'Log', 'Asynchronous', and 'Output'. The 'Log' section contains fields for 'Destination' (set to 'http://'), 'Log Level' (set to 'notice'), and 'Log Type' (set to 'mpgw'). The 'Asynchronous' section contains a 'Method' field set to 'POST'. The 'Output' section has an 'Output' field set to '(auto)'. At the bottom of the dialog are 'Delete', 'Done', and 'Cancel' buttons.

© Copyright IBM Corporation 2014

Figure 5-29. Log action

WE601 / ZE6011.1

Notes:

If you want to capture the message payload (the data in the message), a Log Action must be used.



Unit summary

Having completed this unit, you should be able to:

- Capture information by using system logs for messages that pass through the WebSphere DataPower SOA Appliance
- Configure a multistep probe to examine detailed information about actions within rules
- List the problem determination tools that are available on the WebSphere DataPower SOA Appliance

© Copyright IBM Corporation 2014

Figure 5-30. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: To test a Log Event, you would use the Generate Log Event option in the troubleshooting pane to generate a log message, and verify that it is included or excluded in a log target.
2. A client cannot connect to the XML firewall service. Select the best steps to troubleshoot this problem.
 - A. Check the client URL and Object status (and possibly TCP connection test).
 - B. Ping the DNS to validate the proper XML firewall service. Check the back side connection.
3. Logs can be stored off-device by using (select five):
 - A. SMTP
 - B. SOAP
 - C. NFS
 - D. syslog-ng
 - E. daemon
 - F. syslog
 - G. POP

© Copyright IBM Corporation 2014

Figure 5-31. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

- 1. True.**
- 2. A.**
- 3. A, B, C, D, and F.**

© Copyright IBM Corporation 2014

Figure 5-32. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 3



Creating an advanced multi-protocol gateway

© Copyright IBM Corporation 2014

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

4.0 9.0

Figure 5-33. Exercise 3

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

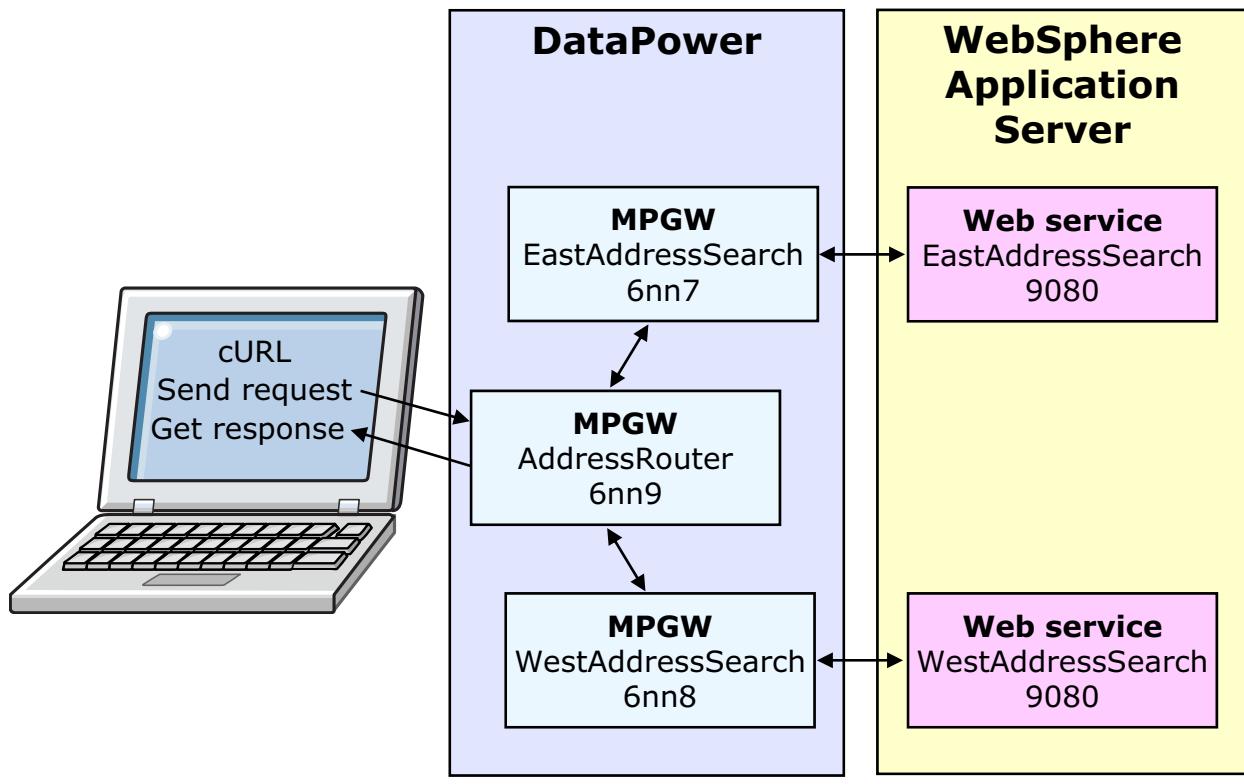
© Copyright IBM Corporation 2014

Figure 5-34. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview



© Copyright IBM Corporation 2014

Figure 5-35. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 6. Handling errors in a service policy

What this unit is about

It is expected that errors occur when messages are processed by the service policy. The developers of service policies need to plan for error handling within the rules of the policy. In this unit, you learn to use the On Error action and Error rule, and how the service policy selects error handling.

What you should be able to do

After completing this unit, you should be able to:

- Configure an On Error action in a service policy
- Configure an Error rule in a service policy
- Describe how On Error actions and Error rules are selected during error handling

How you will check your progress

- Checkpoint
- Exercise 5: Adding error handling to a service policy

Unit objectives

After completing this unit, you should be able to:

- Configure an On Error action in a service policy
- Configure an Error rule in a service policy
- Describe how On Error actions and Error rules are selected during error handling

© Copyright IBM Corporation 2014

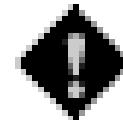
Figure 6-1. Unit objectives

WE601 / ZE6011.1

Notes:

Error handling constructs

- Default error handling procedure is to cancel the current document processing rule and log an error message
- Two methods for handling errors:
 - **On Error** action
 - Lets you either cancel or continue processing
 - If *continue*, then the next action in the rule is executed; otherwise, the rule is canceled
 - **Error rule**
 - Automatically executes if it is configured within the current document processing policy
 - Presence of an **On Error** action precludes the automatic selection of an **error rule** for execution



© Copyright IBM Corporation 2014

Figure 6-2. Error handling constructs

WE601 / ZE6011.1

Notes:

Error handling constructs are used to handle errors that occur during execution of a service policy.

WebSphere Education

Configure an On Error action

- The **On Error** action is used to control what happens when an error is encountered within the rule
 - Optional: execute a named rule to handle the error condition
- Configure the following within an **On Error** action:
 - Error mode:
 - Cancel**: stop executing the current rule
 - Alternative**: invoke an alternative processing rule
 - Continue**: continue with the next sequential action
 - The **Processing Rule** fields specify either:
 - An error rule to execute
 - A custom variable for the processing rule
 - Use the Var Builder to create a custom variable

The screenshot shows the 'On Error' configuration dialog. At the top is a title bar with a diamond icon and the text 'On Error'. Below are several configuration fields:

- Error Mode:** A dropdown menu showing 'Cancel' (selected), 'Cancel', 'Alternative', and 'Continue'.
- Processing Rule:** A dropdown menu showing '(none)'.
- Error Input:** A dropdown menu showing '(none)'.
- Error Output:** A dropdown menu showing '(none)'.
- Asynchronous:** A radio button group with 'on' (selected) and 'off'.

At the bottom are three buttons: 'Delete', 'Done', and 'Cancel'.

© Copyright IBM Corporation 2014

Figure 6-3. Configure an On Error action

WE601 / ZE6011.1

Notes:

To configure an **On Error** action, run the following steps:

- Drag the **Advanced** icon to the rule configuration path.
- Double-click the **Advanced** icon.
- On the Configure Action page, select **On Error** and click **Next**.
- Configure the **On Error** action and click **Done**.
The **Error Input** and **Error Output** context in an **On Error** action provide the context for the actions within the error rule (if selected).
- Use the context **OUTPUT** in the **Error Output** field to return the error message to the client.



Creating an error rule

Rule:

Rule Name: Test_rule_1 Rule Direction: Error

New Rule Delete Rule

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

Filter Sign Verify Validate Encrypt Decrypt Transform Route AAA Results Advanced

ORIGIN SERVER → DATAPOWER ←

Create Reusable Rule

Order	Rule Name	Direction	Actions
↑ ↓	Test_request	Client to Server	◀ ↘
↑ ↓	Test_response	Server to Client	◀ ↗
↑ ↓	Test_rule_1	Error	◀ ↗ ↘

Error rules are used to handle errors in the request or response rule

- Automatically executes when configured in a service policy
- Can be used to log or send a custom error message
 - Use the **Log** action to log entire message
 - Use the **Transform** action to build custom error messages

© Copyright IBM Corporation 2014

Figure 6-4. Creating an error rule

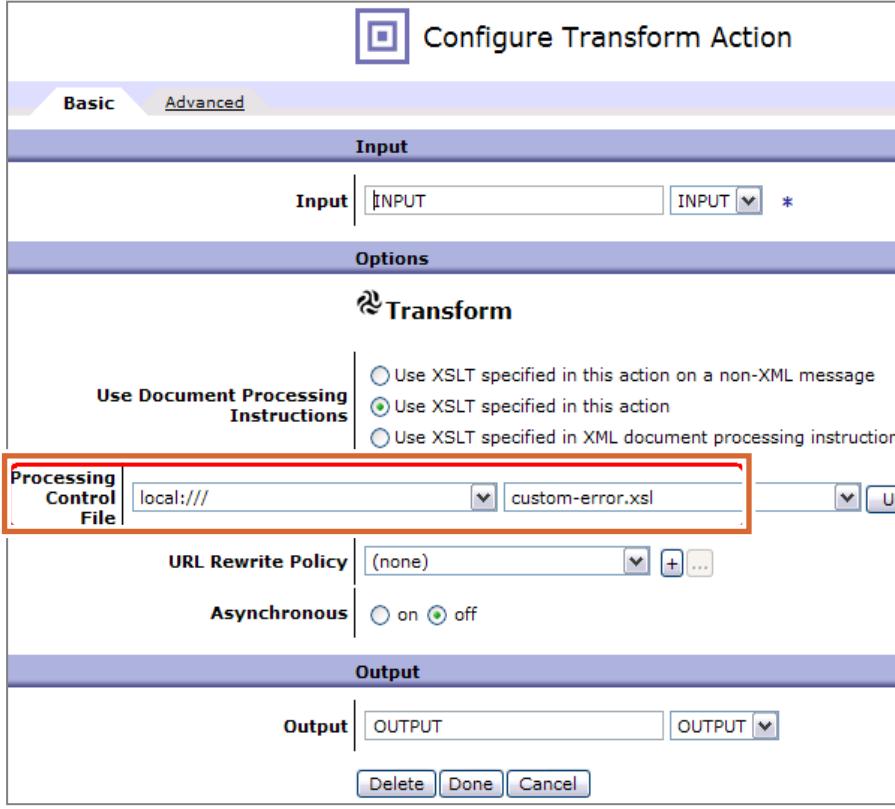
WE601 / ZE6011.1

Notes:

The rule directionality (request or response) does not apply to an error rule; it can run on either the request or the response rule.

Configure Transform action in error rule



Configure Transform Action

Basic Advanced

Input

Input INPUT

Options

Transform

Use Document Processing Instructions

- Use XSLT specified in this action on a non-XML message
- Use XSLT specified in this action
- Use XSLT specified in XML document processing instruction

Processing Control File local:///

URL Rewrite Policy (none)

Asynchronous on off

Output

Output OUTPUT

© Copyright IBM Corporation 2014

Figure 6-5. Configure Transform action in error rule

WE601 / ZE6011.1

Notes:

Style sheet programming that use error variables

- Output log messages with log priority by using `<xsl:message>`

```
<xsl:message
    dp:type='ws-proxy' dp:priority='error'>
        Error: <xsl:value-of select="$errtest"/>
</xsl:message>
```

- The following DataPower variables are useful when generating a custom error message:

`var://service/error-code` DataPower error code

Example: Dynamic execution error

`var://service/error-subcode` DataPower suberror code

Example: Schema validation error

`var://service/error-message`

Error message sent to client

`var://service/transaction-id`

ID used to correlate transactions in the DataPower system logs

`var://service/client-service-address`

Address of the calling client

© Copyright IBM Corporation 2014

Figure 6-6. Style sheet programming that use error variables

WE601 / ZE6011.1

Notes:

The example log message that is generated in the slide has a log priority of **error** with the class name **ws-proxy**. The log message that is generated contains the contents of the variable **errtest**.

The variable that is listed in the slide can also be viewed when you are running the multi-step probe by selecting the **Service Variables** tab.

The `dp:type` attribute in the `<xsl:message>` tag can be caught by a log target, enabling user-defined debug messages to be captured in logs.

Example custom error style sheet

```

<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dp="http://www.datapower.com/extensions"
    extension-element-prefixes="dp" exclude-result-prefixes="dp">

    <xsl:template match="/">
        <!-- Get the error codes set by DP. -->
        <xsl:variable name="dpErrorCode" select=
            "dp:variable('var://service/error-code')"/>
        <xsl:variable name="dpErrorSubcode" select=
            "dp:variable('var://service/error-subcode')"/>
        <xsl:variable name="dpErrorMessage" select=
            "dp:variable('var://service/error-message')"/>
        <xsl:variable name="dpTransactionId" select=
            "dp:variable('var://service/transaction-id')"/>

        <!-- Build custom SOAP fault message -->
        <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
            <env:Body>
                <env:Fault> (details omitted) ... </env:Fault>
            </env:Body>
        </env:Envelope>

    </xsl:template>
</xsl:stylesheet>

```

© Copyright IBM Corporation 2014

Figure 6-7. Example custom error style sheet

WE601 / ZE6011.1

Notes:

This example style sheet includes some common DataPower extension functions that can be used when building a custom error message.

The service variables that are shown are also visible in the multi-step probe.

This style sheet is only a template of an actual error style sheet. A custom error style sheet can customize the amount of detail to include in an error message.

Error rule versus On Error action

- The presence of the **On Error** action precludes an error rule within the same service policy from being selected to handle an error
 - The **On Error** action can optionally execute an error rule
- The error rule executes in the absence of an **On Error** action when an error occurs in the current processing rule
 - The current processing rule is canceled and the execution of the error rule starts
- Multiple **On Error** actions can be defined in a processing rule
 - Each **On Error** action handles errors for subsequent actions within the same processing rule
 - When the next **On Error** action within a rule is executed, it handles errors for the next set of actions
- When no Error Processing is defined in the Service Policy, the default Error Policy is used (if defined)

© Copyright IBM Corporation 2014

Figure 6-8. Error rule versus On Error action

WE601 / ZE6011.1

Notes:



Error Policy

- The Error Policy is a fallback error handler that is used when there is an unhandled error in a Multi-Protocol Gateway transaction
- The Error Policy is available as a configurable property in Multi-Protocol Gateway (MPGW) service
 - Matching Rules must be defined
 - Error Actions must be defined to handle errors in an HTTP or HTTPS request flow
- The Error Policy allows for:
 - Customization of the default error response for non-SOAP/non-XML web applications while the MPGW used to return *SOAP fault*
 - Customization of a default error response (instead of the traditional default SOAP fault) for replying to your non-SOAP/non-XML client applications in a simpler manner
 - Fall back for an error that is not successfully handled by any precedent error handlers (such as multistep error rule)

© Copyright IBM Corporation 2014

Figure 6-9. Error Policy

WE601 / ZE6011.1

Notes:

In DataPower firmware release 6.0.0, a new Error Policy is introduced to the Multi-Protocol Gateway (MPGW) service. Its principal function is being a fallback error handler. If there is an error in the MPGW transaction that is not successfully handled by any precedent error handler, the new Error Policy is executed to generate the final error response.

The needs toward the new feature are described as follows.

Before Release 6.0.0, the MPGW service had a tendency to return a SOAP fault to the client as the default error response. This setting is not an optimal default setting for non-SOAP or non-XML clients; for example, for the MPGW as a web proxy, the client might expect an HTML page to highlight the error cause and suggestions.

Regarding existing error handlers, today, the primary error handler to customize the error response is the multistep error rule that is either designated by an on-error action or fired by a matching procedure. Also, you can manipulate the generation of error response by using service variables (for example, `var://service/error-message, subcode`).

However, even the multistep error rule might not complete, and, when it fails, the client still receives the default SOAP fault message. By using the new Error Policy, when there is no error handler

(such as multistep error rule) or the precedent error handler failed, you can have a *fallback* to generate the error response (such as an HTML, a plaintext, or an XML, or whatever) to the client based on the request's content type.

Typical Error Policy use cases

- A multi-protocol gateway user who:
 - Runs web gateway business and wants to have a default error response that is based on the content type (not always SOAP fault)
 - Wants to have a more convenient configuration than the multistep error rule to produce a non-SOAP fault error response when the error handling logics needs to involve few error actions
 - Has implemented error handling logics on the multistep error rule but wants to have a fallback handler when the multistep error rule might fail somewhere

© Copyright IBM Corporation 2014

Figure 6-10. Typical Error Policy use cases

WE601 / ZE6011.1

Notes:

You can benefit from the new Error Policy in various situations:

- You are running the web business upon the MPGW service and want to have customizable default error response that is based on the runtime request's content type (rather than the SOAP fault message).
- You are developing a new MPGW service and do not need a complex error handling logics (namely, many actions that are involved in the multistep error rule) to generate the error response. For example, think of a case when you need to respond an HTTP URL redirection without a complex error rule configuration, then the Error Policy is a convenient and effective way for this purpose.
- You have implemented error handling logics upon the multistep error rule, and you are now able to use the new Error Policy as a fallback error handler if the multistep error rule failed.



How the Error Policy works (1 of 3)

- Required Configuration:
 - Define the configuration object **Multi-Protocol Gateway Error Policy** and its associated **Multi-Protocol Gateway Error Action(s)** objects
 - In a Multi-Protocol Gateway service configuration, specify the property **Error Policy** with the Multi-Protocol Gateway Error Policy



© Copyright IBM Corporation 2014

Figure 6-11. How the Error Policy works (1 of 3)

WE601 / ZE6011.1

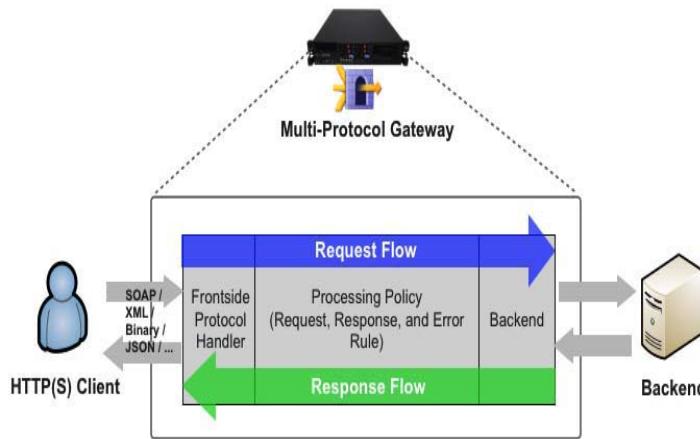
Notes:

The following three slides explain the required configurations to enable the new Error Policy, the pre-conditions when the Error Policy is started, and the expected output when the Error Policy is used for generating the error response.

For configurations, you can see the new property “Error Policy” under the **Advanced** tab in a Multi-Protocol Gateway configuration. To enable the Error Policy for generating the response for an MPGW error, you need to specify it with an existing “Multi-Protocol Gateway Error Policy” object. This object is a new type of object that is introduced by this enhancement. You have implemented error handling logics on the multistep error rule, and you are now able to use the new Error Policy as a fallback error handler if the multistep error rule failed.

How the Error Policy works (2 of 3)

- Error conditions at gateway transaction run time:
 - The MPGW is requested by HTTP/S client
 - An error occurs in front side, request processing, response processing, or backend
 - The error is not successfully handled by any precedent error handlers (typically multistep error rule and special handler like Padding Oracle Protection)
 - The previous conditions collectively leave an “unhandled” error



© Copyright IBM Corporation 2014

Figure 6-12. How the Error Policy works (2 of 3)

WE601 / ZE6011.1

Notes:

The required conditions for the new Error Policy to be run include:

- The MPGW transaction is initiated by an HTTP(S) request representing a web gateway flow. The back-end system can be any type.
- There is an error occurring in the front side (for example, header parsing failure in the frontside), request processing (for example, the Encrypt action in request rule fails), back-end server (for example, failed to establish connection to the back-end server), or response processing (for example, the Filter action rejects the invalid response).
- The multistep error rule, which might be designated by an On-Error action or a Policy Maps matching procedure, is used for handling the error. But neither of them is completed successfully. Or, there is not any error rule to handle the original error.

When the previous conditions are true, the error is not handled by any precedent error handler and then situation of “unhandled error” and then requires a fallback.

For example, you have the error rule that is invoked by the On-Error action, and it failed. Then, the multistep processing rule matching selects the error rule to process, and the matched error rule failed too. Then, it is the time to start the new error policy.

How the Error Policy works (3 of 3)

- Expected results when the Error Policy is invoked:
 - The Error Policy matches the Matching Rule one by one.
 - The Error Action associated with the first-matched Matching Rule is invoked and returns its output to the front side HTTP/S client.

Note: If no Matching Rule is hit or the Error Action fails during its execution, for example, if a connection to the proxy URL cannot be established, then the default SOAP fault is returned to the client.

- Exceptions that the Error Policy is not invoked:
 - The Error Policy is not to be invoked if the error is originated when the user actively initiates the `dp:send-error` extension function to abort the transaction.
 - The Error Policy is not to be invoked if Padding Oracle Protection is enabled.

© Copyright IBM Corporation 2014

Figure 6-13. How the Error Policy works (3 of 3)

WE601 / ZE6011.1

Notes:

When the Error Policy is executed, it evaluates the Matching Rules, selects, and runs the first matched Error Action. At the end, it returns the result to the HTTP/S client.

If an error occurs during the Error Action execution, for example, it failed to fetch the page from the specified URL, then it returns the default SOAP fault to the client.

In the following situations, the Error Policy is not used for generating the response:

- You actively create the transaction failure by using the `dp:send-error` extension function with the specified response message.
- You enable the Padding Oracle Protection setting under the XML Threat Protection tab so that the response messages is obscured.

The screenshot illustrates the configuration of a Multi-Protocol Gateway Error Policy. On the left, the 'Configure Multi-Protocol Gateway Error Policy' window shows a table of policy maps:

Matching Rule	Error Action
XML	redirect2websrv
HTML	LocalHTMLRender
ALL	DefaultResponse

On the right, the 'Configure Multi-Protocol Gateway' window displays various advanced settings:

- Persistent Connections:** Persistent connections are enabled (radio button selected).
- Allow Cache-Control Header:** Cache control headers are disabled (radio button selected).
- Loop Detection:** Loop detection is disabled (radio button selected).
- Follow Redirects:** Follow redirects are disabled (radio button selected).
- Error Policy:** The error policy is currently set to '(none)'.
- Allow Chunked Uploads:** Chunked uploads are disabled (radio button selected).

Figure 6-14. Error Policy Configuration

WE601 / ZE6011.1

Notes:

Multi-Protocol Gateway Error Policy is a new type of config object to be bound to the Multi-Protocol Gateway's property "Error Policy".

It contains an ordered list of Matching Rule with Error Action so that you can define at which particular condition (to be evaluated by the Matching Rule) to run which Error Action for generating the response.

Like the Processing Policy's definition, the Policy Maps evaluate the Matching Rule in order. So the screen capture presents a useful exemplary config for the matching strategy; it works as follows:

- For any HTML request (matched by Content-Type “*html*”), the associated action is to ...
- For any XML request (matched by Content-Type “*xml*”), the associated action is to ...
- If none of the defined matching rule is evaluated to be true, then at the bottom the default action is to ...



Error Policy Configuration – Multi-Protocol Gateway

- Multi-protocol gateway provides four modes:
 - Error Rule
 - Proxy (Remote)
 - Redirect
 - Static (Local)
- Decide which mode to use and configure Response Code, Reason Phrase, and Header Injection to override the current values to be returned to the client

Configure Multi-Protocol Gateway Error Action

Main [HTTP Header Injection](#)

Multi-Protocol Gateway Error Action

[Apply](#) [Cancel](#) [Help](#)

Name	<input type="text"/> *
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Mode	<input checked="" type="checkbox"/> Error Rule <input type="checkbox"/> Proxy (Remote) <input type="checkbox"/> Redirect <input checked="" type="checkbox"/> Static (Local)
Local page location	<input type="text"/> local:/// <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <input type="button" value="Edit..."/> <input type="button" value="View..."/> *
Response Code	<input type="text"/>
Reason Phrase	<input type="text"/>

© Copyright IBM Corporation 2014

Figure 6-15. Error Policy Configuration - Multi-Protocol Gateway

WE601 / ZE6011.1

Notes:

The Multi-Protocol Gateway Error Action is the other new type of config introduced by the feature. Currently, it provides four modes to produce the response message:

- The **Error Rule** mode indicates that the appliance runs the specified Processing Rule and returns its output to the client. You can only choose the Processing Rule with Rule Direction “Error”. The rule allows you to define what you can do (such as logging and rewriting the service variables) in a regular error rule. You can also add, modify, or delete a response header by using the header-related extension functions in the processing rule.
- The **Proxy (Remote)** mode means that the appliance fetches the data from the specified remote HTTP(S) URL and returns the response message to the client.
- The **Redirect** mode indicates that the appliance sends an HTTP redirection to the client with “307 Redirect” and the “Location” header value is as specified in the remote HTTP(S) URL.
- The **Static (Local)** mode is the default mode. It indicates that the appliance fetches the data from the local error page underneath the `local:///` and `store:///` directories and returns the response message to the client.

For **Proxy** and **Static** modes, you can define properties including **Response Code**, **Reason Phrase**, and **Header Injection** to tweak the response. The values override the current values (or default values) to be returned to the client.

Additional Error Policy Processing (1 of 2)

- How to control the HTTP response code and the reason phrase
 - By default “500 Internal Server Error”. Use the Response Code and Reason Phrase to override the defaults.
 - In *Redirect* mode, always “307 Redirect” and cannot be overridden.
- How to control the response headers?
 - Manipulate headers in processing rule (only for *Rule* mode) and use the Header Injection to override headers (for all except for *Redirect* mode).
- “Content-Type” considerations:
 - *Static* mode, you need to statically set the value by using Header Injection.
 - *Proxy* mode, it copies the value that is returned from the Remote URL and you can use Header Injection to override.
 - *Rule* mode, you can manipulate the Content-Type header and use Header Injection to override.

© Copyright IBM Corporation 2014

Figure 6-16. Additional Error Policy Processing (1 of 2)

WE601 / ZE6011.1

Notes:

This slide shows the practical usage notes for the new feature.

For HTTP response code or phrase, the default value is “500 Internal Error”. Except for the *Redirect* mode, which is with fixed value “307 Redirect”, you can use the config including either the Response Code, the Reason Phrase, or both to override the default values.

For response headers, you can always use the Header Injection to override the response headers. And in the *Rule* mode, you can manipulate the response headers by using the extension functions.

“Content-Type” is the most important header to be considered. For *Static* mode, you are usually required to set the value by using the Header Injection. For *Proxy* mode, the appliance copies the value that is returned from the Remote URL and you might use Header Injection to override the value. For *Rule* mode, you might either set the Content-Type on the rule, use the Header Injection, or both to override at the end.



Additional Error Policy Processing (2 of 2)

- In *Proxy* mode, you can use **User Agent** for specifying settings such as SSL Proxy for HTTPS and the Timeout value.
- The feature is only available in Multi-Protocol Gateway services and only applies to HTTP(S) front side protocol handler.
- The display of Web Application Firewall's "Error Policy" in WebGUI is renamed to "Web Application Firewall Error Policy".



© Copyright IBM Corporation 2014

Figure 6-17. Additional Error Policy Processing (2 of 2)

WE601 / ZE6011.1

Notes:

If you are configuring the HTTPS URL for Proxy mode, you need to use the User Agent to set up the required SSL Proxy Profile. The User Agent can be used for setting the timeout value for the connection to the remote URL too.

The feature is only available for MPGW with HTTP(S) traffic and has no impact on other services and flows.

Before release 6.0.0, the web Application Firewall has a similar concept with the object "Error Policy". To eliminate the naming confusion against the new "Multi-Protocol Gateway Error Policy", the previously known "Error Policy" is renamed to "web Application Firewall Error Policy". The change is only taken effective in the displayed name; the config file (.cfg) and exported material in the 6.0.0 firmware and pre-6.0.0 releases are fully compatible.

Unit summary

Having completed this unit, you should be able to:

- Configure an On Error action in a service policy
- Configure an Error rule in a service policy
- Describe how On Error actions and Error rules are selected during error handling

© Copyright IBM Corporation 2014

Figure 6-18. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: When a rule with an **On Error** action encounters an error, the rule is always terminated.
2. True or False: An error rule is unidirectional.
3. A service policy has an error rule and a request rule with an **On Error** action. How does the firmware select the error-handling option?
 - A. The **On Error** radio button is selected from the admin setup page.
 - B. The firmware does not select the error-handling option. Selecting the error-handling option is an off-appliance function.
 - C. If the **On Error** action is already encountered, error processing goes the **On Error** action. If the **On Error** action is not encountered, the error rule gets control.
 - D. None of items A, B, and C.
 - E. All of items A, B, and C.

© Copyright IBM Corporation 2014

Figure 6-19. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. **False.** Continuation of the current rule depends on the setting of Error Mode.
2. **False.** An error rule is active for both request and response rules.
3. **C.**

© Copyright IBM Corporation 2014

Figure 6-20. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 4



Adding error handling to a service policy

© Copyright IBM Corporation 2014

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

9.0

Figure 6-21. Exercise 4

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

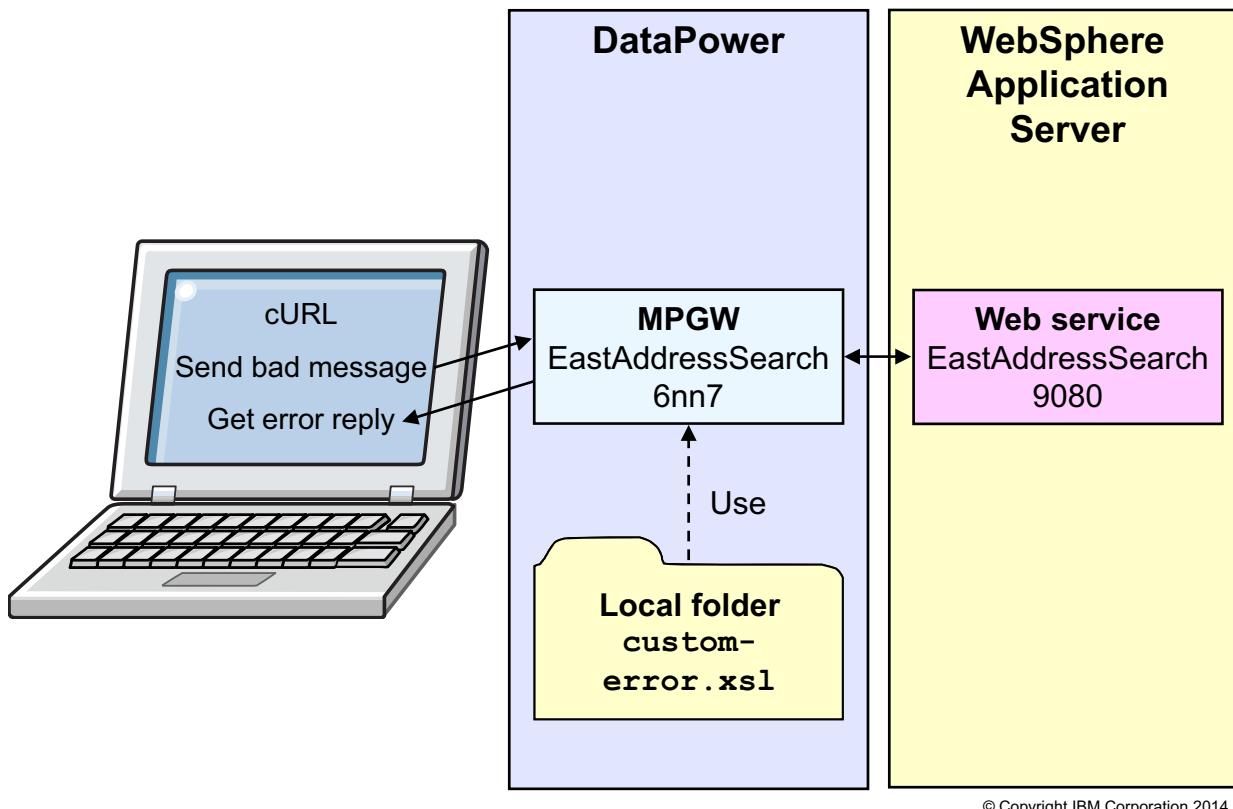
© Copyright IBM Corporation 2014

Figure 6-22. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview



© Copyright IBM Corporation 2014

Figure 6-23. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 7. DataPower cryptographic tools and SSL setup

What this unit is about

This unit describes how to use the cryptographic tools to create keys and certificates. You also set the DataPower objects that are used to validate certificates and configure certificate monitoring to ensure that only valid certificates exist on board.

What you should be able to do

After completing this unit, you should be able to:

- Explain how to use the WebSphere DataPower tools to generate cryptographic keys
- Create a cryptographic identification credential object that contains a matching public and private key
- Create a cryptographic validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up to date
- Describe an SSL proxy profile that accepts an SSL connection request from a client
- Describe an SSL proxy profile that initiates an SSL connection from a DataPower service

How you will check your progress

- Checkpoint
- Hands-on exercise



Unit objectives

After completing this unit, you should be able to:

- Explain how to use the WebSphere DataPower tools to generate cryptographic keys
- Create a cryptographic identification credential object that contains a matching public and private key
- Create a cryptographic validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up to date
- Describe an SSL proxy profile that accepts an SSL connection request from a client
- Describe an SSL proxy profile that initiates an SSL connection from a DataPower service

© Copyright IBM Corporation 2014

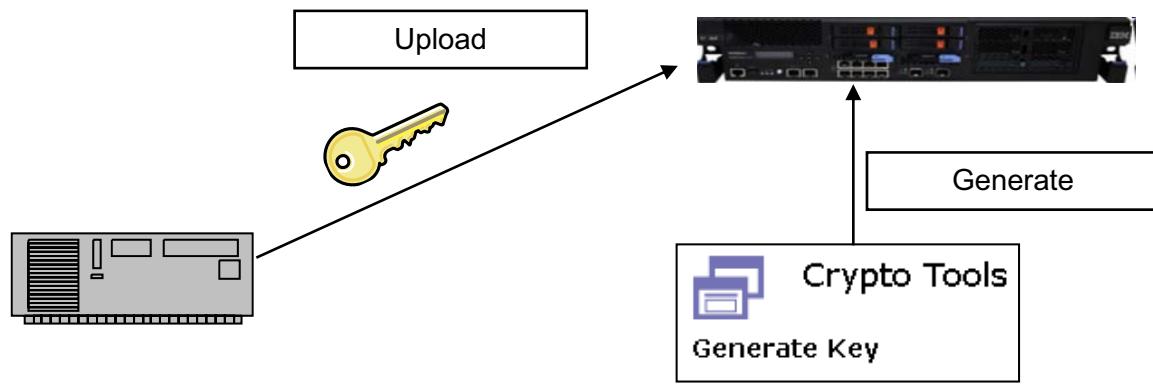
Figure 7-1. Unit objectives

WE601 / ZE6011.1

Notes:

DataPower crypto tools

- There are two methods for creating a private cryptographic key and a self-signed digital certificate:
 - Generated on-board using the DataPower crypto tools
 - Uploading key files to the DataPower appliance



© Copyright IBM Corporation 2014

Figure 7-2. DataPower crypto tools

WE601 / ZE6011.1

Notes:

A self-signed certificate implies that no third-party certificate authority validates the certificate.

All key files are placed in an encrypted storage area on the appliance. The appliance can read them, but the values cannot be displayed to users.

WebSphere Education 

Generating crypto (asymmetric) keys on-board (1 of 2)

- From the WebGUI vertical navigation bar, expand **Administration** and click **Miscellaneous > Crypto Tools**
- Enter key information
 - Only **Common Name (CN)** is required

Generate Key

LDAP (reverse) Order of RDNs	<input type="radio"/> on <input checked="" type="radio"/> off
Country Name (C)	<input type="text"/>
State or Province (ST)	<input type="text"/>
Locality (L)	<input type="text"/>
Organization (O)	<input type="text"/>
Organizational Unit (OU)	<input type="text"/>
Organizational Unit 2 (OU)	<input type="text"/>
Organizational Unit 3 (OU)	<input type="text"/>
Organizational Unit 4 (OU)	<input type="text"/>
Common Name (CN)	<input type="text"/> *
RSA Key Length	<input type="button" value="1024 bits"/>
File Name	<input type="text"/>
Validity Period	<input type="text"/> 365 days

© Copyright IBM Corporation 2014

Figure 7-3. Generating crypto (asymmetric) keys on-board (1 of 2)

WE601 / ZE6011.1

Notes:

The files that are submitted to a certificate authority are created by default.

The fields from **Country Name (C)** down to **Common Name (CN)** are part of the distinguished name.

The file name for the key file that is generated is of the form cert:///name-privkey.pem. If the field is left blank, the system creates this file automatically.

Generating crypto (asymmetric) keys on-board (2 of 2)

- Keys cannot be exported from the DataPower appliance to the workstation
 - Except when **Export Private Key** is selected
 - Also, exported to temporary: directory
- The object name that is entered is used to represent the key and certificate object
- Click **Generate Key** to generate the key and certificate files and objects

The screenshot shows a configuration interface for generating keys. At the top, there are fields for 'Password' and 'Confirm Pass'. Below that is a 'Password Alias' field. The 'Export Private Key' checkbox is checked (on), highlighted with a yellow border. Other checkboxes for 'Generate Self-Signed Certificate' and 'Export Self-Signed Certificate' are unchecked (off). There are also checkboxes for 'Generate Key and Certificate Objects' and 'Object Name', both of which are off. Below these are fields for 'Using Existing Key Object' and 'Generate Key'. The 'Generate Key' button is at the bottom.

© Copyright IBM Corporation 2014

Figure 7-4. Generating crypto (asymmetric) keys on-board (2 of 2)

WE601 / ZE6011.1

Notes:

The password is for the key file that is generated.

Select **on** for **Generate Self-Signed Certificate** to generate a self-signed certificate into the temporary: directory and the store: directory.

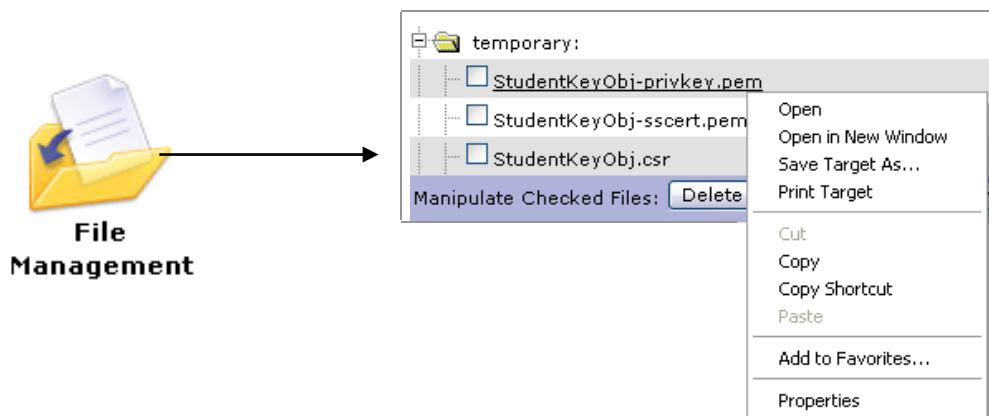
If **Export Self-Signed Certificate** or **Export Private Key** is **off**, then the generated key or certificate is placed in the cert: directory only, where it cannot be edited.

When you click **Generate Key**, you generate a private key file and object, and a certificate file and object.



Download keys from temporary storage

- Keys can be downloaded from temporary storage if **Export Private Key** or **Export Self-Signed Certificate** is on
- From the Control Panel, click the **File Management** icon
- Right-click the file and click **Save Target As...**



© Copyright IBM Corporation 2014

Figure 7-5. Download keys from temporary storage

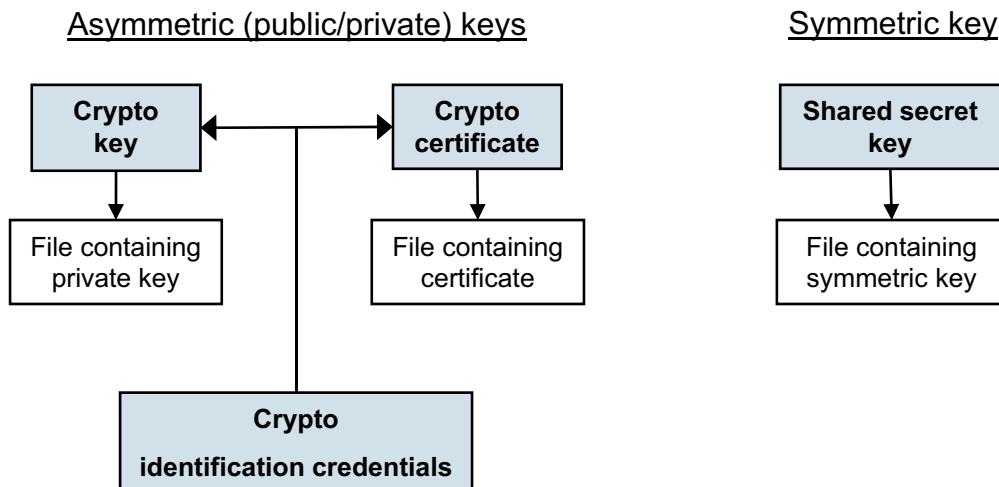
WE601 / ZE6011.1

Notes:

The `temporary:` directory is cleared when the appliance shuts down or restarts.

Key and certificate objects point to files

- The key and certificate objects point to the files on the appliance that are the actual key or certificate
 - Certificate contains the public key



- The **crypto identification credentials** object maintains the relationship between the private key object and its related certificate (public key) object

© Copyright IBM Corporation 2014

Figure 7-6. Key and certificate objects point to files

WE601 / ZE6011.1

Notes:



Crypto shared secret (symmetric) key

- Define a secret key object that points to the key file.
 - From the vertical navigation bar, click **Objects > Crypto Configuration > Crypto Shared Secret Key**
 - The Crypto Shared Secret Key page allows you to define a secret key object for a symmetric key
 - Provides an extra level of security by providing an indirect reference to the file

```

graph TD
    A[Shared secret key] --> B[File containing symmetric key]
  
```

Configure Crypto Shared Secret Key

Main

Crypto Shared Secret Key

Apply Cancel

Name	<input type="text"/> *
Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
File Name	cert: <input type="text"/> dpedu.p12

© Copyright IBM Corporation 2014

Figure 7-7. Crypto shared secret (symmetric) key

WE601 / ZE6011.1

Notes:

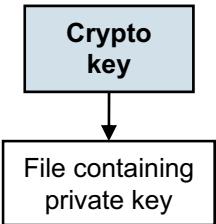
A secret key is used for symmetric key encryption.

DataPower does not have a utility that can generate a symmetric key. Use a tool, such as the Java "keytool" or OpenSSL, to generate a key.

The key file can be uploaded from this page.

Crypto (asymmetric) key

- Define a crypto key object that points to the private key file.
 - From the vertical navigation bar, click **Objects > Crypto Configuration > Crypto Key**
 - The Crypto Key page allows you to define a secret key object for a private asymmetric key
 - Provides an extra level of security by providing an indirect reference to the file



The diagram illustrates the relationship between a 'Crypto key' and a 'File containing private key'. A box labeled 'Crypto key' has a downward-pointing arrow pointing to a box labeled 'File containing private key'.

Crypto Key

Apply Cancel

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

© Copyright IBM Corporation 2014

Figure 7-8. Crypto (asymmetric) key

WE601 / ZE6011.1

Notes:

A crypto key represents the private key that is used for asymmetric key encryption.

The key file can be uploaded from this page.



Crypto certificate

- Create a certificate object from the key file
 - From the vertical navigation bar, click **Objects > Crypto Configuration > Crypto Certificate**
 - Provides an extra level of security by providing an indirect reference to the file

Crypto Certificate

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

Name	<input type="text"/> *
Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
File Name	cert: <input type="button" value="▼"/> (none)
Password	<input type="text"/>
Confirm Password	<input type="text"/>
Password Alias	<input type="radio"/> on <input checked="" type="radio"/> off
Ignore Expiration Dates	<input type="radio"/> on <input checked="" type="radio"/> off

© Copyright IBM Corporation 2014

Figure 7-9. Crypto certificate

WE601 / ZE6011.1

Notes:

The Crypto Certificate page can be accessed from the vertical navigation bar by clicking **Objects > Crypto Configuration > Crypto Key**.

Selecting the **Password Alias** option to be **on** means that the password entered for the key is a password alias that was generated from a password-map.

If **Ignore Expiration Dates** is **off**, the certificate object is placed in a “down” state if it is out of its validity date range. If it is **on**, the certificate object is in an “up” state, but it might be rejected during processing because of an invalid date.



Crypto identification credential

- Create a crypto identification credential
 - Maintains the relationship between a private key and its related certificate that contains the public key
 - Commonly used for SSL authentication
 - From the vertical navigation bar, click **Objects > Crypto Configuration > Crypto Identification Credentials**

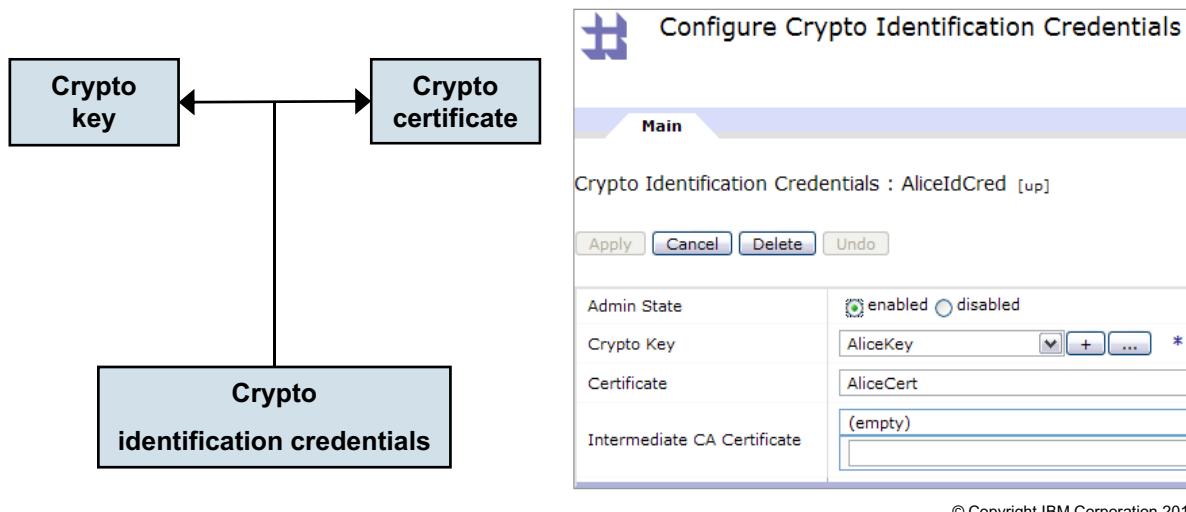


Figure 7-10. Crypto identification credential

WE601 / ZE6011.1

Notes:

Enter a name for the crypto identification credential.

In the **Crypto Key** field, select the crypto key object from the list. You can use the **[+]** and **[...]** to create or edit a crypto key object.

In the **Certificate** field, select a certificate object from the list. You can use the **[+]** and **[...]** to create or edit a certificate object.

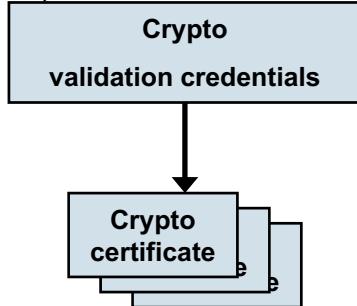
Specify the intermediate certificate authority (CA) certificates, if available, by clicking the **Add**. The process establishes a trust chain that consists of one or more CA certificates.

You can also create a crypto identification credential by clicking **Keys, and Certification Management > Identification Credentials**, from the Control Panel.



Crypto validation credential

- Specifies a list of certificates that a presented certificate or digital signature can be verified against
 - Validates if the presenter is who it says that it is
- The validation mode indicates how to validate against the list:
 - Exact certificate or immediate issuer
 - Full certificate chain checking (PKIX)
- Commonly used for SSL
- Can use certificate revocation lists (CRLs)



Configure Crypto Validation Credentials

Main

Crypto Validation Credentials : pubcert [up]

Admin State: enabled disabled

Certificates:

- ABA-ECOM-Root-CA-pem
- American-Express-Global-CA-pem
- ANX-Network-CA-by-DST-pem
- Asociacion-Nacional-del-Notariado-Mexicano-CA
- Baltimore-EZ-by-DST-pem

Certificate Validation Mode: Match exact certificate or immediate issuer

Use CRL: on off

Require CRL: on off

CRL Distribution Points Handling: Ignore

© Copyright IBM Corporation 2014

Figure 7-11. Crypto validation credential

WE601 / ZE6011.1

Notes:

The certificate validation mode specifies how to validate the presented certificate.

Two options are available:

- Match exact certificates or immediate issuer: The certificate that is presented or the immediate issuer of the certificate must be available on the appliance.
- Full certificate chain checking (PKIX): The certificate that is presented and any intermediate certificates that are chained back to the root certificate must be trusted.

The **Use CRLs** field is used to check whether certificates in the trust chain should be monitored for expiration.

Creating a validation credential that are based on the certificates that are stored in the pubcert directory creates a crypto certificate object for each certificate inside the pubcert directory. The

Create Validation Credential from pubcert: on the Configure Crypto Validation Credentials catalog page does exactly that. An SSL client validates a presented certificate by verifying the issuing CA certificate against its list of common public CA certificates that it contains locally. If the certificate is self-signed, the client must have access to the self-signed certificate. Otherwise, it cannot verify the server identity.

You can create a crypto validation credential that is based on well-known CA certificates that are already stored on the appliance, or imported ones. The option is available on the Crypto Validation Credentials page.



Import and export crypto objects

- Export a **certificate** object to a file
 - The file is exported to `temporary:` directory on the appliance
- Import Crypto Object brings in the exported **certificate** object
 - The file can be in another directory or uploaded
- Private keys can be exported or imported for HSM-equipped appliances only

The screenshot shows the 'Crypto Tools' interface with the 'Export Crypto Object' tab selected. The page title is 'Export Crypto Object'. It has three input fields: 'Object Type' (set to 'Certificate'), 'Object Name' (a required field), and 'Output File Name' (another required field). A 'Help' link is located in the top right corner. At the bottom is a 'Export Crypto Object' button.

© Copyright IBM Corporation 2014

Figure 7-12. Import and export crypto objects

WE601 / ZE6011.1

Notes:

This page is accessed from vertical navigation bar, by clicking **Administration > Miscellaneous > Crypto Tools**.

Certificates are exported to the `temporary:` directory; they can be downloaded by using file management.

Only certificates can be exported and imported.

The object name that is typed must match the name of the exported crypto object exactly.

For an imported crypto object, a password alias can be supplied if the password is not entered.

If the appliance has the Hardware Security Module (HSM) feature installed, private keys can be exported and imported.



Convert Crypto Key Objects / Certificate Objects

- The crypto key object writes a crypto key object to a file in a specific format.
- The certificate object writes a crypto certificate object to a file in a specific format.

The screenshot shows the 'Crypto Tools' interface with the 'Convert Crypto Key Object' tab selected. The interface includes fields for 'Key Name' (with a dropdown menu showing '(none)'), 'Output File Name' (a text input field with an asterisk), 'Output Format' (a dropdown menu set to 'OpenSSH pubkey'), and a 'Convert Crypto Key Object' button. Other tabs visible include 'Import Crypto Object', 'Add SSH Known Host', and 'Convert Crypto Certificate Object'. A 'Help' link is also present.

© Copyright IBM Corporation 2014

Figure 7-13. Convert Crypto Key Objects / Certificate Objects

WE601 / ZE6011.1

Notes:

Crypto Key:

- Convert Crypto Key Object**

Writes a Crypto Key object to a file in a specific format.

- Key Name**

The converted name of the Crypto Key object is Key Name.

- Output File Name**

Specify the name of the output file. When the output format includes private fields of the key, then the file must be in the same directory as the private key object configured file.

- Output Format**

Specify the format of the output file.

OpenSSH pubkey. The OpenSSH public key format that is used in authorized_keys files is OpenSSH pubkey. This format does not contain any private fields of the key (only public ones).

Certificate Object:

- **Convert Crypto Certificate Object**

Writes a Crypto Certificate object to a file in a specific format.

- **Certificate Name**

The converted name of the Crypto Certificate object is Certificate Name.

- **Output File Name**

Specify the name of the output file.

- **Output Format**

Specify the format of the output file.

OpenSSH pubkey. This format is the OpenSSH public key format that is used in authorized_keys files.

“OpenSSH pubkey” format is used with the Secure Cloud Connector.



Uploading keys

- From the vertical navigation bar, click **Objects > Crypto Configuration > Crypto Key**.
- On the Crypto Key page, click the **Upload** to upload key file.

The screenshot shows two windows side-by-side. On the left is the 'Crypto Key' configuration page. It has fields for 'Name' (with a required asterisk), 'Admin State' (radio buttons for 'enabled' and 'disabled'), 'File Name' (dropdown set to 'cert:' with '(none)' selected), 'Password' and 'Confirm Password' (text input fields), and 'Password Alias' (radio buttons for 'on' and 'off'). An 'Upload...' button is highlighted with a yellow box. On the right is a 'Upload File to Directory cert:///'. This dialog has a 'File to upload:' field with a 'Browse...' button, a 'Save as:' field with a required asterisk, and an 'Overwrite Existing File' checkbox. It also has 'Upload' and 'Cancel' buttons. A yellow arrow points from the 'Upload...' button on the main page to the 'Upload' button in the dialog box.

© Copyright IBM Corporation 2014

Figure 7-14. Uploading keys

WE601 / ZE6011.1

Notes:



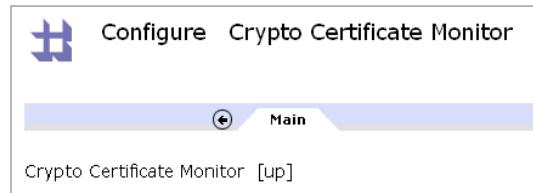
Certificates can expire or get revoked



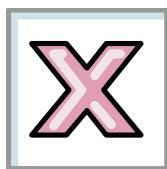
Valid until
02-14-2015

- Certificates are valid only for a certain length of time *and can expire*
- A certificate monitor can constantly check certificates that are stored on the appliance and warn before expiration invalidates the certificate

—This object is ***up*** by default



- Certificates can also be revoked by issuing authority
- The appliance can check certificate revocation lists (CRL) for revoked certificates



© Copyright IBM Corporation 2014

Figure 7-15. Certificates can expire or get revoked

WE601 / ZE6011.1

Notes:

The certificate monitor posts a warning in the system log. Review the system log for warnings.

Expired certificates are not trusted.

Certificate revocation list (CRL) retrieval

- A certificate revocation list (CRL) is a list of certificates from a specific certificate authority (CA) that are revoked and are no longer valid
 - You need to periodically check the validity of certificates
 - Supports CRLs that are in the DER format only
- To set up a CRL list from the vertical navigation bar, click **Objects > Crypto Configuration > CRL Retrieval**
 - Click **CRL Policy** to configure a CRL update policy
- Create a CRL update policy for each CRL to be monitored
 - Can use HTTP or LDAP to retrieve the list
- Is visible and configurable in the **default** domain only

Adding new CRL Policy property of CRL Retrieval

Policy Name	<input type="text"/>
Protocol	http <input type="button" value="..."/> *
CRL Issuer Validation Credentials	<input type="button" value="(...none...)"/> <input type="button" value="+"/> <input type="button" value="..."/>
Refresh Interval	240 minutes *
Cryptographic Profile	<input type="text"/>
Fetch URL	<input type="text"/>

© Copyright IBM Corporation 2014

Figure 7-16. Certificate revocation list (CRL) retrieval

WE601 / ZE6011.1

Notes:

Any trust chain that uses a revoked certificate is broken.

The CRL policy can be configured to fetch CRL lists from a CRL server. The CRL server is checked for validity by using the **CRL Issuer Validation Credential** object that is selected.

The protocol is either **HTTP** or **LDAP**. Appropriate fields display to support the protocol.

The **Cryptographic Profile** identifies the crypto profile to use to connect to the CRL issuer when using SSL.

WebSphere Education



Crypto certificate monitor

- Periodic task runs on the appliance that checks the expiration date of certificates
- To configure a **Crypto Certificate Monitor** from the vertical navigation bar, click **Objects > Crypto Configuration > Crypto Certificate Monitor**
- Expired certificates are written to a log file with a specified warning
- Is visible and configurable in the **default** domain only

 Configure Crypto Certificate Monitor

Main

Crypto Certificate Monitor [up]

Apply Cancel Undo Export

Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Polling Interval	<input type="text"/> 1 day *
Reminder Time	<input type="text"/> 30 day *
Log Level	warning *
Disable Expired Certificates	<input type="radio"/> on <input checked="" type="radio"/> off *

© Copyright IBM Corporation 2014

Figure 7-17. Crypto certificate monitor

WE601 / ZE6011.1

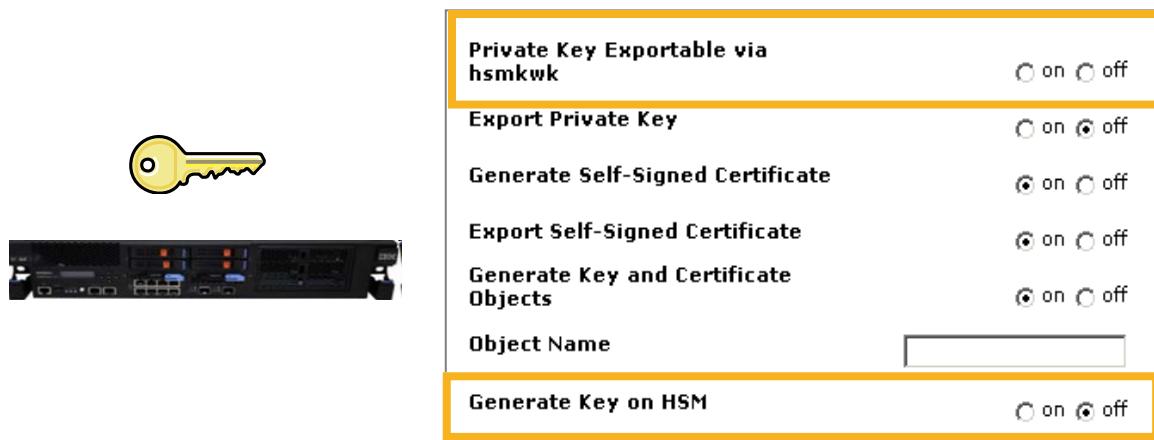
Notes:

Polling interval specifies the frequency with which certificate expiration dates are checked.

Reminder time is the number of days before the certificate expiration event is written to the log file.

Hardware Security Module (HSM)

- Appliances with a hardware security module (HSM) installed can export and import private keys
 - The appliance where the key is exported or imported must also have HSM hardware installed
- DataPower supports FIPS 140-2 level 2 and level 3 security



© Copyright IBM Corporation 2014

Figure 7-18. Hardware Security Module (HSM)

WE601 / ZE6011.1

Notes:

The HSM is a piece of hardware with associated software and firmware that can do a number of security functions. At order time, you can add an HSM to your appliance.

FIPS 140-2 level security is a standard for validating HSMs. For some specialized circumstances, FIPS 140-2 Level 3 security is needed. The appliance supports the process through HSM hardware.

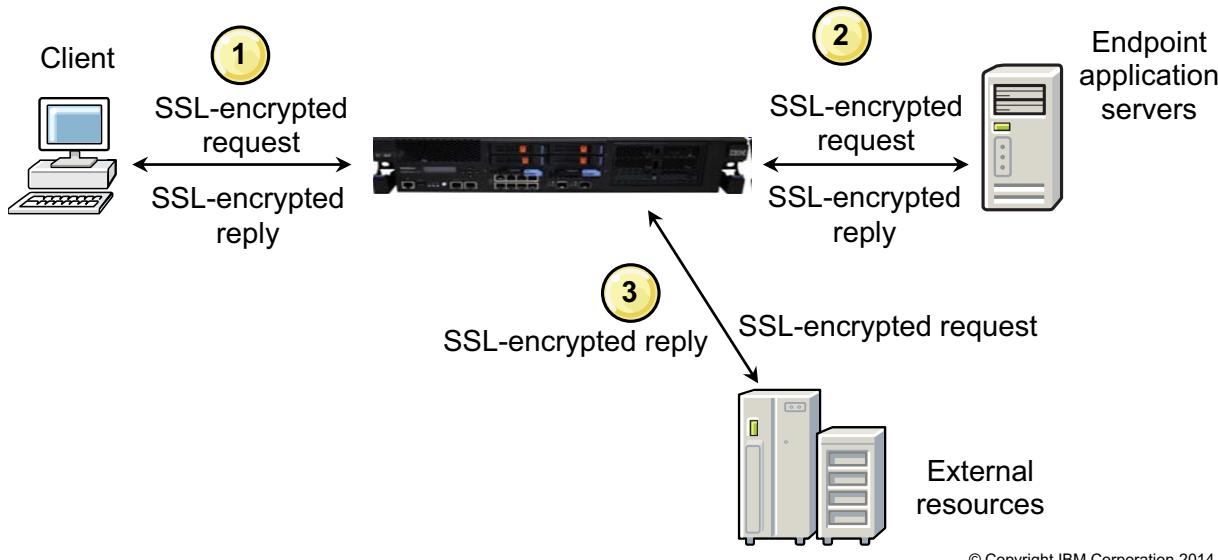
To export private keys on HSM hardware, the **Private Key Exportable via hsmkwk** option must be selected (the location is the Crypto Tools page).

The HSM option is shown only if you have an HSM installed.

HSM is not available on a virtual appliance.

DataPower support for SSL

- DataPower appliance supports SSL:
 1. From remote client to appliance
 2. From appliance to external application server
 3. From appliance to external resource, such as authentication server



© Copyright IBM Corporation 2014

Figure 7-19. DataPower support for SSL

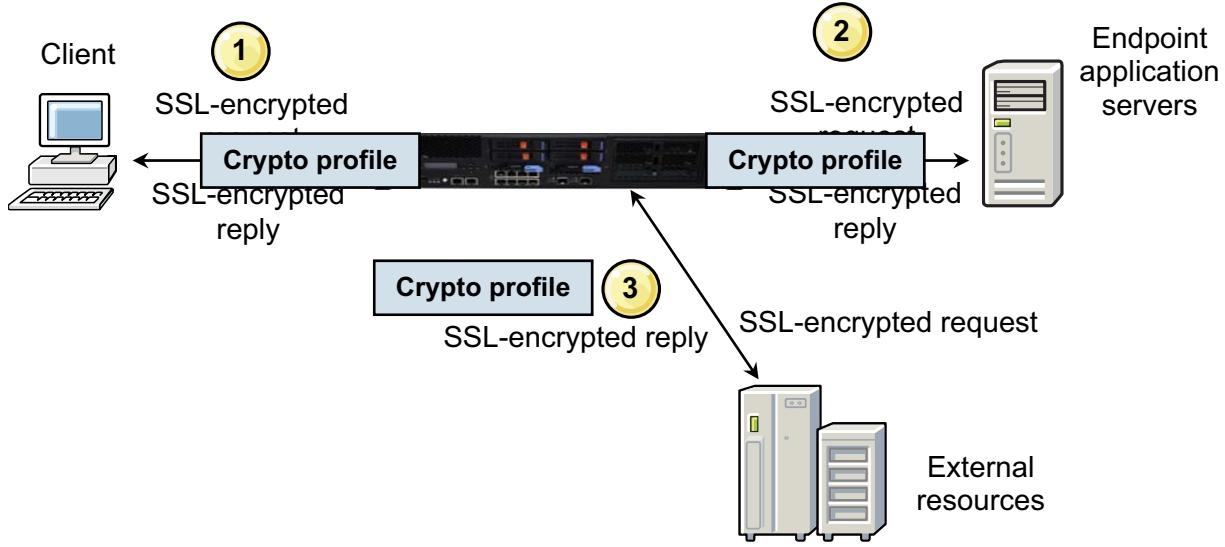
WE601 / ZE6011.1

Notes:

SSL is a point-to-point protocol. A new SSL connection is required for each point. For example, three separate SSL connections are required for connections from remote client to appliance, appliance to endpoint application server, and appliance to external resource.

A crypto profile specifies details of the SSL connection

- It defines:
 - How much DataPower endpoint verification to do, and how to do it
 - What cipher specifications that DataPower can use for this connection
- “1” and “2” are defined directly within the service specification
- “3” is defined within an XML Manager’s **user agent**



© Copyright IBM Corporation 2014

Figure 7-20. A crypto profile specifies details of the SSL connection

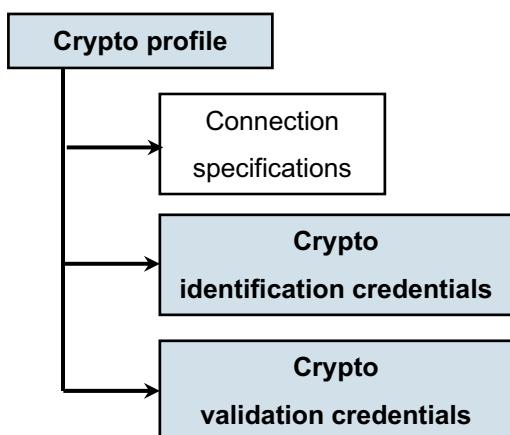
WE601 / ZE6011.1

Notes:



Crypto profile

- Specifies the DataPower end of the SSL connection
- Particulars depend on whether this profile is for an “SSL client” or an “SSL server” end of the connection



Crypto Profile:MyCryptoProfile [up]

Administrative State enabled disabled

Identification Credentials

Validation Credentials

Ciphers

Options

Enable default settings
 Disable SSL version 2
 Disable SSL version 3
 Disable TLS version 1.0
 Permit insecure SSL renegotiation to a legacy client
 Enable compression
 Disable TLS version 1.1
 Disable TLS version 1.2

Send Client CA List on off

© Copyright IBM Corporation 2014

Figure 7-21. Crypto profile

WE601 / ZE6011.1

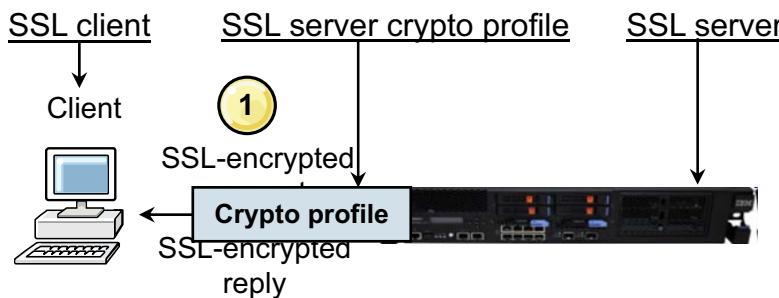
Notes:

The **Ciphers** field specifies what cipher specifications are supported at the DataPower end of the connection. It is composed of one or more cipher suites.

The default cipher string is “HIGH:MEDIUM:!aNULL:!eNULL:@STRENGTH”. The higher preferences are listed first. The default specifies: AES or 3DES (HIGH), 128 bit RC2 or RC4 (MEDIUM), no non-authentication algorithms (anonymous DH) (!aNULL), no non-encryption algorithms (!eNULL), sort list by encryption algorithm key length (@STRENGTH).

Securing connections from client to appliance

- To set up SSL between the client and appliance:
 - DataPower appliance needs to supply a cryptographic certificate that is sent to the Client
 - The appliance maintains the matching private key for the certificate in the **ID credentials**
 - Configure an **SSL server crypto profile** with cryptographic objects that link to certificate-key pair, and specifies the cipher specifications
- The client validates the certificate that is presented by the appliance, which is often included in certificate chain (server-only authentication)
 - Appliance can request a certificate from the client and validate client (mutual authentication)
 - Appliance can use certificates in the **validation credentials** to validate the client certificate



© Copyright IBM Corporation 2014

Figure 7-22. Securing connections from client to appliance

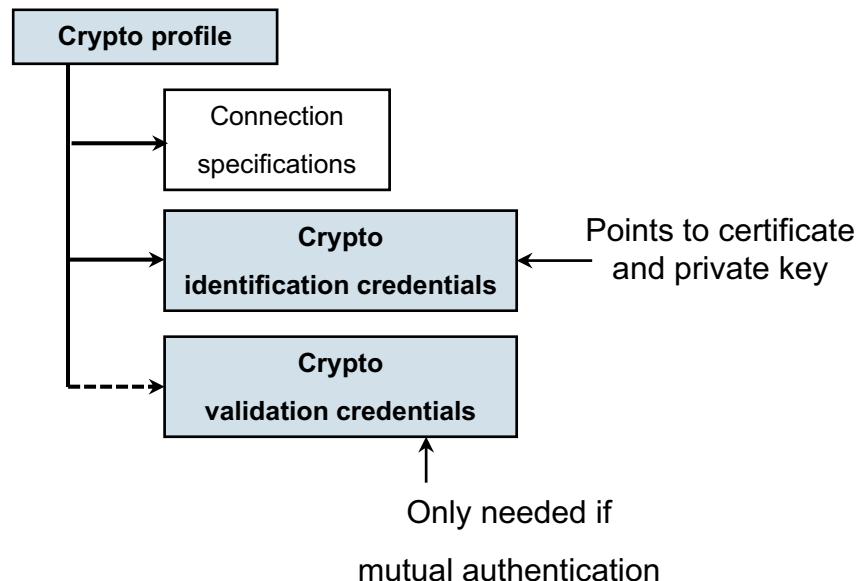
WE601 / ZE6011.1

Notes:

The cryptographic objects that are used by the certificates are linked creating an SSL proxy profile.

Step 1: Appliance supplies cryptographic certificate

SSL server crypto profile



© Copyright IBM Corporation 2014

Figure 7-23. Step 1: Appliance supplies cryptographic certificate

WE601 / ZE6011.1

Notes:

On the Configure XML Firewall page, you create an instance of an SSL server crypto profile that references a crypto identification credential. A crypto identification credential consists of a certificate-key pair that can be used in SSL connections.



Step 2: Configuring SSL server crypto profile

- Configure an **SSL server crypto profile** with cryptographic objects that link to certificate-key pair
 - On the Configure XML Firewall page, select an **SSL Server Crypto Profile** from the drop-down list
 - For a multi-protocol gateway or web service proxy, this profile is set in the front side handler

The screenshot shows the configuration interface for a firewall rule. It is divided into two main sections: 'Back End' and 'Front End'. In the 'Back End' section, there is a 'Server Address' field containing 'localhost' and a 'Server Port' field containing '8080'. In the 'Front End' section, there is a 'Device Address' field containing '0.0.0.0' and a 'Device Port' field containing '8080'. Below these sections are two dropdown menus for 'SSL Client Crypto Profile' and 'SSL Server Crypto Profile'. The 'SSL Server Crypto Profile' dropdown is highlighted with an orange border. Both dropdown menus have a '+' button and a '...' button.

© Copyright IBM Corporation 2014

Figure 7-24. Step 2: Configuring SSL server crypto profile

WE601 / ZE6011.1

Notes:

Configure an SSL server crypto profile with cryptographic objects that link to the certificate-key pair:

- From the Configure XML Firewall page, select an SSL server crypto profile.
- On the Configure XML Firewall page, you can specify the server SSL crypto profile under **Front End**.



Create an SSL server crypto profile

- Create a profile from:
 - **+** (new) on the Configure XML Firewall page
 - **Objects > Crypto Configuration > Crypto Profile**

The screenshot illustrates the process of creating an SSL server crypto profile. On the left, the 'Server Side SSL' configuration page is shown. A red box highlights the '+' button in the 'SSL Server Crypto F' dropdown menu. On the right, the 'Crypto Profile' configuration dialog is displayed, showing a tree view with 'Crypto Profile' selected. The main area of the dialog contains fields for 'Name', 'Admin State' (set to 'enabled'), 'Identification Credentials', 'Validation Credentials', 'Ciphers' (set to 'DEFAULT'), and 'Options' (which includes checkboxes for 'OpenSSL default settings', 'Disabled SSL version 2', 'Disabled SSL version 3', 'Disabled TLS version 1', and 'Send Client CA List').

© Copyright IBM Corporation 2014

Figure 7-25. Create an SSL server crypto profile

WE601 / ZE6011.1

Notes:

The Server Side SSL page is useful when you remember the file names for the key and certificate.

The Crypto Profile page is useful when you have the key, certificate, and credential objects.

This crypto profile uses only an identification credential object because the appliance is not validating the client certificate. If client authentication is required, then a validation credential object needs to be specified.

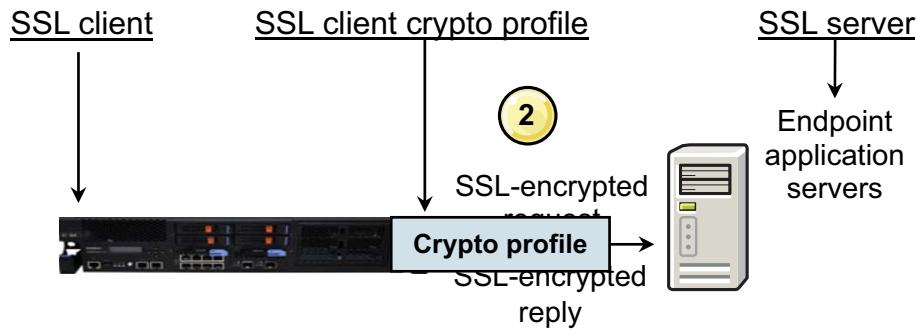
The **Ciphers** field specifies the symmetric key encryption algorithm that is supported by the crypto profile, which is negotiated during an SSL handshake.

The **Options** field allows you to disable support for SSL and TLS versions.

Selecting the **Send Client CA List** enables the transmission of the client CA list to the client. This option is useful when a validation credential is specified. This CA list consists of all the CA certificates that are specified in a validation credential. SSL servers are not required to send a client CA list to clients.

Securing connection from appliance to external application server

- To set up SSL between the appliance and external application server:
 - Certificate that is supplied by the external application server is validated by the DataPower appliance
 - Configure an **SSL client crypto profile** with validation credentials to validate the presented certificate
- The application server might request a certificate the appliance (mutual authentication)
 - Appliance can respond with the certificates in the **identification credentials**



© Copyright IBM Corporation 2014

Figure 7-26. Securing connection from appliance to external application server

WE601 / ZE6011.1

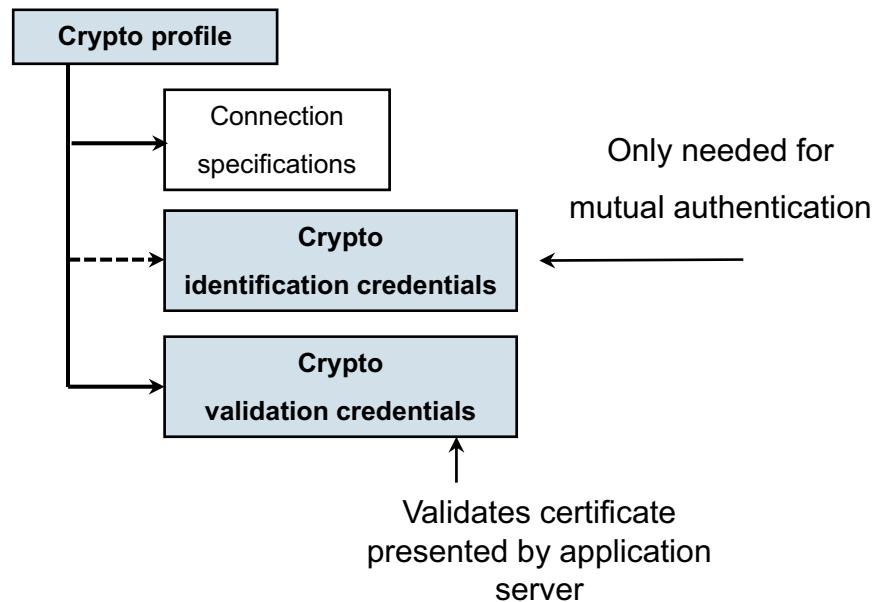
Notes:

The application server can request and validate a certificate from the DataPower appliance.

The SSL client crypto profile specifies a validation credential to validate the certificate that the application server provides.

Step 1: Appliance validates presented certificate

SSL client crypto profile



© Copyright IBM Corporation 2014

Figure 7-27. Step 1: Appliance validates presented certificate

WE601 / ZE6011.1

Notes:

On the Configure XML Firewall page, you create an instance of an SSL client crypto profile that references a crypto validation credential. A crypto validation credential references one or more crypto certificate objects.



Step 2: Configuring an SSL client crypto profile

- Configure an **SSL client crypto profile** that validates the presented certificate
 - On the Configure XML Firewall or Configure Multi-Protocol Gateway page, select an **SSL Client Crypto Profile** from the drop-down list
 - For a web services proxy, the SSL client specification is on the Proxy Settings tab (SSL Proxy)

The screenshot shows a configuration interface with two main sections: 'Back End' and 'Front End'. In the 'Back End' section, there are fields for 'Server Address' (containing 'checkit.west-ibm.com') and 'Server Port' (containing '8080'). In the 'Front End' section, there are fields for 'Device Address' (containing '0.0.0.0') and 'Device Port' (containing '8080'). Below these sections, there are two dropdown menus labeled 'SSL Client Crypto Profile' and 'SSL Server Crypto Profile', each with a '+' button and a '...' button. The 'SSL Client Crypto Profile' dropdown is highlighted with a yellow box.

© Copyright IBM Corporation 2014

Figure 7-28. Step 2: Configuring an SSL client crypto profile

WE601 / ZE6011.1

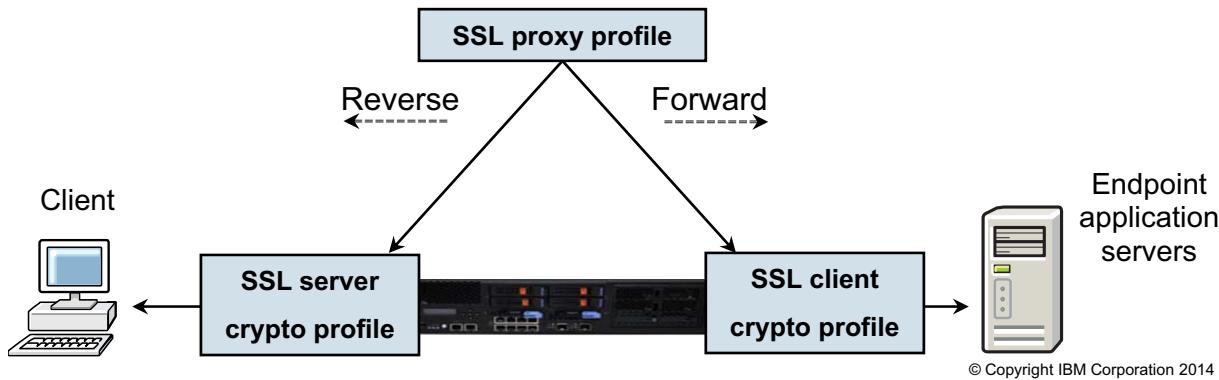
Notes:

As before, if you do not have an SSL client crypto profile, you can use the ... button, or go to **Objects > Crypto Configuration > Crypto Profile**.

The **SSL Proxy** object is covered in later slides.

The SSL proxy profile

- The SSL proxy profile specifies the SSL server and SSL client crypto profiles for a DataPower service
 - Can list 0, 1, or 2 crypto profiles
- The crypto profiles are designated as **forward** or **reverse**
 - Reverse** is when the appliance or service is the SSL server
 - Forward** is when the appliance or service is the SSL client
- Controls SSL session caching
- Specifies whether mutual authentication is required



© Copyright IBM Corporation 2014

Figure 7-29. The SSL proxy profile

WE601 / ZE6011.1

Notes:

Generally, the SSL proxy profile is automatically created when a crypto profile is defined. The name of the generated SSL proxy profile is usually the same name as the service.



SSL proxy profile when the appliance is the SSL server

- **Direction** is indicated as **reverse**
- SSL session caching is **enabled**
 - Cache timeout is 300 seconds
 - Cache maximum size is 20 entries
- Mutual authentication is not supported

SSL Proxy Profile

Name: MPG_Transform

Admin State: Enabled

SSL Direction: Reverse *

Reverse (Server) Crypto Profile: StudentServerCP

Server-side Session Caching: On

Server-side Session Cache Timeout: 300

Server-side Session Cache Size: 20

Client Authentication Is Optional: Off

Always Request Client Authentication: Off

© Copyright IBM Corporation 2014

Figure 7-30. SSL proxy profile when the appliance is the SSL server

WE601 / ZE6011.1

Notes:

Set **Client Authentication is optional** to control when SSL client authentication is optional. When set to **on**, client authentication is not required. When there is no client certificate, the request does not fail. When set to **off** (Default), the SSL server requires client authentication only when the server crypto profile has an assigned Validation Credentials.

Set **Always Request Client Authentication** to control when to request SSL client authentication. When set to **on**, the SSL server always requests client authentication. When set to **off** (Default), the SSL server requests client authentication only when the server crypto profile has an assigned Validation Credentials.



SSL proxy profile when the appliance is the SSL client

- **Direction** is indicated as **forward**
- SSL session caching is **enabled**
- If the appliance or service is acting as both an SSL server and an SSL client (SSL on the front side and the back side of the service), the **Direction** is **two way**
 - Both a forward (client) crypto profile and a reverse (server) crypto profile are entered

SSL Proxy Profile : MyBasicFirewall [up]

Admin State		<input checked="" type="checkbox"/> enabled <input type="checkbox"/> disabled
SSL Direction	Forward	*
Forward (Client) Crypto Profile	StudentClientCP	<input type="button" value="+"/>
Client-side Session Caching	<input checked="" type="radio"/> on <input type="radio"/> off	

© Copyright IBM Corporation 2014

Figure 7-31. SSL proxy profile when the appliance is the SSL client

WE601 / ZE6011.1

Notes:



SSL Proxy Profile list

- **Objects > Crypto Configuration > SSL Proxy Profile**
- The list shows what SSL proxy profiles are using which crypto profiles
 - An SSL proxy profile that functions as a client and a server has both types of crypto profiles

Configure SSL Proxy Profile

Name	Status	Op-State	Logs	Direction	Forward (Client) Crypto Profile	Reverse (Server) Crypto Profile
MyBasicFirewall	saved	up		forward	StudentClientCP	
MyTransformFirewall	saved	up		reverse		StudentServerCP
TwoWayDemo	new	up		two-way	StudentClientCP	StudentServerCP

Add

© Copyright IBM Corporation 2014

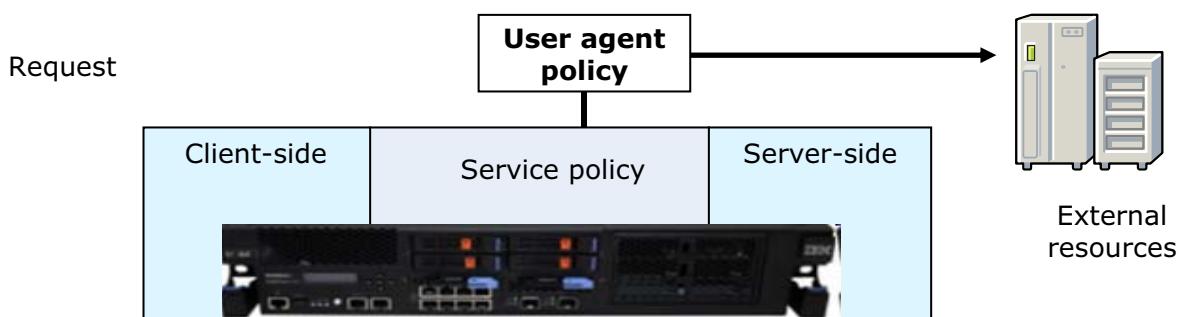
Figure 7-32. SSL Proxy Profile list

WE601 / ZE6011.1

Notes:

User agent

- A client that operates on behalf of a service when establishing a connection to a remote server
 - Commonly used for remote requests initiated from actions in the service policy
 - Example: Configure a user agent to run an SSL profile proxy if matched by a matching expression
 - Policies are applied by using a URL match expression
 - Multiple policies can be associated to a user agent and can be triggered based on different URL strings



© Copyright IBM Corporation 2014

Figure 7-33. User agent

WE601 / ZE6011.1

Notes:

A user agent uses one or more policies to connect to an external server or back side service.



User Agent defined

- The User Agent can be thought of as a utility object to be used by other higher-level DataPower Objects.
- *The User Agent primarily handles the details for network-related outbound calls from the appliance.*
- Tabs on the User Agent and their purposes:
 - **Proxy Policy:** Forward requests to an HTTP proxy server
 - **SSL Proxy Profile Policy:** Automatically initiate SSL Client connections
 - **Basic-Auth Policy:** Inject basic authentication (userid/pw) into requests
 - **Soap-Action Policy:** Inject SOAPAction headers into Web services requests
 - **Pubkey-Auth Policy:** Private key used to validate a certificate being presented in a request for authentication
 - **Allow-Compression Policy:** Allow/disallow compression for certain connections
 - **Restrict to HTTP 1.0 Policy:** Restrict certain connections to HTTP 1.0 based on URL matching
 - **Inject Header Policy:** Inject or override HTTP headers based on URL matching
 - **Chunked Uploads Policy:** Similar to Allow-Compression Policy is used to HTTP 1.1 chunked uploading
 - **FTP Client Policies:** Used to match FTP URLs with client policies to control options for the outgoing FTP connection
 - **SFTP Client Policies:** Same as FTP but with SFTP

© Copyright IBM Corporation 2014

Figure 7-34. User Agent defined

WE601 / ZE6011.1

Notes:

Other tabs exist in configuring a user agent. For more information about these tabs, see the product documentation.



Configuring a user agent

- The User Agent is contained on the XML Manager main page.
- The **default** XML manager object uses a **default** user agent
 - Alternatively, from the vertical navigation bar, click **Network > Other > User Agent** to display or create a user agent
- Create a user agent configuration:
 - In the **Main** tab, enter the user agent name and set HTTP settings
 - Many techniques to set up communication:
 - Proxy Policy: Specifies a URL match expression to forward to a remote address and port
 - Basic-Auth Policy: Associates a user name and password with a set of URLs
 - Soap-Action Policy: Associates a soap-action HTTP header with a set of URLs
 - Pubkey-Auth Policy: Associates a specific private key to use during pubkey authentication

A screenshot of a web-based configuration interface titled "Configure User Agent". At the top, there is a toolbar with icons for back, forward, and search. Below the title, a horizontal navigation bar contains tabs: "Main" (which is highlighted in blue), "Proxy Policy", "SSL Proxy Profile Policy", "Basic-Auth Policy", "Soap-Action Policy", and "Pubkey-Auth Policy". The main content area is currently empty, indicating no configuration details have been entered yet.

Figure 7-35. Configuring a user agent

WE601 / ZE6011.1

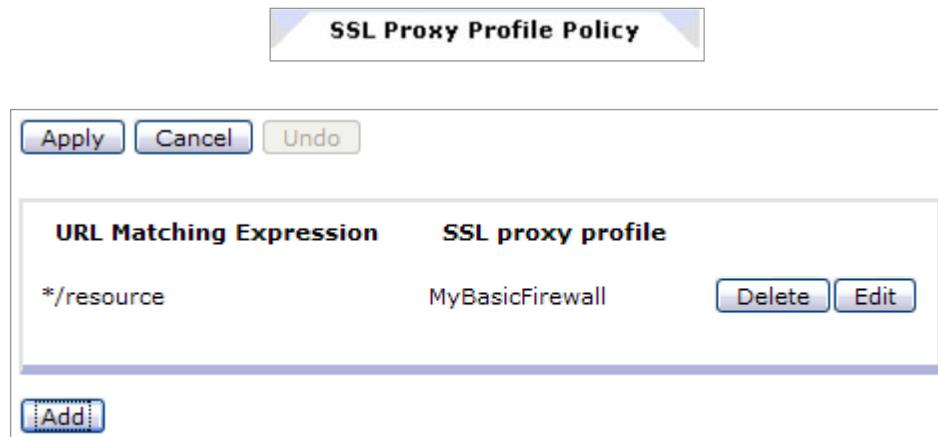
Notes:

Other tabs exist in configuring a user agent. For more information about these tabs, see the product documentation.



Create a user agent configuration

- Configure a user agent to use an SSL proxy profile to communicate with back-end service
 - Click the **SSL Proxy Profile Policy** tab
 - Specify a URL match expression and a corresponding SSL proxy profile



© Copyright IBM Corporation 2014

Figure 7-36. Create a user agent configuration

WE601 / ZE6011.1

Notes:

The SSL proxy profile that is selected is a previously created SSL proxy profile object.

Unit summary

Having completed this unit, you should be able to:

- Explain how to use the WebSphere DataPower tools to generate cryptographic keys
- Create a cryptographic identification credential object that contains a matching public and private key
- Create a cryptographic validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up to date
- Describe an SSL proxy profile that accepts an SSL connection request from a client
- Describe an SSL proxy profile that initiates an SSL connection from a DataPower service

© Copyright IBM Corporation 2014

Figure 7-37. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: The User Agent primarily handles the details for network-related outbound calls from a service policy.
2. What default configuration is provided with DataPower to notify administrator of a certificate expiration?
 - A. DataPower automatically renews expired certificates.
 - B. Expired certificates are removed from the appliance and placed in the expired certificate directory.
 - C. Certificates do not expire.
 - D. Expired certificates are written to a log file with a specified warning
3. True or False: Keys that are generated on-board cannot be exported.
4. True or False: When the remote client initiates an SSL session to a DataPower service, the service end is the “SSL server”.

© Copyright IBM Corporation 2014

Figure 7-38. Checkpoint questions

WE601 / ZE6011.1

Notes:



Checkpoint answers

1. True.
2. D.
3. **False.** Keys can be exported to the temporary: directory if the **Export Private Key** radio button is selected when generating a key on the appliance.
4. True.

© Copyright IBM Corporation 2014

Figure 7-39. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 5



Creating cryptographic objects and
configuring SSL

© Copyright IBM Corporation 2014

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

9.0

Figure 7-40. Exercise 5

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Generate crypto keys by using the WebSphere DataPower cryptographic tools
- Create a crypto identification credential by using a crypto key object and a crypto certificate object
- Validate certificates by using a validation credential object
- Create an SSL proxy profile that accepts an SSL connection request from a client
- Create an SSL proxy profile that initiates an SSL connection from a DataPower service

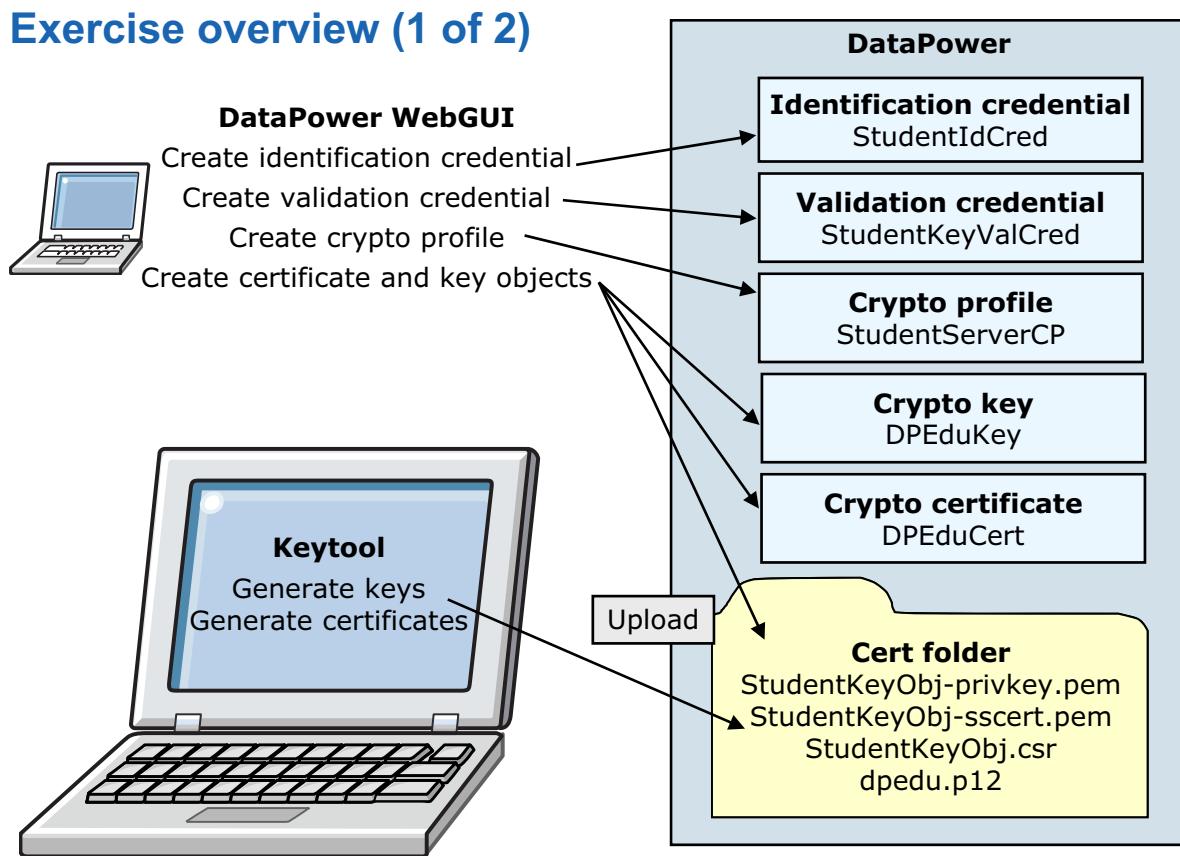
© Copyright IBM Corporation 2014

Figure 7-41. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview (1 of 2)



© Copyright IBM Corporation 2014

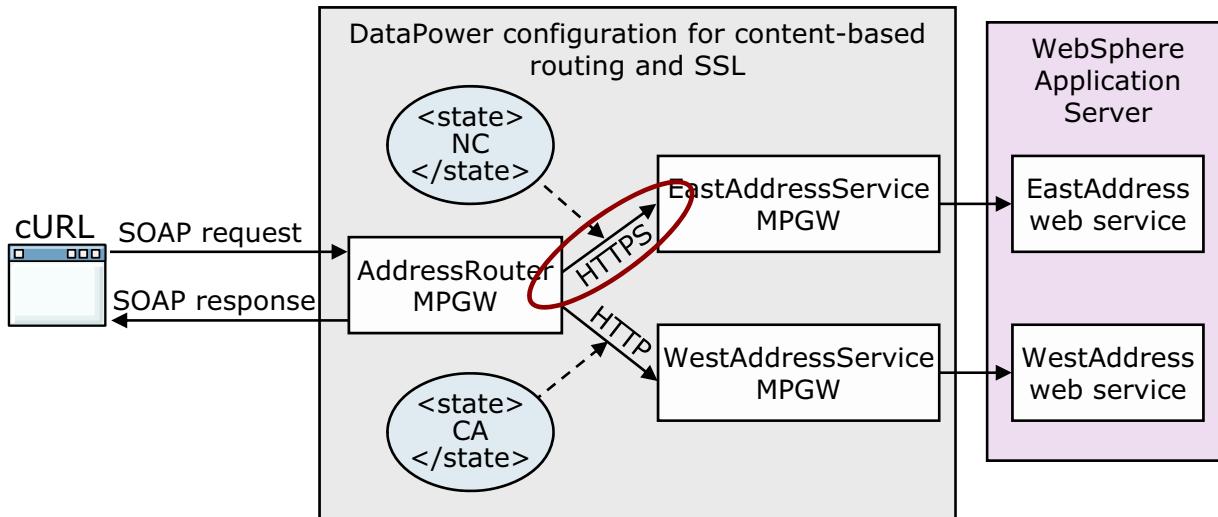
Figure 7-42. Exercise overview (1 of 2)

WE601 / ZE6011.1

Notes:

Exercise overview (2 of 2)

- Create an SSL server configuration on the front side of the EastAddressSearch service
- Create an SSL client configuration on the back side of the AddressRouter service



© Copyright IBM Corporation 2014

Figure 7-43. Exercise overview (2 of 2)

WE601 / ZE6011.1

Notes:

Unit 8. Web service proxy service

What this unit is about

This unit describes the web service proxy service and its role in an XML-aware web-services-based network. You learn the configuration steps that are required to create and manage a web services proxy. You also learn advanced web service configuration steps, such as proxy-level security, SOAPAction policy, and web service endpoint.

What you should be able to do

After completing this unit, you should be able to:

- Describe the web service proxy architecture
- List and explain the configuration steps that are needed to create a web service proxy
- Create and configure a web service proxy policy at various levels of the WSDL file

How you will check your progress

- Checkpoint
- Exercise 9: Configuring a web service proxy



Unit objectives

After completing this unit, you should be able to:

- Describe the web service proxy architecture
- List and explain the configuration steps that are needed to create a web service proxy
- Create and configure a web service proxy policy at various levels of the WSDL file

© Copyright IBM Corporation 2014

Figure 8-1. Unit objectives

WE601 / ZE6011.1

Notes:

Web service proxy overview

- A web service proxy is a middleware component that exists between the client and the web service
 - Decouples web service client from the actual web service
 - Hides web service endpoint address from client
 - Flexibility to change back-end address without affecting client code
 - Can virtualize multiple web services into a single client-facing web service
 - Performs security, validation, and transformation on a request or response to offload these tasks from the back-end web service
- The XG45/XI52/XI50B/XI50x/XB62 DataPower appliances allow you to create a web service proxy to accelerate and mediate communication between a client and a web service
 - Rules that are associated with different parts of a WSDL interface
 - Supports multiple WSDL documents
 - Fine-grained policy control
 - Built-in service level monitoring (SLM) capabilities

© Copyright IBM Corporation 2014

Figure 8-2. Web service proxy overview

WE601 / ZE6011.1

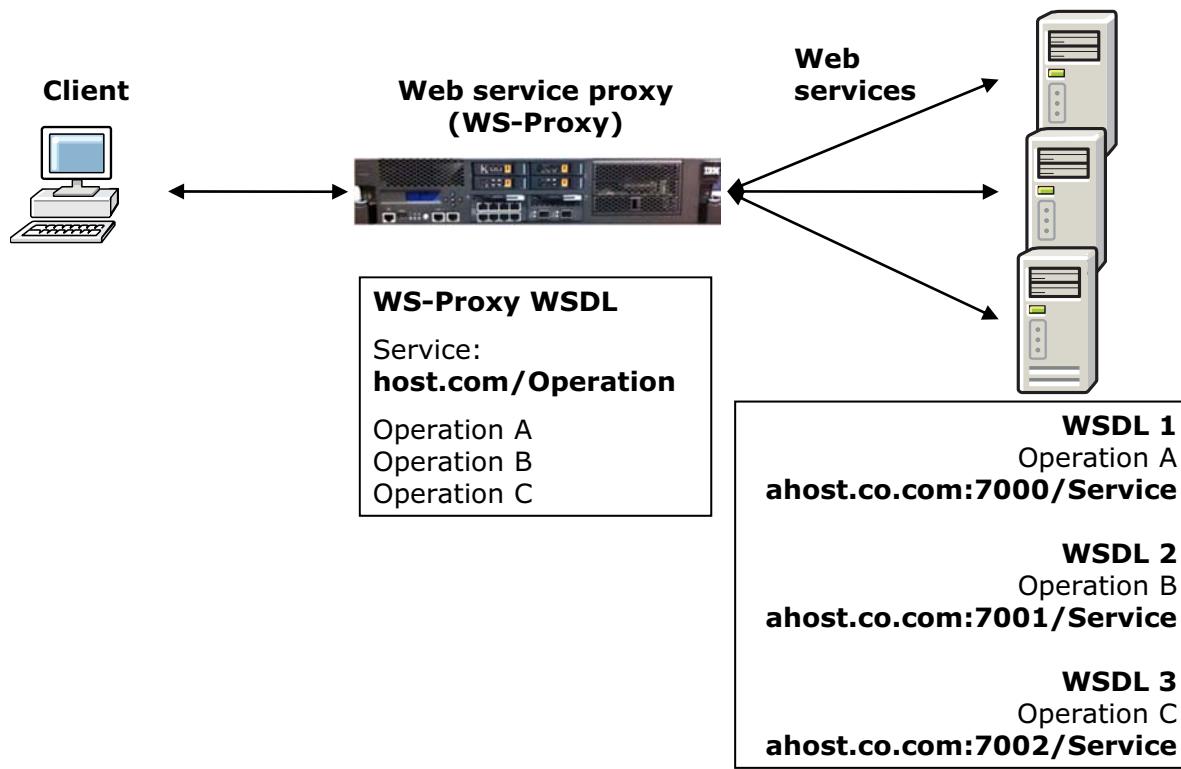
Notes:

It is not necessary for the client to know the endpoint address of the web service. It is always forwarded to the web service proxy. If the web service endpoint changes, only modifications to the web service proxy are required. The client is unaffected.

Performing security, validation, and transformation on the DataPower appliance for web service proxy requests improves application performance because it is done at a hardware level. It is offloaded from the application server, which would perform these tasks in software. You can also apply a standard security policy for your web service proxy on the DataPower appliance because all requests pass through the appliance.



Web service virtualization



© Copyright IBM Corporation 2014

Figure 8-3. Web service virtualization

WE601 / ZE6011.1

Notes:

The web service proxy has a WSDL file that lists the operations it supports. These operations can be aggregated from multiple WSDL files that are in different locations.

The web service proxy maintains a mapping of a local endpoint and remote endpoint for each WSDL file.

Web service proxy benefits

Web service proxy quickly virtualizes your existing services

- Virtualizes a service simply by loading a WSDL document
 - Clients now connect directly to web service proxy and not the back-end service
- Creates processing policy with rules and actions at fine-grained level
 - Rules at a proxy, service, port, or operation level can process request and response messages
- Automatic schema validation of request, response, and fault messages (user policy)
 - User does not need to create a processing policy for this validation
- Service virtualization can occur in real time
 - Web service proxy can update the proxy WSDL automatically when underlying WSDL is updated
 - Integrates with WebSphere Service Registry and Repository and UDDI registries
- Can enforce policy and monitor performance of services
 - Multiple appliance support for virtual services

© Copyright IBM Corporation 2014

Figure 8-4. Web service proxy benefits

WE601 / ZE6011.1

Notes:

A **user policy** allows you to schema validate request, response, and fault messages. It is automatically created when you create a web service proxy.

The web service proxy is built on top of the XML firewall. Therefore, it provides all of the functionality of an XML firewall, such as encryption, validation, AAA, and more.

UDDI is a service repository that is used to search for WSDL files of a service.

Creating a WSDL cache policy enables the proxy WSDL file to be updated automatically when the underlying WSDL changes.

You can create an SLM peer group to share SLM data and enforce SLM policy between multiple DataPower appliances.

The screenshot shows the 'Web service proxy configuration tabs (1 of 2)' section. At the top, there's a navigation bar with a back arrow, the text 'WebSphere Education', and the IBM logo. Below the navigation bar, the title 'Web service proxy configuration tabs (1 of 2)' is displayed in blue. A sidebar on the left contains a square icon and the text 'Configure Web Service Proxy'. The main area features a horizontal tab bar with several tabs: 'WSDL files' (selected), 'SLM Policy', 'Services', 'Policy', 'SLA Policy Details', 'Proxy Settings', 'Advanced Proxy Settings', and 'Headers/Params'. The 'WSDL files' tab is highlighted with a blue background.

- WSDL files
 - Uploads or associates a WSDL document with a web service proxy
 - Configures proxy and remote URI (address, port) of services that are contained in WSDL document
- SLM Policy
 - Monitors and shapes traffic that enters the web service proxy
- Services
 - Listing of services that are defined in each WSDL document
 - Can publish services to UDDI registry
- Policy
 - Configures a web service proxy policy
- SLA Policy Details
 - View WSDL attachments that relate to service level agreement
- Proxy Settings
 - Specifies method of forwarding to service, security, XML manager, and HTTP settings

© Copyright IBM Corporation 2014

Figure 8-5. Web service proxy configuration tabs (1 of 2)

WE601 / ZE6011.1

Notes:

There are configuration options for each tab in the web service proxy GUI.

The WSDL files, services, policy, and proxy settings tabs are covered in this unit.

SLM Policy is covered in more detail in another unit.

Various policies (WS-Policy, WS-MediationPolicy, for example) can be specified in individual files, but point to an attachment point in a WSDL file. The SLA Policy Details tab is not covered in this course.

The image shows a screenshot of the IBM WebSphere Education interface. At the top left is the "WebSphere Education" logo. At the top right is the "IBM" logo. Below the header, the title "Web service proxy configuration tabs (2 of 2)" is displayed in blue. Underneath the title is a section titled "Configure Web Service Proxy". A horizontal navigation bar contains the following tabs: "Proxy Settings" (selected), "Advanced Proxy Settings", "Headers/Params", "WS-Addressing", "WS-ReliableMessaging", "Monitors", "XML Threat Protection", and a small "Help" icon.

- Advanced Proxy Settings
 - Configures advance connection settings
- Headers/Params
 - Add or remove HTTP headers and pass style sheet parameters
- WS-Addressing
 - Specifies the WS-Addressing mode for this service
- WS-ReliableMessaging
 - Toggle to enable WS-ReliableMessaging (deprecated)
- Monitors
 - Identifies any message monitors associated to this service
- XML Threat Protection
 - Provides protection against XML threats

© Copyright IBM Corporation 2014

Figure 8-6. Web service proxy configuration tabs (2 of 2)

WE601 / ZE6011.1

Notes:

The XML Threats and Monitors tabs are covered in other units.



Web service proxy basic configuration steps

1. Create or obtain a WSDL document that describes your web service
2. Use the DataPower WebGUI to create a web service proxy
 - **Web Service Proxy** icon in DataPower WebGUI Control Panel
or
 - Using vertical navigation bar, click **Services > Web Service Proxy > New Web Service Proxy**
3. Upload the WSDL document and add it to the web service proxy
4. Configure endpoint of services that are defined in WSDL document
 - Define both proxy URI and endpoint URI for each service in WSDL document
5. Specify a service policy that consists of rules for the web service (optional)

© Copyright IBM Corporation 2014

Figure 8-7. Web service proxy basic configuration steps

WE601 / ZE6011.1

Notes:

The URI consists of an address and port.

Step 5 is optional because the appliance generates a default service policy. The default service policy applies at the proxy level for each service. You can override the default service policy with a more specific policy at a fine-grained level for each service, port, or operation. Only one policy is executed per request or response.

You can also do these additional configuration steps:

1. Configure how the proxy forwards requests to the back-end web service. By default, the URI defined in the WSDL document is used to determine the back-end web service.
2. Select the SOAP action policy to specify how to consume messages with a SOAPAction header.
3. Configure security settings, such as proxy-wide AAA settings, the decryption key, and the SSL proxy profile, to a back-end service.

Step 1: Obtain WSDL document

- A WSDL document that describes your web service is required before creating a web service proxy
- A WSDL document describes a web service interface by using XML
 - Uses the W3C XML schema type system for type information
 - Contains operations and messages that are bound to a network protocol and message format
 - Includes binding and location information for published web services
- DataPower creates a web service proxy that is based on the structure of a WSDL document
 - WSDL-based configuration consists of a service, ports, and operations
 - SLM and policy configuration can be defined at various levels of the WSDL document

© Copyright IBM Corporation 2014

Figure 8-8. Step 1: Obtain WSDL document

WE601 / ZE6011.1

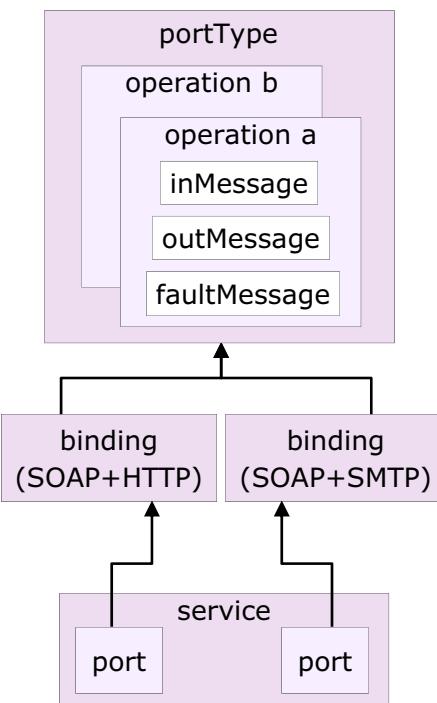
Notes:

A WSDL document describes the service operations that can be invoked together with their messaging protocol, transport, and endpoint address.

Each operation contains an input and output message, whose types are defined by using the XML schema type system.

WSDL structure

- portType
 - Abstract definition of a service
 - Same idea as a Java interface
- binding
 - How to access the portType
 - Multiple bindings per portType
 - HTTP, JMS, SMTP, and more
- port
 - Represents an individual endpoint
- service
 - Something that can be invoked
 - Represents a collection of ports



- All levels of the WSDL are visible and available within the web service proxy

© Copyright IBM Corporation 2014

Figure 8-9. WSDL structure

WE601 / ZE6011.1

Notes:

This diagram shows the general structure of a WSDL and the relationships of the elements to each other.



Step 2: Creating a web service proxy

- You can create a web service proxy by:
 - Clicking the **Web Service Proxy** icon in the DataPower WebGUI Control Panel and clicking **Add** on the “Configure Web Service Proxy” listing page



Web Service Proxy Name	Op-State	Logs	Type	Req-Type	Back Side URL	Resp-Type
AddressSearchProxy	up		static-from-wsdl	soap	NA	soap
<input type="button" value="Add"/>						

- Using the vertical navigation bar, click **Services > Web Service Proxy > New Web Service Proxy**
- You are first prompted for the name of the web service proxy

Web Service Proxy Name
<input type="text"/>
<input type="button" value="Create Web Service Proxy"/>

© Copyright IBM Corporation 2014

Figure 8-10. Step 2: Creating a web service proxy

WE601 / ZE6011.1

Notes:

You can use either approach in creating a web service proxy. The web pages are identical.

From the web service proxy catalog list, you click **Add** to create a service. You are first prompted for the new service name.



An alternative: Web service proxy object editor

- A web service proxy can be created by using a non-graphical, non-wizard approach (not common)
 - From the vertical navigation bar, click **Objects > Service Configuration > Web Service Proxy**
 - All configuration options are available

Configure Web Service Proxy

Web Service Proxy: AddressSearchProxy [up]

Administrative State: enabled disabled

Comments:

Service Priority: 1

© Copyright IBM Corporation 2014

Figure 8-11. An alternative: Web service proxy object editor

WE601 / ZE6011.1

Notes:

The WSDL cache policy and some attachment processing specifications are example configurations that are only possible by using this editor.



1. The **Web Service Proxy Name** is transferred from the earlier prompt
2. The **Add WSDL** option is already active for a new service
3. Add the WSDL file by using **one** of the following approaches:
 - Enter **WSDL File URL** (remote or local URL)
 - Upload WSDL file to **local:** directory
 - Select previously uploaded WSDL document
 - Retrieve from registry
4. Click **Next**

© Copyright IBM Corporation 2014

Figure 8-12. Step 3: Add WSDL document to web service proxy

WE601 / ZE6011.1

Notes:

The Configure Web Service Proxy page is displayed.

The **Edit WSDL or Subscription**, **Add WSDL**, **Add UDDI Subscription**, **Add WSRR Subscription**, and **Add WSRR Saved Search Subscription** options act like a button when they are selected.

Click **Upload** to upload a WSDL file to the DataPower appliance. The WSDL file can be uploaded to the **local:** directory (which is accessible in the current domain) or to the **store:** directory (which is accessible in all domains). The preference is for the "local" files to be in the **local:** directory.

When you upload a WSDL file, the WSDL file URL is automatically populated.

You can upload and add multiple WSDL files.

You can also enter an HTTP URL into the WSDL file URL, and the web page populates the fields with information from the WSDL file.



Step 4: Configure WSDL endpoint

After clicking **Next**, specify the local and remote URI of the WSDL service

- Local (what the client sees):
 - Local endpoint handler
 - Specify URI sent by client
- Remote (where the web service really is):
 - Web service endpoint (protocol, host name, port, and URI)

Web Service Proxy WSDLs			
AddressSearchService - AddressSearch			
Local			
Local Endpoint Handler	URI		
AddressSearchFSH	/EastAddressSearch		
Remote			
Protocol	Remote Endpoint Host	Port	Remote URI
HTTP	transit01.com	9080	/EastAddress/services/AddressSearch
Published	<input checked="" type="checkbox"/> Use Local		

© Copyright IBM Corporation 2014

Figure 8-13. Step 4: Configure WSDL endpoint

WE601 / ZE6011.1

Notes:

The **Local** section contains necessary information for the client to call a service on the web service proxy. You create a local endpoint handler to specify a port number that listens for requests of a particular service and forwards to the remote destination. The endpoint handler is another name for a front side protocol handler. These handlers can be managed individually from **Objects > Protocol Handlers**.

Under **Local**, the URI field is what the client uses prefaced with the host name of the DataPower appliance and the port that is specified in the **Local Endpoint Handler** object.

The **Remote** section contains information about the web service endpoint address that the web service proxy calls. Make sure that you change the default host name of `localhost` to the correct host name.

The **Remote** section **Protocol** choice lists the various protocols available on the back side of the service. Depending on the particular protocol that is selected, the other fields adjust:

- DPMQ
- DPTIBEMS
- DPWASJMS

- HTTP
- HTTPS
- MQ
- TIBEMS

The protocols and URLs are defined as part of the documentation of the url-open extension element. For more information, see the DataPower Information Center.



Step 5: Configure local endpoint handler

- A Local Endpoint Handler (a front side handler) is used to determine the IP address, port, and protocol
- Click the plus sign (+) to create a new local endpoint handler object
 - Specify the appliance local IP address and port number to listen for requests
 - Can also restrict access based on HTTP attributes

Create a New:

- FTP Poller Front Side Handler
- NFS Poller Front Side Handler
- SFTP Poller Front Side Handler
- ... (several other options)
- HTTP Front Side Handler** (highlighted with a red box)
- ... (several other options)

HTTP Front Side Handler

AddressSearchFSH

AddressSearchService - Address Search

Local

Local Endpoint Handler

http_fsh_east_address_search

http_fsh_east_address_search +

Apply Cancel

Name: AddressSearchFSH

Administrative State: enabled

Comments: (empty)

Local IP Address: 0.0.0.0

Port Number: 6885

HTTP Version to Client: HTTP 1.1

© Copyright IBM Corporation 2014

Figure 8-14. Step 5: Configure local endpoint handler

WE601 / ZE6011.1

Notes:

Other choices for local endpoint handlers not visible in the menu in the slide are MQ, SFTP Server, Stateless Raw XML, and Stateful Raw XML.

The Local IP Address of 0.0.0.0 means that the endpoint handler listens for requests on all of the appliance interfaces.

Make sure that the port number you specify here is unique.

Endpoint handlers and front side handlers are synonymous terms, and are configured in the same way.

WebSphere Education

Step 6: Add the WSDL to the service

AddressSearchService - AddressSearch			
Local			
Local Endpoint Handler	URI	Binding (Suffix)	Edit/Remove
AddressSearchFSH	/EastAddressSearch	<input checked="" type="checkbox"/> SOAP 1.1 <input type="checkbox"/> SOAP 1.2 HTTP GET	Add

AddressSearchService - AddressSearch			
Local			
Local Endpoint Handler	URI	Binding (Suffix)	Edit/Remove
AddressSearchFSH	/EastAddressSearch	SOAP 1.1()	Edit Remove
(none)		<input checked="" type="checkbox"/> SOAP 1.1 <input type="checkbox"/> SOAP 1.2 HTTP GET	Add

Remote			
Protocol	Remote Endpoint Host	Port	Remote URI
HTTP	myserver.ibm.com	9080	/EastAddress/services/AddressS

Click **Add** to add the customized WSDL information to the service; then click **Next**

© Copyright IBM Corporation 2014

Figure 8-15. Step 6: Add the WSDL to the service

WE601 / ZE6011.1

Notes:

Clicking **Add** adds the selected WSDL to the service.

As soon as a WSDL is added, it can be edited or removed by selecting the appropriate icon to the right of the WSDL.

Clicking **Next** commits the WSDL to the service.



Initial WSDL completed

- Completed WSDL is listed
 - WSDL status is listed
 - Edit WSDL or Subscription** option is highlighted
- Additional options, such as adding another WSDL, are now available

The screenshot shows the 'Configure Web Service Proxy' interface. At the top, there's a toolbar with tabs: WSDL files, SLM Policy, Services, Policy, SLA Policy Details, Proxy Settings, Advanced Proxy Settings, Headers/Params, and a help icon. Below the toolbar, the 'Web Service Proxy Name' field contains 'AddressSearchProxy'. Underneath are buttons for Apply, Cancel, Delete, and Refresh. To the right are links for Export, View Log, View Status, View Operations, Show Probe, Validate Conformance, and Help. The main area is titled 'WSDLs' and contains a table with one row. The table has columns for 'WSDL Source Location', 'Endpoint Handler Summary', 'WSDL Status', 'WS-I BP Status', and 'Action'. The first column shows 'local:///EastAddressSearch.wsdl'. The second column shows '1 up / 1 configured'. The third column shows 'Okay'. The fourth column shows 'Okay'. The fifth column has a 'Remove' button. The entire row is highlighted with an orange border.

WSDL Source Location	Endpoint Handler Summary	WSDL Status	WS-I BP Status	Action
local:///EastAddressSearch.wsdl	1 up / 1 configured	Okay	Okay	Remove

© Copyright IBM Corporation 2014

Figure 8-16. Initial WSDL completed

WE601 / ZE6011.1

Notes:

The WSDL is now part of the service.

More WSDLs or subscriptions can be added to the service.



Other ways to get a WSDL

- Subscribe to a UDDI registry
- Subscribe to WebSphere Registry and Repository (WSRR)
- Subscribe to a WSRR saved search
 - Can automatically “push” changes to the web service proxy

The screenshot shows the 'Configure Web Service Proxy' interface. In the 'WSDLs' section, there is a table with one row. The table has columns for 'WSDL Source Location', 'Endpoint Handler Summary', 'WSDL Status', 'WS-I BP Status', and 'Action'. The first column contains 'local:///EastAddressSearch.wsdl'. The second column contains '1 up / 1 configured'. The third and fourth columns both show 'Okay'. The fifth column has a 'Remove' link. Above the table, there is a navigation bar with buttons for 'Edit WSDL or Subscription', 'Add WSDL', 'Add UDDI Subscription' (which is highlighted with a red box), 'Add WSRR Subscription', and 'Add WSRR Saved Search Subscription'. There are also links for 'Export', 'View Log', 'View Status', 'View Operations', 'Show Probe', 'Validate Conformance', and 'Help'.

Figure 8-17. Other ways to get a WSDL

WE601 / ZE6011.1

Notes:

A subscription to a registry might also retrieve a WSDL file.

Generally, the registries are polled for the WSDL file on a timed basis, and can also be explicitly polled.

A WSRR Saved Search can be configured to send a WSDL file update from WebSphere Service Registry and Repository to the service.



View WSDL services

- Click the **Services** tab to view the services that are extracted from the WSDL document
 - Click **Publish to UDDI** to configure a connection to a UDDI registry

A screenshot of the DataPower appliance's configuration interface. At the top, there is a navigation bar with tabs: WSDL files, SLM Policy, Services (which is the active tab), Policy, SLA Policy Details, Proxy Settings, and Advanced. Below the navigation bar, there is a section titled "Web Service Proxy Name [up]" with a text input field containing "AddressSearchProxy" and a required indicator (*). Below this are buttons for Apply, Cancel, Delete, Refresh, and Publish to UDDI. Underneath this section, there is a table titled "Services" with one row. The row contains the WSDL Name: "EastAddressSearch.wsdl", the Service name: "AddressSearchService", and a "Publish to UDDI" button. The entire interface has a clean, modern design with a white background and light gray borders for the various components.

© Copyright IBM Corporation 2014

Figure 8-18. View WSDL services

WE601 / ZE6011.1

Notes:

The appliance automatically generates the services in this tab when you add a WSDL file to the web service proxy.

Universal Description, Discovery, and Integration (UDDI) is an XML-based registry that is used to search for WSDL documents.

UDDI is implemented as a web service that you can publish and search for web services.

The DataPower appliance does not provide a UDDI registry, only a connection.

The **View Operations** opens another window that lists the operations that are defined in the WSDLs exposed by this service.

Retrieve the client WSDL from the service

- You can retrieve the client-facing WSDL from the service
 - Edit the endpoint handler to allow HTTP GET
 - Enter:
`http://myDPappliance.com:6999/EastAddressSearch?wsdl`
 - 6999: The port number of the web service proxy
 - /EastAddressSearch: the local or client URI to invoke web service
- Returned WSDL contains the IP address or service port for the appliance as the WSDL <address location= >
 - Original


```
<wsdl:port binding="..." name="AddressSearch">
<address location="http://training.ibm.com:9080/
  EastAddress/services/AddressSearch" /> </wsdl:port>
```
 - Retrieved by ?wsdl


```
<wsdl:port binding="..." name="AddressSearch">
<address
  location="http://192.168.10.41:6999/EastAddressSearch" />
</wsdl:port>
```

Allowed Methods and Versions	<input checked="" type="checkbox"/> HTTP/1.0 <input checked="" type="checkbox"/> HTTP/1.1 <input checked="" type="checkbox"/> POST <input checked="" type="checkbox"/> GET <input checked="" type="checkbox"/> PUT <input type="checkbox"/> HEAD <input type="checkbox"/> OPTIONS
------------------------------	---

© Copyright IBM Corporation 2014

Figure 8-19. Retrieve the client WSDL from the service

WE601 / ZE6011.1

Notes:

The original WSDL used in defining the WS Proxy contains a location that is no longer correct for the web service that is proxied by this service.

When you append ?wsdl to the URL that the client uses to access the web service on the appliance, the appliance returns a WSDL with:

- The appliance IP address
- The service port
- The URI that the client uses to access the web service

These values are not the values in the original WSDL.

Modifying the location in the client WSDL

- The WSDL retrieved from the web service proxy by using `?wsdl` by default places the IP address and port for the appliance in the “location”

```
<wsdl:port binding="..." name="AddressSearch">
  <address
    location="http://192.168.10.41:6999/EastAddressSearch" />
</wsdl:port>
```

- You can specify a different host name or port to place in the WSDL
 - Clear **Use Local** to enter your own values
 - Now retrieved by `?wsdl`

```
<wsdl:port binding="..." name="AddressSearch">
  <address
    location="http://myDPappliance.com:6999/EastAddressSearch
    " /> </wsdl:port>
```

© Copyright IBM Corporation 2014

Figure 8-20. Modifying the location in the client WSDL

WE601 / ZE6011.1

Notes:

The WSDL retrieved by `?wsdl` contains the IP address and port of the appliance and web service proxy service.

By clearing the **Use Local** check box, you can explicitly specify the host name, port, and URI that are included in the retrieved WSDL.

This feature becomes especially useful if you have a load balancer that fronts the appliance. By using the explicit approach, you can specify the load balancer details in the retrieved WSDL so the clients send their requests to the correct host name, port, or URI on the load balancer. The load balancer must be configured to forward requests to the appliance host name, port, or URI.



Step 7: Configuring web service proxy policy (optional)

WSDL files SLM Policy Services Policy SLA Policy Details Proxy Settings Advanced Proxy Settings Headers/Para

Web Service Proxy Name [up] AddressSearchProxy *

Apply Cancel Delete Refresh Export | View Log | View Status | View Show Probe | Validate Conform

Policy

Use this pane to define the processing policies to implement at various levels in the WSDL hierarchy.

WSDL Policy Tree Representation

Show portType and binding nodes

Define the policies to apply in the tree.

proxy: AddressSearchProxy

- WS-Policy (default) WS-I Conformance (none) Priority Normal
 - Processing Rules** Request rules:1 ,Response rules:1
- +| WSAR: westAddressSearch.wsdl ✓
- +| WS-Policy (default) WS-I Conformance (none) Priority Normal
 - Processing Rules**
- +| WSAR: eastAddressSearch.wsdl ✓
- +| WS-Policy (default) WS-I Conformance (none) Priority Normal
 - Processing Rules**

- Click the **Policy** tab to view web service proxy policy
 - The web service proxy policy consists of rules
 - Rules can be executed at a specific level per request or response
- Click **Processing Rules** to open policy editor for rules at that level

© Copyright IBM Corporation 2014

Figure 8-21. Step 7: Configuring web service proxy policy (optional)

WE601 / ZE6011.1

Notes:

The **Policy** tab shows the rules that are defined at the various levels within any WSDLs defined for this service.

Default proxy-level rules are provided.

Clicking **Processing Rules** opens the policy editor in a lower section of the page.

A toggle is available to show the portType and binding nodes in the WSDL levels. The default is to not display them in the policy tree.

The **more** links display more help text on the page.

WebSphere Education

Configure web service proxy policy rule

The screenshot shows the WebSphere DataPower Policy Editor interface. At the top, there's a navigation bar with 'WebSphere Education' and the IBM logo. Below it, the main title is 'Configure web service proxy policy rule'. The interface is divided into several sections:

- Policy Tree:** Shows a tree structure with nodes like 'proxy: addressSearchProxy', 'WS-Policy (default)', 'WS-I Conformance (none)', 'Priority Normal', 'Processing Rules (Request rules:1, Response rules:1)', and 'wsdl: EastAddressSearch.wsdl'.
- Policy Configuration:** A large panel where rules are defined. It includes fields for 'Rule Name' (set to 'addressSearchProxy_default_request-rule') and 'Rule Direction' (set to 'Client to Server'). Below these are buttons for 'New Rule' and 'Delete Rule'. A note says 'Create rule: Click New, drag action icons onto line.' and 'Edit rule: Click on rule, double-click on action'.
- Action Toolbar:** A horizontal bar with icons for Filter, Sign, Verify, Validate, Encrypt, Decrypt, Transform, Route, AAA, Results, SLM, and Advanced.
- Flow Diagram:** A sequence diagram showing a 'CLIENT' icon connected by a line with a diamond-shaped connector to a square connector, which then connects to another square connector.
- Configured Rules Table:** A table listing configured rules:

Order	Rule Name	Direction	Actions
1	addressSearchProxy_default_request-rule	Client to Server	Filter, SLM, Results (with delete rule)
2	addressSearchProxy_default_response-rule	Server to Client	Route, SLM (with delete rule)
- Notes:** Two callout boxes provide additional information:
 - The first box, titled 'Processing Rules', explains how to view rule configurations at different levels and expand/collapse WSDL levels.
 - The second box, titled 'Default Rules', describes the two default rules: a request rule enabling SLM and a reply rule passing messages through.
- Copyright Information:** © Copyright IBM Corporation 2014.

Figure 8-22. Configure web service proxy policy rule

WE601 / ZE6011.1

Notes:

You can add, view, or modify rules by selecting **Processing Rules** at the intended level.

To show or hide the levels of the WSDLs, click the plus sign (+) or the minus sign (-).

The default proxy-level rule contains two actions, an **SLM** action and a **Results** action. The **SLM** action is a checkpoint event that calls the web service proxy SLM policy. You can verify the **SLM** action by double-clicking it and noting the SLM policy name. Click the **SLM Policy** tab to verify that the proxy name listed in the page is the same as the **SLM** action.



Adding a rule

1. Add a rule to **findByName** operation by clicking **Processing Rules**
2. Click **New Rule** in policy editor (default name can be typed over)
3. When finished, click **Apply** at the web service proxy (page) level

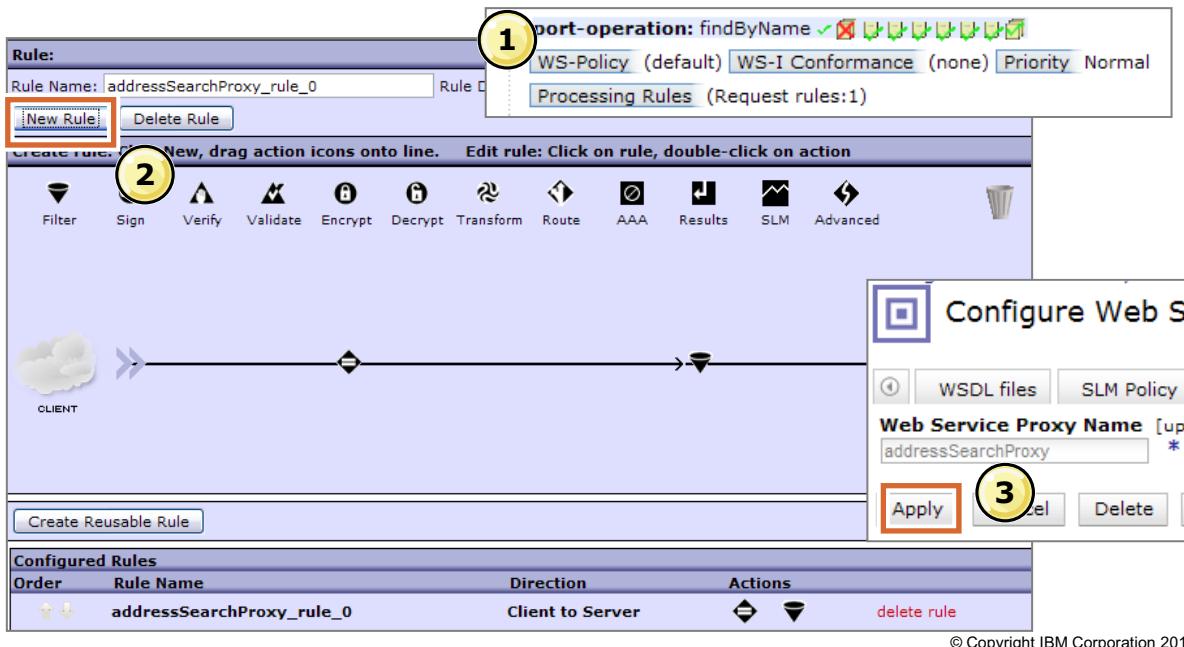


Figure 8-23. Adding a rule

WE601 / ZE6011.1

Notes:

The number and type of rules that are defined at that level in the WSDL are displayed next to the **Processing Rules** button.

You create and configure the actions in the rule as you normally would.

All rules that are configured at this level are in the **Configured Rules** section of the policy editor.

Clicking **Apply** at the page (service) level commits this rule to the policy.



Default validation (user policies)

- By default, each level of the proxy (proxy, WSDL, service, port, operation) defines a user policy to:
 - Schema validates messages (against schema in WSDL): request, response, and fault
 - Schema validates SOAP headers (against schema in WSDL)
 - Enable or disable a component
 - Publish a component in the proxy WSDL file
 - Use WS-Addressing
 - Use WS-ReliableMessaging
 - Accept MTOM/XOP messages during validation

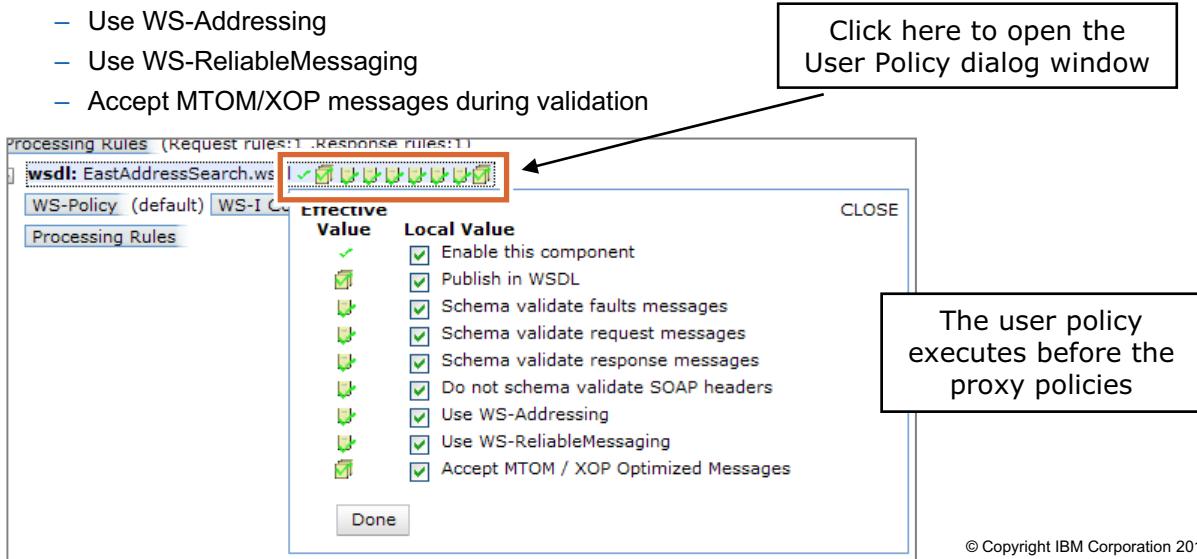


Figure 8-24. Default validation (user policies)

WE601 / ZE6011.1

Notes:

Click any of the icons at each level to view the user policy pop-up dialog box.

The first check mark enables the component. Each option that is shown in the pop-up dialog box maps to an icon with a green check mark or red X.

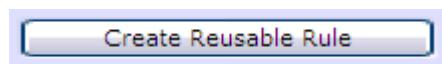
Each policy level contains a user policy that can be enabled or disabled.

The web service proxy policy and user policy are separate from each other; the user policy is executed before the web service proxy policy.



Create reusable rule

- The rule configuration area allows you to select a set of actions to be invoked as a reusable rule
 - Click **Create Reusable Rule**.
 - Draw a box around the actions to include in the reusable rule.
 - Click **Apply**. A gray rectangle appears around the reusable rule in the rule configuration area and generates a new rule name for the new reusable rule.
 - Use the **Advanced – Call Processing Rule** action to reuse the rule.



© Copyright IBM Corporation 2014

Figure 8-25. Create reusable rule

WE601 / ZE6011.1

Notes:

Reusable rules are useful for applying a common set of actions at many levels of the web service proxy. Additional actions can be added before or after the reusable rules. It allows you to more easily manage a set of actions that repeat across many levels of the web service proxy.

Reusable rules can be defined in the other service type processing policies, such as an XML firewall policy.



Advanced web service proxy configuration

More options and tabs available for advanced web services proxy configuration:

- Proxy settings
 - Web service proxy type
 - Security settings (AAA, cryptographic key)
 - SOAP Action Policy
 - XML Manager
- Advanced proxy settings
 - HTTP connection settings
- Headers, parameters
 - Adds or removes HTTP headers and passes style sheet parameters
- WS-Addressing
 - Indicates support for WS-Addressing for front-end or back-end server
- WS-ReliableMessaging
 - Indicates whether to use WS-ReliableMessaging
- XML threat protection
 - Provides protection against XML threats

© Copyright IBM Corporation 2014

Figure 8-26. Advanced web service proxy configuration

WE601 / ZE6011.1

Notes:

In this presentation, only the proxy settings are examined, in later slides. See the various service guides for information about the settings that are contained in the **Advanced Proxy Settings**, **Headers/Params**, **WS-Addressing**, and **WS-ReliableMessaging** tabs.

The XML threat protection settings are explained in the XML threat protection presentation.

WS-Policy

- WS-Policy is a specification that defines metadata to enable interoperability between web service consumers and web service providers
- The WS-Policy specifications enable organizations to automate their service governance models by creating a concrete instance of web service governance
- Behaviors:
 - Parse WSDL with policy elements already included in the WSDL and recognize standardized policy “domains” (WS-Security Policy, WS-ReliableMessaging Policy)
 - DataPower supports retrieving WSDL by using WebSphere Service Registry and Repository queries
 - DataPower supports retrieving WSDL by using a UDDI interface



© Copyright IBM Corporation 2014

Figure 8-27. WS-Policy

WE601 / ZE6011.1

Notes:

WS-Policy is used to assert policies on security, quality of service (QoS), required security tokens, privacy, and other items. A web service can stipulate what it can provide, and a consumer can stipulate its requirements.



Conformance policy

- Defines which profiles to use to validate whether received messages are in conformance to the selected interoperability profiles
- When a client sends nonconforming requests for a conforming back-end server:
 - The conformance policy can be used to fix nonconforming requests during message processing
- For signed and encrypted nonconforming data:
 - The cryptographic protection must be removed before and after conformance correction
- It can be added to a WS-Proxy in the Policy editor



© Copyright IBM Corporation 2014

Figure 8-28. Conformance policy

WE601 / ZE6011.1

Notes:

Supported profiles:

- WS-I Basic Profile Version 1.0
- WS-I Basic Profile Version 1.1
- WS-I Attachments Profile Version 1.0
- WS-I Basic Security Profile Version 1.0

Any conformance correction must be coded in a style sheet; the firmware does not automatically provide it.



Conformance policy object

1. **Profiles:** Profiles against which conformance is checked
2. **Ignored Requirements:** Conformance requirements to ignore
3. **Corrective Stylesheets:** XSL sheets are invoked after conformance analysis
4. **Record Report:** Degree of nonconformance that causes a report to be recorded
5. **Reject non-conforming messages:** Degree of nonconformance that causes a rejected message
6. **Other Options:** Deliver conformance analysis report as an action result

The screenshot shows two instances of the 'Operation Conformance Policy' dialog box. The left instance is titled 'Basic' and the right one is titled 'Advanced'. Both dialogs have tabs for 'Profiles', 'Record Report', 'Ignored Requirements', 'Corrective Stylesheets', and 'Other Options'. The 'Profiles' tab is selected in both. The 'Record Report' tab is selected in the right dialog. The 'Ignored Requirements' tab is selected in the left dialog. The 'Corrective Stylesheets' tab is selected in the right dialog. The 'Other Options' tab is selected in the left dialog. Numbered circles (1 through 6) are overlaid on the dialog boxes to correspond with the list items above.

Section	Description
1	Profiles: WS-I BP 1.0, WS-I BP 1.1, WS-I AP 1.0, WS-I BSP 1.0
2	Ignored Requirements: (empty)
3	Record Report: Never (radio button selected)
4	Record Report: Failure, Warning, Always (radio buttons available)
5	Reject non-conforming messages: Never (radio button selected)
6	Other Options: Use analysis as result (radio buttons: on/off)

© Copyright IBM Corporation 2014

Figure 8-29. Conformance policy object

WE601 / ZE6011.1

Notes:

Ignored requirements are entered as a text string. For example, `BSP1.0:R4221` would ignore requirement R4221 in the Basic Security Profile V1.0.

Record report options include:

- **Never:** Never record reports
- **Failure:** Record reports with conformance failures
- **Warning:** Record reports with conformance warnings
- **Always:** Record reports for all outcomes

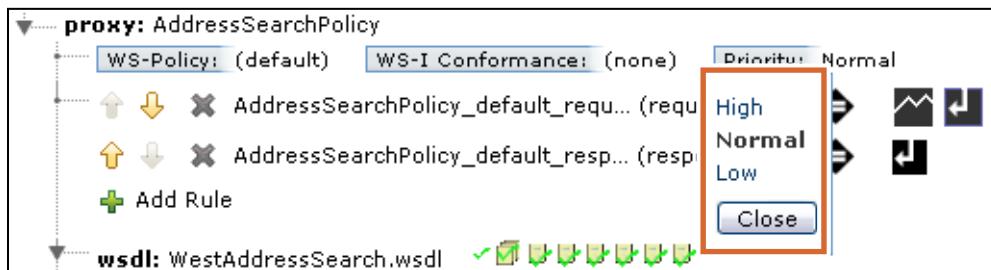
Reject nonconforming messages include:

- **Never:** Never reject messages
- **Failure:** Reject messages with conformance failures
- **Warning:** Reject messages with conformance warnings or failures



Service priority

- The policy editor has a **Priority** field
 - Sets priority for resource allocation and scheduling



- The different levels of priority are:
 - High: Receives higher than normal priority
 - Low: Receives lower than normal priority
 - Normal: (default) Receives normal priority

© Copyright IBM Corporation 2014

Figure 8-30. Service priority

WE601 / ZE6011.1

Notes:

The priority has no effect until the appliance encounters a resource constraint.



Proxy settings (1 of 4)

- Click the **Proxy Settings** tab to view the proxy settings
 - Many options have default values
- Type**
 - Static Backend:** Web service proxy forwards to single back-end server
 - Dynamic Backend:** Web service proxy determines back-end server during service processing
 - Static from WSDL (default):** Service section in the WSDL file determines the back-end server

The screenshot shows the 'Proxy Settings' tab selected in the top navigation bar. The 'Web Service Proxy Name' is set to 'AddressSearchProxy'. In the 'General Configuration' section, the 'Comments' field is empty. The 'Type' section is highlighted with a red box, showing three options: 'Dynamic Backend', 'Static Backend', and 'Static from WSDL', with 'Static from WSDL' selected. Other sections visible include 'XML Manager' (set to 'default') and 'Authorization AAA Policy' (set to '(none)').

Figure 8-31. Proxy settings (1 of 4)

WE601 / ZE6011.1

Notes:

When the web service proxy receives requests from a client, it forwards them to a back-end server for a service request.

The **Type** section specifies how that back-end server is determined. A back-end server is identified with a URL and port. The default option is **Static from WSDL**, which uses the WSDL file to determine the back-end server. The **Dynamic Backend** option determines the back-end server during document processing, and the **Static Backend** option always forwards to a single back-end server.

If the **Static Backend** type is selected, the page reloads, and you are supplied fields in which you can enter the back-end information. Several URL Helper buttons (WebSphere MQ, TibcoEMS, WebSphereJMS, and IMS Connect) are presented to help build the back-end URL.



Proxy settings (2 of 4)

- Decrypt Key
 - Selects a cryptographic key object to decrypt the message payload
- EncryptedKeySHA1 Cache Lifetime
 - Cache Lifetime for the decrypted generated key
- Preserve EncryptedKey Chain
 - Whether to output the element chain that is used to decrypt
- Decrypt with Key from EncryptedData
 - Enable decrypt action to attempt decryption with the key that is inside the EncryptedData element
- Client Principal
 - The client principal name when decrypt is required
 - Used when the encryption uses a Kerberos session key or uses a key that is derived from the session key
- Server Principal
 - The server principal name when decrypt is required
 - Used when the encryption uses a Kerberos session key or uses a key that is derived from the session key

© Copyright IBM Corporation 2014

Figure 8-32. Proxy settings (2 of 4)

WE601 / ZE6011.1

Notes:

The message payload refers to the message body.

Encrypting a message introduces new elements into the SOAP message that would cause automatic message validation to fail, since a typical schema validation does not check for these elements.

An example SOAP message with encrypted payload might look like:

```
<SOAP:Body>
<EncryptedData ...>
```

Using a cryptographic key ensures that a message can pass automatic validation by decrypting the message payload before validation. The entire message must be encrypted, not only fields within the message.

EncryptedKeySHA1 Cache Lifetime is the cache lifetime for the decrypted generated key. Setting the value to 0 means that the decrypted generated key is not cached.

Preserve EncryptedKey Chain, if it is on, outputs the chain of elements that the decrypted Encrypted Data uses, such as xenc:EncryptedKey, wsc:DerivedKeyToken. Otherwise, all

xenc:EncryptedKey elements are removed after decryption, and even some of the encrypted data might not be decrypted successfully.

Decrypt with Key from EncryptedData: In scenarios in which the key is inside an EncryptedData element (such as "encrypted SAML Assertion"), the decrypt action cannot locate the key to decrypt the corresponding EncryptedData elements. Select **on** to enable the decrypt action to attempt decryption with the key that is inside the EncryptedData element.

The **Client Principal** field contains the full name of the client principal when the web service proxy must automatically decrypt encrypted requests. Use this property when the encryption uses a Kerberos session key, or a key that was derived from the session key.

In a similar fashion, the **Server Principal** field specifies the full name of the server principal when the web service proxy must automatically decrypt encrypted responses.



Proxy settings (3 of 4)

- **Kerberos Keytab**
 - Select the Kerberos keytab file that contains the principals
- **SOAP Action Policy:** Validates messages that contain a SOAPAction HTTP header
 - **Lax:** Validates messages with empty SOAPAction HTTP header or empty string within SOAPAction HTTP header
 - **Off:** SOAPAction HTTP header is ignored
 - **Strict:** Message must contain exact match of SOAPAction header that is specified in WSDL file
- **Monitor via Web Services Management Agent:** Allows for autonomous monitoring of this service by a WS-Management agent

The screenshot shows a configuration panel for proxy settings. It includes fields for selecting a Kerberos Keytab (with '(none)' selected), choosing a SOAP Action Policy (with 'Lax' selected), and enabling monitoring via a Web Services Management Agent (with 'on' selected).

© Copyright IBM Corporation 2014

Figure 8-33. Proxy settings (3 of 4)

WE601 / ZE6011.1

Notes:

Select the Kerberos Keytab object that contains the principals for the **Kerberos Keytab** list. The web service proxy uses these principals to automatically decrypt encrypted requests and responses.

The WSDL file for a service defines the value that a SOAPAction header must contain for a SOAP request. The SOAPAction header is defined in the HTTP header, not the SOAP header.

The **SOAP Action Policy** setting specifies how to validate messages with a SOAPAction HTTP header.

A WS-Management agent can monitor the web service proxy without any monitors that are defined on the Monitors tab.



Proxy settings (4 of 4)

- **XML Manager:** Assigns an XML manager to the web service proxy
- **Authorization AAA Policy:** Selects or creates a AAA policy to apply to all service endpoints configured for this web service proxy
 - AAA policy can also be applied at a fine-grained level in the **Policy** tab

The screenshot shows the 'Proxy Settings' tab of a web-based configuration interface. At the top, there are tabs for WSDL files, SLM Policy, Services, Policy, SLA Policy Details, Proxy Settings (which is selected), and Advanced Proxy. Below the tabs, the 'Web Service Proxy Name' is set to 'AddressSearchProxy'. There are buttons for Apply, Cancel, Delete, and Refresh. On the right, there are links for Export, View Log, and Show. The main area is divided into sections: 'General Configuration' (Comments, Type: Static from WSDL selected), 'XML Manager' (set to default), and 'Authorization AAA Policy' (set to none). The 'Authorization AAA Policy' section is highlighted with a red box.

© Copyright IBM Corporation 2014

Figure 8-34. Proxy settings (4 of 4)

WE601 / ZE6011.1

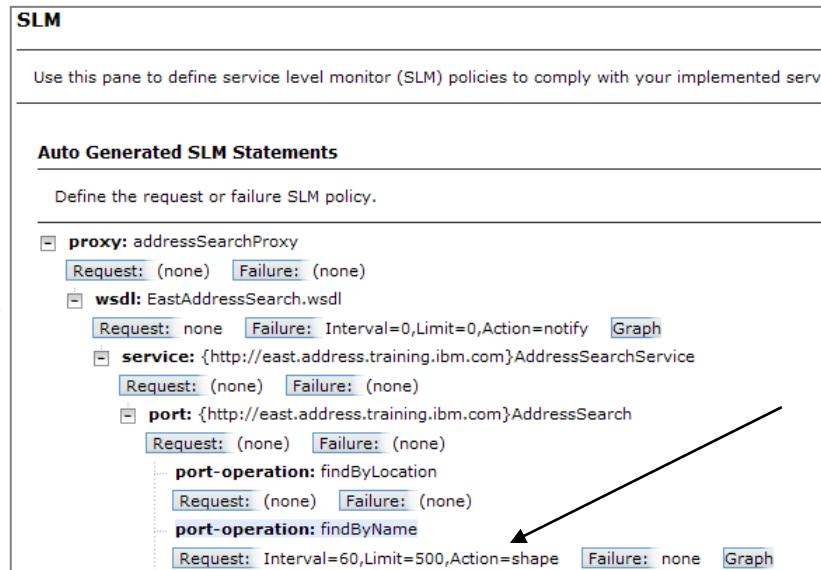
Notes:

The Authorization AAA policy specifies how incoming messages are authenticated and authorized. The last A is for Audit.

The proxy AAA policy is applied for all service endpoints within the proxy.

Web service proxy: SLM Policy tab

- Click the **SLM Policy** tab to monitor requests that enter the web service proxy
 - Provides monitoring at a fine-grained level
 - Controls traffic entering the web service proxy by using the **Throttle** and **Shape** action
 - Can view graph to see results of the traffic



© Copyright IBM Corporation 2014

Figure 8-35. Web service proxy: SLM Policy tab

WE601 / ZE6011.1

Notes:

Under **Request**, you can count the number of transactions that occur with a specific **interval** (in seconds). If the transaction **limit** is exceeded, you can specify an **action** to:

- Notify:** Generate a log message when the transaction limit is exceeded.
- Throttle:** Additional transactions above the limit are rejected, and log messages are generated.
- Shape:** The first 2500 transactions in excess of the maximum transaction rate are queued for later transmission, and subsequent transactions in excess of the 2500 limit are rejected. Log messages are generated.

Under **Failure**, you can specify the same information as **Request**, except that these settings apply to error messages.



WSDL cache policy

- Create a WSDL cache policy to update the WSDL proxy with changes from underlying WSDL file
 - Scheduled poll of underlying WSDL
 - If changes are detected, then the proxy WSDL is automatically updated
- Option is available only from **Objects** view:
 - Click **Objects > Service Configuration > Web Service Proxy**; then click an existing web service proxy
 - Use the arrows at the top to select the **WSDL Cache Policy** tab

The screenshot shows the 'Configure Web Service Proxy' interface. At the top, there are tabs: 'Header Suppression', 'Stylesheet Parameter', 'Probe Triggers', 'WSDL Cache Policy' (which is highlighted with a red box), 'WSDL File', and 'User policy'. Below the tabs, it says 'Web Service Proxy: AddressSearchProxy [up]'. There are buttons for 'Apply', 'Cancel', 'Delete', and 'Undo'. To the right, there are links: 'Export', 'View Log', 'View Status', 'Show Probe', 'Validate Conformance', 'Help', 'Quiesce', and 'Unquiesce'. A copyright notice at the bottom right reads '© Copyright IBM Corporation 2014'.

URL Match expression	TTL
http://myWSDLserver.com/AddressSearch/*	900

Add

Figure 8-36. WSDL cache policy

WE601 / ZE6011.1

Notes:

Click **Add** to create a WSDL cache policy.

The URL match expression is used to match the URL of the WSDL file (that is, its location).

Time to Live (TTL) is expressed in seconds. It specifies how long the current WSDL file exists until it is automatically refreshed when a corresponding URL match expression is matched.

The WSDL file might exist on an external server.



Troubleshooting a web service proxy

- Check active web service operations by using **Status > Web Service > Web Services Operations**

WSProxy	Index	Interface	Port	URL	SOAP Action	SOAP Body	Status	
AddressSearchProxy	0	172.16.78.44	6975	/EastAddressSearch		{http://east.address.training.ibm.com}retrieveAll	Registered	http://
AddressSearchProxy	1	172.16.78.44	6975	/EastAddressSearch		{http://east.address.training.ibm.com}findByLocation	Registered	http://
AddressSearchProxy	2	172.16.78.44	6975	/EastAddressSearch		{http://east.address.training.ibm.com}findByName	Registered	http://
AddressSearchProxy	3	172.16.78.44	6975	/WestAddressSearch		{http://west.address.training.ibm.com}retrieveAll	Registered	http://
AddressSearchProxy	4	172.16.78.44	6975	/WestAddressSearch		{http://west.address.training.ibm.com}findByLocation	Registered	http://

- List of validation checks for web service proxy

- Request

- Web service proxy active and listening on port
 - Verify that client submitted correct URI
 - Web service proxy received request (system log, probe)
 - SOAPAction header must agree with operation name in SOAP body
 - Passed automatic schema validation (user policy)
 - Back-end service active and available (system log)
 - Request that is transmitted to correct back-end URL (system log)

- Response

- Response that is received from back-end service (system log)
 - Response passed automatic schema validation (user policy)
 - Response that transmitted completely to client (system log, probe)

© Copyright IBM Corporation 2014

Figure 8-37. Troubleshooting a web service proxy

WE601 / ZE6011.1

Notes:

The default error messages that the web service proxy returns are intentionally vague so that no clues are provided to an intruder who is trying to compromise the system. For example:

```
HTTP/1.0 500 Error
X-Backside-Transport: FAIL
Connection: close
Content-Type: text/xml
<?xml version='1.0' ?>
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/envelope/'>
<env:Body>
<env:Fault>
<faultcode>General</faultcode>
<faultstring>Internal Error</faultstring>
</env:Fault>
</env:Body>
</env:Envelope>
```



Unit summary

Having completed this unit, you should be able to:

- Describe the web service proxy architecture
- List and explain the configuration steps that are needed to create a web service proxy
- Create and configure a web service proxy policy at various levels of the WSDL file

© Copyright IBM Corporation 2014

Figure 8-38. Unit summary

WE601 / ZE6011.1

Notes:



Checkpoint questions

1. True or False: A web service proxy and SLM policy can be defined at a fine-grained level.
2. Which of the following levels can be configured with a web service proxy policy?
 - A. Proxy
 - B. Message
 - C. Service
 - D. Port
3. True or False: A WSDL must be uploaded onto the appliance when creating a web service proxy.
4. List the three options under the SOAPAction policy:
 - A. lax: This option validates messages with an empty SOAPAction HTTP header or an empty string within the SOAPAction HTTP header.
 - B. strict: The message must contain an exact match of the SOAPAction header that is provided in the WSDL file.
 - C. off: The SOAPAction HTTP header is ignored.
 - D. lazy: The SOAPAction allows all messages through.

© Copyright IBM Corporation 2014

Figure 8-39. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.



Checkpoint answers

- 1. True.**
- 2. A, C, and D.**
- 3. False.** A WSDL can be retrieved from a subscription.
- 4. A, B, and C.**

© Copyright IBM Corporation 2014

Figure 8-40. Checkpoint answers

WE601 / ZE6011.1

Notes:



Exercise 6



Configuring a web service proxy

© Copyright IBM Corporation 2014

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

9.0

Figure 8-41. Exercise 6

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Configure a web service proxy to virtualize an existing set of web services
- Create a policy within the web service proxy

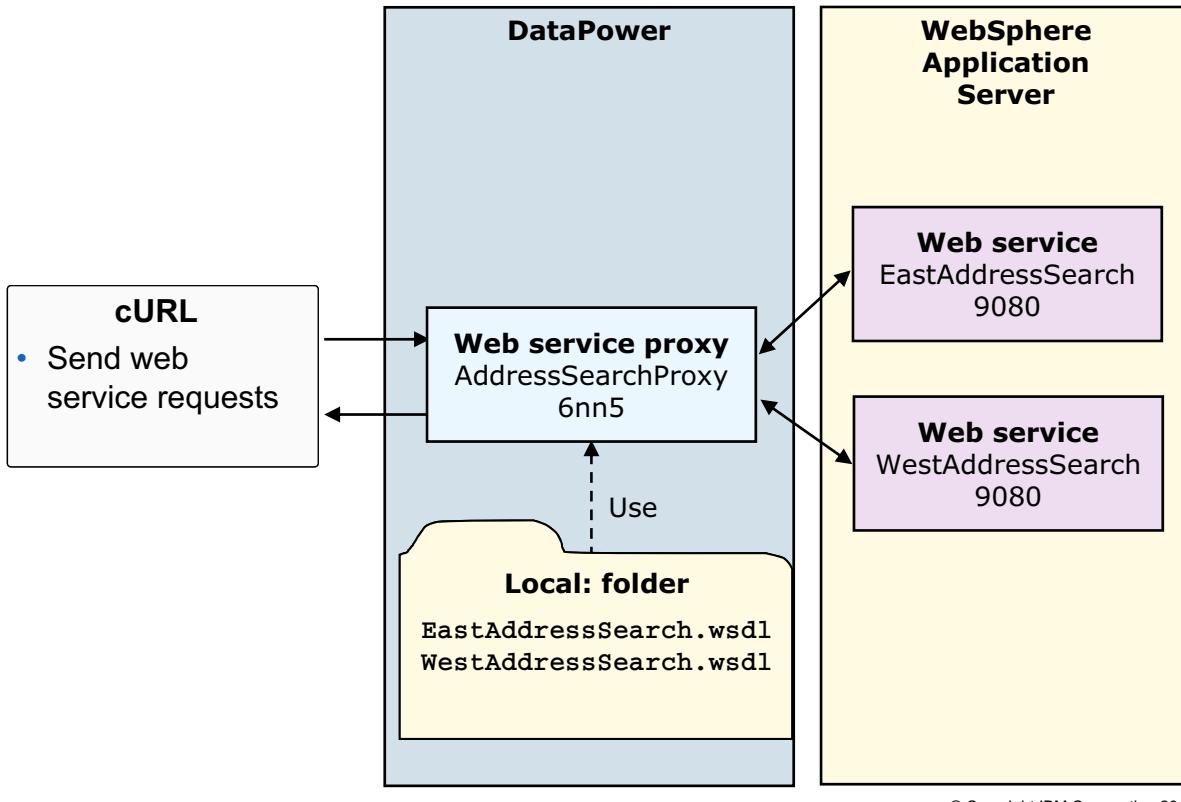
© Copyright IBM Corporation 2014

Figure 8-42. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview



© Copyright IBM Corporation 2014

Figure 8-43. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 9. Service level monitoring

What this unit is about

Service level management is the monitoring and management of message traffic that concerns quality of service (QoS) indicators such as throughput, response time, and availability. Within DataPower, service level monitoring (SLM) is a tool that helps support those activities. This unit defines the DataPower version of SLM and describes various ways to configure it.

What you should be able to do

After completing this unit, you should be able to:

- Identify the SLM functions that the WebSphere DataPower appliance provides
- Create an SLM policy object by using the WebGUI
- Create a custom SLM Statement
- Use the SLM Policy tab in the web service proxy to create a basic SLM policy

How you will check your progress

- Checkpoint
- Exercise 14. Implementing an SLM monitor in a web service proxy



Unit objectives

After completing this unit, you should be able to:

- Identify the SLM functions that the WebSphere DataPower appliance provides
- Create an SLM policy object by using the WebGUI
- Create a custom SLM statement
- Use the SLM Policy tab in the web service proxy to create a basic SLM policy

© Copyright IBM Corporation 2014

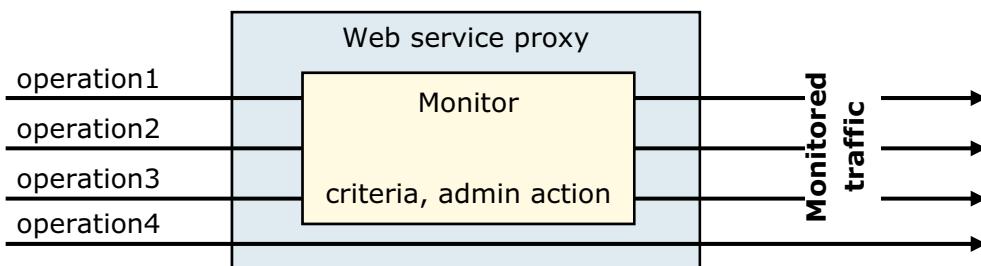
Figure 9-1. Unit objectives

WE601 / ZE6011.1

Notes:

Service level monitoring (SLM) in DataPower

- A concept of monitoring message traffic regarding a predefined set of criteria, and possibly managing the throughput
- Criteria can be traffic rate, client ID, target resource, time, and others
 - Might be related to a service level agreement (SLA) with a client
- If thresholds are reached, “administrative” actions are taken
 - Log, buffer, reject
- Monitoring and actions are applied to selected messages, within a web service proxy or multi-protocol gateway



© Copyright IBM Corporation 2014

Figure 9-2. Service level monitoring (SLM) in DataPower

WE601 / ZE6011.1

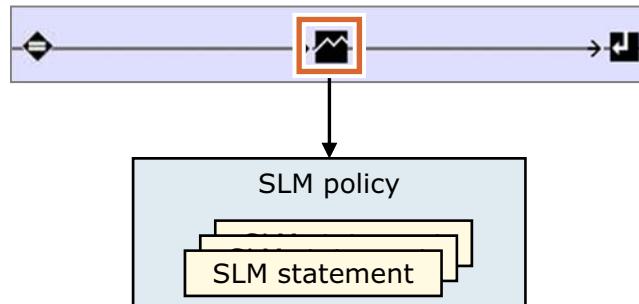
Notes:

Service level monitoring (SLM) within DataPower is a subset of service level management at the enterprise level. Service level management means monitoring and managing the availability and quality of the relevant services that are being provided. In this context, it generally implies the availability and performance of the associated web services.

There might be a service level agreement (SLA) between the client and the service provider. DataPower SLM is a tool to help deliver on the agreement. SLM is available for web service proxies and multi-protocol gateways.

The pieces of SLM (1 of 2)

- The presence of an **SLM processing action** in a rule enables the monitor



- The SLM action specifies an SLM policy object
- The **SLM policy** consists of one or more SLM statements
- An **SLM statement** defines the measurement criteria and administrative action
- A message is processed through the statements in order
 - If any thresholds are exceeded, the specified administrative actions are taken

© Copyright IBM Corporation 2014

Figure 9-3. The pieces of SLM (1 of 2)

WE601 / ZE6011.1

Notes:

SLMs differ from message monitors in that they are not directly associated with a service. Rather, the SLM is implemented by using an SLM policy, which, in turn, is associated with the service.

Statements that measure execution durations are configured for messages that pass through the appliance during a configured measurement window and that also match a set of selection criteria.

Approaches to define SLM policies

1. Add an SLM action to a request rule
 - An SLM policy is specified in the action
 - Applies to both web service proxies and multi-protocol gateways
 - An SLM action and SLM policy are auto-generated for a web service proxy
2. Specify SLM criteria to the levels of the WSDL
 - Applies only to a web service proxy
 - Auto-generates the SLM statements
3. Attach WS-MediationPolicy policies to the WSDL
 - Applies only to a web service proxy
 - Auto-generates the SLM statements

© Copyright IBM Corporation 2014

Figure 9-4. Approaches to define SLM policies

WE601 / ZE6011.1

Notes:

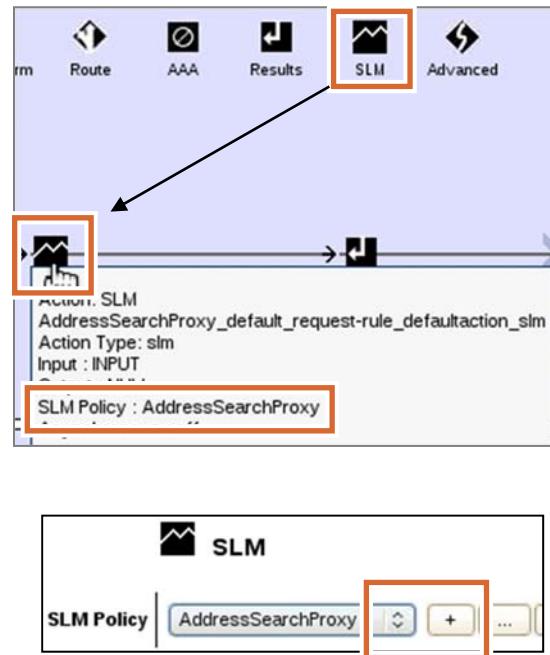
The first two approaches have been supported for many years.

WS-MediationPolicy is an IBM proposed web service standard for quality of service (QoS) specifications. WS-MediationPolicy statements can be a policy attachment for a WSDL, and be stored in WebSphere Service Registry and Repository. WS-MediationPolicy statements auto-generate SLM-related processing rules. These rules execute before the developer-specified rules within the web service proxy. WS-MediationPolicy is not explained in any detail in this course.



Approach 1: Add an SLM action to a request rule

- An **SLM** action identifies an SLM policy for execution
 - Web service proxy: the **SLM** action has its own icon
 - Multi-protocol gateway: the **SLM** action is selected from the **Advanced** icon
- When configuring the SLM action, you must specify an existing SLM policy, or create one
- Without an SLM action and SLM policy, no monitoring occurs



© Copyright IBM Corporation 2014

Figure 9-5. Approach 1: Add an SLM action to a request rule

WE601 / ZE6011.1

Notes:

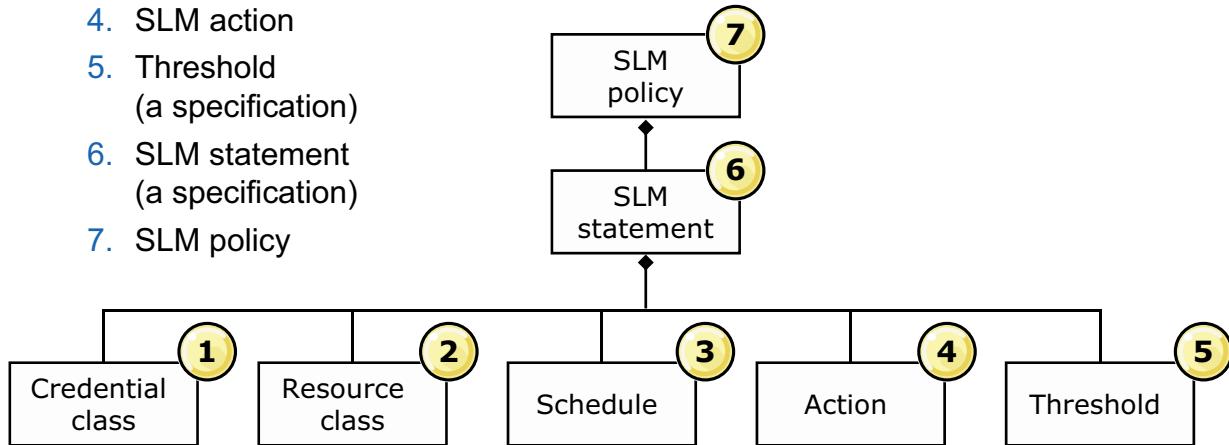
The **SLM** action screen capture is from a web service proxy.

Compare this action with the SLM action object, which is explained later.

The pieces of SLM (2 of 2)

- An SLM policy requires the following objects, if they affect the policy:

1. SLM credential class
2. SLM resource class
3. SLM schedule
4. SLM action
5. Threshold
(a specification)
6. SLM statement
(a specification)
7. SLM policy



© Copyright IBM Corporation 2014

Figure 9-6. The pieces of SLM (2 of 2)

WE601 / ZE6011.1

Notes:

A threshold and an SLM statement are not separate objects. They are specifications. A threshold is a specification within an SLM statement. An SLM statement is a specification within an SLM policy object.

Depending on what criteria are needed for a specific SLM statement, only certain SLM objects are needed. For example, if you are monitoring only the target resource, then the SLM credential and SLM schedule objects are not needed.

The SLM credential class

- Defines which clients are subject to an SLM statement
 - Click **Objects > Monitoring > SLM Credential Class** to define individually
- A credential class consists of:
 - Credential Type:** Specifies what to use for a credential
 - Match Type:** Specifies how a successful match is determined
 - Credential Value:** (optional) Is used to specify exact values when match type is **exact**
 - Request header** (not shown): Name of a header when the credential type is request header

Configure SLM Credential Class

Main

SLM Credential Class

Name *

Administrative State enabled disabled

Comments

Credential Type *

Match Type *

© Copyright IBM Corporation 2014

Figure 9-7. The SLM credential class

WE601 / ZE6011.1

Notes:

An SLM credential class is used to select messages for inclusion in the SLM policy statement. A credential class obtains a credential (that is, a user identity) from a message.

The **Credential Type** determines the method that is used to obtain the identity. Examples are **Client IP**, **Mapped Credential**, **Extracted Identity**, and **IP from Header**. It can also be a custom style sheet. If **Mapped Credential** or **Extracted Identity** is used, a previous AAA policy must exist to provide these values.

The **Match Type** setting determines the method that is used to match the credential that is obtained. For a Match Type of **Per Extracted Value**, all configured SLM policies apply to each extracted value. A list of all unique values of the specified type are extracted and reported. For a Type of **Exact**, an SLM policy applies to only values that match. Another field appears that lists the accepted values. For the Type of **Regular Expression**, an SLM policy applies to only values that match. Instead of a list of specific values to match, a field appears that lists PCRE-style expressions to determine if a presented value matches.

The **Credential Value** setting determines specific values when it is an exact match or regular expression type.

If a match is made, the message is included in the set of messages that the SLM policy affects.

WebSphere Education

The SLM resource class

- Identifies a set of resources subject to an SLM policy statement
 - Click **Objects > Monitoring > SLM Resource Class** to define individually

- A resource class consists of:
 - Resource Type:** Specifies a method used to identify the resource
 - Match Type:** Specifies how a successful match is determined
 - Resource Value:** Values to match

Configure SLM Resource Class

Main

SLM Resource Class

Apply Cancel

Name	*
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	
Resource Type	Mapped Resource
Match Type	Per Extracted Value *

© Copyright IBM Corporation 2014

Figure 9-8. The SLM resource class

WE601 / ZE6011.1

Notes:

An SLM resource class is used to select messages for inclusion in the SLM policy statement. A resource class obtains a resource identifier from a message.

The **Resource Type** determines the method that is used to obtain the resource. Examples are **Mapped Resource**, **Destination URL**, **WSDL Operation**, and **XPath Expression**. The list is extensive; consult the product documentation for a complete list. If **Mapped Resource** is used, a previous AAA policy must exist to provide these values.

The **Match Type** setting determines the method that is used to match the resource that is obtained, which is the same as for the credential class.

If a match is made, the message is included in the set of messages that the SLM policy affects.



SLM resource class example

- SLM policy applies to incoming message that contains the following attributes:
 - WSDL operation: **AddressSearch/findByName**
 - Namespace: **http://east.address.training.ibm.com**

SLM Resource Class : rsrc-AddressSearchProxy-wsdl-operation22 [up]

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

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Resource Type	WSDL Operation *
Match Type	Exact *
Resource Value	{http://east.address.training.ibm.com}AddressSearch/findByName ↑ ↓ X

© Copyright IBM Corporation 2014

Figure 9-9. SLM resource class example

WE601 / ZE6011.1

Notes:

Here are some Resource Type examples:

- **WSDL**: Specifies that a WSDL file defines membership in this resource class
- **WSDL Service**: Specifies that WSDL service names define membership in this resource class
- **WSDL Operation**: Specifies that WSDL operations define membership in this resource class
- **Destination URL**: Specifies the URL output to the destination server, which might not be identical to the URL that the client requests

The SLM schedule

- Specifies a time period during which the associated SLM statement is enforced
 - Click Objects > Monitoring > SLM Schedule to define individually
- Schedule elements
 - Week Days**
 - Start Time**
 - Duration**
 - Start Date**
 - End Date**
 - Time Zone**

© Copyright IBM Corporation 2014

Figure 9-10. The SLM schedule

WE601 / ZE6011.1

Notes:

An SLM schedule restricts the hours and days of operation of an SLM statement.

Schedules allow the application of different policies during the different clock hours of a 24-hour day.

If no schedule is specified, this policy statement is enforced always.

Use the check boxes to specify the days of the week that are included in the SLM schedule.

The **Start Time** and **Duration** apply to all checked days.

The **Start Date** and **Stop Date** indicate which dates this schedule is in effect. The Stop Date is non-inclusive.

The **Time Zone** (not visible in the screen capture) offers the choice of all the worldwide time zones, or “appliance local time”. This setting indicates what time zone the Start Time is applied to.

The screenshot shows the WebSphere Education interface with the title 'The SLM action'.

SLM Action Configuration Dialog:

SLM Action	
Name:	<input type="text"/>
Admin State:	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments:	<input type="text"/>
Type:	Log Only <input type="button" value="▼"/>
Log Priority:	debug <input type="button" value="▼"/>

Configure SLM Action List:

Name	Status	Op-State	Logs	Admin State	Comments
notify	saved	up		enabled	
shape	saved	up		enabled	
throttle	saved	up		enabled	

Actions:

- When an SLM statement detects a threshold violation, an SLM action defines the response
 - Click **Objects > Monitoring > SLM Action** to define individually
- Default SLM Action objects
 - Notify:** Creates log message when action is fired
 - Shape:** Buffers request to meet traffic threshold up to limit: otherwise, it rejects
 - Throttle:** Reject outright
- New SLM actions can be defined to change log priority of logged message

Figure 9-11. The SLM action

WE601 / ZE6011.1

Notes:

An SLM action defines a behavior that is triggered when a threshold value is attained. It specifies the administrative operations or sanctions that are taken when the configured threshold is exceeded.

Default SLM Action objects:

- Log only:** After the action is triggered, writes a log entry and continues to process subsequent transactions.
- Reject:** After the action is triggered, writes a log entry and rejects traffic until the monitored entity is within conformance levels.
- Shape:** After the action is triggered, writes a log entry. The next 2500 transactions are queued for later transmission when the monitored entity is within conformance levels. After 2500 transactions are queued, further transactions are rejected.

Do not confuse the **SLM action object** that is used within an SLM statement with the **SLM processing action** that is used in a processing rule to enable SLM monitoring.

WebSphere Education

SLM statement (1 of 2)

- An SLM statement can consist of:
 - Credential Class:** Defines a possible client group subject to this SLM statement
 - Resource Class:** Identifies a possible resource group subject to this SLM statement
 - Schedule:** Time frame during which this SLM statement is enforced
 - Action:** Administrative action (sanction) to take if threshold violated (required)
- SLM statements exist only within the SLM policy object

Edit Statement

Identifier	2 *
User Annotation	Auto generated
Credential Class	(none) + ...
Resource Class	AddressSearchPolicy_port-operation_findByName + ...
Schedule	(none) + ...
SLM Action	shape + ... *
Threshold Interval Length	0
Threshold Interval Type	Fixed
Threshold Algorithm	Greater Than

© Copyright IBM Corporation 2014

Figure 9-12. SLM statement (1 of 2)

WE601 / ZE6011.1

Notes:

An **SLM statement** establishes criteria for selecting messages, sets a measurement interval, sets thresholds, and determines the action to take when the threshold is exceeded for the selected messages.

Messages are selected based on a credential class, a resource class, or both. If neither is configured, all messages are selected.

The **Identifier** field gives this SLM statement a unique name within the SLM policy object that it is a part of. It also is displayed in any log entries that are generated because this statement is in effect.

SLM statements are not objects that can be created, reviewed, or edited as stand-alone objects. They are available only within the SLM policy object.



SLM statement (2 of 2)

Thresholds

- Usage level that triggers an SLM action

Threshold Interval Length	0	Seconds
Threshold Interval Type	Fixed	
Threshold Algorithm	Greater Than	
Threshold Type	Count All	
Threshold Level	300	
Reporting Aggregation Interval	0	Minutes
Maximum Records Across Intervals	5000	Records *
Auto Generated by GUI	<input type="radio"/> on <input checked="" type="radio"/> off	
Maximum Credentials-Resource Combinations	5000	Records *
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>		

Threshold fields

- Threshold Interval Length:** Length of measurement interval
- Threshold Interval Type**
 - Fixed: a discrete block of time, for example, 8 a.m. to 9 a.m.
 - Moving: a moving window, for example, the last 60 minutes
- Threshold Algorithm:** Greater than, less than, token bucket, high-low threshold
- Threshold Type:** Count all, count errors, back-end latency, internal latency, total latency, request message payload, response message payload
- Threshold Level:** Value that triggers the threshold

© Copyright IBM Corporation 2014

Figure 9-13. SLM statement (2 of 2)

WE601 / ZE6011.1

Notes:

The threshold algorithm specifies how the threshold is evaluated within the current interval. **Greater Than** and **Less Than** are simple relational operations. **Token-bucket** is based on a rate and allows bursting. High and low thresholds trigger at the high threshold and continue to trigger until the low threshold is achieved.

The high-low-thresholds algorithm allows the user to specify when to start the sanction and when to stop in cases where those two values are not the same. The threshold level is the "high" starting point. The **High Low Release Level** (not shown) configures the "low" stopping point.

Threshold Type specifies how the **Threshold Level** is applied to the count.

Reporting Aggregation Interval is the base aggregation level in minutes for the reporting statistics. This property is independent of the thresholding interval.

Maximum Records Across Intervals is the total number of records for a reporting interval. A single reporting aggregation interval can contain multiple records; for example, one record per resource or credential. This property allows you to define a maximum memory-consumption threshold. The default is 5000.

Auto Generated by GUI is a read-only property that, when on, indicates that the WebGUI created the statement is part of a default SLM configuration (**SLM Policy** tab in a web service proxy).

Maximum Credentials-Resource Combinations is the maximum number of records for the combination of credentials and resources. This property limits the maximum number of combinations and allows the setting of a maximum memory-consumption threshold. The default is 5000.



SLM policy: Main tab

- An SLM policy consists of one or more SLM statements and an evaluation method
 - Click **Objects > Monitoring > SLM Policy** to define individually
- Evaluation method: determines how the SLM policy evaluates the remaining SLM statements if a threshold is exceeded in the current SLM statement
 - Execute all statements**
 - Terminate at first action**
 - Terminate at first reject**

SLM Policy: AddressSearchProxy [up]

Administrative State: enabled disabled

Comments:

Evaluation Method: Terminate at First Reject

Peer Group: (none) + ...

© Copyright IBM Corporation 2014

Figure 9-14. SLM policy: Main tab

WE601 / ZE6011.1

Notes:

The **Evaluation Method** field allows control over execution of the statements within the policy.

- Execute all statements:** Causes the policy to execute all policy statements regardless of what action those statements take
- Terminate at first action:** Causes the policy to stop executing any statement after the *first statement* that takes an *action* because a threshold is met
- Terminate at first reject** (the default): Causes the policy to stop executing any statement after the *first statement* that *rejects a message* because a threshold is met

An SLM policy can be enforced across a group of appliances that handle load-balanced traffic that is destined for the same resources by using a **Peer Group**.

Peer groups establish a data sharing protocol among appliances so that each appliance includes the traffic that passed through the other peers when calculating whether a threshold is reached. SLM monitors are the only monitor types that do so.



SLM policy: Statements tab

- Lists the SLM statements that are part of this SLM policy, and the order of evaluation

Statement							
Identifier	User Annotation	Credential Class	Resource Class	Schedule	SLM Action	Thread Inter.	Length
1	Auto Generated		AddressSearchProxy_a552283b-ce25-442f-b609-14de727fbe6e		notify	60	
2	Auto Generated		AddressSearchProxy_91bec166-0909-4e55-96d2-711ae2d29ae6		throttle	60	
3	limit on web service		EastAddressSeach_URI		throttle	60	

© Copyright IBM Corporation 2014

Figure 9-15. SLM policy: Statements tab

WE601 / ZE6011.1

Notes:

Getting SLM statements into the Statement list

- Since SLM statements do not exist as separate objects, they cannot be selected from a drop-down list
- SLM statements are added to the list by:
 - Specifying SLM criteria on the SLM Policy tab of a web service proxy (auto-generates SLM statements)
 - Clicking **Add** beneath the list to create a custom SLM statement
- The first two statements are auto-generated
- The third statement is custom

Hold Al	Threshold Algorithm	Threshold Type	Threshold Level	High-Low Release Level	Burst Limit	Reporting Aggregation Interval	Maximum Records Across Intervals	Auto Generated by GUI	Maximum Credentials-Resource Combinations	
	Greater Than	Count All	2	0	0	0	5000	on	5000	 
	Greater Than	Count All	5	0	0	0	5000	on	5000	 
	Greater Than	Count All	200	0	0	0	5000	off	5000	 
										Add

© Copyright IBM Corporation 2014

Figure 9-16. Getting SLM statements into the Statement list

WE601 / ZE6011.1

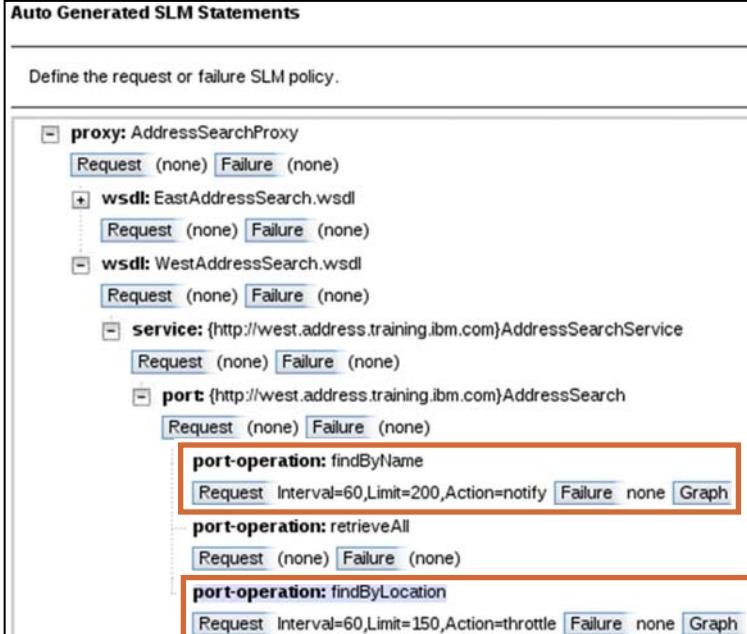
Notes:

This graphic is the right side of the WebGUI page from the previous slide.

Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2)

- Web service proxy has an **SLM Policy** tab to allow simple definitions of SLM monitoring criteria
- Specifying this criterion creates the auto-generated SLM statements
- SLM criteria can be uniquely specified at the different levels of the WSDL (proxy, wsdl, service, port, port-operation)
- Criteria can be set for successful transactions (Request) and errors (Failure)



© Copyright IBM Corporation 2014

Figure 9-17. Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2)

WE601 / ZE6011.1

Notes:

For the auto-generated SLM statements, you specify the measurement interval, the threshold value, and the SLM action to take if the threshold is exceeded.

The **Graph** button is explained in a later slide.

The screen capture shows a service-level policy for the `findByName` operation of 200 transactions per 60 seconds, which if exceeded results in a `notify` action. It also dictates that five failed transactions within 60 seconds get logged. For the `findByLocation` operation, a lower limit of 150 transactions per 60 seconds results in the `throttle` action.

The image shows a screenshot of the IBM WebSphere Education interface. At the top, there is a blue header bar with the text "WebSphere Education" and the IBM logo. Below the header, the title "Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2)" is displayed in large blue text. The main content area is divided into two sections: "SLM Peers" and "SLM Statements".

SLM Peers: This section contains a description: "Define the collection of SLM peers that monitor an SLA." It includes a dropdown menu labeled "Type" set to "SLM Unicast" with a required asterisk (*). Below it is a "URL" field with "(empty)" and an "Add" button.

SLM Statements: This section contains a description: "SLM Statements define custom SLM policies to monitor transactions that meet specific credential or resource criteria." It features a table with columns: ID, Credential Class, Resource Class, Schedule, Threshold Level, Threshold Type, and Action. A button labeled "Create New Statement" is located at the bottom left of this section.

© Copyright IBM Corporation 2014

Figure 9-18. Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2)

WE601 / ZE6011.1

Notes:

This graphic is the lower part of the SLM Policy tab for a web service proxy.

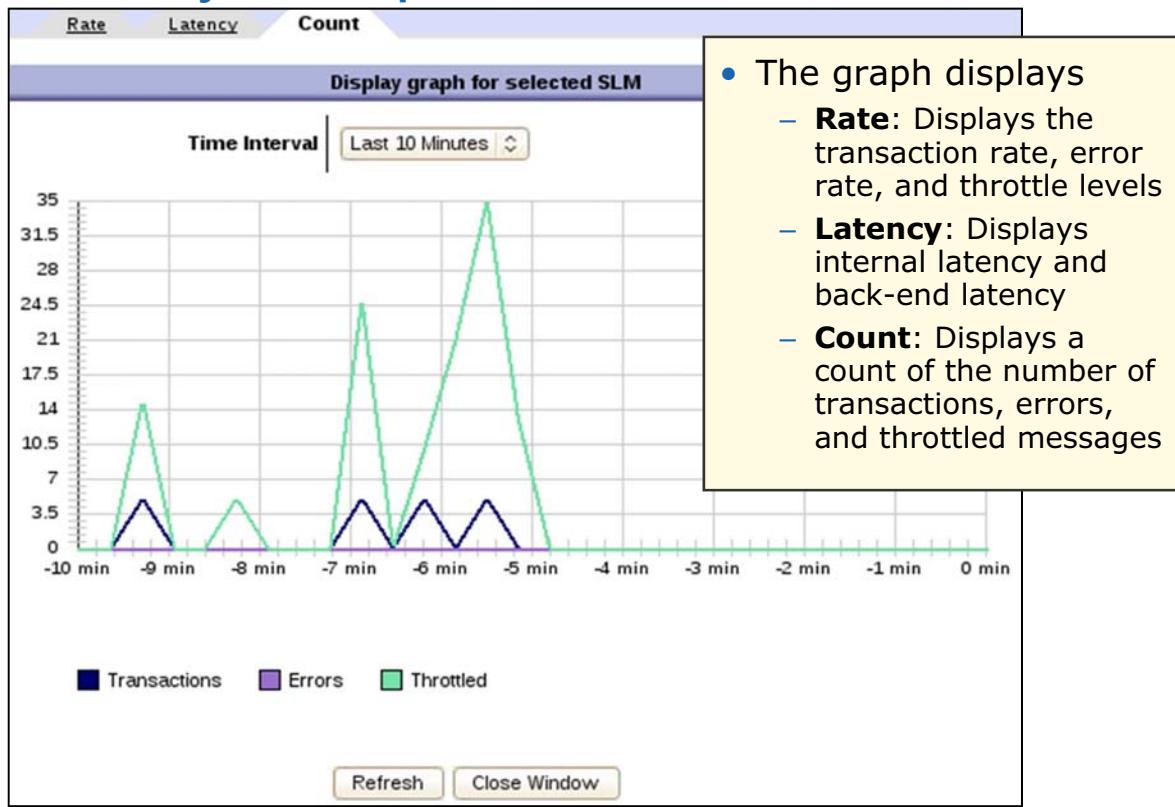
Configuring SLM peers is an administrative task.

SLM Statements lists only custom SLM statements that exist within the SLM policy that has the same name as the web service proxy. The specifications on this page define the default SLM policy object that is created for the web service proxy.

If you click **Create New Statement**, the page refreshes with a section that contains the same fields as exist in an SLM statement configuration page.

WebSphere Education

SLM Policy tab: Graphs



- The graph displays

- Rate**: Displays the transaction rate, error rate, and throttle levels
- Latency**: Displays internal latency and back-end latency
- Count**: Displays a count of the number of transactions, errors, and throttled messages

© Copyright IBM Corporation 2014

Figure 9-19. SLM Policy tab: Graphs

WE601 / ZE6011.1

Notes:

The SLM graph is displayed by selecting the appropriate **Graph** radio button in the web service proxy **SLM Policy** tab.

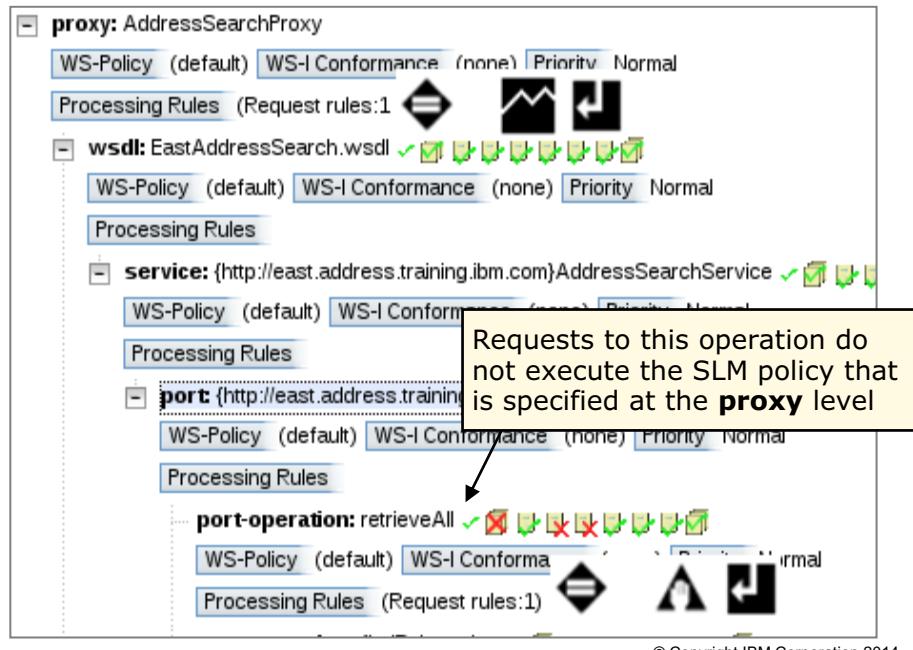
The possible time intervals are last 10 minutes, last 30 minutes, last hour, and last 3 hours.

This option is only for development time monitoring. For production monitoring, use software such as the IBM Tivoli Monitoring Family.

SLM action granularity

- A web service proxy or multi-protocol gateway service policy must explicitly define an **SLM action** in order for client requests to participate in service level monitoring

- The default web service proxy request policy contains an SLM action
- A fine-grained policy without an SLM action does not participate in service level monitoring



© Copyright IBM Corporation 2014

Figure 9-20. SLM action granularity

WE601 / ZE6011.1

Notes:

For both a web service proxy and a multi-protocol gateway, an **SLM action** must be in a rule of the service policy for any SLM monitoring to occur.

Each request to a web service proxy executes the most specific rule that it can find, starting at the port-operation level. Only one rule is executed per request. The default **proxy** rule contains an SLM action for the request rule. Therefore, all web service requests participate in service level monitoring by default. However, if a more specific rule is defined, the default proxy level rule is not executed. Hence, no SLM action is "inherited". For the more specific rule to support SLM monitoring, it must also contain its own SLM action.

For a multi-protocol gateway, there is no such rule "inheritance". Each rule must contain its own SLM action to participate in SLM monitoring.



Unit summary

Having completed this unit, you should be able to:

- Identify the SLM functions that the WebSphere DataPower appliance provides
- Create an SLM policy object by using the WebGUI
- Create a custom SLM statement
- Use the SLM Policy tab in the web service proxy to create a basic SLM policy

© Copyright IBM Corporation 2014

Figure 9-21. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. What are the five constructs that make up the **SLM Statement** object?
 - A. Credential class, resource class, schedule, threshold, and action
 - B. Service policy, processing rules, actions, rules, and filter
 - C. Client class, resource class, schedule, threshold, and sanction

2. Match the functionality to the **Reject** and **Shape** action types:

Description	Definition
1. Reject action	A. Log and drop traffic
2. Shape action	B. Log, queue traffic to meet threshold, otherwise reject

3. True or False: SLM monitors are implemented as part of a service policy.

© Copyright IBM Corporation 2014

Figure 9-22. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. A.
2. 1 – A, 2 – B.
3. True.

© Copyright IBM Corporation 2014

Figure 9-23. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 7



Implementing an SLM monitor in a web service proxy

© Copyright IBM Corporation 2014

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

9.0

Figure 9-24. Exercise 7

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Specify service level monitoring criteria for a web service proxy
- Inspect and edit an SLM policy object
- Explain the need for an operation-level SLM action in a web service proxy
- Create a custom log target for SLM events

© Copyright IBM Corporation 2014

Figure 9-25. Exercise objectives

WE601 / ZE6011.1

Notes:

Unit 10. XML and web services security overview

What this unit is about

This unit describes the features of the web services security specification. This specification provides message level security to ensure message confidentiality and integrity by using XML encryption and XML signatures. You learn how to use the DataPower device to encrypt and decrypt, and to sign and verify messages.

What you should be able to do

After completing this unit, you should be able to:

- Describe the features of the WS-Security specification
- Enable message confidentiality by using XML Encryption
- Provide message integrity by using XML Signature

How you will check your progress

- Checkpoint
- Exercise 8: Web service encryption and digital signatures



Unit objectives

After completing this unit, you should be able to:

- Describe the features of the WS-Security specification
- Enable message confidentiality by using XML Encryption
- Provide message integrity by using XML Signature

© Copyright IBM Corporation 2014

Figure 10-1. Unit objectives

WE601 / ZE6011.1

Notes:

Review of basic security terminology

- **Authentication** verifies the identity of a client
- **Authorization** decides a client's level of access to a protected resource
- **Integrity** ensures that a message is not modified while in transit
- **Confidentiality** ensures that the contents of a message are kept secret
- **Auditing** maintains records to hold clients accountable to their actions
- **Nonrepudiation** is the condition where you are assured that a particular message is associated with a particular individual



© Copyright IBM Corporation 2014

Figure 10-2. Review of basic security terminology

WE601 / ZE6011.1

Notes:

Authentication is the act of verifying that the identity asserted by the client is valid. Normally, a security token that is attached to the message makes a claim about the client identity. Plaintext user name and password tokens, X.509 certificates, and Kerberos tickets are all examples of identity claims.

Authorization is the process of deciding whether a client has access to a protected resource. This process also determines the level of access that the server grants the client. In most cases, the authorization decision requires that the client identity is known and verified. That is, authorization occurs after authentication.

Integrity, also known as **data integrity**, makes sure that a message is not altered or tampered with while it travels between the client and the server. Digital signatures and hash codes can prove whether a message was modified in transit.

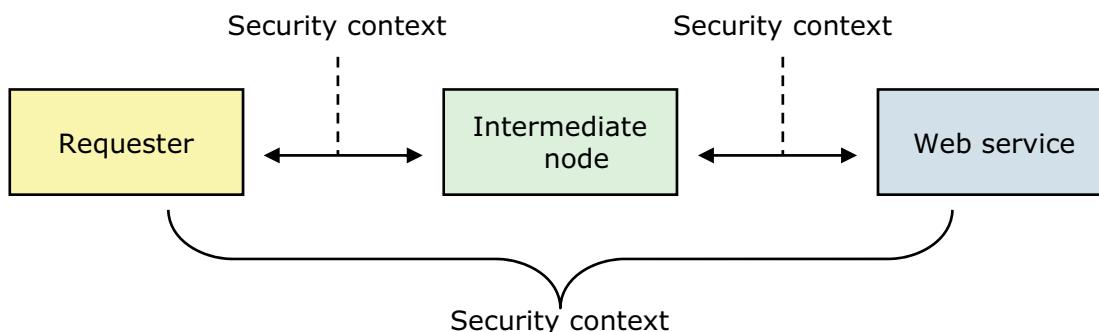
Confidentiality ensures that only authorized parties have access to protected resources. The effect of confidentiality is to keep private data or resources secret. This quality is often implemented through the encryption of data, in which only authorized parties have the means of making obscured data into legible information.

Auditing is the process of maintaining irrefutable records for holding clients accountable to their actions. Signed security logs provide one way to audit a security system. The concept of nonrepudiation is tied closely to auditing. It is the ability of one party of the communication to prove that the other party received its message. **Nonrepudiation** is often split into two concepts: nonrepudiation of origin proves that one party sent a message, while nonrepudiation of receipt proves that one party received a message.

Verifying the digital signature and the expiration date on the message enforces nonrepudiation of origin. Nonrepudiation of receipt depends on the software environment.

Web services security

- Web Services Security (WS-Security) provides a standard, platform-independent way for specifying **message-level** security information
- Flexible set of mechanisms for using a range of security protocols:
 - Does **not** define a set of security protocols
 - Provides **end-to-end** security



© Copyright IBM Corporation 2014

Figure 10-3. Web services security

WE601 / ZE6011.1

Notes:

WS-Security does not describe specific security protocols. This model can use different security mechanisms, and can be configured to match the requirements of new ones as they are developed. By separating the security constraints from the actual implementation, developers can change security technologies without needing to adopt another web services security specification.

Each arrow between two boxes shows a point-to-point security context. Transport level security, such as SSL/TLS, provides a security context that persists only from one intermediate node to another.

The curved line that spans multiple boxes is an example of end-to-end security. WS-Security provides this security context.

WS-Security provides message-level security. SSL/TLS secures the entire HTTP request, and is at the transport layer. WS-Security allows security to be applied to specific message parts of the request payload.

Components of WS-Security

- Associates security tokens with a message
 - Username token profile
 - X.509 token profile
 - Kerberos token profile
 - SAML token profile: Security Assertion Markup Language
 - REL token profile: Rights Expression Language
- Confidentiality (XML encryption)
 - Process for encrypting data and representing the result in XML
- Integrity (XML signature)
 - Digitally sign the SOAP XML document, providing integrity and signer authentication
- XML canonicalization
 - Normalizes XML document
 - Ensures that two semantically equivalent XML documents contain the same octet stream

© Copyright IBM Corporation 2014

Figure 10-4. Components of WS-Security

WE601 / ZE6011.1

Notes:

An XML digital signature is based on the W3C recommendation specification for XML-signature syntax and processing. For more information, see: <http://www.w3.org/TR/xmldsig-core/>

XML encryption is based on the W3C recommendation for XML encryption syntax and processing. For more information, see: <http://www.w3.org/TR/xmlenc-core/>

The security token profiles that are listed are for WS-Security V1.1. For links to the list of specifications, see <http://www.oasis-open.org/specs/index.php#wssv1.0>

Specifying security in SOAP messages

- Attach security-related information to SOAP messages in the `<wsse:Security>` header element

```

<env:Envelope
    xmlns:env="http://www.w3.org/2001/12/soap-envelope">
<env:Header>

    <wsse:Security
        env:actor="http://www.example.com/secManager"
        env:mustUnderstand="1"
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd">
        <!-- WS-Security header here -->
    </wsse:Security>

</env:Header>
<env:Body>
    <!-- SOAP message body here -->
</env:Body>
</env:Envelope>

```

© Copyright IBM Corporation 2014

Figure 10-5. Specifying security in SOAP messages

WE601 / ZE6011.1

Notes:

The **actor** and **mustUnderstand** are special attributes that the SOAP specification defines. The **actor** attribute contains a URL of the targeted recipient for the SOAP header. The **mustUnderstand** attribute is used to specify that the tags in the header must be understood; otherwise, a fault is thrown.

Scenario 1: Ensure confidentiality with XML encryption

- Use XML encryption to keep messages secret
 - Encrypt with the recipient certificate: only the recipient can decrypt with associated private key
 - XML encryption specification does not describe how to create or exchange keys
- XML encryption supports:
 - Message encryption at different levels of granularity, from a single element value to a tree of XML elements
 - Secure message exchange between more than two parties: a message might pass through intermediate handlers that read only the parts of the message relevant to them

© Copyright IBM Corporation 2014

Figure 10-6. Scenario 1: Ensure confidentiality with XML encryption

WE601 / ZE6011.1

Notes:

By encrypting message content, the privacy of the content becomes decoupled from the transport mechanism. For example, messages sent over an SSL connection are encrypted. They are thus provided with some degree of privacy, but no further privacy is provided after the message exits the SSL connection. By encrypting the content of the message, the message can travel across transport boundaries, such as HTTP and WebSphere MQ, and remain private.

The `<Envelope>`, `<Header>`, and `<Body>` elements cannot be encrypted.

XML encryption and WS-Security

- The XML encryption standard uses symmetric encryption algorithms for the actual data encryption
 - The symmetric key is passed with the message
 - The public key in the recipient's certificate is used to encrypt the symmetric key before transmission
 - Only the recipient has the private key to decrypt the symmetric key
 - Encrypted data is inside <enc:EncryptedData> element
- The WS-Security standard uses XML encryption
 - Places encryption metadata in SOAP header <wsse:Security>
 - Supports passing the symmetric key in multiple ways

© Copyright IBM Corporation 2014

Figure 10-7. XML encryption and WS-Security

WE601 / ZE6011.1

Notes:

DataPower support for XML encryption

- Applies XML encryption to a message by defining a processing rule that contains:
 - Encrypt** action: performs full or field-level message encryption
 - Decrypt** action: performs full or field-level message decryption
- Acts as **client** to *encrypt* a message sent to the server



- Acts as **server** to *decrypt* a message that the client sent



© Copyright IBM Corporation 2014

Figure 10-8. DataPower support for XML encryption

WE601 / ZE6011.1

Notes:



Encrypt action

The **Encrypt** action performs full or field-level encryption

- Envelope Method: controls placement of generated security elements
- Message and Attachment Handling: encrypt message, attachment, or both
- Encryption Key Type: how the symmetric key is protected
- Use Dynamically Configured Recipient Certificate: uses passed certificate, if it exists
- One Ephemeral Key: causes all encryption in this step to use the same ephemeral key
- Recipient Certificate: the certificate that is used to encrypt the encryption key

Encrypt	
Envelope Method	<input checked="" type="radio"/> WSSec Encryption <input type="radio"/> Standard XML Encryption <input type="radio"/> Advanced
Message Type	<input checked="" type="radio"/> SOAP Message <input type="radio"/> Raw XML Document <input type="radio"/> Selected Elements (Field-Level) <input type="radio"/> Advanced
Asynchronous	<input type="radio"/> on <input checked="" type="radio"/> off
Message and Attachment Handling	Message Only <input type="button" value="Save"/>
Encryption Key Type	Use Ephemeral Key Transported by Asymmetric Algorithm <input type="button" value="Save"/>
Use Dynamically Configured Recipient Certificate	<input type="radio"/> on <input checked="" type="radio"/> off <input type="button" value="Save"/>
One Ephemeral Key	<input type="radio"/> on <input checked="" type="radio"/> off <input type="button" value="Save"/>
Recipient Certificate	(none) <input type="button" value="Save"/>
WS-Security Version	1.0 <input type="button" value="Save"/>

© Copyright IBM Corporation 2014

Figure 10-9. Encrypt action

WE601 / ZE6011.1

Notes:

The **Advanced** choice for Envelope Method and Message Type is not selectable.

An ephemeral key is a key that is generated each time encryption occurs. Basically, it is the symmetric key that is used for encryption.

The DataPower device supports the following encryption schemas:

WSSec encryption (OASIS) standard puts the signature and key information in the SOAP header.

Standard XML encryption (W3C) puts the signature and key information in the body of message.

The WS-Security standard puts the signature and key information in the WS-Security header of the SOAP message. This standard does not add elements to the body of the message, and therefore does not violate the underlying schema.

Standard XML encryption was originally designed to handle any XML message, including those messages that are not formatted to the SOAP specification. It puts the signature and key information in the body of the message, thus adding more elements to the body of the message.

The DataPower SOA appliance supports both methods of encryption. The appliance can use either standard for full message or partial encryption.

The following message types are supported:

- **SOAP message**: an encrypted SOAP document
- **Raw XML document**: an encrypted XML document (it cannot be used with WSSec encryption)
- **Selected elements (field-level)**: a partially encrypted SOAP document

The following options are in the **Message and Attachment Handling** menu:

- **Attachments only**: only the attachments of the message are encrypted
- **Message only**: only the message (root part) is encrypted
- **Message and attachments**: message (root part) and attachments are encrypted

The encryption key type specifies how the symmetric encryption key is protected. Depending on the selection, the fields in the page might change:

- Use Ephemeral Key Transported by Asymmetric Algorithm: The X509 key-cert pair transports the ephemeral key with an asymmetric algorithm.
- Use Symmetric Key Directly: A security token protects the session key.
- Use Ephemeral Key Wrapped by a Symmetric Key: The ephemeral key is encrypted by a symmetric key from a security token.

If **Use Dynamically Configured Recipient Certificate** is set to **on**, the Encrypt action uses a certificate that is used in a previous Verify action. This option supports use of the certificate in a Verify action for the request message as the encrypting certificate in an Encrypt action in the response.



Decrypt action

- The **Decrypt** action performs full or field-level decryption
 - Message Type: specifies how to decrypt the message
 - Decrypt Key: private key object that is used to decrypt

Basic [Advanced](#)

Input

Input INPUT
INPUT *

Options

Decrypt

Message Type Entire Message/Document
 Selected Elements (Field-Level)
 Advanced
*

Asynchronous on off

Decrypt Key (none) [+](#) [...](#) [Save](#)

Output

Output OUTPUT
OUTPUT

[Delete](#) [Done](#) [Cancel](#)

© Copyright IBM Corporation 2014

Figure 10-10. Decrypt action

WE601 / ZE6011.1

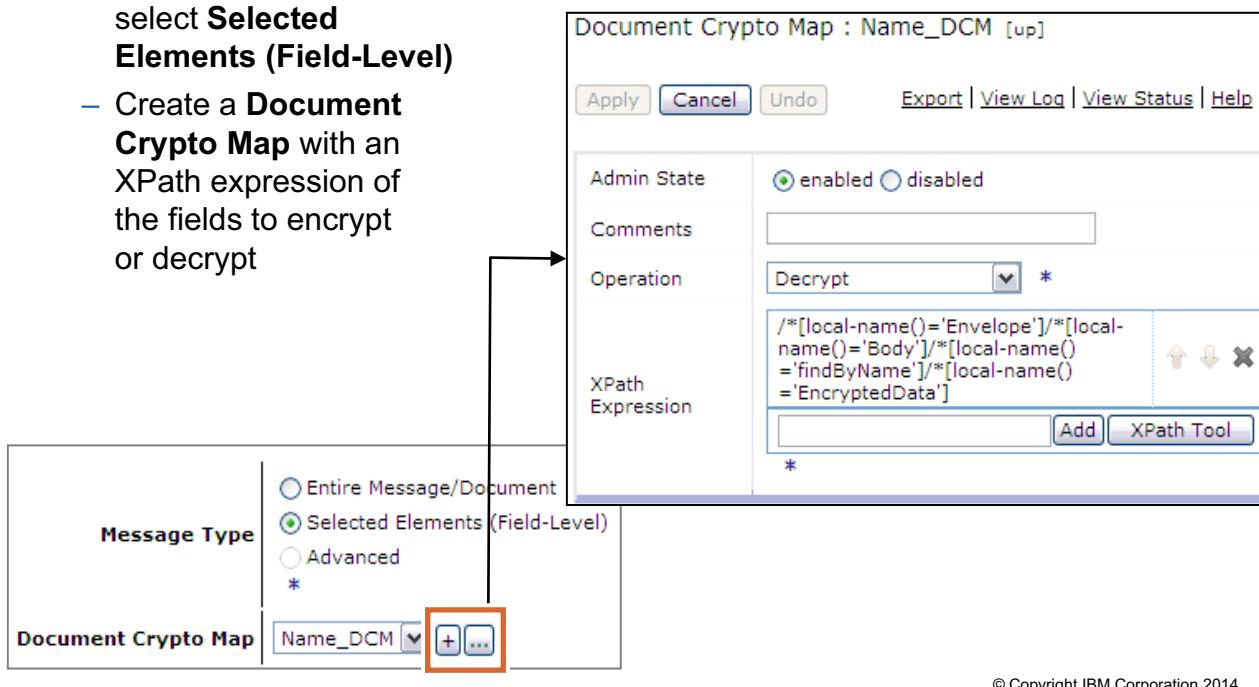
Notes:

The **Advanced** tab allows you to override the style sheet that is used to decrypt. The default file that is used is `store:///decrypt.xsl`.

WebSphere Education 

Field-level encryption and decryption

- Performs field-level encryption and decryption on messages
 - Under **Message Type**, select **Selected Elements (Field-Level)**
 - Create a **Document Crypto Map** with an XPath expression of the fields to encrypt or decrypt



© Copyright IBM Corporation 2014

Figure 10-11. Field-level encryption and decryption

WE601 / ZE6011.1

Notes:

The XPath expression can be created from an XML file by selecting the elements to encrypt or decrypt. The XPath expression for field-level decryption is different from the XPath expression for encrypting the same field. Encryption occurs on an element in the original message, for example, <name>. When it is time to decrypt, the field is no longer known as <name>, but as something else, such as <EncryptedData>. Thus, the XPath expression to get to the apparently identical element differs depending on whether you are encrypting the original field or decrypting the encrypted field.

XPath tool

- In the document crypto map, click **XPath Tool** to create an XPath expression by using an XML file
 - URL of Sample XML Document:** upload or select an XML document
 - Namespace handling:** how the XPath statement matches namespace declarations
 - XPath:** generated XPath statement

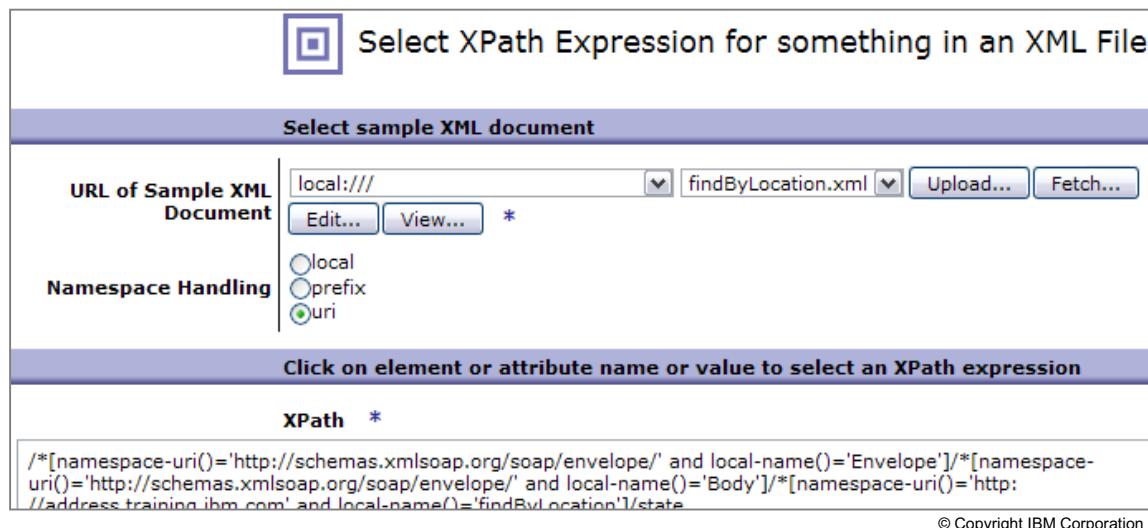


Figure 10-12. XPath tool

WE601 / ZE6011.1

Notes:

The content of the selected XML file is omitted from the slide and is shown below the three buttons (**Refresh**, **Done**, and **Cancel**). Click the elements in the XML file to generate an XPath expression.

The three options for namespace handling are:

- local:** This option compares only the local name (element name), ignoring the namespace.
- prefix:** This option compares the qualified name, including the namespace prefix. It can be used when the mapping from the namespace prefix to the URI is specified on the **Namespace Mappings** tab on an object configuration page.
- uri:** This option compares the local name and namespace URI.

Sample encrypted SOAP message

```

<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header>
        <wsse:Security soapenv:mustUnderstand="1">
            <xenc:EncryptedKey>
                ...
            </xenc:EncryptedKey>
        </wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
        <q0:findByName>
            <xenc:EncryptedData>
                <EncryptionMethod />
                <CipherData>
                    <CipherValue>
                        ...
                    </CipherValue>
                </CipherData>
            </xenc:EncryptedData>
        </q0:findByName>
    </soapenv:Body>
</soapenv:Envelope>

```

The diagram illustrates the structure of an encrypted SOAP message. It shows the XML code for the envelope, header, and body. In the header, there is a security block containing an encrypted key. In the body, there is a 'findByName' operation which contains an encrypted data block. This encrypted data block includes an encryption method, cipher data, and multiple cipher values. A large brace on the right side groups the entire body under the heading 'Field-level XML encryption'. Another brace within the body groups the 'xenc:EncryptedData' element under the heading 'Key that is used to encrypt message'.

© Copyright IBM Corporation 2014

Figure 10-13. Sample encrypted SOAP message

WE601 / ZE6011.1

Notes:

This example message does field-level encryption on the child elements of the `<q0:findByName>` element. If full-message encryption is applied, then this element would also be encrypted.

Namespace declarations were removed in this example.

When XML encryption is applied to the original SOAP message, a web services security header is inserted into the SOAP header with information about the key that was used to encrypt the message body. In this example, the child element of `<q0:findByName>` is encrypted.

Scenario 2: Ensure integrity with XML signatures

- Sign SOAP message parts to provide message integrity
 - Provide guarantee that message is not changed
 - Mismatch between decrypted hash (from signature) and computed hash (from cleartext) indicates that message is modified
 - Signatures provide strong indication of identity
 - Only holder of private key can create signature that matches the enclosed certificate (if asymmetric)
- XML signature standard provides a schema for storing digital signature information within XML messages
 - Does not describe how to digest and sign messages
 - Supports symmetric and asymmetric signing key
- Recipient validates the signature by repeating the same steps that are used to generate a digital signature
 - Compute message digest of received message
 - Obtain original message digest by decrypting signature from received message by using certificate, which holds public key
 - Compare computed message digest against original message digest

© Copyright IBM Corporation 2014

Figure 10-14. Scenario 2: Ensure integrity with XML signatures

WE601 / ZE6011.1

Notes:

The XML digital signature (XMLDS) is a joint effort between the World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF). For more information about signatures, see <http://www.w3.org/signature>

An XML signature is transport-independent; it can cross multiple transport protocol boundaries.

A message digest is a hash value that is generated by applying a digest algorithm to a message part. A private key is used to generate a digital signature. Depending on the algorithm, either the same private key or a public key is used to verify the signature.

A sender can choose to sign only specific portions of the XML tree rather than the complete document.

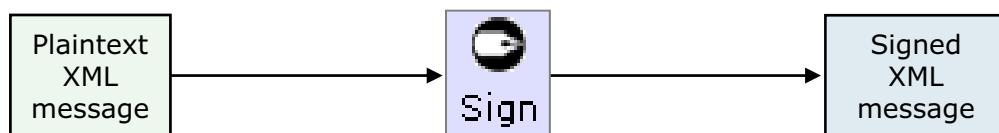
Consider the following example:

```
<transaction-info>
<user-id>jsmith</user-id>
<action>buy</action>
<symbol>IBM</symbol>
</transaction-info>
```

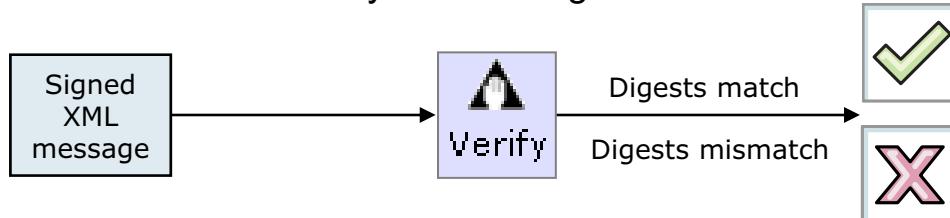
You can sign the value within the symbol element, the entire symbol element (including the value), or a group of elements within transaction-info.

DataPower support for XML signature

- Apply XML signature to a message by defining a document processing rule that contains:
 - **Sign** action: digitally signs a message
 - **Verify** action: verifies the digital signature in the message
- Acts as sender to sign message that is sent to the server



- Acts as receiver to verify the message that the client sent



© Copyright IBM Corporation 2014

Figure 10-15. DataPower support for XML signature

WE601 / ZE6011.1

Notes:

A client signs a message by using its private key. The message is verified with the client public key by using the client certificate, if asymmetric. The public key that is used to verify that the message is associated with the private key that was used to sign the message.

WebSphere Education

Sign action

- The **Sign** action signs specific elements or the entire message by using a crypto key object
 - Envelope Method:** determines placement of signature in message
 - Message Type**
 - Key:** crypto key object that is used to sign message
 - Certificate:** crypto certificate object that is associated with crypto key object

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

© Copyright IBM Corporation 2014

Figure 10-16. Sign action

WE601 / ZE6011.1

Notes:

Digital signatures might occur anywhere in a message. The signature can be in either the header or the body of the message, depending on the style that was chosen to sign the message. For non-SOAP XML messages, the signature element might occur anywhere in the message.

The choice of envelope method determines the placement of the XML signature (from DataPower WebGUI documentation):

- Enveloped Method:** The signature is over the XML content that contains the signature as an element. The content provides the root XML document element (not considered a good idea).
- Enveloping Method:** The signature is over content that is found within an object element of the signature. The object, or its content, is identified by using a reference through a URI fragment identifier or transform (not considered a good idea).
- SOAPSec Method:** The signature is included in a SOAP header entry.
- WSSec Method:** The signature is included in a WS-Security security header.

If an Envelope Method of **WSSec Method** and a Message Type of either **SOAP Message** or

SOAP With Attachments are selected, then the page shows a **Use Asymmetric Key** option. If the choice is:

- **On**, then the Signing Algorithm shows asymmetric choices
- **Off**, then the Signing Algorithm shows symmetric HMAC choices



Verify action

- The **Verify** action verifies a digital signature
- Signature Verification Type
 - Use asymmetric only, symmetric only, or either
- Optional Signer Certificate
 - Used instead of passed certificate
- Validation credential object
 - One or more certificate objects that are used to validate the signer certificate

Verify

Asynchronous	<input type="radio"/> on <input checked="" type="radio"/> off
Signature Verification Type	RSA/DSA Signatures <input type="button" value="▼"/> <input type="checkbox"/> Save
Optional Signer Certificate	<input type="text"/>
Validation Credential	AliceValidCred <input type="button" value="▼"/> <input type="button" value="+"/> <input type="button" value="..."/> <input checked="" type="checkbox"/> Save
Output	
Output	<input type="text"/>
<input type="button" value="Delete"/> <input type="button" value="Done"/> <input type="button" value="Cancel"/>	

© Copyright IBM Corporation 2014

Figure 10-17. Verify action

WE601 / ZE6011.1

Notes:

By default, a digital signature is verified by using the certificate (public key) that is contained in the signature. No additional configuration steps are required. The validation credential object validates the included certificate. If the certificate that is supplied in the signature does not validate against the validation credential object, the signature verification fails.



Verify action: Advanced tab

Verify

Action Type	Verify
Transform File	store://l1 verify.xsl <input type="button" value="Upload..."/> <input type="button" value="Fetch"/> Stylesheet Summary: Verify RSA, DSA or HMAC signatures.
Asynchronous	<input type="radio"/> on <input checked="" type="radio"/> off
Signature Verification Type	RSA/DSA Signatures <input type="checkbox"/> Save
Optional Signer Certificate	<input type="text"/> <input type="checkbox"/> Save
Validation Credential	AliceValidCred <input type="button" value="..."/> <input type="checkbox"/> Save
Check Timestamp	<input checked="" type="radio"/> on <input type="radio"/> off <input checked="" type="checkbox"/> Save
Check Timestamp Created	<input type="radio"/> on <input checked="" type="radio"/> off <input type="checkbox"/> Save
Check Timestamp Expiration	<input checked="" type="radio"/> on <input type="radio"/> off <input type="checkbox"/> Save
Timestamp Expiration Override Period	0 sec <input type="button" value="..."/>

© Copyright IBM Corporation 2014

Figure 10-18. Verify action: Advanced tab

WE601 / ZE6011.1

Notes:

The **Verify** action uses an advanced **Check Timestamp Expiration** property, which is **on** by default. Valid signatures might expire and thus fail verification.



Field-level message signature and verification

- The **Sign** action supports signing of specific elements by using a document crypto map
 - Similar to the **Encrypt** and **Decrypt** actions
- Select **Selected Elements (Field-Level)**
 - Create a Document Crypto Map with an XPath expression of the fields to sign

Document Crypto Map : Sign_DCM [up]

Admin State: enabled

Comments:

Operation: Sign (WS-Security) *

XPath Expression: /SOAP-ENV:Envelope/SOAP-ENV:Body/q0:findByName/name/title

Message Type:

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

Document Crypto Map: Sign_DCM [+] ...

© Copyright IBM Corporation 2014

Figure 10-19. Field-level message signature and verification

WE601 / ZE6011.1

Notes:

The **Verify** action does not include a **field-level** radio button. In the WSSec envelope method, an ID is inserted into the element of the message that is signed. For example, if the entire message is signed, then the child element of the SOAP body contains the ID attribute. The ID attribute can be used to determine the elements that are signed.

This ID might cause messages to fail schema validation.

Sample signed SOAP message

```

<soapenv:Envelope xmlns:q0="http://east.address.training.ibm.com">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1">
      <wsse:BinarySecurityToken
        wsu:Id="SecurityToken-abf72a2b-3118-4aa2-98e7-462fa3208f5a">
      </wsse:BinarySecurityToken>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          ...
        </SignedInfo>
        <SignatureValue>rFHK9ixdAm6Mq0</SignatureValue>
        <KeyInfo>
          <wsse:SecurityTokenReference xmlns="">
            ...
          </wsse:SecurityTokenReference>
        </KeyInfo>
      </Signature>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body wsu:Id="Body-441d82cd-1613-4905-8aab-ebc7a91d8121">
    <q0:findByLocation>
      <city/>
      <state>NY</state>
    </q0:findByLocation>
  </soapenv:Body>
</soapenv:Envelope>

```



© Copyright IBM Corporation 2014

Figure 10-20. Sample signed SOAP message

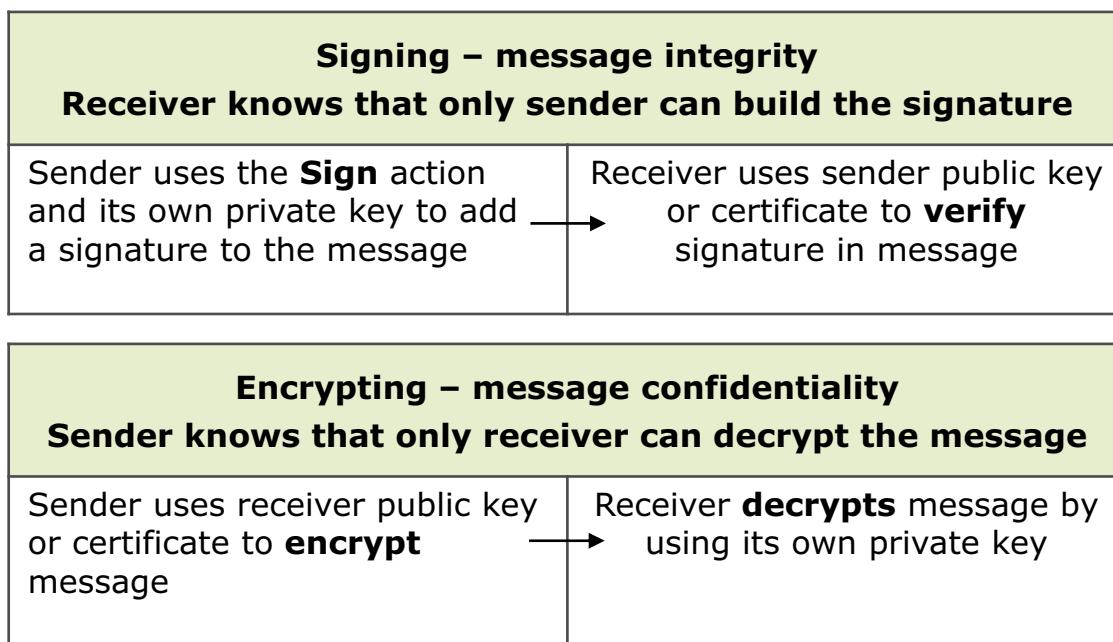
WE601 / ZE6011.1

Notes:

This message is condensed to fit on the slide.

Signing messages might rewrite attributes in the message. Notice the `wsu:id` attribute added by the **Sign** action to the SOAP body. This action might cause the body of the message to invalidate against a schema, depending upon how the schema is written.

Summary of security and keys



Sign message, then encrypt, for best confidentiality and integrity

© Copyright IBM Corporation 2014

Figure 10-21. Summary of security and keys

WE601 / ZE6011.1

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe the features of the WS-Security specification
- Enable message confidentiality by using XML Encryption
- Provide message integrity by using XML Signature

© Copyright IBM Corporation 2014

Figure 10-22. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: A document crypto map is used to specify an XPath expression that contains the elements to encrypt, decrypt, sign, and verify.
2. True or False: Encryption and decryption can occur at both the message and field levels, but sign and verify occur at the message level only.
3. True or False: The validation credential object validates the signer certificate, which is the public key that is used to generate the digital signature. This certificate is usually included in the message, but an alternative certificate can be specified in the **Signer Certificate** field.

© Copyright IBM Corporation 2014

Figure 10-23. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. **False.** A document crypto map is used to specify an XPath expression that contains the elements to encrypt, decrypt, and sign. The Verify action does not use a map since it can determine the signed elements from the headers.
2. **False.** Both scenarios are supported, even though the Verify action does not have a selected field-level radio button.
3. **True.**

© Copyright IBM Corporation 2014

Figure 10-24. Checkpoint answers

WE601 / ZE6011.1

Notes:



Exercise 8



Web service encryption and digital signatures

© Copyright IBM Corporation 2014

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

9.0

Figure 10-25. Exercise 8

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Create a multi-protocol gateway to generate a message with XML encryption
- Create a multi-protocol gateway to generate a message with an XML digital signature
- Perform field-level encryption and decryption on XML messages
- Create a rule to decrypt messages and verify digital signatures that are contained in a message within a web service proxy policy

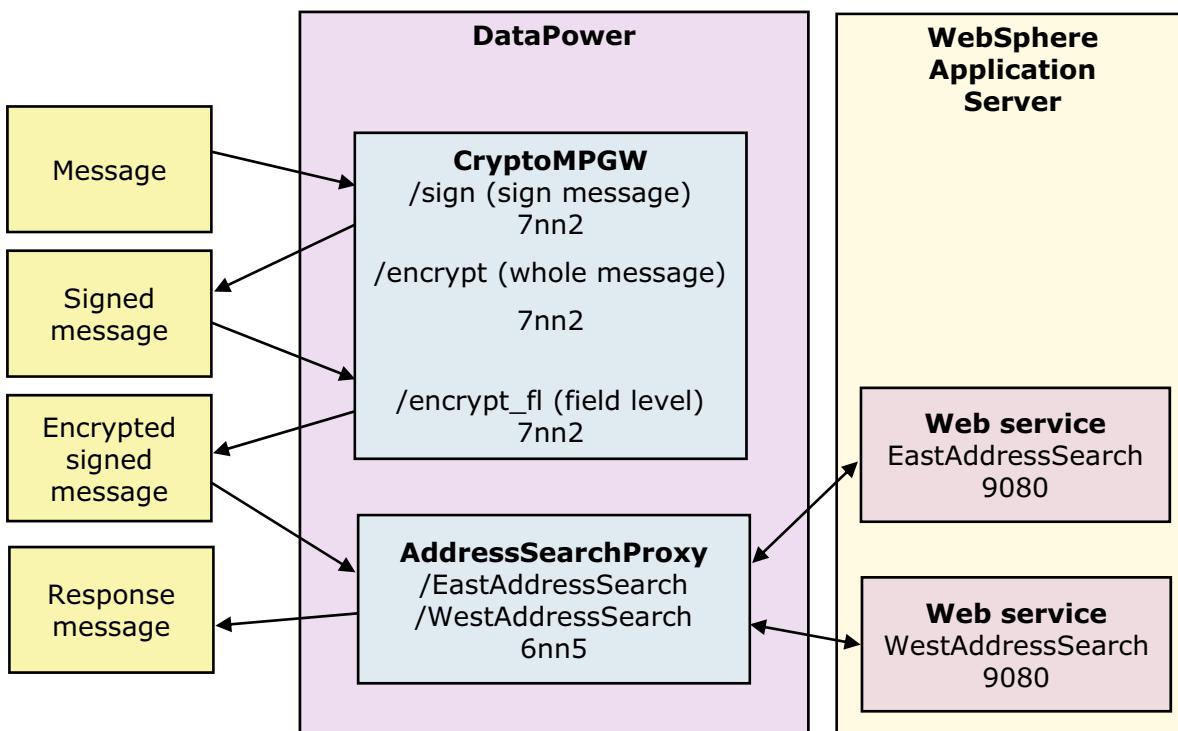
© Copyright IBM Corporation 2014

Figure 10-26. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview



© Copyright IBM Corporation 2014

Figure 10-27. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 11. Authentication, authorization, and auditing (AAA)

What this unit is about

This unit describes the authentication, authorization, and auditing (AAA) framework within the WebSphere DataPower SOA Appliances. These three facets of security both monitor and restrict access to resources.

What you should be able to do

After completing this unit, you should be able to:

- Describe the AAA framework within the WebSphere DataPower SOA Appliance
- Explain the purpose of each step in an access control policy
- Authenticate and authorize web service requests with:
 - WS-Security Username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

How you will check your progress

- Checkpoint
- Exercise 9: Web service authentication and authorization



Unit objectives

After completing this unit, you should be able to:

- Describe the AAA framework within the WebSphere DataPower SOA Appliance
- Explain the purpose of each step in an access control policy
- Authenticate and authorize web service requests with:
 - WS-Security Username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

© Copyright IBM Corporation 2014

Figure 11-1. Unit objectives

WE601 / ZE6011.1

Notes:

Authentication, authorization, and auditing

- In the DataPower appliance, AAA represents three security processes: **authentication, authorization, and auditing**.



- **Authentication** verifies the identity of the request sender
- **Authorization** determines whether the client has access to the requested resource
- **Auditing** keeps records of any attempts to access resources

© Copyright IBM Corporation 2014

Figure 11-2. Authentication, authorization, and auditing

WE601 / ZE6011.1

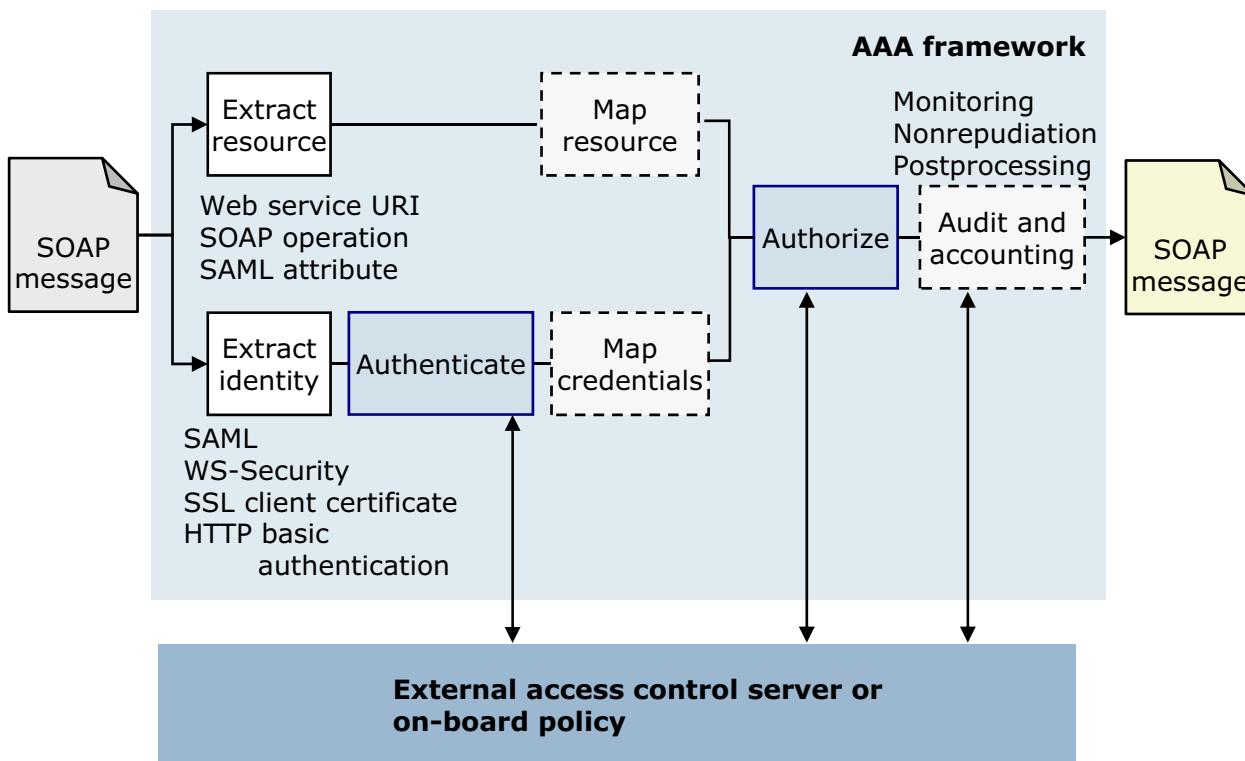
Notes:

Authentication always precedes authorization. A policy cannot decide whether a request proceeds if it does not know the identity of the requester. For example, a security guard first determines whether someone is an employee of the company. After this step, the guard determines whether that employee has access to the building. Together, authentication and authorization restrict access to resources.

Although auditing does not directly protect resources against unauthorized access, this third process has an important role in securing resources. A record of successful and unsuccessful access attempts allows the security infrastructure to detect suspicious activity in the system. Historical logs also enforce nonrepudiation; clients cannot deny accessing the system in the past.

In literature these three steps are commonly known as "AAA", which is pronounced "triple A."

Authentication and authorization framework



© Copyright IBM Corporation 2014

Figure 11-3. Authentication and authorization framework

WE601 / ZE6011.1

Notes:

The AAA action combines three security processes into a single style sheet transform. In the first step, the style sheet extracts the identity token from the message. To verify the claims that the token makes, the style sheet either authenticates the token against an on-board policy or queries an external access control server. As soon as the client identity is confirmed, the style sheet maps the client credentials to one of the users or groups that the service defines.

In the second step, the style sheet extracts the requested resource from the message. For web services, a resource represents a service or service operation. If the requested resource is an alias for one or more back-end resources, the style sheet maps the alias to the actual resource names as well.

When the style sheet determines the requested back-end resource and confirms the client identity, it decides whether the client has permission to access the requested resource. In other words, the style sheet authorizes access to a back-end resource.

The final step is auditing and accounting. The style sheet records any access attempts, successful or unsuccessful, for monitoring and nonrepudiation. The style sheet can also do post processing steps, such as generating various tokens for the outgoing SOAP message. A custom style sheet can also be specified.

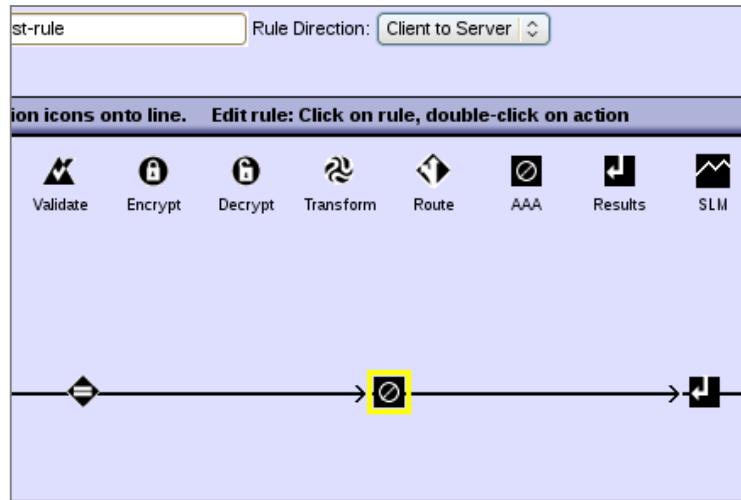
The various tokens that can be generated are:

- SAML assertion
- WS-Security Kerberos AP-REQ
- WS-Security user name
- LTPA
- Kerberos SPNEGO
- ICRX



AAA action and access control policy

- To restrict access to resources, add a **AAA action** to a document processing rule
 - AAA action invokes an **access control policy**, or **AAA policy**
- An **access control policy**, or a **AAA policy**, determines whether a requesting client is granted access to a specific resource
 - These policies are filters that accept or deny specific client requests



© Copyright IBM Corporation 2014

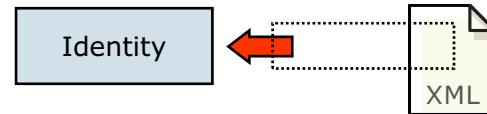
Figure 11-4. AAA action and access control policy

WE601 / ZE6011.1

Notes:

How to define an access control policy (1 of 2)

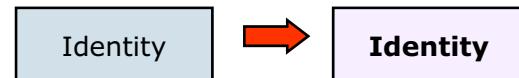
1. Define one or more identity extraction methods



2. Define the authentication method



3. Map authentication credentials (optional)



© Copyright IBM Corporation 2014

Figure 11-5. How to define an access control policy (1 of 2)

WE601 / ZE6011.1

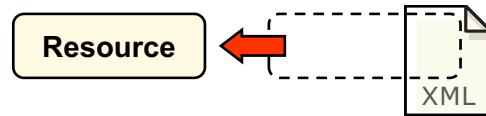
Notes:

The access control policy steps relate directly to the processing stages within the AAA framework. In the first step, the policy defines how the framework retrieves information about the client identity. The framework can treat the requested URL, the client IP address, the HTTP header, or any part of the message, as a client identifier. When it is extracted, the second step describes how to verify the claimed identity that is stored in the message. If the authorization method (which is described on the next slide) expects a different client identifier, the policy can apply a custom style sheet to convert the authentication credentials.

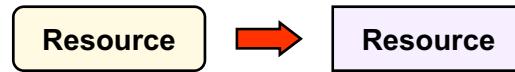


How to define an access control policy (2 of 2)

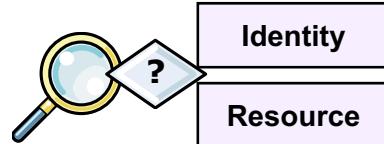
4. Define resource extraction methods



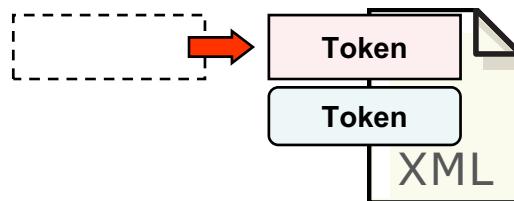
5. Map requested resources (optional)



6. Define the authorization method



7. Specify postprocessing actions (optional)



© Copyright IBM Corporation 2014

Figure 11-6. How to define an access control policy (2 of 2)

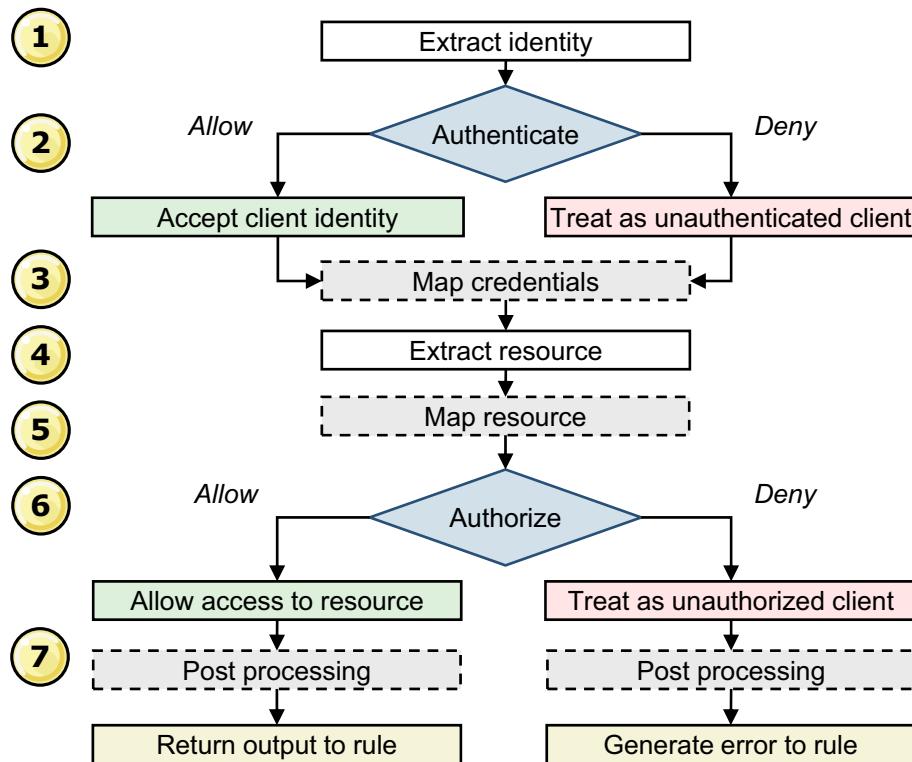
WE601 / ZE6011.1

Notes:

If the authentication step succeeds, the policy determines the resources that the client requests before making a final decision on whether to authorize access. An optional mapping step matches the resource request with a type expected by the authorization method.

When authentication and authorization succeed, monitoring and post processing steps can take place. The monitoring step records whether the access control policy as a whole succeeds or fails. Such information can be used for auditing purposes. Unlike the monitoring step, post processing occurs only if the policy authorizes the resource request. This final step can add more security tokens to the message header.

Access control policy processing



© Copyright IBM Corporation 2014

Figure 11-7. Access control policy processing

WE601 / ZE6011.1

Notes:

The numbers correspond to the access control policy steps detailed on the previous two slides. Keep in mind that the output message is returned to the processing rule, not back to the actual client itself. Similarly, an **On Error** action or an error rule suppresses or handles errors that are generated from a AAA action.

The only part of the post processing step that occurs when authorization fails is the incrementing of the authorization failures counter (if one exists).

Within the post processing step, monitors track the requests.

Scenario 1: Authorize authenticated clients

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - The client communicates to the DataPower SOA appliance over a Secure Sockets Layer (SSL) connection
 - A WS-Security UsernameToken element holds the requesting client identity
 - Verifies the claimed identity of the client against a list that is stored on the DataPower SOA appliance itself
 - The requested resource is the web service operation
 - Allows any authenticated client access to the web service operation

© Copyright IBM Corporation 2014

Figure 11-8. Scenario 1: Authorize authenticated clients

WE601 / ZE6011.1

Notes:

In this scenario, the client includes a WS-Security username token with a password or password digest as a proof of identity. As a good practice, clients send plain text tokens, such as the WS-Security username token, within a secure channel, such as an SSL connection.

The access control policy on the DataPower SOA appliance verifies the user name and password against a built-in user list. It assumes that all authenticated users have full access to any resource protected by the policy.

Scenario 1: Sample SOAP request message

```
<?xml version="1.0" encoding="UTF-8">
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsse="http://...wssecurity-secext-1.0.xsd"
    xmlns:q0="http://east.address.training.ibm.com">
    <soap:Header>
        <wsse:Security>
            <wsse:UsernameToken>
                <wsse:Usernamewsse:Usernamewsse:Passwordwsse:Password>
            </wsse:UsernameToken>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <q0:retrieveAll />
    </soap:Body>
</soap:Envelope>
```

© Copyright IBM Corporation 2014

Figure 11-9. Scenario 1: Sample SOAP request message

WE601 / ZE6011.1

Notes:

The WS-Security username token provides a basic method to transport user credentials to a web service. The password field can be in plain text, or the Secure Hash Algorithm (SHA1) can hash it. Since SHA1 is a well-known algorithm, a hashed password provides a minimal level of security by obfuscating the password. Messages with these identity credentials are sent only over a secure connection.

For the sake of brevity, the URI for the **wsse** namespace declaration is truncated. For the URI, see the WS-Security V1.1 specification.

Within the SOAP body, the child element describes the requested web service operation. In effect, this element identifies the resource that is requested in this call.

 WebSphere Education



Scenario 1: Identify the client

1. Create a AAA policy object on the DataPower SOA appliance
2. Extract the client's identity with the **Password-carrying UsernameToken Element from WS-Security Header** option
3. For the authentication method, **Use AAA information file**
 - Specify the name of the AAA information file in the **URL** field
4. Leave the identity mapping method at **None**

AAA Policy Name: AddressAdmin

Define how to extract a user's identity from an incoming request.

HTTP Authentication Header
 Password-carrying UsernameToken Element from WS-Security Header
 Derived-key UsernameToken Element from WS-Security Header
 BinarySecurityToken Element from WS-Security Header

Define how to authenticate the user.

Accept LTPA token
 Accept SAML assertion with valid signature
 Bind to LDAP server
 Contact ClearTrust server
 Contact IBM Security Access Manager
 Contact Netegrity SiteMinder
 Contact NSS for SAF authentication
 Contact SAML server for SAML Authentication statement
 Contact WS-Trust server for WS-Trust token
 Custom template
 Pass identity token to authorization phase
 Retrieve SAML assertions that corresponds to SAML Browser Artifact
 Use AAA information file
 Use certificate from BinarySecurityToken

URL	<input type="text" value="store:III"/>
	<input type="text" value="AAAInfo.xml"/>
	<input style="margin-right: 5px;" type="button" value="+"/> <input type="button" value="..."/>

© Copyright IBM Corporation 2014

Figure 11-10. Scenario 1: Identify the client

WE601 / ZE6011.1

Notes:

The choices for identity extraction are:

- HTTP Authentication header
- Password-carrying UsernameToken element from WS-Security header
- Derived-key UsernameToken element from WS-Security header
- BinarySecurityToken element from WS-Security header
- WS-SecureConversation identifier
- WS-Trust Base or Supporting token
- Kerberos AP-REQ from WS-Security header
- Kerberos AP-REQ from SPNEGO token
- Subject DN of SSL certificate from connection peer
- Name from SAML Attribute assertion
- Name from SAML Authentication assertion

- SAML Artifact
- Client IP address
- Subject DN from certificate in message signature
- Token extracted from message
- Token extracted as cookie value
- LTPA token
- Processing metadata
- Custom style sheet
- HTML forms-based authentication
- Oauth

The choices for authentication method are:

- Accept LTPA token
- Accept SAML assertion with valid signature
- Bind to LDAP server
- Contact ClearTrust server
- Contact IBM Security Access Manager
- Contact Netegrity SiteMinder
- Contact NSS for SAF authentication
- Contact SAML server for SAML Authentication statement
- Contact WS-Trust server for WS-Trust token
- Custom template
- Pass identity token to authorization phase
- Retrieve SAML assertions that correspond to SAML Browser Artifact
- Use AAA information file
- Use certificate from BinarySecurityToken
- Use established WS-SecureConversation security context
- Use RADIUS server
- Validate Kerberos AP-REQ for server principal
- Validate signer certificate for digitally signed message
- Validate SSL certificate from connection peer



Scenario 1: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method

- The name of the child element in the SOAP body of the request is the request element name

6. Leave the resource mapping method at **None**

7. For the authorization method, allow any request from an authenticated client to proceed

Resource Identification Methods	<input type="checkbox"/> URL Sent to Back End <input type="checkbox"/> URL Sent by Client <input type="checkbox"/> URI of Toplevel Element in the Message <input checked="" type="checkbox"/> Local Name of Request Element <input type="checkbox"/> HTTP Operation (GET/POST) <input type="checkbox"/> XPath Expression <input type="checkbox"/> Processing Metadata
Define how to map resources. Method <input type="text" value="None"/> *	
Define how to authorize a request. <ul style="list-style-type: none"> <input type="radio"/> AAA information file <input checked="" type="radio"/> Allow any authenticated client <input type="radio"/> Always allow <input type="radio"/> Check membership in LDAP group <input type="radio"/> Contact ClearTrust server <input type="radio"/> Contact IBM Security Access Manager <input type="radio"/> Contact Netegrity SiteMinder <input type="radio"/> Contact NSS for SAF authorization <input type="radio"/> Contact OAuth STS <input type="radio"/> Custom template <input type="radio"/> Generate SAML Attribute query <input type="radio"/> Generate SAML Authorization query <input type="radio"/> Use SAML attributes from authentication <input type="radio"/> Use XACML Authorization decision 	

© Copyright IBM Corporation 2014

Figure 11-11. Scenario 1: Authorize access to resources

WE601 / ZE6011.1

Notes:

The authorization choices are:

- AAA information file
- Allow any authenticated client
- Always allow
- Check membership in LDAP group
- Contact ClearTrust server
- Contact IBM Security Access Manager
- Contact Netegrity SiteMinder
- Contact NSS for SAF authorization
- Contact OAuth STS
- Custom template
- Generate SAML Attribute query

- Generate SAML Authorization query
- Use SAML attributes from authentication
- Use XACML Authorization decision



Scenario 2: Security token conversion

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - The client communicates to the DataPower SOA appliance over a Secure Sockets Layer (SSL) connection
 - The HTTP BASIC-AUTH header information holds the identity of the requesting client
 - Generates a WS-Security UsernameToken element corresponding to the HTTP BASIC-AUTH header
 - Defers the authentication and authorization tasks to the back-end web service

© Copyright IBM Corporation 2014

Figure 11-12. Scenario 2: Security token conversion

WE601 / ZE6011.1

Notes:

HTTP BASIC-AUTH is the basic authentication scheme. See the following slide for an example of an HTTP request message with a basic authentication header.

Scenario 2: Sample HTTP request message

```

POST /EastAddress/services/AddressSearch HTTP/1.1
Host: www.example.com
Content-type: text/xml; charset=utf-8
Content-length: 237
Authorization: Basic T3phaxI6U2hlaWtoTkJha2U=

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:q0="http://east.address.training.ibm.com">
    <soap:Header />
    <soap:Body>
        <q0:retrieveAll />
    </soap:Body>
</soap:Envelope>
```

© Copyright IBM Corporation 2014

Figure 11-13. Scenario 2: Sample HTTP request message

WE601 / ZE6011.1

Notes:

In this scenario, the HTTP authorization header field is used for authentication. Remember that the client encoded the user name and password in Base64. This encoding method is known, hence, the client uses an SSL connection to keep the contents of this message private.

Base64 is a binary-to-text encoding scheme by printable (mostly alphanumeric) characters. As a MIME content transfer encoding, it is used to encode binary data into email messages.

In the HTTP basic authentication scheme, the user name and password are concatenated with a colon (:) before it is encoded byBase64. For example, the user name "Alice" and the password "ond3mand" become "Alice:ond3mand". In Base64 encoding, the user name and password string is "QWxpY2U6b25kM21hbmQ=".

 WebSphere Education 

Scenario 2: Identify the client

1. Create a AAA policy object on the DataPower SOA appliance
2. Extract the client's identity with the **HTTP Authentication header** option
 - The value within the Authorization HTTP header represents the HTTP authentication header
3. For the authentication method, specify **Pass identity token to authorization phase**
4. Leave the identity mapping method at **None**

1. Create a AAA policy object on the DataPower SOA appliance	<input checked="" type="checkbox"/> HTTP's Authentication Header <input type="checkbox"/> Password-carrying UsernameToken Element from WS-Security Header <input type="checkbox"/> Derived-key UsernameToken Element from WS-Security Header <input type="checkbox"/> BinarySecurityToken Element from WS-Security Header <input type="checkbox"/> WS-SecureConversation Identifier <input type="checkbox"/> WS-Trust Base or Supporting Token <input type="checkbox"/> Kerberos AP-REQ from WS-Security Header <input type="checkbox"/> Kerberos AP-REQ from SPNEGO Token <input type="checkbox"/> Subject DN of the SSL Certificate from the Connection Peer <input type="checkbox"/> Name from SAML Attribute Assertion <input type="checkbox"/> Name from SAML Authentication Assertion
--	---

2. Extract the client's identity with the HTTP Authentication header option	<input type="radio"/> Contact SAML server for SAML Authentication statement <input type="radio"/> Contact WS-Trust server for WS-Trust token <input type="radio"/> Custom template <input checked="" type="radio"/> Pass identity token to authorization phase <input type="radio"/> Retrieve SAML assertions that corresponds to SAML Browser Artifact <input type="radio"/> Use AAA information file <input type="radio"/> Use certificate from BinarySecurityToken
--	---

Define how to map credentials.	
Method	<input type="text" value="none"/> *

© Copyright IBM Corporation 2014

Figure 11-14. Scenario 2: Identify the client

WE601 / ZE6011.1

Notes:



Scenario 2: Authorize access to resources

5. Select **Local Name of Request Element** as the resource extraction method
 - The name of the child element in the SOAP body of the request is the request element name
6. Leave the resource mapping method at **None**
7. Set the authorization method to always allow requests
8. In the postprocessing step, add the WS-Security Username Token

Resource Identification Methods	<input type="checkbox"/> URL Sent to Back End <input type="checkbox"/> URL Sent by Client <input type="checkbox"/> URI of Toplevel Element in the Message <input checked="" type="checkbox"/> Local Name of Request Element <input type="checkbox"/> HTTP Operation (GET/POST) <input type="checkbox"/> XPath Expression <input type="checkbox"/> Processing Metadata *
Define how to authorize a request.	
<input type="radio"/> AAA information file <input type="radio"/> Allow any authenticated client <input checked="" type="radio"/> Always allow <input type="radio"/> Check membership in LDAP group <input type="radio"/> Contact ClearTrust server	
Choose any post processing.	
Include Password WS-Security UsernameToken Password Type Actor/Role Identifier	<input checked="" type="radio"/> on <input type="radio"/> off Digest [Text input field]

© Copyright IBM Corporation 2014

Figure 11-15. Scenario 2: Authorize access to resources

WE601 / ZE6011.1

Notes:

The **Run Custom Post Processing Stylesheet** setting applies a custom style sheet to the outgoing request message. This setting does not require enablement for a built-in post processing step, such as adding a WS-Security username token.

The slide does not show the setting of the "Add WS-Security Username Token" to **on**. When this option is set, the page repaints to include the fields that are shown ("Include Password" and the others).



Scenario 3: Multiple identity extraction methods

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - Uses either a WS-Security UsernameToken element or a BinarySecurityToken element from the WS-Security header to determine the client's identity
 - Verifies the identity of the client
 - The requested resource is the web service operation
 - Allows any authenticated client access to the web service operation

© Copyright IBM Corporation 2014

Figure 11-16. Scenario 3: Multiple identity extraction methods

WE601 / ZE6011.1

Notes:

For identity extraction methods, the policy runs **all** checked methods. The system runs the methods in the order that is presented in the check box list. Afterward, the system concatenates all identities that are found for authentication. This scheme allows different clients to use different identification methods.

However, if a client includes more than one identifier in the message, **both** identifiers must pass the authentication stage.



Scenario 3: Identify the client

1. Create a AAA policy object on the DataPower SOA appliance
2. Extract the client's identity from the UserName element or a BinarySecurityToken
 - Separate WS-Security token profiles describe the structure of the UsernameToken and the BinarySecurityToken
3. For the authentication method, specify **Bind to LDAP server**
 - The LDAP directory server provides an external list of authenticated users
4. Leave the identity mapping method at **None**

Define how to extract a user's identity from an incoming request.
<input type="checkbox"/> HTTP Authentication header <input checked="" type="checkbox"/> Password-carrying UsernameToken element from WS-Security header <input type="checkbox"/> Derived-key UsernameToken element from WS-Security header <input checked="" type="checkbox"/> BinarySecurityToken element from WS-Security header <input type="checkbox"/> WS-SecureConversation identifier <input type="checkbox"/> WS-Trust Base or Supporting token
Define how to authenticate the user.
<input type="radio"/> Accept LTPA token <input type="radio"/> Accept SAML assertion with valid signature <input checked="" type="radio"/> Bind to LDAP server <input type="radio"/> Contact ClearTrust server <input type="radio"/> Contact IBM Security Access Manager
Define how to map credentials.
Method <input type="text" value="none"/> *

© Copyright IBM Corporation 2014

Figure 11-17. Scenario 3: Identify the client

WE601 / ZE6011.1

Notes:

Scenario 3: Authorize access to resources

5. Select **Local Name of Request Element** as the resource extraction method
- The name of the child element in the SOAP body of the request is the request element name

Resource Identification Methods	<input type="checkbox"/> URL Sent to Back End <input type="checkbox"/> URL Sent by Client <input type="checkbox"/> URI of Toplevel Element in the Message <input checked="" type="checkbox"/> Local Name of Request Element <input type="checkbox"/> HTTP Operation (GET/POST) <input type="checkbox"/> XPath Expression <input type="checkbox"/> Processing Metadata *
--	--

6. Leave the resource mapping method at **None**

Define how to map resources.

Method	<input style="width: 100px; height: 25px; border: 1px solid black; padding: 2px; margin-right: 5px;" type="button" value="none"/> *
--------	--

7. For the authorization method, allow any request from an authenticated client to proceed

Define how to authorize a request.

<input type="radio"/> AAA information file <input checked="" type="radio"/> Allow any authenticated client <input type="radio"/> Always allow <input type="radio"/> Check membership in LDAP group <input type="radio"/> Contact ClearTrust server
--

© Copyright IBM Corporation 2014

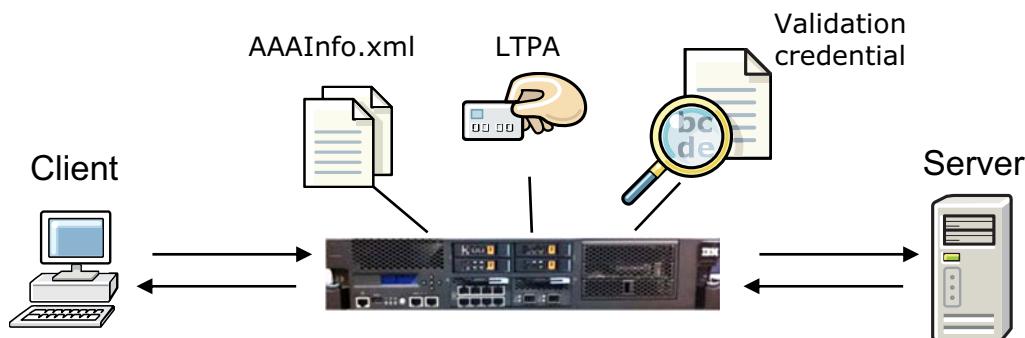
Figure 11-18. Scenario 3: Authorize access to resources

WE601 / ZE6011.1

Notes:

Internal access control resources

- Authentication and authorization can be performed on the DataPower box by:
 - AAA file: XML file that contains validation information for the AAA steps (authenticate, authorize, map credentials, map resource)
 - LTPA: token type that the IBM WebSphere Application Server and Lotus Domino products use
 - Validation credential object: list of certificates that are used to validate the incoming digital signature



© Copyright IBM Corporation 2014

Figure 11-19. Internal access control resources

WE601 / ZE6011.1

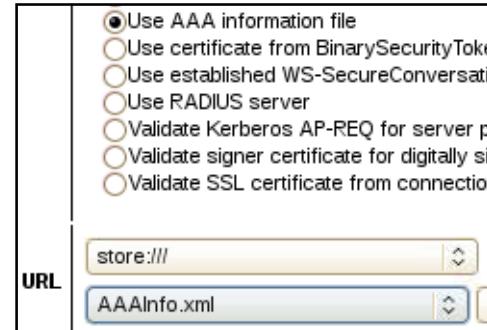
Notes:

The validation credential object references a list of certificates on the appliance that validate the incoming digital signature. This object is also used when configuring client-side SSL.



AAA XML file

- The AAA XML file is used to validate the credentials in a AAA policy
- Used by the following AAA steps:
 - Authenticate
 - Authorize
 - Map credentials
 - Map resource
- Useful for testing of AAA policy when off-box resources not available
 - Use in production to maintain small list of AAA credentials
- For the authenticate or authorize step in the AAA policy, select **Use AAA information file**
 - Select an existing XML file or create a AAA file



© Copyright IBM Corporation 2014

Figure 11-20. AAA XML file

WE601 / ZE6011.1

Notes:

The DataPower WebGUI includes a set of wizard pages that make it easy to create a AAA XML file. When you attempt to create a AAA Info file, or edit an existing one, a AAA Info file editor opens.

Example AAA XML file

```
<aaa:AAAInfo xmlns:aaa="http://www.datapower.com/AAAInfo">
    <aaa:FormatVersion>1</aaa:FormatVersion>
    <aaa:Filename>local:///AddressInfo.xml</aaa:Filename>
    <aaa:Summary>
        AAA file to validate credentials for Address users
    </aaa:Summary>

    <aaa:Authenticate>
        <aaa:Username>AddressAdmin</aaa:Username>
        <aaa:Password>password</aaa:Password>
        <aaa:OutputCredential>
            AddressUser
        </aaa:OutputCredential>
    </aaa:Authenticate>
</aaa:AAAInfo>
```

© Copyright IBM Corporation 2014

Figure 11-21. Example AAA XML file

WE601 / ZE6011.1

Notes:

The Authenticate step uses this AAA XML file to validate the extracted identity. The incoming identity has a user name of AddressAdmin and password **of** password.



Lightweight Third Party Authentication

- Lightweight Third Party Authentication (LTPA) is a single sign-on (SSO) credential format for distributed, multiple application server environments
 - LTPA is a proprietary token type that the IBM WebSphere Application Server and Lotus Domino products use
- The purpose of LTPA is threefold:
 - Propagates the caller identity through a unique identifier of the client
 - Establishes a trust relationship between two servers, with one as the client and one as the server, through a signed token
 - Keeps the information within the token secret by signing and encrypting the token
 - A set of key files must be uploaded to the DataPower SOA appliance to decrypt and validate the digital signature within the token

© Copyright IBM Corporation 2014

Figure 11-22. Lightweight Third Party Authentication

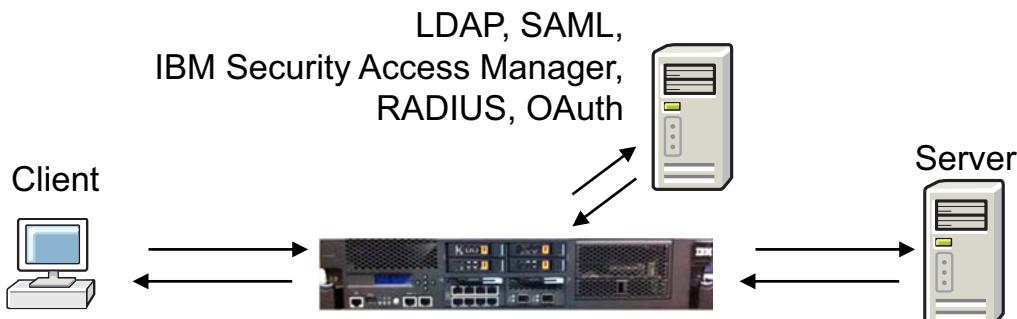
WE601 / ZE6011.1

Notes:

For more information, see the article *WS-Policy security integration between DataPower and WebSphere Application Server*, which includes a section on using the LTPA token:

http://www.ibm.com/developerworks/websphere/library/techarticles/0911_rasmussen/0911_rasmussen.html

External access control resource



- Delegates the authentication and authorization task to an external security system
- The authentication and authorization tasks can be delegated to the same system or to separate systems
 - For example, an LDAP directory tracks client identities, while IBM Tivoli Access Manager determines whether the client has access to the specified resource
 - The **map credentials** and **map resource** steps convert the security token to match the input that the authorization step requires

© Copyright IBM Corporation 2014

Figure 11-23. External access control resource

WE601 / ZE6011.1

Notes:

It is also possible to do authentication and authorization on an IBM Security Access Manager system. IBM Security Access Manager can be configured to use its own user repository for authentication instead of using a separate, external Lightweight Directory Access Protocol (LDAP) server.

The list of external access controls on this slide is merely an example. For a full list of security products and specifications that are supported, see the product documentation.



Lightweight Directory Access Protocol

- LDAP provides a means of storing and retrieving information about people, groups, or objects on a centralized X.500 or LDAP directory server
 - **X.500** enables the information to be organized and queried, by LDAP, from multiple web servers by various attributes
 - **LDAP** reduces system resources by including only a functional subset of the original X.500 Directory Access Protocol (DAP)
- A few facts about LDAP:
 - An LDAP directory is a tree of directory entries
 - The **distinguished name (DN)** is a unique identifier for entries
 - A **bind** operation authenticates the client by sending the clients distinguished name and password in cleartext
 - Use an SSL connection to keep LDAP queries secret

© Copyright IBM Corporation 2014

Figure 11-24. Lightweight Directory Access Protocol

WE601 / ZE6011.1

Notes:

Security Assertion Markup Language

- SAML provides an XML-based framework for exchanging authentication, authorization, and attribute assertions between the entities
 - Provides a standard, platform-neutral way for exchanging security information between a security system and an application that trusts the security system
 - Expands the authentication and authorization trust model from existing systems by allowing new systems to delegate trust management to other systems
 - Includes protocol for requesting this information from security authorities
 - For example, SOAP and HTTP bindings

© Copyright IBM Corporation 2014

Figure 11-25. Security Assertion Markup Language

WE601 / ZE6011.1

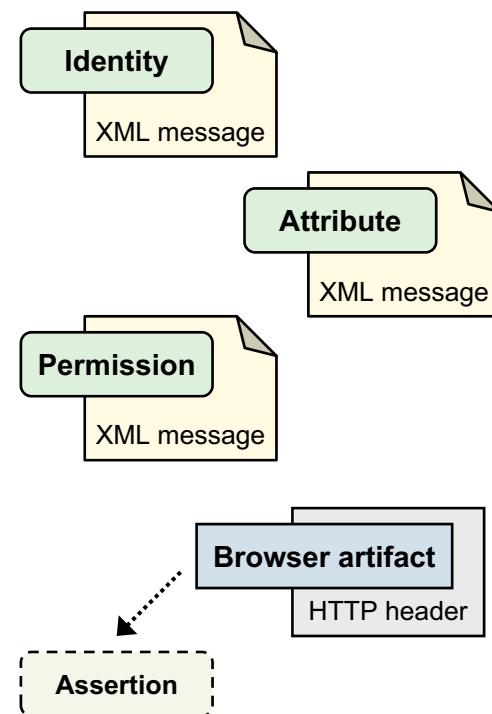
Notes:

Federated Security Systems require an interoperable way of sending security information from one system to another. The Security Assertion Markup Language (SAML) is designed specifically for this purpose. It is analogous to how the SOAP specification defines a messaging model for transferring information between web service clients and servers.

SAML allows clients or intermediaries to embed claims, or assertions, into the message. One common use for assertions is single sign-on: after a security server authenticates a client, a SAML authentication statement is tagged to the client request. Subsequent systems that process the request need only to trust the assertion instead of authenticating the client again.

Types of SAML assertions

- Three main types of XML-based SAML assertions exist:
 - **Authentication** assertions represent the identity of the specified subject that is verified by another entity
 - **Attribute** assertions represent any attributes that are associated with the specified subject
 - **Authorization** decision assertions represent whether the specified subject is granted or denied access to a specified resource
- In addition, the HTTP binding provides a non-XML reference:
 - A **SAML artifact** that is embedded in the URL query string provides a reference to an actual SAML assertion that is stored in a remote site



© Copyright IBM Corporation 2014

Figure 11-26. Types of SAML assertions

WE601 / ZE6011.1

Notes:

In plain terms, here are some typical statements that the three types of SAML assertions make:

- Authentication statement: "I am Bob Smith."
- Attribute statement: "Bob Smith is a payroll manager."
- Authorization decision statement: "Payroll managers can run the Payroll Update web service."

These assertions avoid repeating the same checks on the same message as it passes through different systems. In addition, assertion statements delegate the authentication and authorization task to a separate server.

The last point describes the HTTP binding for SAML. Remember that SAML is not only used for web services. For example, a web application server might want to verify a SAML assertion in a single sign-on (SSO) scenario. Without even examining the HTTP request message, the server extracts and dereferences a SAML assertion from the URL query string.

Scenario 4: Authorize valid SAML assertions

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - A SAML authentication assertion holds the requesting client identity
 - Accepts the claimed identity of the client if the digital signature of the SAML assertion is valid
 - The requested resource is defined as an attribute in the SAML assertion
 - Allows any authenticated client with a specific SAML attribute access to the web service operation

© Copyright IBM Corporation 2014

Figure 11-27. Scenario 4: Authorize valid SAML assertions

WE601 / ZE6011.1

Notes:

In this example, the request message contains a SAML authentication statement and a SAML attribute statement. The authentication statement claims that the current requester is verified during a previous processing step. The access control policy accepts this claim if and only if the digital signature that was used to sign the claim is valid.

An application-specific SAML attribute describes the resource that the client requests. The policy authorizes the request if the current requester is an authorized user.

Scenario 4: SAML authentication statement (1 of 2)

```

<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
  AssertionID="IDd600a593-4e13-44d9-829a-3055600c46ca"
  IssueInstant="2006-07-28T18:51:02Z"
  Issuer="http://training.ibm.com/security/"
  MajorVersion="1" MinorVersion="1">
  <saml:Conditions NotBefore="2006-07-28T18:51:02Z"
    NotOnOrAfter="2006-07-28T18:54:02Z"/>
  <saml:AuthenticationStatement
    AuthenticationInstant="2006-07-28T18:51:02Z"
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:unspecified">
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
        NameQualifier="http://address.training.ibm.com">
        admin
      </saml:NameIdentifier>
    . . . (continued on next slide)
  
```

© Copyright IBM Corporation 2014

Figure 11-28. Scenario 4: SAML authentication statement (1 of 2)

WE601 / ZE6011.1

Notes:

This example is a SAML assertion that is generated in the post-processing step of an access control policy.

The **Conditions** element defines a window of time in which this statement is valid. This time limit reduces the likelihood of a replay attack.

Within the **AuthenticationStatement**, the **Subject** element describes the identity of the client through a **NameIdentifier** element.

Scenario 4: SAML authentication statement (2 of 2)

. . . (*continued from previous slide*)

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
  </saml:ConfirmationMethod>
</saml:SubjectConfirmation>
</saml:Subject>
<saml:SubjectLocality IPAddress="127.0.0.1"/>
</saml:AuthenticationStatement>
</saml:Assertion>
```

© Copyright IBM Corporation 2014

Figure 11-29. Scenario 4: SAML authentication statement (2 of 2)

WE601 / ZE6011.1

Notes:

The **SubjectConfirmation** element describes which party backs up the claim. In this example, the message sender vouches for the validity of this claim.

It is a good practice to sign SAML assertions digitally to maintain the integrity of the claim.

Scenario 4: SAML attribute statement

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    ... MajorVersion="1" MinorVersion="1">
    <saml:Conditions NotBefore="2006-07-28T18:51:02Z"
        NotOnOrAfter="2006-07-28T18:54:02Z"/>
    <saml:AttributeStatement>
        <saml:Subject>
            <saml:NameIdentifier
                Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
                NameQualifier="http://address.training.ibm.com">
                admin
            </saml:NameIdentifier>
        </saml:Subject>
        <saml:Attribute
            AttributeName="EastAddressSearch"
            AttributeNamespace="http://address.training.ibm.com">
            <saml:AttributeValue>
                Query
            </saml:AttributeValue>
        </saml:Attribute>
    </saml:AttributeStatement>
</saml:Assertion>
```

© Copyright IBM Corporation 2014

Figure 11-30. Scenario 4: SAML attribute statement

WE601 / ZE6011.1

Notes:

This example is a SAML attribute statement, holding application-specific information.

Similar to a SAML authentication statement, the **NameIdentifier** element describes the subject that added the attribute.

The **Attribute** element describes application-specific information. For example, a SAML attribute element can encapsulate fields from an LDAP directory entry. The system can use this additional information about the subject to make an authorization decision.

Again, it is a good practice to sign SAML assertions digitally to maintain the integrity of the claim.



Scenario 4: Identify the client

1. Create a AAA policy object on the DataPower SOA appliance
2. Extract the client's identity by the **Name from SAML authentication assertion** option
3. For the authentication method, select **Accept a SAML Assertion with a Valid Signature**
 - Specify the validation credential for the SAML signature
4. Leave the identity mapping method at **None**

Identification Methods	<input type="checkbox"/> Name from SAML attribute assertion <input checked="" type="checkbox"/> Name from SAML authentication assertion <input type="checkbox"/> SAML artifact <input type="checkbox"/> Client IP address <input type="checkbox"/> Subject DN from certificate in the message's signature
Define how to authenticate the user.	
Method	<input checked="" type="radio"/> Accept a SAML Assertion with a Valid Signature <input type="radio"/> Accept an LTPA token <input type="radio"/> Bind to Specified LDAP Server <input type="radio"/> Contact a SAML Server for a SAML Authentication
SAML Signature Validation Credentials <div style="display: flex; align-items: center;"> <div style="border: 1px solid #ccc; padding: 2px 5px; margin-right: 5px;">pubcert</div> <div style="border: 1px solid #ccc; padding: 2px 5px; margin-right: 5px;">+</div> <div style="border: 1px solid #ccc; padding: 2px 5px; margin-right: 5px;">...</div> </div>	

© Copyright IBM Corporation 2014

Figure 11-31. Scenario 4: Identify the client

WE601 / ZE6011.1

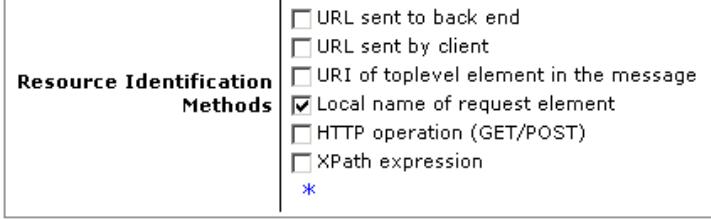
Notes:

To verify the signature of the SAML assertion, the access control policy needs the validation credential.

Scenario 4: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method
 - The name of the child element in the SOAP body of the request is the request element name
6. For the authorization method, **Use SAML Attributes from Authentication**
 - Set the SAML attribute that matches type as **Must match at least one name and value**
7. Select **SAML Attributes** from the authentication method page



Define how to authorize a request.	
Method	<input type="radio"/> Generate a SAML Authorization Query <input type="radio"/> Generate a SAML Attribute Query <input checked="" type="radio"/> Use SAML Attributes from Authentication <input type="radio"/> XPath <input type="radio"/> Any <input type="radio"/> All <input checked="" type="radio"/> Any-Value <input type="radio"/> All-Values
Type	

SAML Attributes

[Back](#) |
 [Next](#) |
 [Advanced](#) |
 [Cancel](#)

© Copyright IBM Corporation 2014

Figure 11-32. Scenario 4: Authorize access to resources

WE601 / ZE6011.1

Notes:

When authorizing requests based on SAML attributes, you must specify one or more expected attributes in a separate page. The following slide describes how to enter in the list of expected SAML attributes.



Scenario 4: Match SAML attributes

8. On the **SAML Attributes** page, click **Add**
9. Declare the expected SAML attribute values within an SAML attribute statement

- The namespace URI and local name represent the qualified name for the SAML attribute
- The attribute value is application-specific; it can be used to represent the identity of the client or the name of a requested resource

Add a SAML Attribute

Namespace URI	<input type="text" value="http://address.trainir"/>
Local Name	<input type="text" value="EastAddressSearch"/>
Attribute Value	<input type="text" value="Query"/>

Submit **Cancel**

Configure an Access Control Policy

SAML Attributes

Namespace URI	Local Name	Attribute Value
<input type="text" value="http://address.training.ibm.com"/>	<input type="text" value="EastAddressSearch"/>	<input type="text" value="Query"/>

© Copyright IBM Corporation 2014

Figure 11-33. Scenario 4: Match SAML attributes

WE601 / ZE6011.1

Notes:



Access control policy by SAML information

- Identity extraction methods:
 - Name from SAML attribute assertion <saml:Subject> element
 - Name from SAML authentication assertion <saml:Subject> element
 - SAML browser artifact from the URL query string
- Authentication methods:
 - Accept a SAML assertion with a valid signature
 - Retrieve SAML assertions corresponding to a SAML browser artifact
 - Contact a SAML server for a SAML authentication statement
- Authorization methods:
 - Generate a SAML authorization query
 - Generate a SAML attribute query
- Postprocessing:
 - Generate a SAML V1.0, V1.1, or V2.0 assertion

© Copyright IBM Corporation 2014

Figure 11-34. Access control policy by SAML information

WE601 / ZE6011.1

Notes:

In addition to the previously covered scenarios, an access control policy can parse any token type and make a SAML authorization or attribute query to an external server. To avoid repeating security checks, the policy can generate a SAML assertion during the post processing stage.



Unit summary

Having completed this unit, you should be able to:

- Describe the AAA framework within the WebSphere DataPower SOA Appliance
- Explain the purpose of each step in an access control policy
- Authenticate and authorize web service requests with:
 - WS-Security Username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

© Copyright IBM Corporation 2014

Figure 11-35. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: To authenticate a client without using an external access control resource, compare the client's credentials against a custom DataPower AAA XML file or validate the digital signature that is used to sign the credential.
2. True or False: Mapping credentials and requested resources allows an access control policy to use different methods for authorization. The authorization step can use tokens that were not part of the original request message.
3. True or False: The postprocessing step in an access control policy adds more information to the outgoing request message or transforms the message itself.

© Copyright IBM Corporation 2014

Figure 11-36. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. True.

2. True.

3. True.

© Copyright IBM Corporation 2014

Figure 11-37. Checkpoint answers

WE601 / ZE6011.1

Notes:



Exercise 9



Web service authentication and
authorization

© Copyright IBM Corporation 2014

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

9.0

WE601 / ZE6011.1

Figure 11-38. Exercise 11

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Configure an action to enforce authentication and authorization policies
- Configure an action to verify an SAML assertion token for authentication and authorization purposes

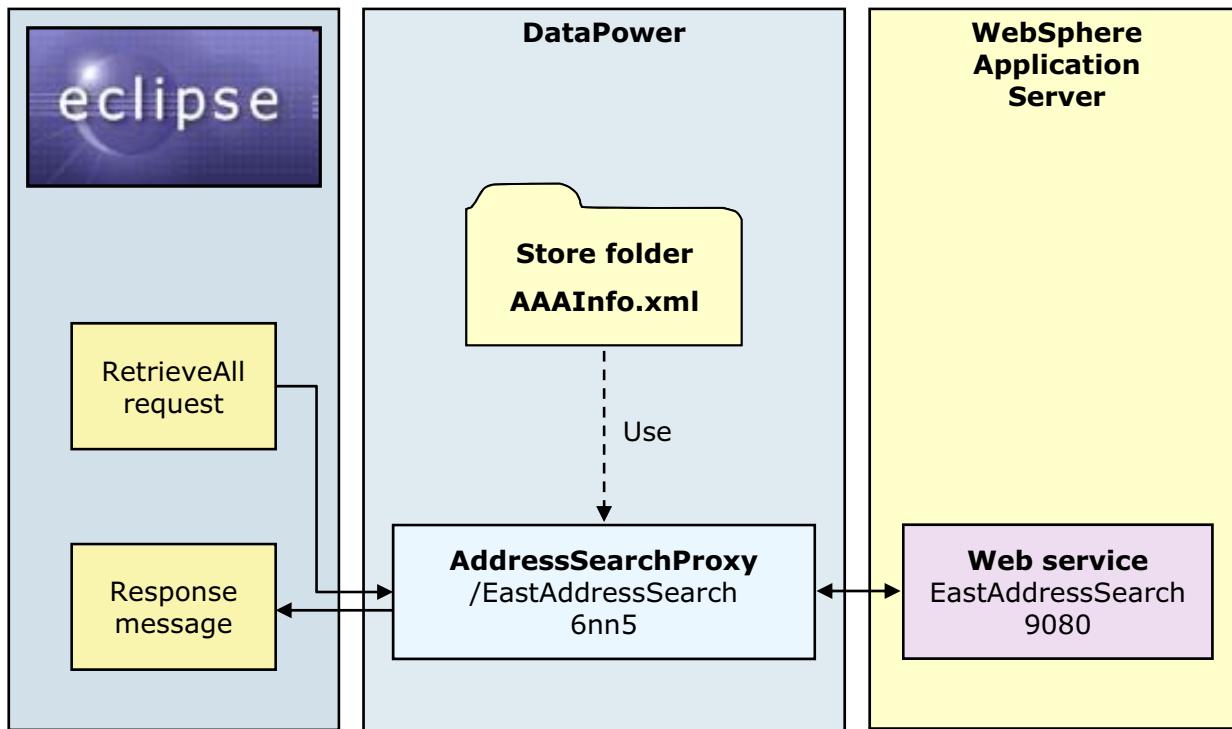
© Copyright IBM Corporation 2014

Figure 11-39. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview



© Copyright IBM Corporation 2014

Figure 11-40. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 12. OAuth overview and DataPower implementation

What this unit is about

This unit introduces the OAuth framework and its DataPower implementation.

What you should be able to do

After completing this unit, you should be able to:

- Describe the OAuth framework
- Describe why OAuth is useful in security scenarios
- Describe the OAuth three-legged scenario
- Explain the role that a DataPower appliance performs in an OAuth framework
- Describe the OAuth configuration options on DataPower: the web token service, the AAA action, the OAuth client profile, and the OAuth client group

How you will check your progress

- Checkpoint
- Exercise: Define a three-legged OAuth scenario that uses DataPower services

References

<http://pic.dhe.ibm.com/infocenter/wsdatap/v6r0m0/index.jsp>

<http://tools.ietf.org/html/rfc6749>



Unit objectives

After completing this unit, you should be able to:

- Describe the OAuth framework
- Describe why OAuth is useful in security scenarios
- Describe the OAuth three-legged scenario
- Explain the role that a DataPower appliance performs in an OAuth framework
- Describe the OAuth configuration options on DataPower: the web token service, the AAA action, the OAuth client profile, and the OAuth client group

© Copyright IBM Corporation 2014

Figure 12-1. Unit objectives

WE601 / ZE6011.1

Notes:

What is OAuth?

- OAuth defines a way for a client to access server resources **on behalf** of another party.
- It provides a way for the user to authorize a third party to their server resources **without** sharing their credentials.
- It **separates** the identity of the resource owner (user) from a third-party application that acts on behalf of the user.
- It allows a user to grant different levels of access to different third-party applications, for a specified duration of time.
- OAuth 2.0 authorization framework:
<http://tools.ietf.org/html/rfc6749>

© Copyright IBM Corporation 2014

Figure 12-2. What is OAuth?

WE601 / ZE6011.1

Notes:

The OAuth specification solves a specific problem: how to delegate access rights to a third-party client that is working on behalf of the user. Before OAuth, third-party applications ask and store the user's user name and password within the application. This process is risky because the server cannot distinguish between the user and the third-party application. One analogy in the real world is to hand over your house keys to a cleaning service. You must have a high degree of trust in the client to give them complete access to your home.

With OAuth, the client does not use your credentials. Instead, an authorization service gives a temporary pass to the client, so it can perform a limited set of tasks in a fixed time period. As the user, you can tell the authorization service to revoke the temporary pass at any time.

Although OAuth is more complicated than handing over your credentials to the client, it is a safer mechanism that gives the user control over the third-party client's actions.

Delegated authorization example

- When you purchase a car, you receive a main key and a **valet** key.



- The valet can use your car with the valet key, with some restrictions:
 - Speed restriction
 - Distance restriction
 - Cannot alter some car functions, such as radio stations
 - Cannot open car storage areas
- When you let the valet borrow the valet key, you are delegating access to certain features of the car to a third party.

© Copyright IBM Corporation 2014

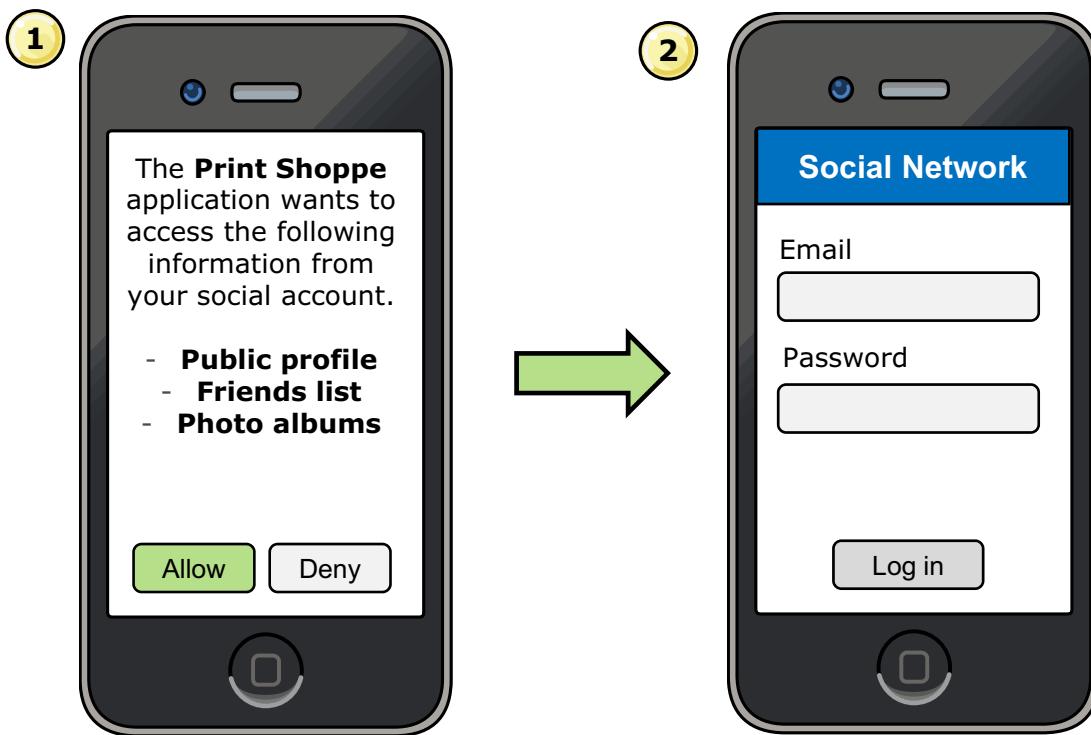
Figure 12-3. Delegated authorization example

WE601 / ZE6011.1

Notes:



Example: Allow third-party access to social account



© Copyright IBM Corporation 2014

Figure 12-4. Example: Allow third-party access to social account

WE601 / ZE6011.1

Notes:

Whenever you sign up for a web-based application or a mobile application, you create an account on the server with a user name and password. The process becomes tedious for the user when they sign up for dozens of applications.

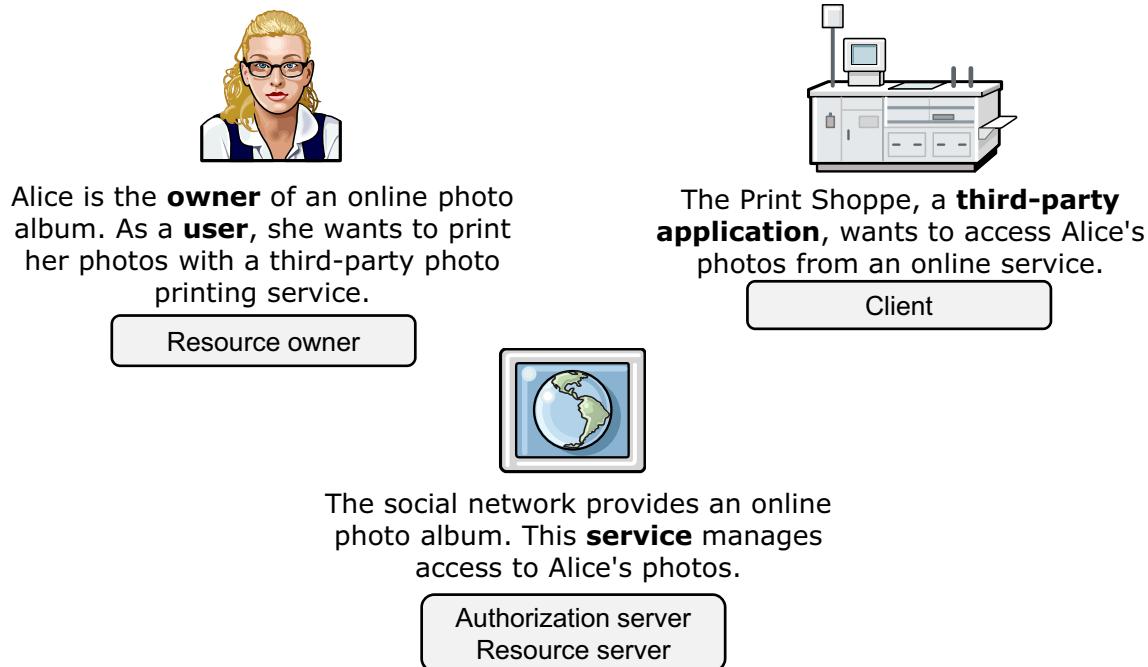
Social networks, such as Facebook and Twitter, already link your identity to a user account. Therefore, many applications use your social network account to create a new account on the application's server.

There are three players in this scenario. You as the user; the third-party application as a client; and the social network as a web-based service. You want the third-party application to access some (but not all) of your information from the service. That is, you want the client to act on your behalf to access resources on the service.

In this example, the third-party application, the Print Shoppe, wants to access your online photo album from your social network account. The application opens a new page from the social network site. After you log in to the social network, the social network service grants an authorization token to the application. At no time does the third-party application see your user name or password on the social network.

Example: Third-party access to online photo album

- OAuth allows social network applications to share resources.



© Copyright IBM Corporation 2014

Figure 12-5. Example: Third-party access to online photo album

WE601 / ZE6011.1

Notes:

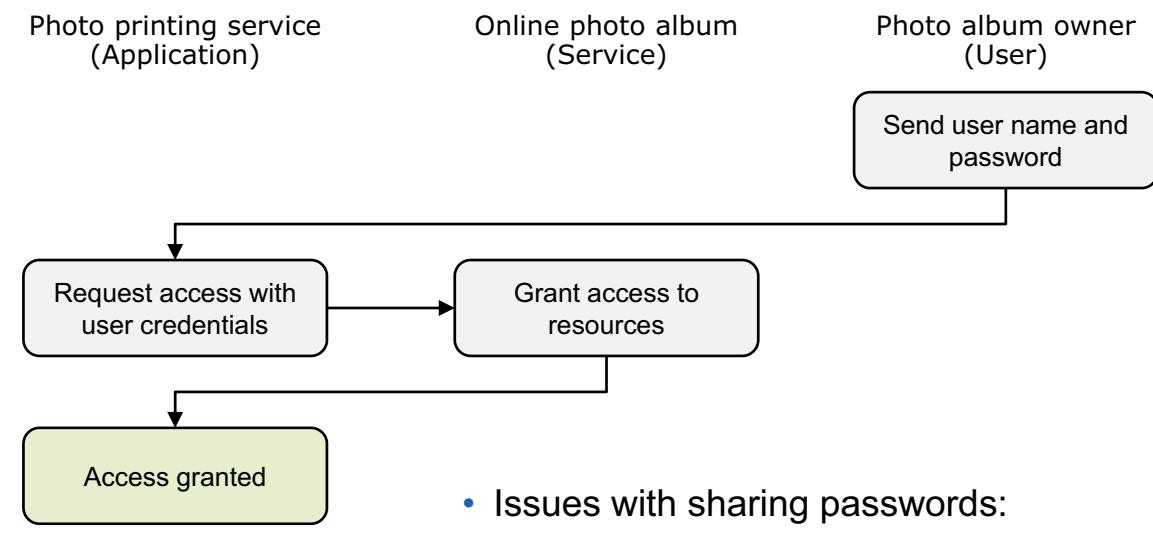
Take a closer look at the three actors in the OAuth scenario. Alice is the **owner** of an online photo album. As the **user**, she wants to print her photos with a third-party photo printing service. The Print Shoppe is a **third-party client application** that wants to access Alice's photos from the online service. Last, the social network is a service that securely stores Alice's photos. This service also manages access to the photos from Alice and third-party applications that are acting on Alice's behalf.

In common usage, this usage pattern is known as the **three-legged scenario**.

The names within the rectangles are the official OAuth roles that each participant plays. In this example, the social network service acts in two roles.



Before OAuth – sharing user passwords



- **Issues with sharing passwords:**

- The service cannot distinguish between the user (resource owner) and the third-party application.
- No method to revoke access for just the third-party application.

© Copyright IBM Corporation 2014

Figure 12-6. Before OAuth - sharing user passwords

WE601 / ZE6011.1

Notes:

Without OAuth, the user must give their user name and password to the third-party application. In turn, the third-party application sends these credentials while posing as the user. For convenience's sake, the application saves a copy of the user name and password.

There are several issues with this scenario. First, the service cannot distinguish between the owner of the resource, and the third-party application. To the service, it is the same user that is accessing the application. This practice is not safe; the user does not know what the application reads or modifies on the service. Second, there is no simple way to revoke access for one particular third-party application. The user must reset their password, which breaks access from all third-party applications.



OAuth – Three-legged scenario

- The three-legged scenario involves three parties: the **user**, the third-party **application**, and the **service**.
 1. The third-party application redirects the web page to the service's page.
 2. The user enters login credentials to the service.
 3. The service asks whether the user wants to grant access to the user's information on the service.
 4. The service gives the third-party application an authorization grant code.
 5. The third-party application sends the grant code to the service and requests an access token.
 6. The service gives the third-party application an access token.
 7. On subsequent requests, the third-party application sends the access token in order to access the user's resources on the service.
- You must use the three-legged scenario if you want to have the third-party application perform actions on the user's protected resources without the application having the user credentials.

© Copyright IBM Corporation 2014

Figure 12-7. OAuth - Three-legged scenario

WE601 / ZE6011.1

Notes:

The term **three-legged scenario** refers to the three main actors in this scenario: the user, the third-party client application, and the service. The service holds the resources that belong to the user. The service also controls access to the resources. In practice, the service is playing two OAuth roles, which is a common practice. The OAuth 2.0 specification does allow the two roles to be implemented separately. In this case, the three-legged scenario is really a four-legged scenario. For example, DataPower automatically separates the two roles as two separate, but related DataPower services.

In the first step, the third-party application redirects the web page to the service's page. This action ensures that the user enters their login credentials directly to the service and not the application.

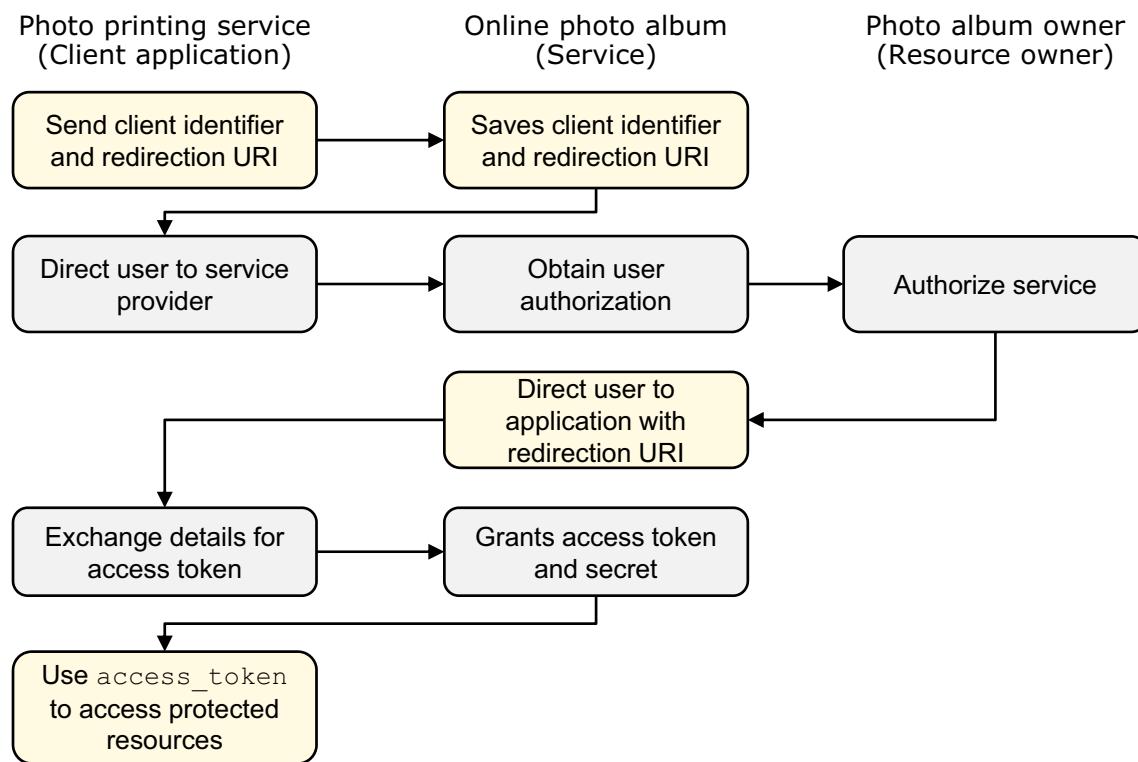
After the service authenticates the user, it asks whether the user wants to authorize the third-party application to access some of the user's protected resources.

The service returns control to the third-party application. It also gives the third-party application an access token: permission to perform some tasks on behalf of the user on the service.

On subsequent requests, the third-party application sends the access token to read or modify the user's resources on its behalf.

The OAuth specification supports other scenarios. However, you must use the three-legged scenario if you want to allow a third-party application to perform actions on the user's protected resources without the application having the user credentials.

OAuth 2.0 – Authorization code grant and access token



© Copyright IBM Corporation 2014

Figure 12-8. OAuth 2.0 - Authorization code grant and access token

WE601 / ZE6011.1

Notes:

The three columns represent the work that each actor does: the third-party client application, the resource server and authorization service, and the user/resource owner.

Before the third-party application prompts the user, it sends an identifier that represents the user. If no such user exists on the service, the authorization workflow stops.

The application also sends a redirection URI so that the service can call the application after it authenticates the user.

The client application directs the user to enter their credentials to the service provider. After the server successfully authenticates the user; it asks whether the user wants to grant access to protected resources, such as the user's private photo album.

After the user authorizes access to the client application, the service directs the user back to the client application. The service also sends an access token to the client application.

To access the protected resources, the client application sends the access token to read or modify protected resources. As an added layer of security, the layer can contact the service and revoke the access token at any time.



OAuth 2.0 roles

Resource owner

An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as a user.

Authorization server

The server that issues access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

Client

An application that makes protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics.

Resource server

The server that hosts the protected resources, capable of accepting and responding to protected resource requests by using access tokens.

Definitions from the OAuth 2.0 specification

© Copyright IBM Corporation 2014

Figure 12-9. OAuth 2.0 roles

WE601 / ZE6011.1

Notes:

These items are the role definitions that are directly from the specification.



OAuth 2.0 roles in the DataPower world (1 of 2)

Resource owner

Usually a user that interacts with the client and the various servers by using a user agent. A user agent can be a web browser, or an interface on a tablet or smartphone.

Authorization server

DataPower supplies a special service, the **Web Token Service**, which acts as an authorization server and token endpoint.

Client

Client code that is running on a web server, application that is running on a tablet or smartphone, client code that is running within a web browser. Varying levels of client credential confidentiality. Defined in DataPower as an **OAuth client** object.

Resource server

The resource server acts as a normal DataPower authentication server in front of the real back-end application, and can also perform other typical DataPower functions.

OAuth roles from a DataPower perspective

© Copyright IBM Corporation 2014

Figure 12-10. OAuth 2.0 roles in the DataPower world (1 of 2)

WE601 / ZE6011.1

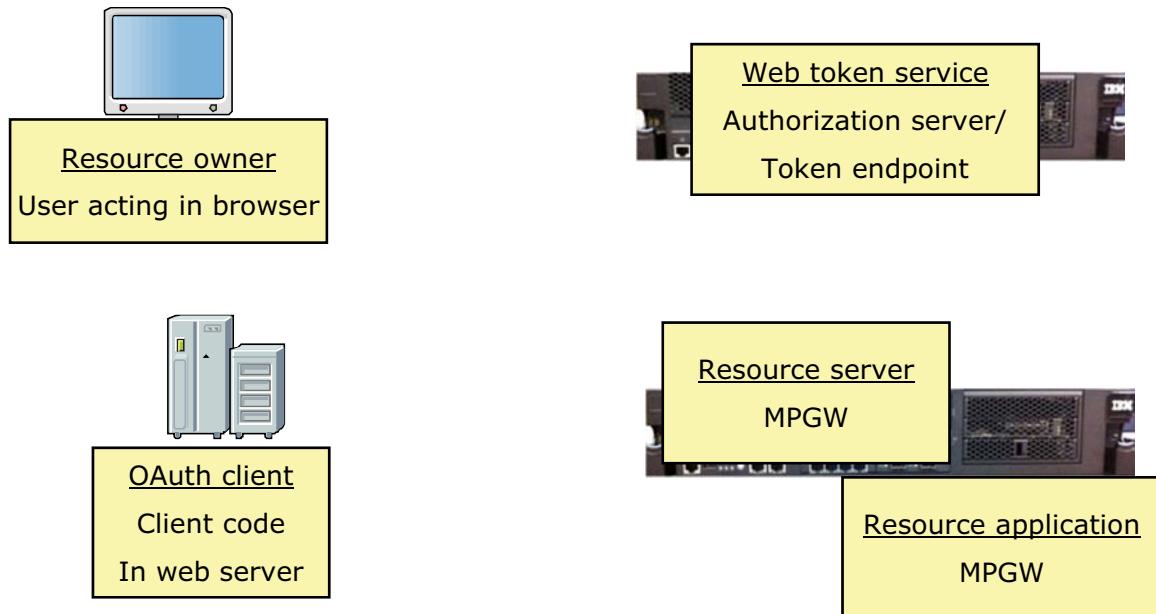
Notes:

These names are the OAuth roles as they can be implemented in a DataPower environment.

DataPower does separate the authorization server function from the resource server function, contrary to many online systems that perform the two functions within the same system.

A web token service is defined as an authorization server and a token endpoint. The authorization behavior is as you expect. The token endpoint refers to the service's ability to supply an access token back to the client.

OAuth 2.0 roles in the DataPower world (2 of 2)



© Copyright IBM Corporation 2014

Figure 12-11. OAuth 2.0 roles in the DataPower world (2 of 2)

WE601 / ZE6011.1

Notes:

The resource server is also considered as the enforcement point (EP) of the secured access to the actual back-end application.

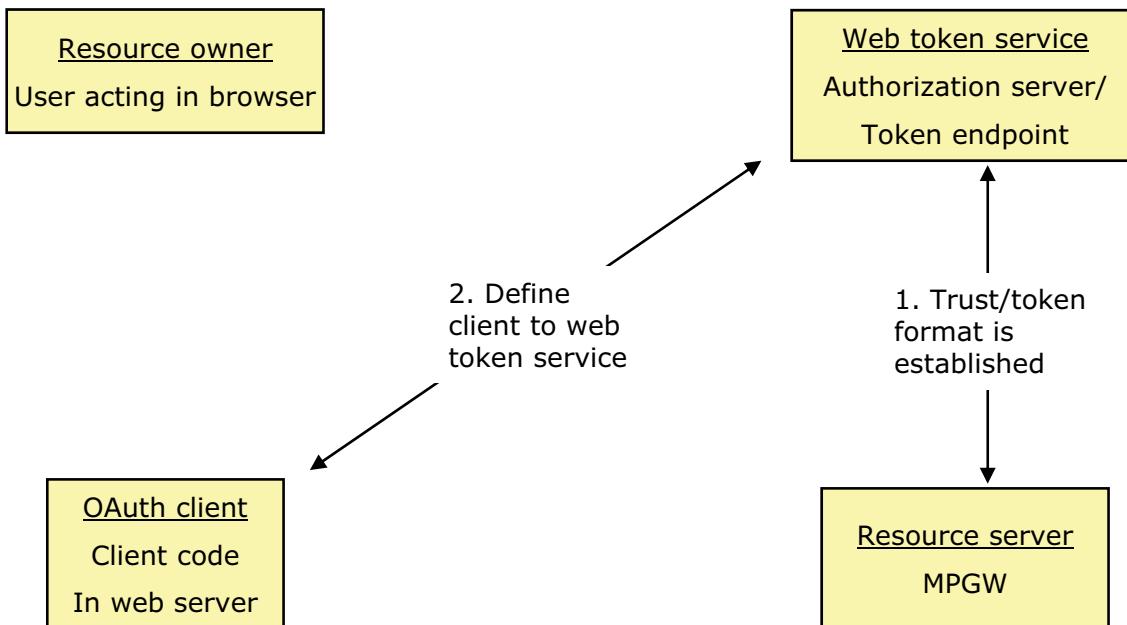
The resource application can be another service on the appliance, or an application service that runs on a server in the trusted domain.

DataPower also supports scenarios where the authorization server is **not** a DataPower service while the resource server is on DataPower, and the reverse situation.

Tivoli Federated Identity Manager (TFIM) also provides an OAuth authorization server implementation. A DataPower based resource server can interact with the Tivoli Federated Identity Manager-based authorization server. The DataPower resource server sends a WS-Trust request to Tivoli Federated Identity Manager to have the access token validated.

Sample three-legged scenario in DataPower (1 of 4)

Activities before the first access



© Copyright IBM Corporation 2014

Figure 12-12. Sample three-legged scenario in DataPower (1 of 4)

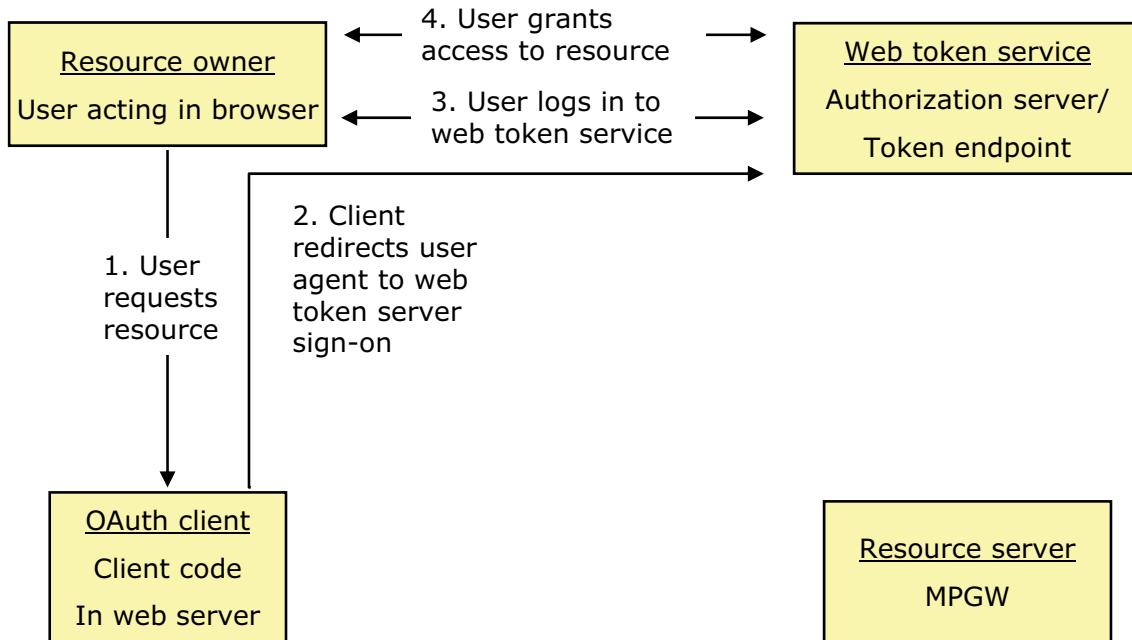
WE601 / ZE6011.1

Notes:

1. The OAuth 2.0 specification does not specify how the access token is formed or validated. DataPower defines its own implementation of the token format and validation, so both the web token service and the resource server use the same implementation.
2. The required parameters for the actual OAuth client are defined in an OAuth client object that the web token service and the resource server can access.

Sample three-legged scenario in DataPower (2 of 4)

User requests the resource, grants access



© Copyright IBM Corporation 2014

Figure 12-13. Sample three-legged scenario in DataPower (2 of 4)

WE601 / ZE6011.1

Notes:

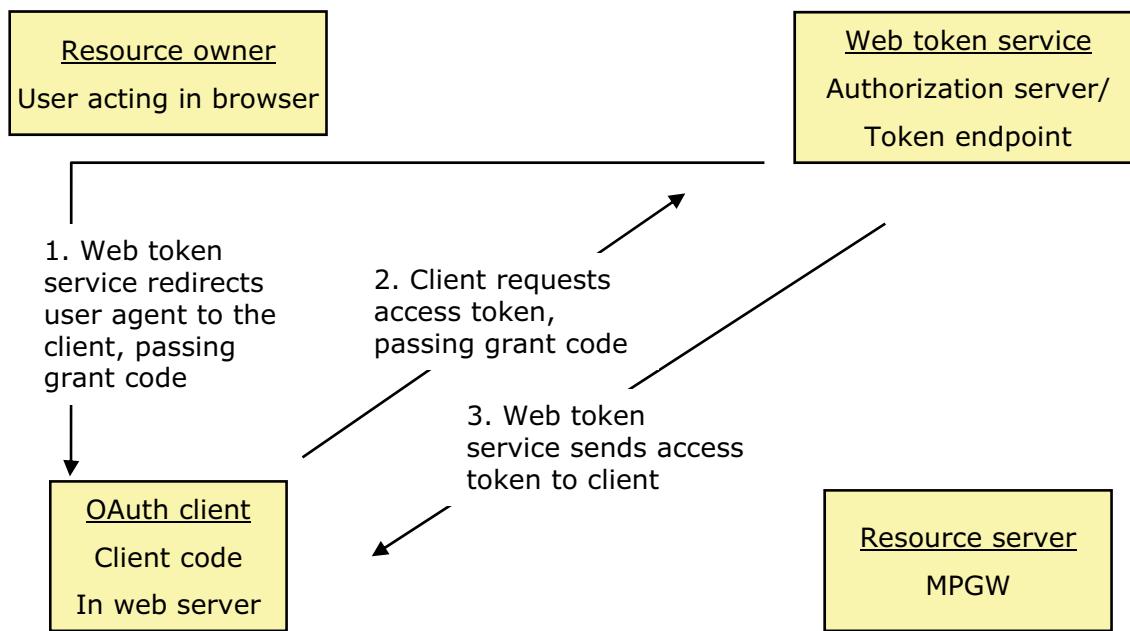
This diagram is a sample flow of the grant type of authorization code. There are more grant types: implicit, resource owner password credentials, and client credentials. For more information, see the specification.

1. The user requests the resource from the client.
2. The client returns a redirection to the user agent that refers to the web token service's URL.
3. The behavior to authenticate and authorize the user depends on the options that are configured in the web token service.
4. The web token service sends an access request to the user (user agent), who chooses to grant access permission or to deny access.

Notice that the client never sees the user's login to the web token service, and does not participate in the access grant.

Sample three-legged scenario in DataPower (3 of 4)

Grant code is used to get access token



© Copyright IBM Corporation 2014

Figure 12-14. Sample three-legged scenario in DataPower (3 of 4)

WE601 / ZE6011.1

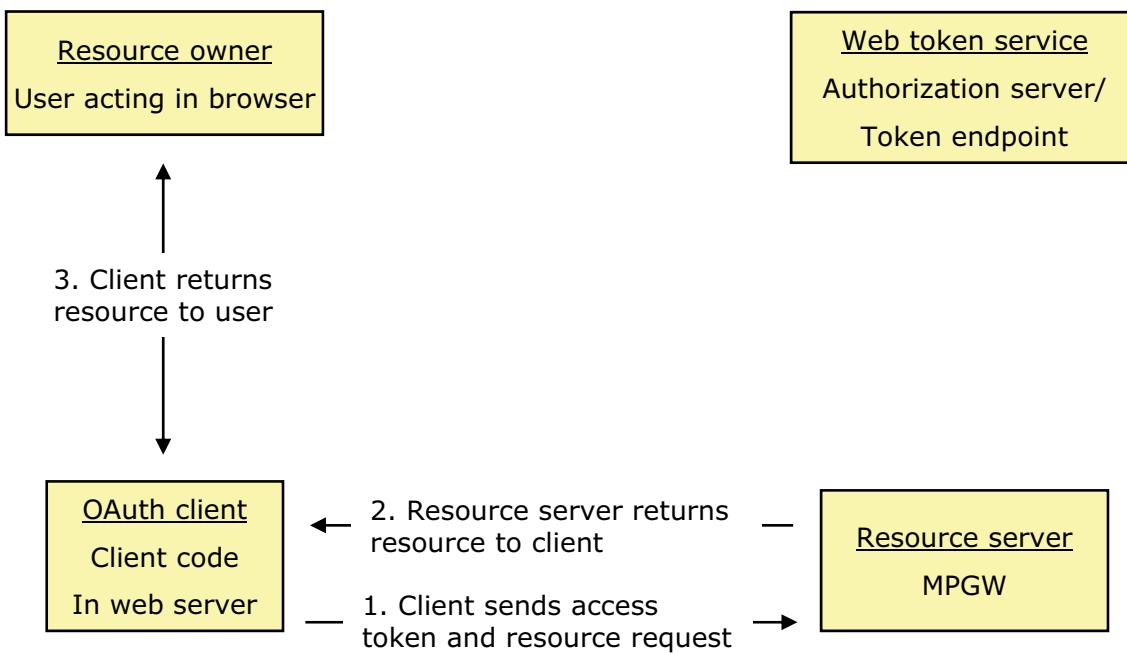
Notes:

1. The web token service redirects the user agent to the client, and passes the authorization grant code.
2. The client sends the grant code back to the web token service and asks for an access token. The client also sends its registered client ID and client secret to the web token service to verify its identity.
3. The web token service sends an access token to the client.

The user (resource owner) is not involved in the grant code and access token interactions.

Sample three-legged scenario in DataPower (4 of 4)

Access token is used to get resource



© Copyright IBM Corporation 2014

Figure 12-15. Sample three-legged scenario in DataPower (4 of 4)

WE601 / ZE6011.1

Notes:

1. The client sends the access token and the resource request to the resource server.
2. The DataPower service, acting as the resource server, validates the access token. It might perform other DataPower functions before passing the request to the actual back-end application. It takes the resource response and sends it back to the client.
3. The client usually returns the requested resource to the user.



Authorization request

- The **OAuth client** issues a request for an **authorization grant** from the **authorization server**, on behalf of the user
- The parameters:
 - **response_type**: A value of "code" identifies it as a request for an authorization grant.
 - **client_id**: The client identifier of the initiating OAuth client. This parameter is the client identifier that the authorization server knows each particular OAuth client by.
 - **redirect_uri**: A URL that refers to the OAuth client "entry point". The authorization server uses this URL for an HTTP redirection on the response.
 - **scope**: The scope of the access request.
 - **state**: A value that the OAuth client can use to maintain state between the request and the callback. The authorization server includes this value when redirecting the user-agent (browser) back to the client. The parameter should be used for preventing cross-site request forgery.
- The parameters are sent as part of the URI string

© Copyright IBM Corporation 2014

Figure 12-16. Authorization request

WE601 / ZE6011.1

Notes:

These next few slides are the definitions of the requests and responses for an authorization grant request and access token request from the client.

They show the types of information that are passed.



Authorization response

- The **authorization server** responds with an **authorization grant code** to the **OAuth client**, by using an HTTP redirection to the user's browser (user agent)
- The parameters:
 - **code**: The authorization code that the authorization server generates. The authorization code must expire shortly after it is issued to mitigate the risk of leaks, and the client must not reuse it.
 - **state**: The "state" parameter that was present in the client authorization request.
- The parameters are returned in the URI string as part of the Location header in the redirection

© Copyright IBM Corporation 2014

Figure 12-17. Authorization response

WE601 / ZE6011.1

Notes:



Access token request

- The **OAuth client** issues a request for an **access token** from the **authorization server**, on behalf of the OAuth client
 - The authorization server is acting as a token endpoint
- The parameters:
 - **grant_type**: The value must be set to "authorization_code".
 - **code**: The authorization code that was received from the authorization server.
 - **redirect_uri**: The "redirect_uri" parameter that was included in the authorization request. The authorization server requires that the values must be identical.
 - **client_id**: The client identifier of the initiating OAuth client.
 - **client_secret**: The client secret that is defined in the OAuth client.
- The parameters are sent in the HTTP request body

© Copyright IBM Corporation 2014

Figure 12-18. Access token request

WE601 / ZE6011.1

Notes:



Access token response

- The **authorization server** responds with an **access token** to the **OAuth client**
 - The authorization server is acting as a token endpoint
- The parameters:
 - access_token: The access token that the authorization server generates.
 - token_type: The type of the token.
 - expires_in: The lifetime in seconds of the access token.
 - refresh_token: (optional) The refresh token, which can be used to obtain new access tokens by using the same authorization grant code.
 - scope: The scope of the request
- The response is returned in the HTTP request body as JSON data

© Copyright IBM Corporation 2014

Figure 12-19. Access token response

WE601 / ZE6011.1

Notes:

The token type for the exercise is “bearer”, which indicates that the access token is included in the request as an HTTP Authorization header.

Resource request

- The **OAuth client** sends the request to the **resource server**, and includes the **access token**
- The specification does not explicitly indicate the parameters on the call, but it does indicate what must happen:
 - The resource server must validate the access token.
 - The resource server must ensure that the token has not expired.
 - The resource server must validate that the token scope covers the requested resource.
- The methods that the resource server uses to validate the access token are beyond the scope of the specification but generally involve an interaction or coordination between the resource server and the authorization server

© Copyright IBM Corporation 2014

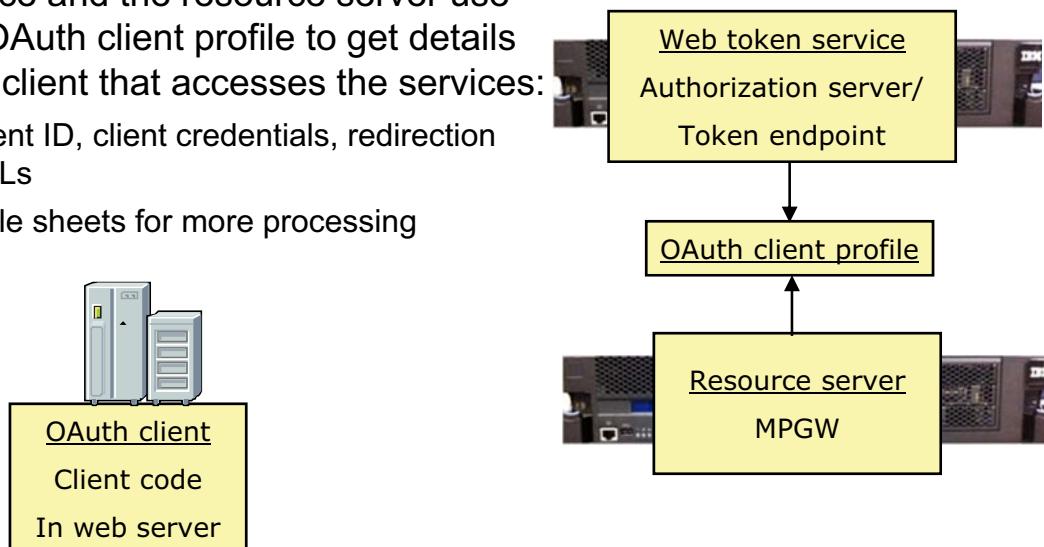
Figure 12-20. Resource request

WE601 / ZE6011.1

Notes:

OAuth client and the OAuth Client Profile object

- The AAA policies in the web token service and the resource server use the OAuth client profile to get details on a client that accesses the services:
 - Client ID, client credentials, redirection URLs
 - Style sheets for more processing



- The actual OAuth client is running on some remote platform:
 - Web server
 - Smartphone

© Copyright IBM Corporation 2014

Figure 12-21. OAuth client and the OAuth Client Profile object

WE601 / ZE6011.1

Notes:



DataPower OAuth objects: OAuth Client (1 of 4)

The object name is used as the OAuth client ID

Defines why the client is contacting the DataPower service

Specifies how this client can ask for a grant

Specifies how this client can ask for a grant

© Copyright IBM Corporation 2014

Figure 12-22. DataPower OAuth objects: OAuth Client (1 of 4)

WE601 / ZE6011.1

Notes:

If Customized OAuth is enabled, a style sheet must be specified that handles all of the implementation details of the OAuth roles that the client is playing.

The OAuth Role choices controls which of the other options on the page are visible. In a situation where the authorization server (web token service) and the resource server are on the same domain in the same appliance, both service types use the same client profile, so both roles are selected. In another situation where the authorization server is centralized on one appliance and one domain in a cluster of appliances that are running numerous resource servers, the choices differ. For the centralized authorization server, only the options that pertain to authorization and token requests are needed. For the distributed resource servers, they need to see only the specific options of the client that pertain to the actual resource request.

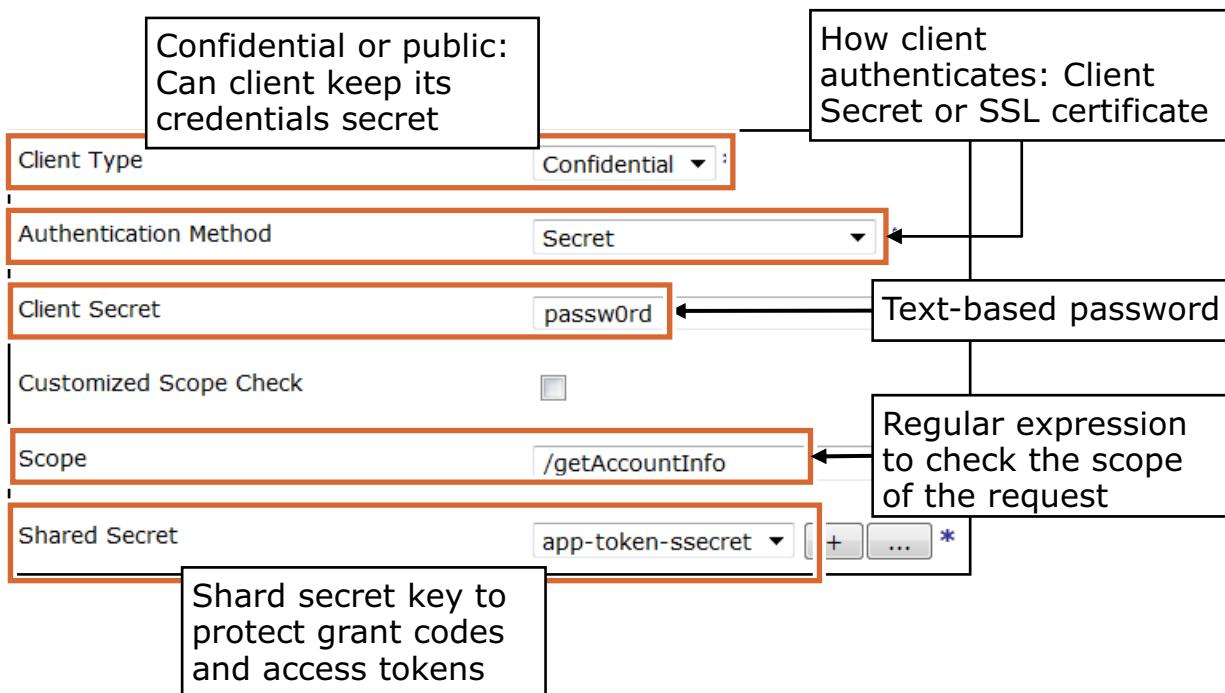
Although most of the information in this unit focuses on authorization grant code, the OAuth specification supports four types of grants:

- **Authorization Code Grant:** Optimized for confidential clients; requires a client that can send and receive redirections; uses grant codes, access tokens, and refresh tokens

- **Implicit Grant:** Optimized for public clients; requires a client that can send and receive redirections, typically implemented in the browser by using a scripting language; uses access tokens
- **Resource Owner Password Credentials Grant:** Optimized for when the client can see the resource owner credentials; the client is typically implemented as a highly privileged application or within the operating system of the platform; uses access tokens and refresh tokens
- **Client Credentials Grant:** For confidential clients only, client can request access by using only its client credentials, resource owner is not directly involved, uses access tokens

The OAuth 2.0 specification also supports the definition of custom grant types: extension grants.

DataPower OAuth objects: OAuth Client (2 of 4)



© Copyright IBM Corporation 2014

Figure 12-23. DataPower OAuth objects: OAuth Client (2 of 4)

WE601 / ZE6011.1

Notes:

If the client type is public, it means that the client does not keep its credentials a secret. In that case, the Authentication Method and Client Secret fields are not displayed.

If the Authentication Method is specified as “client certificate from SSL”, a Client SSL Credentials field replaces the Client Secret field, and a crypto validation credential is selected.

If the Customized Scope Check is enabled, a Scope Customized Process field replaces the Scope field, and a style sheet that checks the scope is specified.

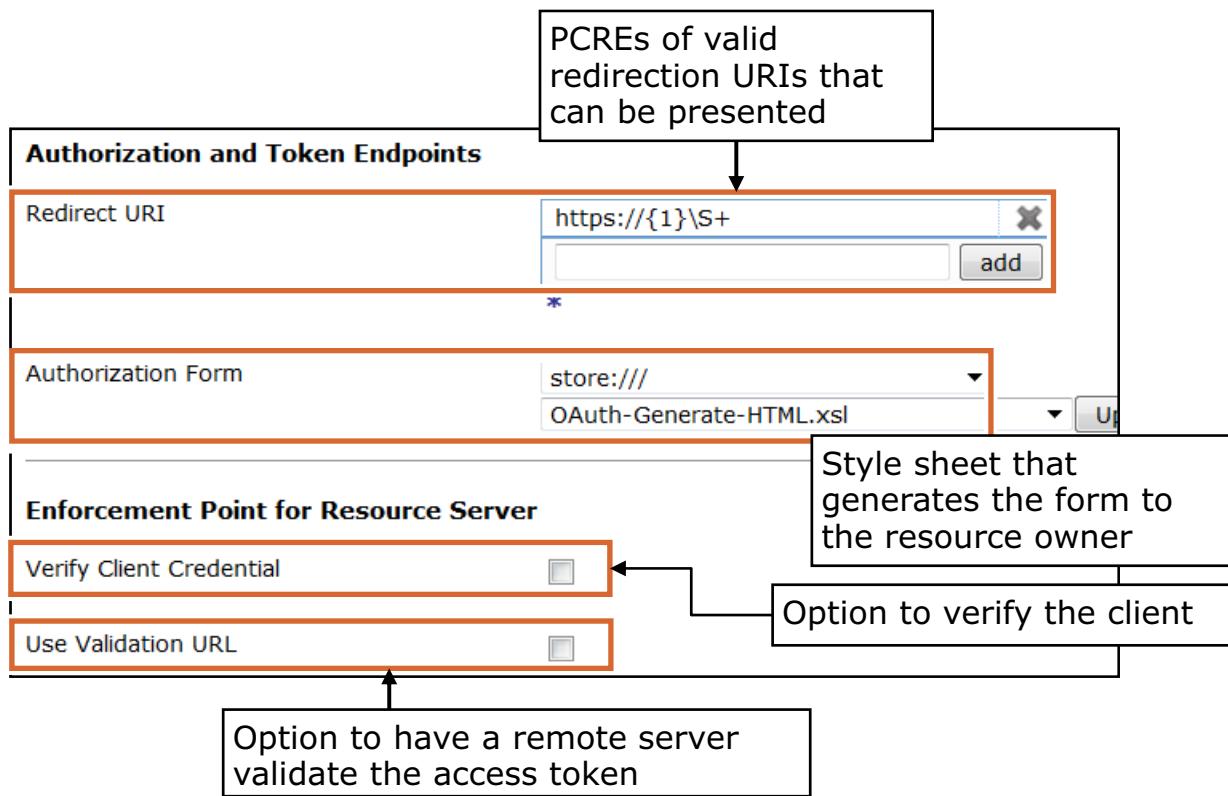
If the Customized Scope Check is disabled, the Scope field contains a Perl-compatible regular expression (PCRE) that checks the scope. For example, If the requested scope must contain the string “Pictures” or “Videos”, then a PCRE is “`^Pictures$ | ^Videos$`”.

Scope checking occurs during authorization, token, and resource requests.

The shared secret (symmetric) key is used to sign and encrypt the grant codes, access tokens, and refresh tokens that are passed during the OAuth protocol. The symmetric key must be at least 32 bytes long.



DataPower OAuth objects: OAuth Client (3 of 4)



© Copyright IBM Corporation 2014

Figure 12-24. DataPower OAuth objects: OAuth Client (3 of 4)

WE601 / ZE6011.1

Notes:

Valid redirection URIs are listed to detect malicious clients.

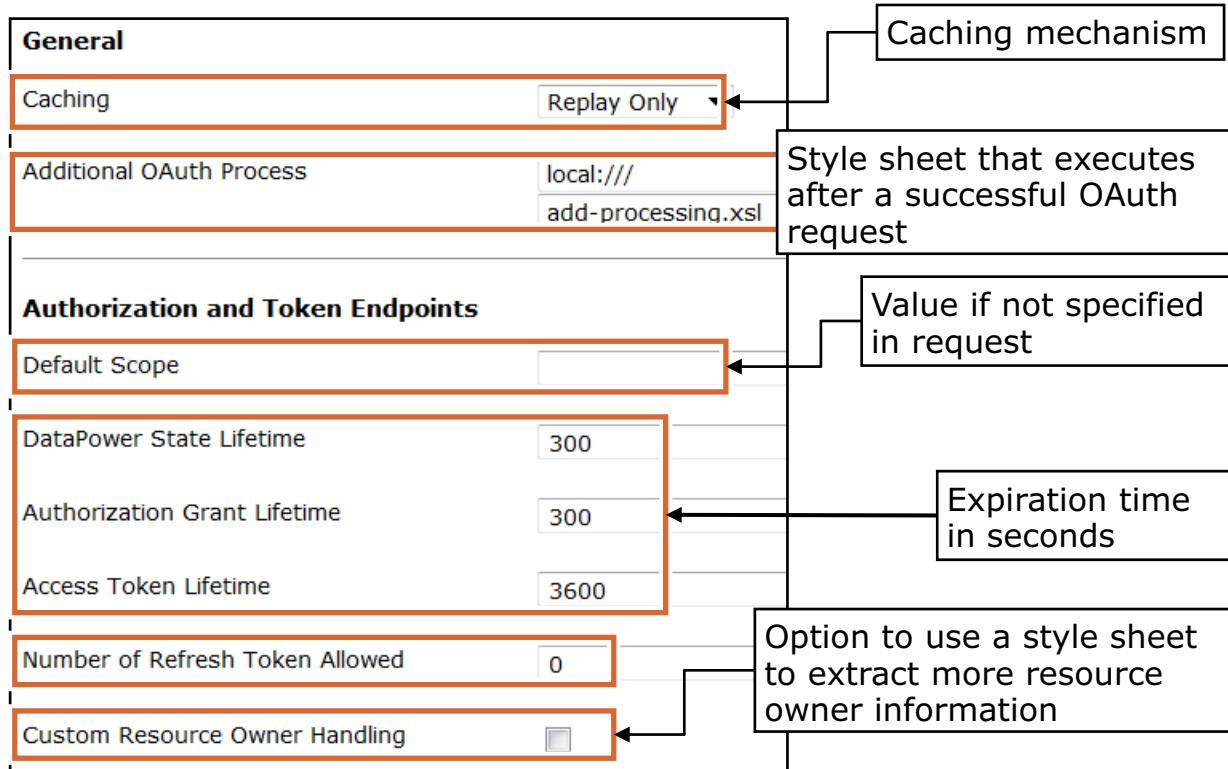
The Authorization Form specifies the style sheet that generates the authorization form that is sent to the resource owner, and handles any related errors. The listed sample style sheet can be copied to the local: directory and customized.

The Verify Client Credential option indicates whether to verify the client credentials (client secret or client credential).

If the Use Validation URL is enabled, a Validation URL field is displayed to allow entry of the location of some remote server that validates access tokens. This field can be used when the authorization server is not a DataPower web token service. A Validation SSL Proxy Profile choice is also presented, where you can specify the SSL proxy profile that manages the SSL interaction.



DataPower OAuth objects: OAuth Client (4 of 4)



© Copyright IBM Corporation 2014

Figure 12-25. DataPower OAuth objects: OAuth Client (4 of 4)

WE601 / ZE6011.1

Notes:

This screen capture is from the **Advanced** tab.

The **Caching** option indicates the mechanism in effect. "Replay Only" uses the replay cache to prevent replay attacks. "Token Cache" uses system memory to support token revocation. "Custom" uses a style sheet to handle revocation.

The Additional OAuth Process style sheet allows extra processing after generating a grant code, after generating an access token, and after verifying an access token.

The Default Scope is the default scope that is used for any request that does not contain a scope.

The DataPower State Lifetime is the expiration time for the authorization form that is presented to the resource owner. The Authorization Grant Lifetime is time limit for the client to use the grant code. The Access Token Lifetime is the lifetime of the access token.

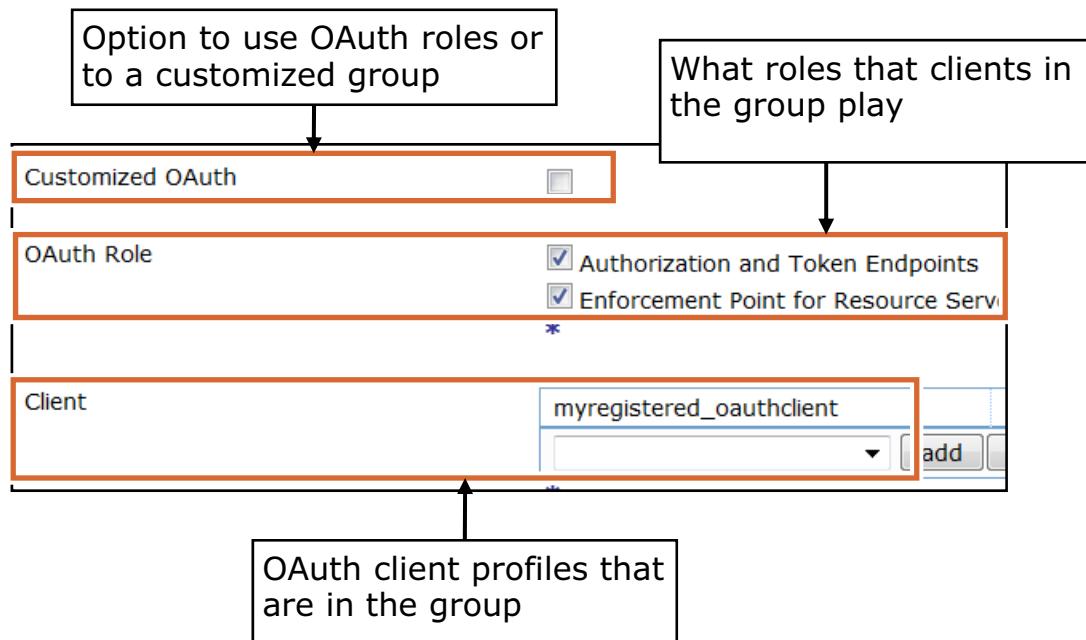
The OAuth client uses refresh tokens to request more access tokens when the access token expires. The authorization server can send a refresh token to the client along with the requested access token. When the client detects that the access token is rejected because of an expiration, it can use the refresh token to ask the authorization server for a new access token. The new token

can be of the same or of a narrower scope. The Number of Refresh Token Allowed field indicates how many refresh tokens the authorization server can generate for a specific OAuth client.

If Custom Resource Owner Handling is enabled, you can specify a style sheet that can extract extra resource owner information. By default, the resource owner is the user name from the extracted identity. For custom handling, the style sheet can provider a different resource owner name. This style sheet applies only when it is run from the authorization server.

DataPower OAuth objects: OAuth Client Group

- The AAA policies in the web token service and the resource server refer to a **group** of OAuth client profiles, rather than individual ones



© Copyright IBM Corporation 2014

Figure 12-26. DataPower OAuth objects: OAuth Client Group

WE601 / ZE6011.1

Notes:

If the Customized OAuth is enabled, the OAuth Role check boxes are not displayed.



DataPower OAuth objects: Web Token Service

- The web token service is a specialized loopback service that acts as an authorization and token endpoint for OAuth
- Other tabs and fields allow specification of other service parameters, such as timeouts, request type, HTTP version, and more

The screenshot shows the XML manager interface for a DataPower Web Token Service. The configuration includes:

- Endpoints**: A table titled "Source addresses" with one row:

Local address	Port	SSL	Assign the SSL proxy profile	Credential Class Set
0.0.0.0	7070	on	oauth-server	Protocol
- Processing Policy**: Set to "oauth-wts".
- Ports and addresses that this web token service listens on**: A callout box pointing to the "Port" column in the table.
- Processing policy that is implemented in the service**: A callout box pointing to the "Processing Policy" dropdown.

© Copyright IBM Corporation 2014

Figure 12-27. DataPower OAuth objects: Web Token Service

WE601 / ZE6011.1

Notes:

It is easiest to create a web token service by using the **add wizard**. In the wizard, you specify a source address, and the AAA policy that is used in a generated processing policy. The AAA policy must specify OAuth protocol. If you do not use the wizard, you must code your own processing policy.

You can code your own authorization and token endpoint service, if required.



AAA policy: Key to OAuth behavior

- The AAA policy within the processing policy of the web token service or the resource server controls the OAuth support

© Copyright IBM Corporation 2014

Figure 12-28. AAA policy: Key to OAuth behavior

WE601 / ZE6011.1

Notes:



AAA policy for the web token service

- Along with the regular Identity extraction selection, you must also select **OAuth** and identify an **OAuth client group**

The screenshot shows a configuration panel for an AAA policy. At the top right, there is a checked checkbox labeled "OAuth" with an asterisk (*) indicating it is required. Below this, there is a field labeled "HTTP Basic Authentication Realm" containing the value "login". Further down, there is a section labeled "Registered OAuth clients" with a dropdown menu set to "OAuth-Code-Client".

- In the Resource extraction phase, you select **Processing metadata**, and specify the **oauth-scope-metadata**

The screenshot shows a configuration panel for resource extraction. At the top right, there is a checked checkbox labeled "Processing metadata" with an asterisk (*) indicating it is required. Below this, there is a section labeled "Processing metadata items" with a dropdown menu set to "oauth-scope-metadata".

© Copyright IBM Corporation 2014

Figure 12-29. AAA policy for the web token service

WE601 / ZE6011.1

Notes:

By specifying the OAuth client group in Identity extraction, the client information is available to the web token service. In the screen capture, the HTTP Basic Authentication Realm field is displayed because the HTTP Authentication Header was also selected for an identity extraction method.

The oauth-scope-metadata contains the scope that is specified in the OAuth requests.



AAA policy for the resource server (1 of 2)

- In Identity extraction, you must select **OAuth** and identify an **OAuth client group**

OAuth
*

Registered OAuth clients OAuth-Code-Client ▾

- In the Resource extraction phase, you select **URL sent by client** and **Processing metadata**, and specify the **oauth-scope-metadata**

Resource information	<input type="checkbox"/> URL sent to back end <input checked="" type="checkbox"/> URL sent by client <input type="checkbox"/> URI of top level element in <input type="checkbox"/> Local name of request elem <input type="checkbox"/> HTTP operation (GET or PO <input type="checkbox"/> XPath expression <input checked="" type="checkbox"/> Processing metadata *
Processing metadata items	oauth-scope-metadata ▾

© Copyright IBM Corporation 2014

Figure 12-30. AAA policy for the resource server (1 of 2)

WE601 / ZE6011.1

Notes:

By specifying the OAuth client group in Identity extraction, the client information is available to the resource server.

The oauth-scope-metadata contains the scope that is specified in the OAuth requests. The URL sent by the client is also retrieved.

For v6.0.0, the firmware does a string compare of the scope value from the oauth-scope-metadata to the value from the other selected extracted resource. A successful compare results in a validate request. For v5.0.0, further processing is required (next slide).



AAA policy for the resource server (2 of 2)

- In the Map resources phase, select a Method of **Custom**, and specify the **style sheet** to perform the custom mapping

Method	Custom
Custom URL	local:///accesstoken-check-resource

© Copyright IBM Corporation 2014

Figure 12-31. AAA policy for the resource server (2 of 2)

WE601 / ZE6011.1

Notes:

For v5.0.0, a map resources style sheet must be coded to determine a successful match.

For later versions of the firmware, this style sheet can be used to do a custom compare.

Unit summary

Having completed this unit, you should be able to:

- Describe the OAuth framework
- Describe why OAuth is useful in security scenarios
- Describe the OAuth three-legged scenario
- Explain the role that a DataPower appliance performs in an OAuth framework
- Describe the OAuth configuration options on DataPower: the web token service, the AAA action, the OAuth client profile, and the OAuth client group

© Copyright IBM Corporation 2014

Figure 12-32. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: With OAuth, resource owners allow third-party access to the resource **without** sharing their credentials.
2. True or False: Three-legged OAuth is the traffic and data pattern that OAuth is designed to solve.
3. DataPower implementation of OAuth includes (select 3):
 - a. Web Token Service
 - b. One-legged authentication
 - c. AAA action
 - d. SSL
 - e. OAuth Client Profile and OAuth Client Group
4. True or False: OAuth configuration on DataPower does not allow for the use of custom style sheets.

© Copyright IBM Corporation 2014

Figure 12-33. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.



Checkpoint answers

- 1. True.**
- 2. True.**
- 3. A, C, and E.**
- 4. False.** OAuth configuration on DataPower **does** allow for the use of custom style sheets.

© Copyright IBM Corporation 2014

Figure 12-34. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 10



Define a three-legged OAuth scenario that uses DataPower services

© Copyright IBM Corporation 2014

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

9.0

Figure 12-35. Exercise 10

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

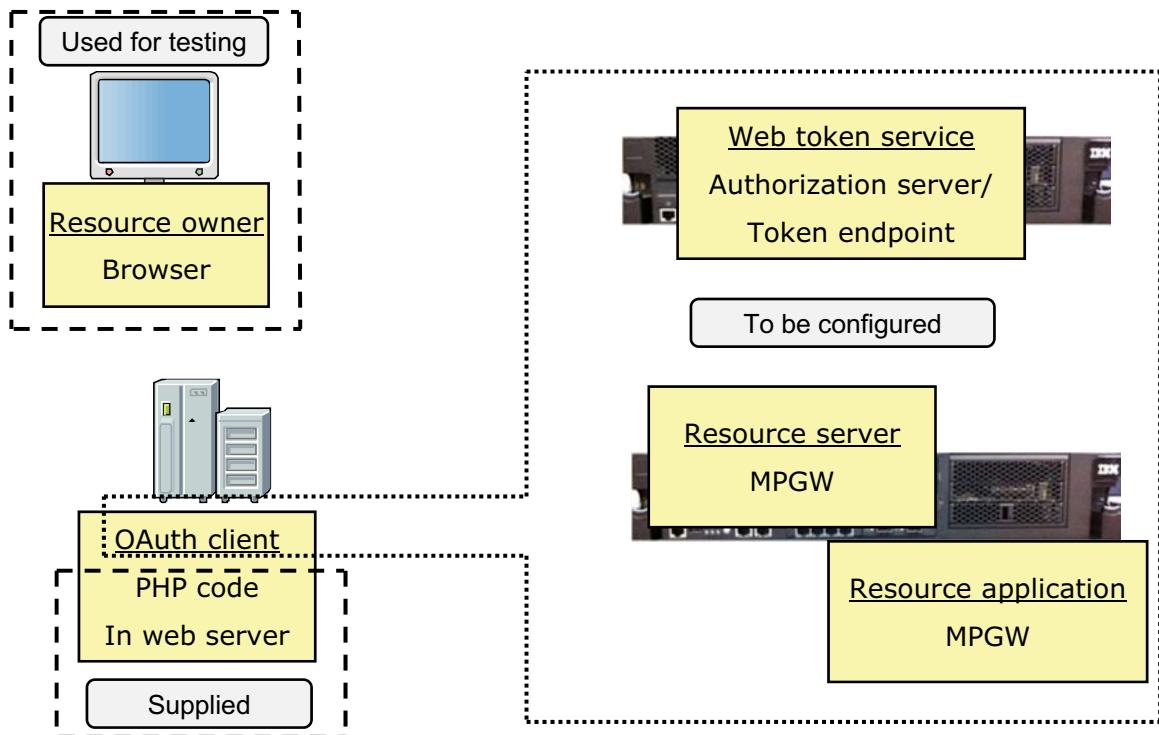
© Copyright IBM Corporation 2014

Figure 12-36. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview (1 of 3)



© Copyright IBM Corporation 2014

Figure 12-37. Exercise overview (1 of 3)

WE601 / ZE6011.1

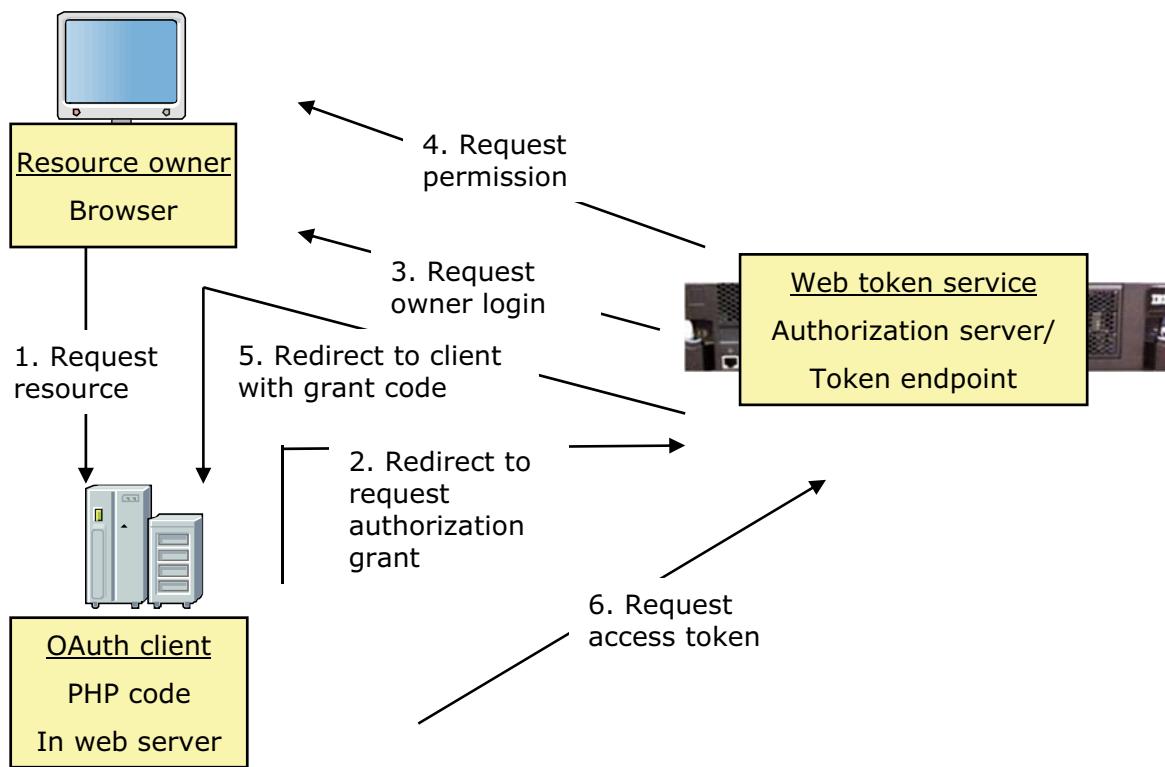
Notes:

The student configures an OAuth client profile, an OAuth client group, a web token service, a resource server that is implemented as a multi-protocol gateway, and another multi-protocol gateway that is the back-end resource application.

The OAuth client code is supplied as a PHP module that is running under a web server.

The resource owner operates through the browser as its user agent during the testing section.

Exercise overview (2 of 3)



© Copyright IBM Corporation 2014

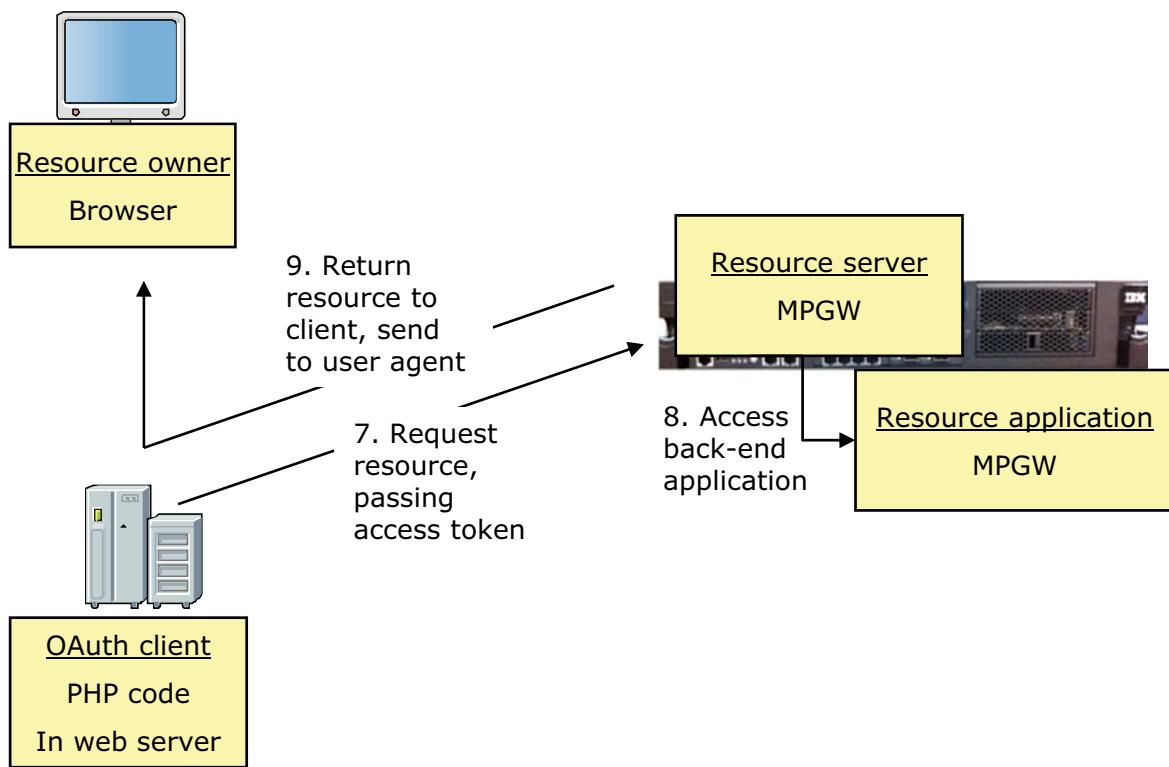
Figure 12-38. Exercise overview (2 of 3)

WE601 / ZE6011.1

Notes:

The OAuth client responsibilities are split between the PHP client code and the browser. This setup is not accurate OAuth interaction, but it does allow the student to see the OAuth flows that normally are encapsulated. The flows in themselves are accurate for the specification and the DataPower implementation. The client itself usually performs Steps 2 and 6, but for education purposes, the PHP code sends a page to the browser to show some of the internal details. For those pages, the student is acting as the OAuth client.

Exercise overview (3 of 3)



© Copyright IBM Corporation 2014

Figure 12-39. Exercise overview (3 of 3)

WE601 / ZE6011.1

Notes:

Step 7 is exposed to the student for educational purposes, but the client normally performs it.

Unit 13. Patterns for service configuration

What this unit is about

This unit describes patterns as used by DataPower. It explains how a pattern is initially created and made available for use. The focus of the presentation is on how to create a new service from an existing pattern.

What you should be able to do

After completing this unit, you should be able to:

- Explain what a DataPower pattern is, and its purpose
- Describe how a pattern is created
- Generate a new service from a pattern

How you will check your progress

- Checkpoint
- Hands-on exercise

References

IBM WebSphere DataPower Version 6.0 Information Center:

<http://pic.dhe.ibm.com/infocenter/wsdatap/v6r0m0/index.jsp>



Unit objectives

After completing this unit, you should be able to:

- Explain what a DataPower pattern is, and its purpose
- Describe how a pattern is created
- Generate a new service from a pattern

© Copyright IBM Corporation 2014

Figure 13-1. Unit objectives

WE601 / ZE6011.1

Notes:

What is a pattern?

- A pattern is an importable/exportable DataPower object that is a template of an existing service configuration (the “source service”)
- It is used to generate new services that are based on the source service, but differ by a limited set of variables/specifications
- The pattern presents the limited set of variables/specifications only
- The new service is generated as a set of new DataPower objects
- The generated service can be further modified as needed
- There is no backward or forward connection between the generated service and the generating pattern
- A **pattern creator** creates a pattern, a **pattern deployer** generates a new service from a pattern
- Patterns can be further edited, cloned, and deleted
- A separate “pattern console” GUI is used to work with patterns
- Several sample patterns are supplied with the firmware

© Copyright IBM Corporation 2014

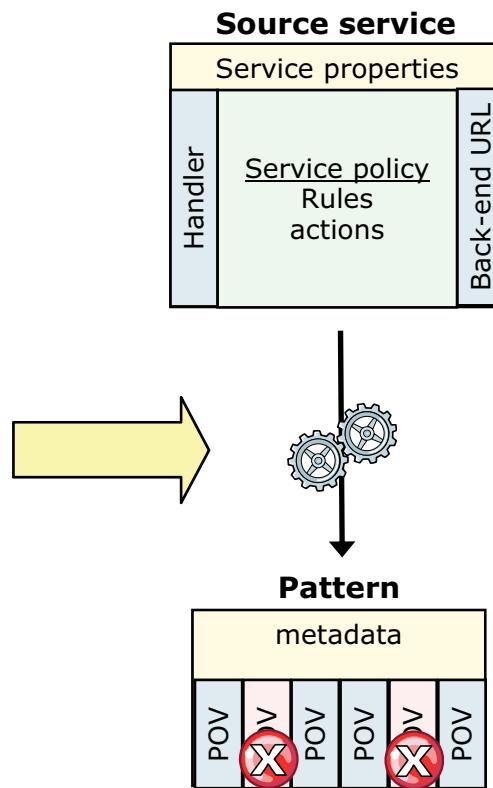
Figure 13-2. What is a pattern?

WE601 / ZE6011.1

Notes:

Creating a pattern

- In the pattern creation dialog box, the pattern creator identifies the “source service”
 - An existing service that is the model for modified versions
- As the source service is scanned, the “points of variability” (POVs) are listed, and the creator selects which ones must be visible to the deployer
 - A POV is some configuration variable that the pattern framework allows to be exposed



© Copyright IBM Corporation 2014

Figure 13-3. Creating a pattern

WE601 / ZE6011.1

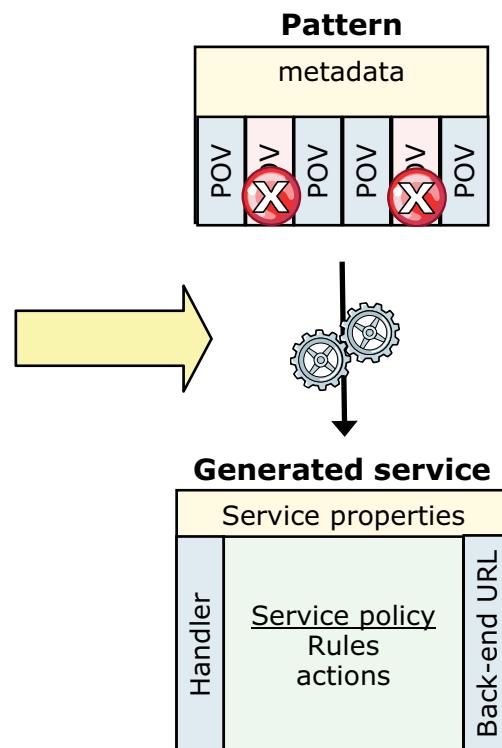
Notes:

There is a list of what configuration variables in a service and its related objects that can be exposed by a pattern. The pattern creation dialog box displays them.

The pattern creator decides to expose the variable in the pattern, or locks the value so the deployer cannot change it.

Deploying a pattern

- The pattern deployer chooses the appropriate pattern, and selects “Deploy”
- The wizard displays each of the exposed POVs, and the deployer enters the appropriate value for the new service
- The service is generated as a set of new DataPower objects



© Copyright IBM Corporation 2014

Figure 13-4. Deploying a pattern

WE601 / ZE6011.1

Notes:

Only the POVs that the pattern creator exposed in the pattern can be used by the deployer. The non-exposed POVs are hidden from the deployer.



The pattern console

- Any work that involves patterns must be done in the **pattern console**
- The pattern console UI is launched into a single, separate browser tab or browser window
 - Existing WebGUI remains intact
- Pattern console includes the following capabilities
 - Browse and deploy patterns
 - Customize and create patterns
 - Clone patterns
 - View endpoint or service status
- Several predefined “best practice” patterns are included in the firmware
 - Read-only, but they can be cloned

© Copyright IBM Corporation 2014

Figure 13-5. The pattern console

WE601 / ZE6011.1

Notes:



Getting to the pattern console

- Two ways to get to the pattern console:
 - Its own URL
`https://<appliance_address>:<WebGUI_port>/dp`
 - Click **Pattern Console** from the WebGUI:



- It opens a second browser tab or browser window
 - Shared session between pattern console and WebGUI, so WebGUI login and timeout applies to both
 - Pattern console has its own Login page that you can log in from if a WebGUI timeout occurs

© Copyright IBM Corporation 2014

Figure 13-6. Getting to the pattern console

WE601 / ZE6011.1

Notes:

Get Started **Endpoints** **Patterns**

Menu bar

Shortcuts

Create a Pattern 	Deploy a Pattern 	View Endpoints
Create a pattern from a service in the domain. Learn more	Create a service from a pattern. Learn more	View endpoints of all services, configurations in the domain Learn more

© Copyright IBM Corporation 2014

Figure 13-7. The pattern console

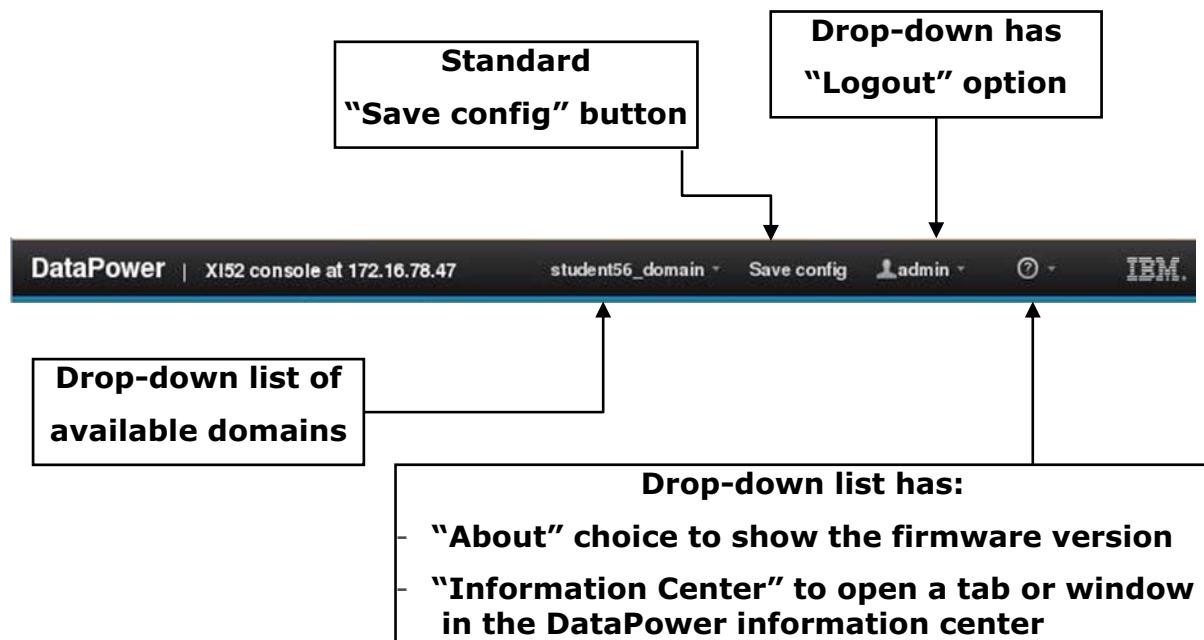
WE601 / ZE6011.1

Notes:



The pattern console menu bar

- Slight change from WebGUI



© Copyright IBM Corporation 2014

Figure 13-8. The pattern console menu bar

WE601 / ZE6011.1

Notes:

The screenshot shows the 'Getting Started' tab of the WebSphere Education pattern console. At the top, there's a blue header bar with the 'WebSphere Education' logo on the left and the 'IBM' logo on the right. Below the header, the title 'The pattern console “Getting Started” tab' is displayed in blue. A bulleted list follows, detailing the features of this tab:

- Shortcuts to tasks available under the other tabs
- Clicking **Learn more** opens a text pane that provides some help on the task

Below the list, there's a navigation bar with three tabs: 'Get Started' (which is selected), 'Endpoints', and 'Patterns'. The main content area is titled 'Get started' and contains a welcome message: 'Welcome to DataPower. The configuration pattern console provides the service pattern functionality to help you create services. The tasks described in this page get you started with the console. More information is available in the IBM WebSphere DataPower SOA Appliances Information Center that is accessed through the help icon in the banner.'

The page is divided into three sections:

- Create a Pattern**: Features a pencil icon and the text 'Create a pattern from a service in the domain.' with a 'Learn more' link.
- Deploy a Pattern**: Features a play button icon and the text 'Create a service from a pattern.' with a 'Learn more' link.
- View Endpoints**: Features two interlocking gears icon and the text 'View endpoints of all service configurations in the domain' with a 'Learn more' link.

At the bottom right of the page, there's a copyright notice: '© Copyright IBM Corporation 2014'.

Figure 13-9. The pattern console "Getting Started" tab

WE601 / ZE6011.1

Notes:



The pattern console “Endpoints” tab

- Read-only list of **all** the endpoints defined in a specific domain
- List can be sorted by clicking each column header

The screenshot shows the WebSphere Pattern Console interface. At the top, there are three tabs: "Get Started", "Endpoints" (which is selected and highlighted in blue), and "Patterns". Below the tabs, the title "Services on student54_domain domain" is displayed. To the right of the title is a search bar with a magnifying glass icon and an input field containing a placeholder character. The main area is a table titled "Endpoints" with the following columns: Endpoint Status, Protocol, Endpoint Details, Handler, Service Class, and Service. The table contains four rows of data:

Endpoint Status	Protocol	Endpoint Details	Handler	Service Class	Service
Error	HTTP	:6959	AddressSearch		AddressRouter
Up	HTTP	172.16.78.48:6547	EastAddressSearch		EastAddressSearch
Error	HTTP	:6958	WestAddressSearch		WestAddressSearch
Up	HTTP	172.16.80.79:6535	JimTestWSPLDAP_AddressS		JimTestWSPLDAP

© Copyright IBM Corporation 2014

Figure 13-10. The pattern console "Endpoints" tab

WE601 / ZE6011.1

Notes:

The list displays all the endpoints in the domain, regardless of whether they were created from a pattern or not.

The screenshot shows the 'Patterns' tab of the WebSphere Education pattern console. At the top, there's a navigation bar with 'Get Started', 'Endpoints', and 'Patterns'. Below it is a toolbar with icons for 'New Pattern...', 'Deploy...', and others. The main area has a sidebar on the left with categories like 'All Categories', 'Search for patterns', and a list of patterns including 'Web application', 'Web application with Form-based authentication and LTPA SSO', 'Web application with ISAM authentication and authorization', 'Web application with OAuth 2.0 authentication and authorization', 'Web application with OpenID Connect authentication and authorization', and 'Web application with SAML 2.0 authentication and authorization'. A red box highlights the 'List of patterns in this domain' section. The main content area has a 'Documentation' section with text about a Multi-Protocol Gateway (MPGW) service, a 'Documentation on selected pattern as written by the pattern creator' box (also highlighted with a red box), and a 'Next Steps' section with two items.

Documentation:

This pattern, when deployed, creates a Multi-Protocol Gateway (MPGW) service that is intended to proxy simple web application, mobile web, and web API traffic. The pattern reduces the time and risk of developing a web application proxy by implementing proven techniques for optimizing and securing web traffic and pre-establishing many of the required configuration settings. The resulting MPGW service has the following behavioral characteristics:

- Receives messages from the client by using the HTTPS protocol
- Caches, on the appliance, response headers that are set by the backend servers based on the HTTP cache control
- Provides default protection for HTTP header value, and total header URL string, HTTP header name, number of headers allowed
- Provides a pre-configured error handling mechanism to the client when an error occurs
- Does not perform user authentication

Documentation on selected pattern as written by the pattern creator

Next Steps

After you deploy this pattern, consider the following next steps:

1. If you want the service to receive messages by using the HTTP protocol instead of HTTPS, update the corresponding FSH object.
2. If you want to establish a different response caching policy (for example, use an IBM DataPower XC10 caching appliance for distributed caching), update the corresponding XML manager object.

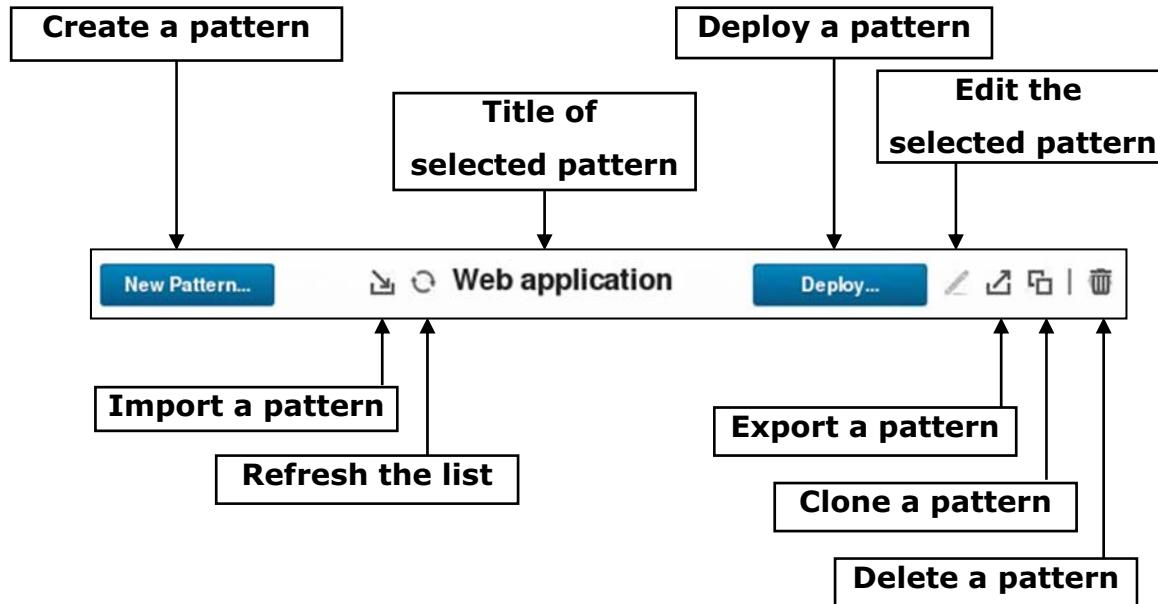
© Copyright IBM Corporation 2014

Figure 13-11. The pattern console "Patterns" tab

WE601 / ZE6011.1

Notes:

The pattern toolbar



© Copyright IBM Corporation 2014

Figure 13-12. The pattern toolbar

WE601 / ZE6011.1

Notes:

The supplied sample patterns in the default domain cannot be edited.



Steps to generate a service from a pattern

1. Select a pattern
2. Click **Deploy...**
3. Enter name of new service that is generated from the pattern
 - Used as a prefix for all objects generated from the pattern for this service
4. Supply properties or variables that the pattern requests
 - As specified by the pattern creator
5. Click **Deploy Pattern**
 - The new service and related objects are generated into the domain

© Copyright IBM Corporation 2014

Figure 13-13. Steps to generate a service from a pattern

WE601 / ZE6011.1

Notes:



Deployment: selecting a pattern

- Patterns must exist in the domain to be listed
 - Patterns are DataPower objects themselves, and can be imported or exported
- The list can be “filtered” by category and by keywords
- Pattern creator designates a category and any keywords during pattern creation

The screenshot shows two panels of a web application. The left panel is a search interface with a dropdown menu labeled 'All Categories' which is currently selected. Other options include 'Web Application' and 'Web Service'. Below the dropdown are two search results: 'Web application with Form-based authentication and LTPA SSO' and 'Web application with ISAM'. The right panel shows a search results list with three items: 'Web application with WSRR Saved Search subscription', 'Web application with WSRR Service Version subscription', and 'Web service with WSRR Saved Search subscription'. The first item in the list is highlighted with a red box.

- Select the pattern in the list
- Click **Deploy...** on the page to start the deployment wizard

© Copyright IBM Corporation 2014

Figure 13-14. Deployment: selecting a pattern

WE601 / ZE6011.1

Notes:



Deployment: filling out the wizard

- Specify the name of the service to be generated
- Fill in any remaining POVs that the pattern creator exposed
- Click **Deploy Pattern**
- The new service is generated into the domain
- After generation, the new service can be modified similar to any other service
 - The service name is used as the prefix for the name of the generated dependent objects

Deploy Pattern
Web application with OAuth authorization enforcement

* Service name:

Description:

Step 2: Back-end endpoint details
Specify the URL of the back-end server for which DataPower acts as a web proxy. Also, optionally, provide an existing SSL proxy profile to use when communicating to the back-end using SSL.

* URL:

SSL proxy profile:

Step 3: Front-end endpoint details
Specify the ID number or port number of the UTTOC front-end binder/CCU. Select an existing CCU.

© Copyright IBM Corporation 2014

Figure 13-15. Deployment: filling out the wizard

WE601 / ZE6011.1

Notes:



Points of variability

- The pattern creation wizard exposes only a limited subset of the configuration options for a multi-protocol gateway or a web service proxy:
 - Front side handler specifics
 - WSRR subscription, WSRR saved search subscription
 - Multi-protocol gateway back-end URL
 - Authentication with LTPA token, SSL certificate
 - Authorization and authentication with LDAP
 - Authorization and authentication with ISAM
 - Identity extraction from OAuth
- The pattern creator decides which POVs in the source service are exposed to a pattern deployer

© Copyright IBM Corporation 2014

Figure 13-16. Points of variability

WE601 / ZE6011.1

Notes:

ISAM is IBM Security Asset Manager, which was called TAM (Tivoli Access Manager).



Unit summary

Having completed this unit, you should be able to:

- Explain what a DataPower pattern is, and its purpose
- Describe how a pattern is created
- Generate a new service from a pattern

© Copyright IBM Corporation 2014

Figure 13-17. Unit summary

WE601 / ZE6011.1

Notes:



Checkpoint questions

1. True or False: If a pattern is updated, all services generated from that pattern are also updated.
2. True or False: Patterns are available in the default domain only.

© Copyright IBM Corporation 2014

Figure 13-18. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

1.

2.



Checkpoint answers

1. **False.** Once a service is generated from a pattern, there is no further connection between the pattern and the service.
2. **False.** The sample patterns are in the default domain, but patterns can be created and deployed in any domain.

© Copyright IBM Corporation 2014

Figure 13-19. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 11



© Copyright IBM Corporation 2014

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

9.0

Figure 13-20. Exercise 11

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2014

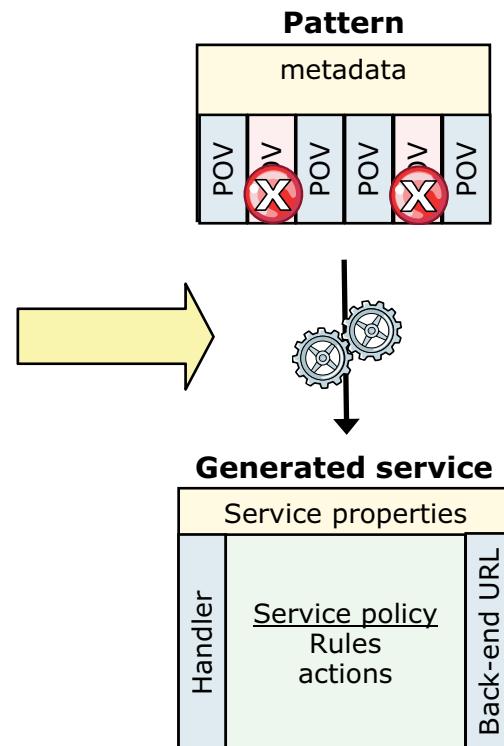
Figure 13-21. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview

- Import a pattern into the application domain
- Deploy the pattern by supplying the needed values in the wizard
- Test the generated service



© Copyright IBM Corporation 2014

Figure 13-22. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 14. Integration with WebSphere MQ

What this unit is about

This unit describes how to configure the DataPower appliance to communicate with WebSphere MQ. You learn how to receive and put messages on WebSphere MQ queues. You also learn how DataPower manages transactions between WebSphere MQ queue managers.

What you should be able to do

After completing this unit, you should be able to:

- Create a multi-protocol gateway with a WebSphere MQ front-side handler
- Configure a WebSphere MQ back-end Uniform Resource Locator (URL)
- Manage transactionality between WebSphere MQ queue managers

How you will check your progress

- Checkpoint
- Exercise 12: Configuring a multi-protocol gateway service with WebSphere MQ



Unit objectives

After completing this unit, you should be able to:

- Create a multi-protocol gateway with a WebSphere MQ front-side handler
- Configure a WebSphere MQ back-end Uniform Resource Locator (URL)
- Manage transactions between WebSphere MQ queue managers

© Copyright IBM Corporation 2014

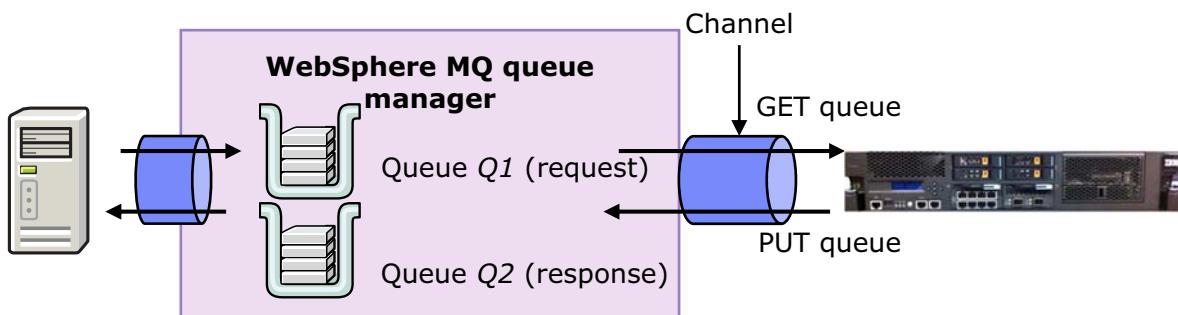
Figure 14-1. Unit objectives

WE601 / ZE6011.1

Notes:

WebSphere MQ fundamentals

- A **queue manager** manages a container for messages that are sent over a WebSphere MQ network
 - In a publish/subscribe model, **queues** represent a message destination for messages that are organized in FIFO order
 - Queue managers send messages over a communications link known as a **channel**
 - A WebSphere MQ client (such as the WebSphere MQ front side handler) must poll the queue manager for new messages
 - The queue manager itself does not initiate connections to the clients



© Copyright IBM Corporation 2014

Figure 14-2. WebSphere MQ fundamentals

WE601 / ZE6011.1

Notes:

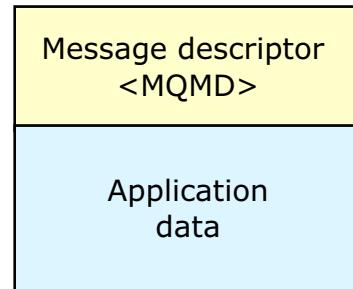
IBM WebSphere MQ allows asynchronous message communication across a network. If HTTP communication is analogous to telephone calls, then message delivery over WebSphere MQ is analogous to a courier service. For point-to-point communications, messages are deposited in a queue and used by a service later. The queue manager maintains a set of queues in one node on the network. Separate queues store and forward request and response messages.

FIFO stands for first-in first-out. Queues mainly work in a FIFO fashion unless a special weighting for messages is implemented.

IBM WebSphere MQ does all network communications over a **channel**. More specifically, the software program that allows network communication between a WebSphere MQ client and the WebSphere MQ queue manager is known as a **client channel**. The channel is a program that runs on the same host as the WebSphere MQ queue manager that provides network connectivity, rather than the connection itself. If a client application is local to the queue manager, then a channel is not necessary, but it is allowed. The DataPower appliance is always remote to the queue manager. Hence, communication is always over a channel.

WebSphere MQ message

- WebSphere MQ messages are divided into two parts:
 - Message descriptor: contains message ID and control information
 - Application data: message payload
- Data that is contained within the message descriptor is encapsulated within an <mqmd> header
 - Message metadata: contains information about the message
- Application data
 - Contains application-specific data, such as an XML message
- Example: create a reply message by using the message ID of the sender and copy it into the correlation ID field



© Copyright IBM Corporation 2014

Figure 14-3. WebSphere MQ message

WE601 / ZE6011.1

Notes:

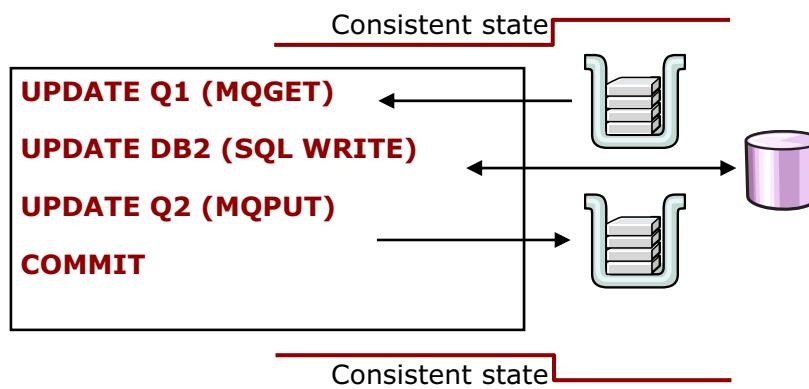
Controlling information within a WebSphere MQ message can include the message priority, reply queue name, correlation ID, and more.

Every message has a message identifier, which is determined by the value of the field `MsgId` in its message descriptor. When an application puts a message on a queue, either the application can supply a message identifier, or it can ask the queue manager to generate a unique one.

The correlation identifier is normally used to provide an application with some means of matching a reply with the original message. In a reply message, therefore, the value of the `CorrelId` field is normally copied from the `MsgId` field of the original message.

Transactions

- A transaction is a sequence of operations that either commit or roll back their work
 - A transaction *rolls back* if any one of the operations in the transaction fails
 - A transaction *commits* if all the operations in the transaction succeed
- A **local unit of work** is defined as when only the queue manager resources are being updated
- A **global unit of work** is defined as when resources of other resource managers are also being updated



© Copyright IBM Corporation 2014

Figure 14-4. Transactions

WE601 / ZE6011.1

Notes:

The terms **transaction** and **unit of work** are interchangeable.

It can happen that failure occurs during a unit of work, or the application might determine that it cannot complete the unit of work for any reason. In such cases, the changes to resources that are already made are **backed out**, or **rolled back**.

The point at which changes to the resources within a unit of work are committed or backed out is known as a **point of synchronization**, or a **sync point**. At a sync point, the data within the resources is in a consistent state from the point of view of the business and its applications.

Resource managers such as WebSphere MQ queue manager can participate in a global unit of work, which involves the processing of resources from multiple resource managers. A transaction manager is required to coordinate such a transaction. It uses the two-phase commit protocol, with a prepare and commit phase. The prepare phase ensures that all resources marked for commitment can be redistributed. The commit phase sends a request to all resource managers to commit their work. The standard interface that is used between the transaction and resource manager is the X/Open XA interface. Global units of work are sometimes referred to as XA transactions.

DataPower support for WebSphere MQ

- The DataPower XI52 device can exchange messages with WebSphere MQ systems
 - DataPower XI52 provides an enhanced implementation of the IBM WebSphere MQ client
 - DataPower WebSphere MQ client configuration is performed by using the DataPower management interfaces
 - Supports both point-to-point and publish/subscribe modes
- Bridges disparate transport protocols, such as HTTP to WebSphere MQ
 - Messages originating within or outside of WebSphere MQ can flow easily to and from another WebSphere MQ messaging bus or other messaging system, such as HTTP or TIBCO EMS
- The multi-protocol gateway service allows for the implementation of multiple transport protocols by using:
 - Front side handlers
 - Back-end URL

© Copyright IBM Corporation 2014

Figure 14-5. DataPower support for WebSphere MQ

WE601 / ZE6011.1

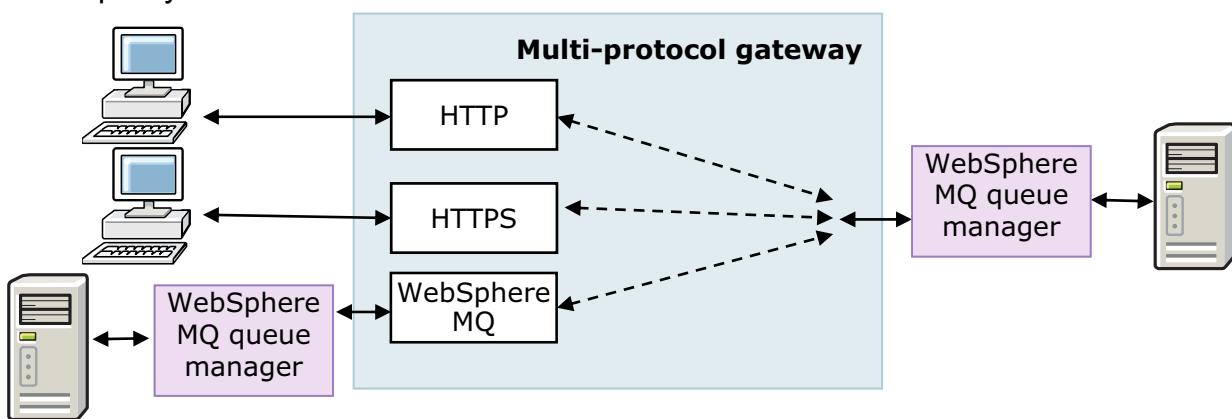
Notes:

The publish/subscribe mode is available with firmware V3.8.0, and WebSphere MQ V7.

Support "fire and forget" or one-way messaging by setting the reply queue to an empty string and the message type to **pass-through**.

Provide WebSphere MQ access

- The multi-protocol gateway can be configured to accept requests from an IBM WebSphere MQ system
 - Request and response messages reside in queues on a WebSphere MQ queue manager
 - All requests are sent to the back-end web service over another set of WebSphere MQ queues
 - Web service request messages that pass through the gateway execute a service policy



© Copyright IBM Corporation 2014

Figure 14-6. Provide WebSphere MQ access

WE601 / ZE6011.1

Notes:

Contact your IBM WebSphere MQ administrator for the host name, port, and queue names in your application. Setting up a WebSphere MQ queue manager is beyond the scope of this presentation.

The DataPower appliance can obtain responses that are associated to a request. The DataPower appliance polls the reply-to queue to find a correlated response message. The gateway examines the correlation ID value in the WebSphere MQ header of messages on the reply-to queue. When this ID is the same as the message ID assigned to the request, the gateway takes the message as the response.

If such a message is found, the multi-protocol gateway can again apply any configured processing policy actions to the response and returns the reply to the requesting HTTP client. This message includes error responses from the back-end application server. If no response is found, the MPGW generates an error to the front side client.

The web service proxy can also use the WebSphere MQ front side handler.

Step 1: Create a WebSphere MQ queue manager (1 of 2)

1. Create a WebSphere MQ queue manager object from the control panel
2. Provide the host name or IP address of the queue manager
 - If the port is not specified, the default value 1414 is used
3. Provide the **queue manager name** if it is different from the default
4. Provide an alternate **channel name** if necessary
5. Enter a user name that identifies the client (the WebSphere MQ queue manager object) to the queue manager

Name	EastAddressQM
Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	WebSphere MQ Queue Manager
Host Name	[REDACTED] (1414)
Queue Manager Name	WBRK6_DEFAULT_QUEUE_MANAGER
Channel Name	EastAddress.Channel
Channel Heartbeat value	300
User Name	MUSR_MQADMIN
Maximum Message Size	1048576

© Copyright IBM Corporation 2014

Figure 14-7. Step 1: Create a WebSphere MQ queue manager (1 of 2)

WE601 / ZE6011.1

Notes:

A WebSphere MQ queue manager object allows a back side handler on the DataPower SOA appliance to access an IBM WebSphere MQ queue manager. The same object also allows WebSphere MQ queue managers to connect to a DataPower service through a front side handler.

The **Queue Manager Name** field is necessary only if a nondefault queue manager name is assigned to this queue manager.

SYSTEM.DEF.SVRCONN represents the default server connection channel.

The **user name** field is used to provide a plain text string that identifies the client to the WebSphere MQ queue manager. You provide a user name with administrative permissions on the local operating system.

During the installation of WebSphere MQ, it creates a user in the local Windows registry that is called MUSR_MQADMIN in the WebSphere MQ user group, which has local OS administrative permissions. In Linux, the default user is "mqm".

The default port number, if it is not specified in the host name field, is 1414.

Step 1: Create a WebSphere MQ queue manager (2 of 2)

6. Specify whether the WebSphere MQ queue manager participates in a transaction
 - 0 (zero), undeliverable messages are silently discarded
 - 1, the queue manager commits or rolls back the transaction
7. Set the total number of open connections
8. Specify the encryption key and type for an SSL connection
9. Configure an automatic retry interval to automatically reconnect to the queue manager

The screenshot shows the 'Queue Manager Properties' dialog box. The following fields are highlighted with yellow circles containing numbers:

- Cache Timeout:** A text input field with a dropdown menu for 'seconds'. Number 6 is overlaid on the input field.
- Units of Work:** A text input field with value '0'. Number 7 is overlaid on the input field.
- SSL Key Repository:** A dropdown menu set to 'cert:///'. Buttons for 'Upload...' and 'Fetch...' are visible. Number 8 is overlaid on the dropdown.
- SSL Cipher Specification:** A dropdown menu set to 'None'. Number 8 is overlaid on the dropdown.
- Convert Input:** A radio button group where 'on' is selected. Number 9 is overlaid on the radio buttons.
- Automatic Retry:** A radio button group where 'on' is selected. Number 9 is overlaid on the radio buttons.
- Retry Interval:** A text input field with value '1' and a 'seconds' unit indicator. Number 9 is overlaid on the input field.
- Reporting Interval:** A text input field with value '1' and a 'seconds' unit indicator. Number 9 is overlaid on the input field.
- Alternate User:** A radio button group where 'on' is selected. Number 9 is overlaid on the radio buttons.
- Local Address:** An empty text input field.
- XML Manager:** A dropdown menu set to 'default'. Buttons for '+', '...', and '*' are visible. Number 9 is overlaid on the dropdown.
- SSL proxy profile:** A dropdown menu set to '(none)'. Buttons for '+', '...', and '*' are visible. Number 9 is overlaid on the dropdown.

© Copyright IBM Corporation 2014

Figure 14-8. Step 1: Create a WebSphere MQ queue manager (2 of 2)

WE601 / ZE6011.1

Notes:

The **SSL Key Repository** specifies the location of the key database file. Set this field to **none** to disable SSL support.

Set **Convert Input** to **on** to convert messages to the default coded character set identifier (CCSID) in use by the IBM WebSphere MQ queue manager. For example, the character encoding that is used on a zSeries system might not match an Intel Linux server that is based on Intel. The CCSID helps the system to interpret the message properly.

The **Local Address** field specifies the Ethernet address to use for the outbound message.

When it is turned on, the **Alternate User** field contains a flag inside a WebSphere MQ message. It allows WebSphere MQ to run a system exit call that calls on custom C/C++ code in WebSphere MQ.



Step 1: Use SSL in mutual authentication mode

- The WebSphere MQ queue manager can be configured to use SSL in mutual authentication mode with a remote WebSphere MQ queue manager
- Execute the following steps:
 - Configure the remote WebSphere MQ queue manager to use SSL
 - In DataPower, generate a self-signed certificate
 - DataPower generates the certificate key pair in the PEM format
 - Use an external application to convert the PEM format to pkcs12
 - This step is required since WebSphere MQ does not understand the PEM format
 - Import the converted certificate key into WebSphere MQ
 - Obtain the WebSphere MQ key database file and import it into DataPower and select it in the **SSL Key Repository** field

SSL Key Repository	cert: <input type="button" value="▼"/> <input type="text" value="EastAddress.kdb"/> <input type="button" value="▼"/>
SSL Cipher Specification	<input type="text" value="TRIPLE_DES_SHA_US"/> <input type="button" value="▼"/>

© Copyright IBM Corporation 2014

Figure 14-9. Step 1: Use SSL in mutual authentication mode

WE601 / ZE6011.1

Notes:

To set up SSL between the DataPower WebSphere MQ client and a WebSphere MQ queue manager, both parties exchange keys that are used during SSL communication.

The first step is to enable SSL for the WebSphere MQ queue manager. In DataPower, you generate the certificate key pair that is imported into WebSphere MQ. DataPower generates certificates in the PEM format, which WebSphere MQ does not support. You convert the PEM format into the pkcs12 format. You can use the OpenSSL tool to convert between certificate formats.

When the certificate key pair is converted, you import it into the WebSphere MQ key database. Finally, you export the WebSphere MQ key database and import it into DataPower. When the key database is imported, you can select it in the **SSL Key Repository** field.

For more information, see the technote "Configuring DataPower WebSphere MQ client to use SSL in mutual authentication mode" at:

<http://www.ibm.com/support/docview.wss?&fdoc=aimwdp&&rs=2362&&uid=swg21260155>

Step 2: Add a WebSphere MQ front side handler

1. Open the multi-protocol gateway from the previous scenario
2. Create a **WebSphere MQ Front Side Handler** to accept requests from a WebSphere MQ system
3. Select the queue manager object that was defined in the previous step
4. Specify the queue for the request messages (Get Queue) and the response messages (Put Queue)
5. Configure the coded character set identifier (CCSID) for the messages on the queue

Create a New:

Name	(Yellow circle with 2)
General	
Administrative State	
Comments	
Queue Manager	(Yellow circle with 3)
Get Queue	(Yellow circle with 4)
Put Queue	
The number of concurrent MQ connections	1
Get Message Options	4097
Polling Interval	30
Retrieve Backout Settings	<input type="radio"/> on <input checked="" type="radio"/> off
Use Queue Manager in URL	<input type="radio"/> on <input checked="" type="radio"/> off
CCSI	(Yellow circle with 5)

Example for point-to-point mode

© Copyright IBM Corporation 2014

Figure 14-10. Step 2: Add a WebSphere MQ front side handler

WE601 / ZE6011.1

Notes:

The administrator on the WebSphere MQ queue manager defines the names of the Get and Put queues.

The publish/subscribe mode uses different fields on the web page.

There is an option to support an asynchronous Put operation (Async Put on or off).



Step 3: Configure a WebSphere MQ back-end transport

1. Click **MQHelper** in the **Back side settings**
2. Select a new or existing queue manager object
3. Set the **URI** that identifies the service on the final back-end destination
4. Specify the request and response queues
5. Set **Transactionality** to **on** if the queues participate in a unit of work
6. Enable the **UserIdentifier** header field, if a processing action is added an identifier
7. Set **ReplyToQ** to **on** to copy the reply ObjectName in MQOD to ReplyToQ in MQMD

The screenshot shows the URL Builder interface with numbered steps:

1. Backend URL: http://9.65.242.185:9080/EastAdd1 *
2. Queue Manager: AddressQM
3. URI: stAddress/services/AddressSearch
4. RequestQueue: SEARCH_REQ
5. Transactionality: on (radio button selected)
6. User Identifier: on (radio button selected)
7. ReplyToQ: on (radio button selected)

Build URL button is at the bottom.

Example for point-to-point mode

© Copyright IBM Corporation 2014

Figure 14-11. Step 3: Configure a WebSphere MQ back-end transport

WE601 / ZE6011.1

Notes:

The **MQHelper** button is displayed if the back-end transport is static.

The queue manager can be the same as the one that is used in the front side handler.

When the **Transaction** option is set to **on**, DataPower does not consider the message successfully posted onto the queue until it receives a response from the queue manager. With the option set to **off**, no confirmation is requested, and successful posting of the message is assumed.

Although it has a different name, the **Transaction** option is similar to the **units of work** field in the WebSphere MQ queue manager object.

The **User Identifier** setting allows the WebSphere MQ back-end transport to add a value to the user identifier header field. This setting adds `?PMO=2052` to the URL. The actual header value itself must be set by header injection or another processing action.

The back-end URL for a WebSphere MQ uses a URI syntax specific to DataPower. For example, the settings in the slide would create a URL of: `dpmq://AddressQM/EastAddress/services/AddressSearch?RequestQueue=SEARCH_REQ;ReplyQueue=SEARCH_RESP`

Publish/subscribe: WebSphere MQ front side handler support

1. Rather than **Get Queue** and **Put Queue** fields, a Topic String needs to be entered
2. Messages are published to a Topic String
3. Subscribers are delivered messages that were published to the Topic Strings to which they are subscribed
4. Specify **Subscription Name** for durable subscription
5. If response is needed, specify **Publish Topic String**
6. If both Get Queue and Subscribe Topic String are present, the Get Queue overrides
7. If both Put Queue field and Publish Topic String are present, the Put Queue overrides

The screenshot shows a user interface titled "Publish and Subscribe". It contains three text input fields:

- Subscribe Topic String**: An empty input field.
- Subscription Name**: An empty input field.
- Publish Topic String**: An empty input field.

© Copyright IBM Corporation 2014

Figure 14-12. Publish/subscribe: WebSphere MQ front side handler support

WE601 / ZE6011.1

Notes:

Publish/subscribe applies to WebSphere MQ V7.



Publish/subscribe: WebSphere MQ back-end transport support

1. Use **MQHelper** in the **Back side settings** as before
2. A queue manager object is still required
3. Similar to the front side handler:
 - The service publishes requests to the **PublishTopicString**
 - Use the **SubscribeTopicString** to receive messages
 - For durable subscriptions, specify the **SubscriptionName**
4. If both RequestQueue and PublishTopicString are entered, the RequestQueue overrides

© Copyright IBM Corporation 2014

Figure 14-13. Publish/subscribe: WebSphere MQ back-end transport support

WE601 / ZE6011.1

Notes:

The helper constructs a DataPower WebSphere MQ URL like:

dpmq://QM/?SubscribeTopicString=yyyy;SubscriptionName=zzz

In the MQHelper dialog:

If both the RequestQueue and PublishTopicString are entered, the RequestQueue overrides.

If both the ReplyQueue and SubscribeTopicString are entered, the ReplyQueue overrides.

If the WebSphere MQ URL is entered manually, whatever is entered, the latest in the URL string overrides.



Message properties

- Name-value pairs that are associated with the message (WebSphere MQ V7)
- Allow DataPower awareness of properties by enabling parsing in WebSphere MQ front side handler
- Messages can be selectively retrieved by specifying an SQL92-style statement as a “message selector”



- Message properties can be manipulated in a style sheet within a Transform Binary action

© Copyright IBM Corporation 2014

Figure 14-14. Message properties

WE601 / ZE6011.1

Notes:

Messages are retrieved from the queue only if the message property satisfies the selector statement.

A selector can also be specified in the WebSphere MQ URL.

WebSphere Education 

Ordered processing of WebSphere MQ messages

- Enforces serial processing of WebSphere MQ messages:
 - Retrieves from the front side queue and is presented to the request rule
 - Exits the request rule and is sent to the back side queue
 - Retrieves from the back side queue and is presented to the response rule
- The message order is the sequence in which messages are pulled from the front side request queue
- Messages are buffered, if needed, to maintain order
- Messages exiting a response rule might get buffered
 - Messages sent to the front-end queue are always delivered in the order that they are pulled from the back side queue

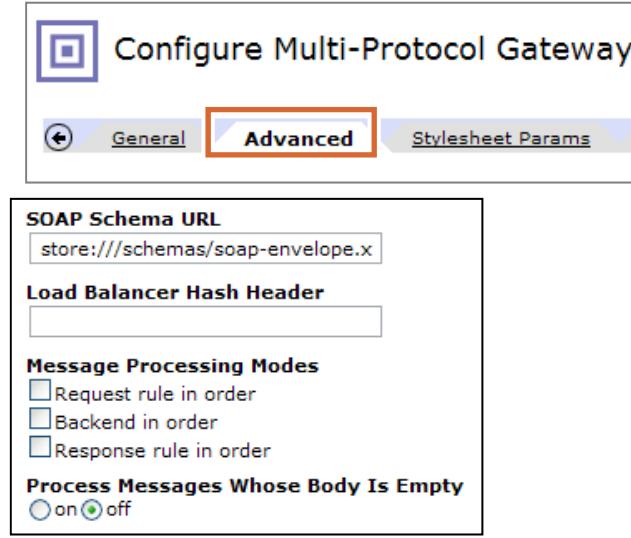


Figure 14-15. Ordered processing of WebSphere MQ messages

WE601 / ZE6011.1

Notes:

Request rule in order: Enforces first-in first-out serial processing of messages for actions in the request rule. The appliance initiates and completes request rule processing for messages in the order in which they were pulled from the front-end request queue. The appliance starts the request rule for the second message in the request queue only after it completes the processing of the first message. The back-end request queue accepts whatever message arrives first, except when you enforce the back-end system to order serial processing. In that case, the appliance buffers messages so that it sends messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Back-end in order: Enforces the serial processing of messages that are delivered to the back-end request queue. If necessary, the appliance buffers messages to complete the request rule that processes out of order. It also delivers messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Response rule in order: Enforces serial processing of messages for actions in the response rule. The appliance initiates and completes response rule processing for messages in the order in which they were pulled from the back-end reply queue. The appliance starts the response rule for the second message in the reply queue only after it completes the processing of the first message. The

appliance always buffers messages so that it sends messages to the front-end reply queue in the same order in which they were pulled from the back-end reply queue.



Controlling backout of WebSphere MQ messages

- WebSphere MQ queue manager object
 - **Units of Work** must be 1
 - **Backout Threshold** indicates the number of reprocessing attempts per message
 - **Backout Queue Name** indicates the queue where messages are placed after they are attempted to the threshold limit

Units of Work	<input type="text" value="1"/>
Automatic Backout	<input checked="" type="radio"/> on <input type="radio"/> off
Backout Threshold	<input type="text" value="3"/>
Backout Queue Name	<input type="text" value="EastAddressQueueError99"/>

- WebSphere MQ front side handler
 - Setting **Retrieve Backout Settings** to **on** causes the appliance to retrieve the backout settings from the actual WebSphere MQ server

<input type="button" value="Retrieve Backout Settings"/>	<input checked="" type="radio"/> on <input type="radio"/> off
--	---

© Copyright IBM Corporation 2014

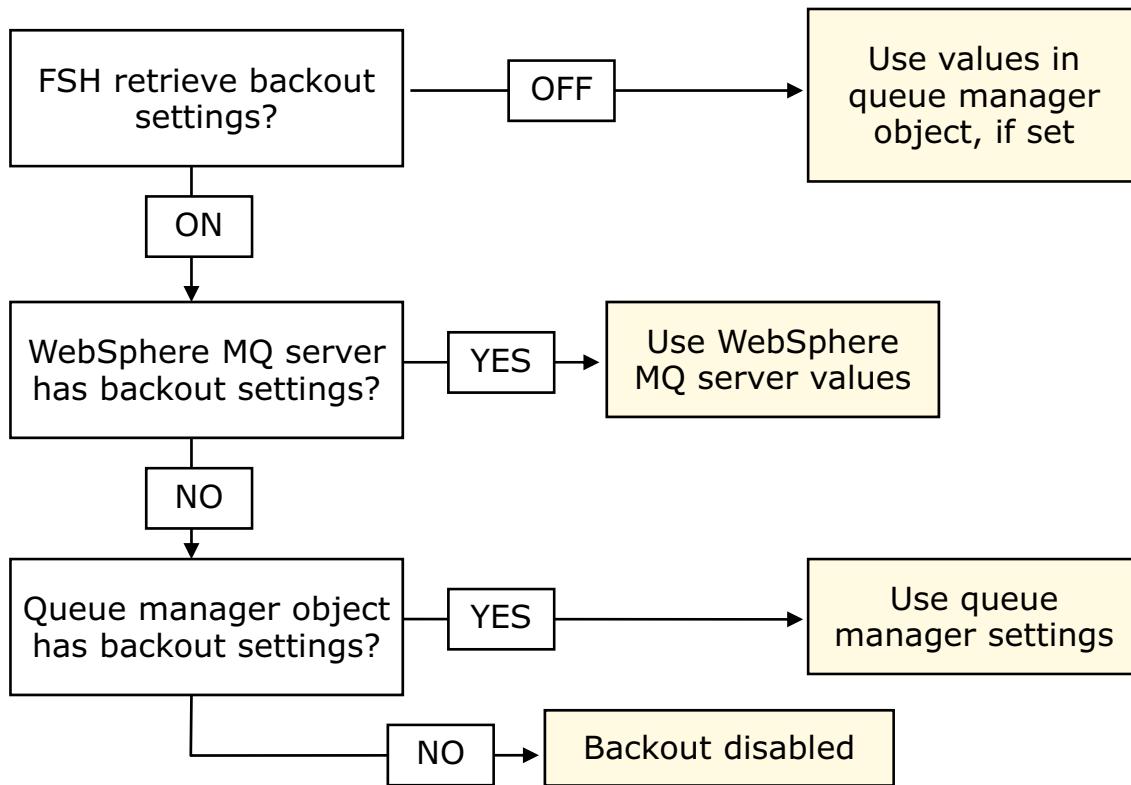
Figure 14-16. Controlling backout of WebSphere MQ messages

WE601 / ZE6011.1

Notes:

The settings in the WebSphere MQ queue manager object determine what the backout settings are for a specific queue manager, unless the handler option can potentially override the setting.

Decision tree for the backout settings



© Copyright IBM Corporation 2014

Figure 14-17. Decision tree for the backout settings

WE601 / ZE6011.1

Notes:



WebSphere MQ header action in service policy

- In policy editor:
Advanced > MQ Header

- Enables manipulation of WebSphere MQ headers without requiring a style sheet
- Allows modification of MQMD request headers, MQMD response headers, queue manager and reply queue for response, message retrieval by message ID, or correlation ID

MQ Header

Asynchronous	<input type="radio"/> on <input checked="" type="radio"/> off
MQ Processing Type	<input checked="" type="radio"/> request <input type="radio"/> response <input type="checkbox"/> Save
MQ Request Header Processing	MQMD for PUT <input type="checkbox"/> Save
Override Existing MQMD	<input type="radio"/> on <input checked="" type="radio"/> off <input type="checkbox"/> Save
Message Id	<input type="text"/>
Correlation Id	<input type="text"/>
Character Set Id	<input type="text"/>
Format Name	<input type="text"/>
ReplyToQ	<input type="text"/>
ReplyToQMgr	<input type="text"/>

© Copyright IBM Corporation 2014

Figure 14-18. WebSphere MQ header action in service policy

WE601 / ZE6011.1

Notes:

The message ID for the current message can be retrieved from `var://service/message-identifier`. You can enter that variable into the entry field on the page. It is similar to the correlation ID and `var://service/correlation-identifier`.

Typical uses of a WebSphere MQ header action

- Retrieve a response message by WebSphere MQ message ID or WebSphere MQ correlation ID
 - Retrieve either value from a DataPower variable and enter it in the appropriate entry field
 - The field with an entry determines which ID is used for retrieval
- Modify MQMD request message headers
 - Selectively override any of the listed headers, or replace them with new and default headers
- Modify MQMD response message headers
 - Selectively override any of the listed headers, or replace them with new and default headers
- Change the queue manager for response messages
 - Modify the destination reply queue manager that is defined in the response header
- Change the reply queue for response messages
 - Modify the destination reply queue that is defined in the response header

© Copyright IBM Corporation 2014

Figure 14-19. Typical uses of a WebSphere MQ header action

WE601 / ZE6011.1

Notes:



Transactions and WebSphere MQ

DataPower allows control of transactions at both ends:

- Front side
 - **Units of Work** parameter in WebSphere MQ Queue Manager object

Configure MQ Queue Manager

Main Advanced

MQ Queue Manager : EastAddressQM [down]

Apply Cancel Undo

Units of Work 1

- Back side
 - **Backend URL:**
sync query parameter,
transactional query parameter

Configure Multi-Protocol Gateway

General Advanced Stylesheet Params

Back side settings

Backend URL
dpmq://EastAddressQM/?RequestC *

© Copyright IBM Corporation 2014

Figure 14-20. Transactions and WebSphere MQ

WE601 / ZE6011.1

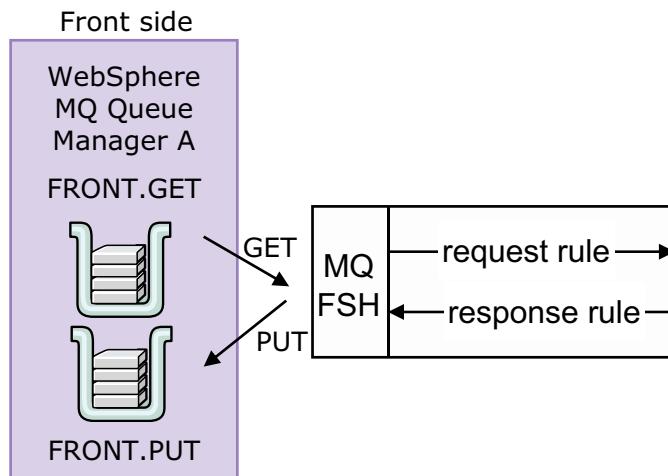
Notes:

A client application generally assumes that a message was PUT successfully, and that a transaction is not explicitly started. It is possible to request acknowledgment of the receipt of the message on the queue.

A transaction (unit of work) can be requested if, for example, multiple messages are PUT to queues within the same transaction.

DataPower does propagate a transaction between two different WebSphere MQ queue managers.

WebSphere MQ front side transactions



- The WebSphere MQ front side handler (MQFSH) gets the message from FRONT.GET
 - If Units of Work = 1, then the transaction with Queue Manager A begins
- When the response rule completes, the MQFSH puts the message to FRONT.PUT
 - If Units of Work = 1, then the transaction with Queue Manager A ends

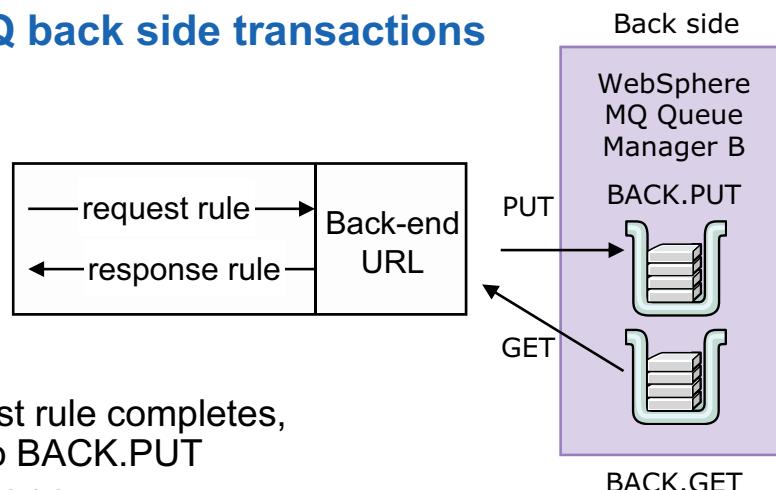
© Copyright IBM Corporation 2014

Figure 14-21. WebSphere MQ front side transactions

WE601 / ZE6011.1

Notes:

WebSphere MQ back side transactions



- When the request rule completes, MQPUT is put to BACK.PUT
 - If Sync = true, MQCOMMIT is then sent
 - Message is available on BACK.PUT and visible to back side WebSphere MQ application
- Service retrieves response message from BACK.GET
 - If Transactional = true, then the transaction with Queue Manager B starts
- Response rule completes
- FSH writes message to FRONT.PUT
 - If Transactional = true, then the transaction with Queue Manager B ends

© Copyright IBM Corporation 2014

Figure 14-22. WebSphere MQ back side transactions

WE601 / ZE6011.1

Notes:

Sync=true is necessary if Queue Manager B is using transactions.

Sync = true|false cannot be set by the WebSphere MQ URL Builder. You must edit the back-end URL field to add the Sync parameter.



WebSphere MQ DataPower URL

- The WebSphere MQ DataPower URL is of the following form:
`dpmq://QueueManager/URI?RequestQueue=PUTQ;ReplyQueue=GETQ;Sync=true;Transactional=true;PMO=2048`
 - QueueManager: name of the WebSphere MQ queue manager object
 - URI: string to include in the URL
 - RequestQueue: name of the queue where messages are sent
 - ReplyQueue: name of the queue to poll for messages
 - Transactional (true or false): used to enforce transactional units of communication with back side queue managers
 - Sync (true or false): used to send MQCOMMIT to back side queue manager after MQPUT
 - PMO: set options on the MQPUT call
- The url-open extension element supports the WebSphere MQ DataPower URL
 - A style sheet can GET and PUT to queues

© Copyright IBM Corporation 2014

Figure 14-23. WebSphere MQ DataPower URL

WE601 / ZE6011.1

Notes:

Existing MQMD headers can be accessed from the variable:

`var://context/contextname/_extension/header/MQMD`



WebSphere MQ queue manager Group object

- Provides failover of WebSphere MQ queue manager objects
 - Consists of a single primary WebSphere MQ queue manager and many backup WebSphere MQ queue managers
- If the primary queue manager becomes unavailable, the DataPower appliance can attempt to place the message on one of the backup WebSphere MQ queue managers

MQ Queue Manager Group

Name	<input type="text" value="EastAddressQMGroup"/> *
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Primary MQ Queue Manager	<input type="text" value="EastAddressQM"/> <input type="button" value="..."/> *
Backup MQ Queue Managers	<input type="text" value="AddressQM"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="text" value="AddressQM"/> <input type="button" value="Add"/> <input type="button" value="..."/>

© Copyright IBM Corporation 2014

Figure 14-24. WebSphere MQ queue manager Group object

WE601 / ZE6011.1

Notes:

If the primary queue manager becomes available again, it returns to "primary" status.



Unit summary

Having completed this unit, you should be able to:

- Create a multi-protocol gateway with a WebSphere MQ front-side handler
- Configure a WebSphere MQ back-end Uniform Resource Locator (URL)
- Manage transactionality between WebSphere MQ queue managers

© Copyright IBM Corporation 2014

Figure 14-25. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: WebSphere MQ support is only available on the multi-protocol gateway.
2. True or False: The DataPower MQ client implementation supports one-way messaging.
3. Match the definitions between local and global units of work:

Description	Definition
1. Local units of work	A. Involves the updating of resources on multiple resource or queue managers
2. Global units of work	B. Involves updating resources of a single resource or queue manager C. DataPower supports

© Copyright IBM Corporation 2014

Figure 14-26. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. **False.** The Web Service Proxy can also use an MQ front side handler.
2. **True.**
3. **1 – B and C, 2 – A.**

© Copyright IBM Corporation 2014

Figure 14-27. Checkpoint answers

WE601 / ZE6011.1

Notes:

Unit 15. DataPower and WebSphere JMS

What this unit is about

This unit describes how to configure a WebSphere JMS front-side handler and WebSphere JMS back-end URL to connect to the default messaging provider in WebSphere Application Server V7.

What you should be able to do

After completing this unit, you should be able to:

- Configure a WebSphere JMS front-side handler to send JMS messages to the default messaging provider in WebSphere Application Server V7
- Configure a back-end WebSphere JMS URL to communicate with the default messaging provider in WebSphere Application Server V7
- Describe the components of the service integration bus on WebSphere Application Server V7

How you will check your progress

- Checkpoint
- Appendix Exercise B: Configuring WebSphere JMS (optional)



Unit objectives

After completing this unit, you should be able to:

- Configure a WebSphere JMS front-side handler to send JMS messages to the default messaging provider in WebSphere Application Server V7
- Configure a back-end WebSphere JMS URL to communicate with the default messaging provider in WebSphere Application Server V7
- Describe the components of the service integration bus on WebSphere Application Server V7

© Copyright IBM Corporation 2014

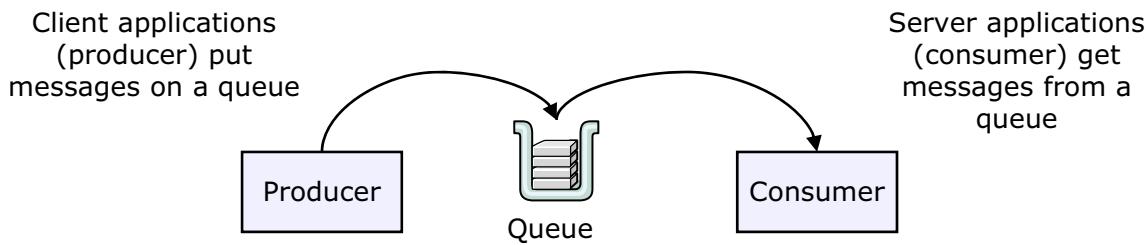
Figure 15-1. Unit objectives

WE601 / ZE6011.1

Notes:

Messaging middleware

- Messaging middleware acts as a broker to provide asynchronous delivery of data between applications
 - Acts as an intermediary between the producer and consumer of the message
 - Ensures reliable message delivery
 - WebSphere MQ is an IBM messaging middleware product
 - WebSphere Application Server includes messaging middleware
 - Enterprise Message Service (EMS) is the TIBCO messaging middleware product
- Messages are sent to destinations and stored until delivered
 - The messaging middleware manages delivery
 - Consumer and producer do not need to be active at the same time



© Copyright IBM Corporation 2014

Figure 15-2. Messaging middleware

WE601 / ZE6011.1

Notes:

There are many different types of middleware: transaction processing monitors, remote procedure calls, and object request brokers.

Messaging middleware stores, routes, and manages messages for delivery. Imagine a courier service that allows you to drop off a package for delivery. That company ensures that your package is delivered before a certain time. If the other party is not available, it holds the package for later delivery or pickup. Messages are routed through the services of messaging middleware in this manner.

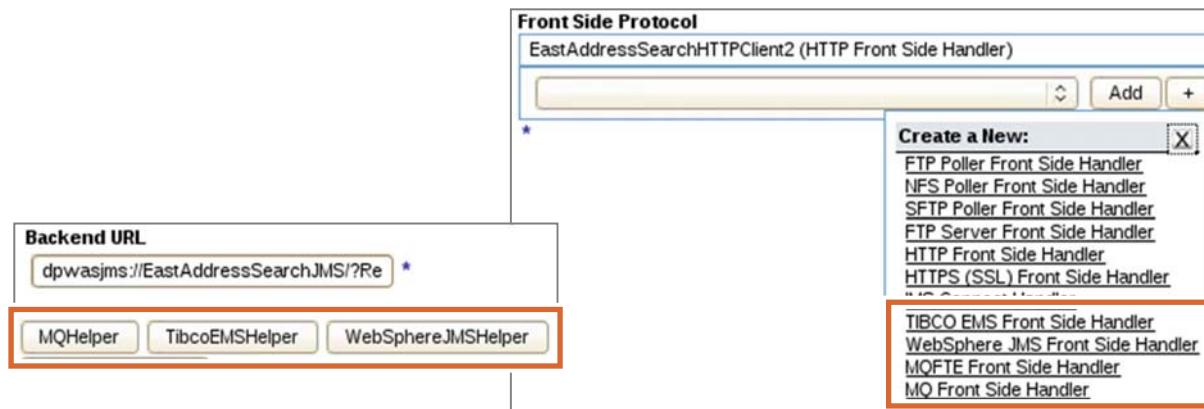
The application that produces the messages is known as the producer, and the application that receives the messages is known as the consumer.

Messages can remain on the queue for some length of time.



DataPower support of messaging middleware

- Message queuing
 - WebSphere MQ
- Java Message Service (JMS)
 - WebSphere Application Server
 - TIBCO Enterprise Message Service (EMS)



© Copyright IBM Corporation 2014

Figure 15-3. DataPower support of messaging middleware

WE601 / ZE6011.1

Notes:

DataPower uses WebSphere MQ to support the MQ model.

It supports the JMS model by using two of the implementations: WebSphere Application Server and TIBCO EMS.

No other implementations of JMS are supported currently.

The support is at both the front side and the back-end system. It is also supported within style sheets. The url-open extension element can call the MQ and JMS implementations from within a style sheet.

Java Message Service

- Java Message Service (JMS) is a Java API for accessing messaging middleware
- JMS provides:
 - Programming interface: does not provide the run time
 - Vendor-neutral and standard approach to use messaging middleware, such as WebSphere MQ
- A messaging middleware that supports JMS is known as a JMS provider
- Java Enterprise Edition requires that applications servers:
 - Include a JMS provider
 - Support JMS to access messaging middleware
- DataPower does not contain a Java run time, so it cannot connect directly to a JMS resource
 - Uses other protocols to connect to the Java resource
 - Customized approach available for only WebSphere Application Server and TIBCO
 - Supports JMS messages within WebSphere MQ messages

© Copyright IBM Corporation 2014

Figure 15-4. Java Message Service

WE601 / ZE6011.1

Notes:

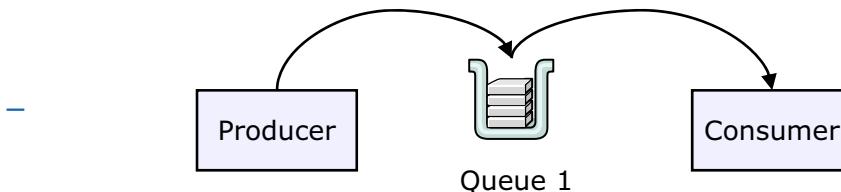
Before JMS, you needed to use a proprietary client API to access messaging middleware.

WebSphere Application Server V7 is a Java EE V5-compliant server that also provides a pure Java JMS V1.1 provider.

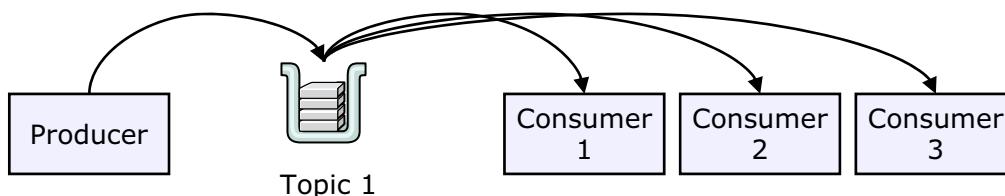
JMS models

JMS provides two messaging models:

- Point-to-point
 - Only one consumer might receive a particular message



- Publish/subscribe
 - Many-to-many
 - Consumers register to receive messages on a topic



© Copyright IBM Corporation 2014

Figure 15-5. JMS models

WE601 / ZE6011.1

Notes:

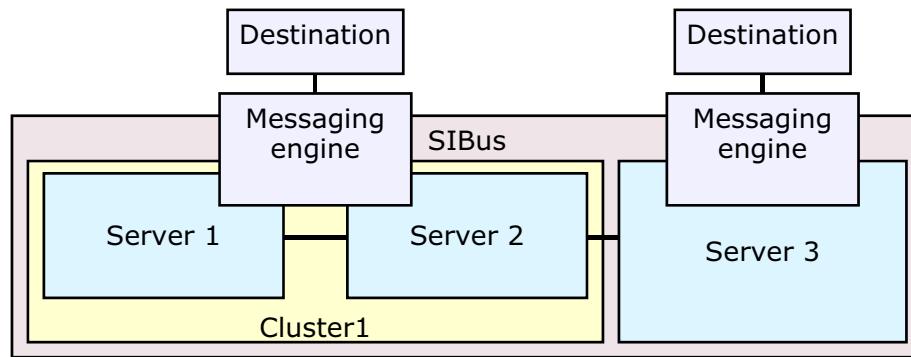
In the point-to-point messaging domain, it is not necessary for the message consumer and producer to be available at the same time. If either one is unavailable, the JMS queue keeps the message.

The publish/subscribe model defines one or more message consumers that subscribe to a particular topic. The JMS sender transmits a message to a particular topic.

Use **durable subscriptions** to deliver messages even if the message consumer is temporarily unavailable. This option stores the message in the topic, ready to be retrieved when the consumer becomes available.

WebSphere service integration bus (SIBus)

- Communication infrastructure that provides service integration through synchronous and asynchronous messaging; is part of WebSphere Application Server
- SIBus is an interconnection of bus members
 - Bus member is either a server cluster or server
 - Each bus member has an associated messaging engine
 - Destinations (queue, topic) are linked to a messaging engine
 - Logical construct
- When SIBus is used for JMS applications, it is referred to as a messaging bus



© Copyright IBM Corporation 2014

Figure 15-6. WebSphere service integration bus (SIBus)

WE601 / ZE6011.1

Notes:

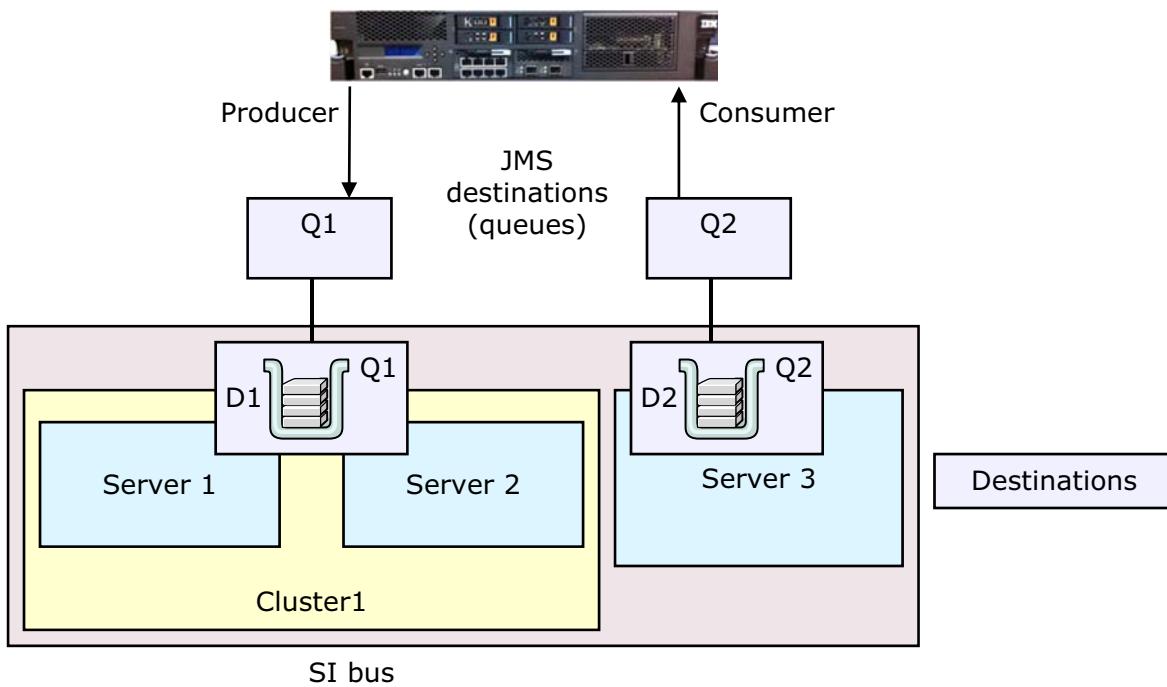
Any application can exchange messages with any other application by using a **destination**.

A message-producing application, that is, a **producer**, can produce messages for a destination regardless of which messaging engine the producer uses to connect to the bus.

A message-consuming application, that is, a **consumer**, can consume messages from a destination (whenever that destination is available) regardless of which messaging engine the consumer uses to connect to the bus.

JMS queue resources on SIBus

- JMS applications use JMS **queues** for point-to-point messaging



© Copyright IBM Corporation 2014

Figure 15-7. JMS queue resources on SIBus

WE601 / ZE6011.1

Notes:

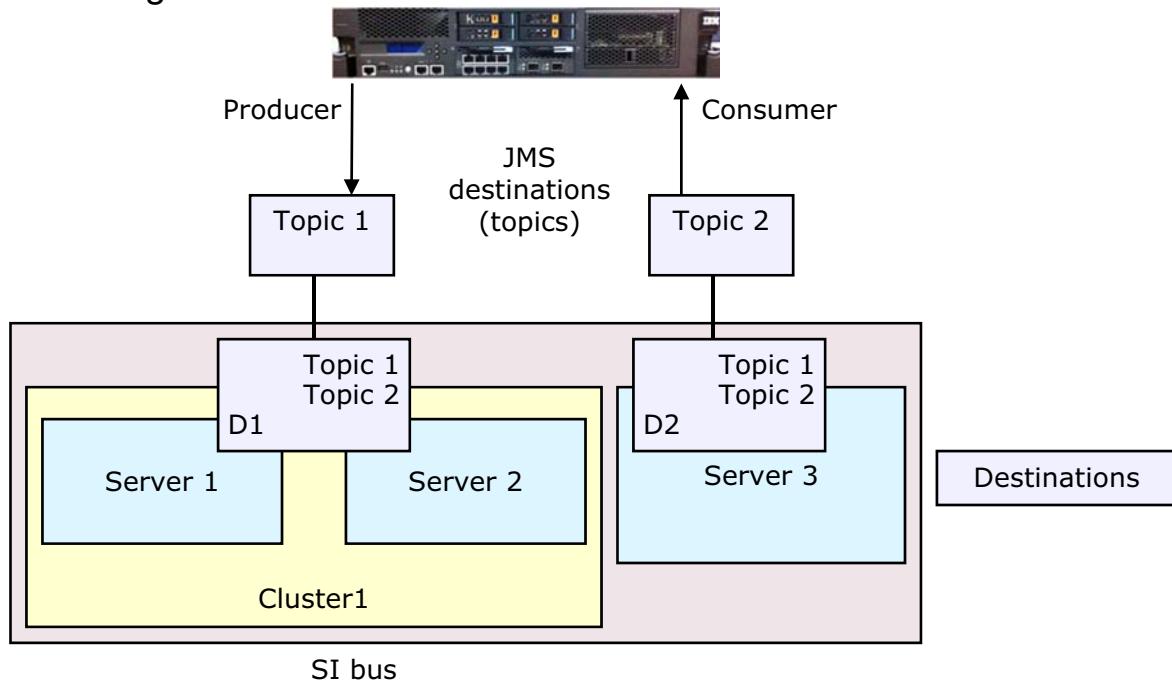
JMS destinations are configured on the application server. They encapsulate the name of the actual queue destination on the service integration bus.

D1 and D2 are destinations.

Q1 and Q2 are JMS queues.

JMS topic resources on SIBus

- JMS applications use JMS **topics** to publish and subscribe to messages



© Copyright IBM Corporation 2014

Figure 15-8. JMS topic resources on SIBus

WE601 / ZE6011.1

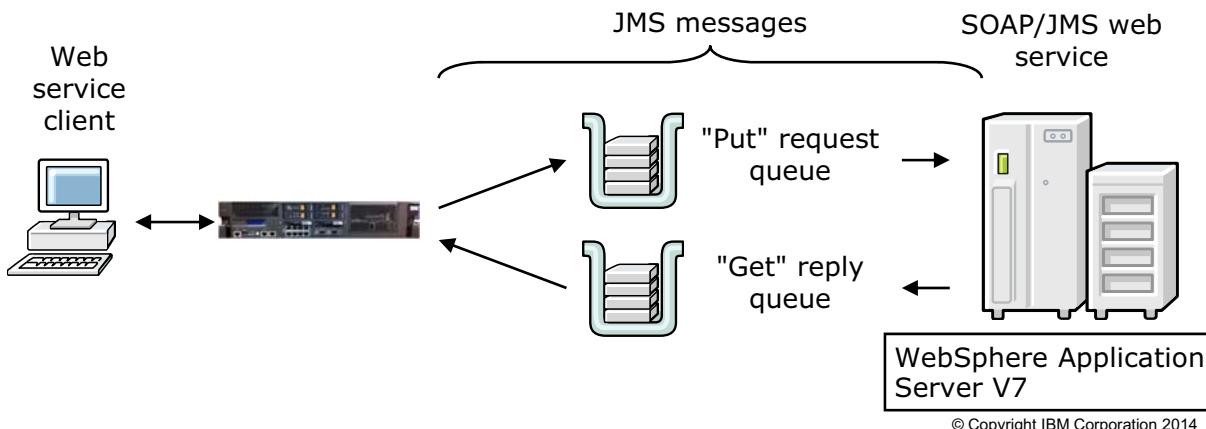
Notes:

An application such as DataPower interacts with a JMS topic, which is a JMS resource that is configured on the default messaging provider.

JMS applications receive messages only from a topic when it is connected to a server.

WebSphere JMS support

- DataPower supports JMS messaging to the default messaging provider in WebSphere Application Server V7 by using the **service integration bus**
 - Firmware uses the IBM JetStream Formats and Protocols (JFAP) to communicate with the service integration bus
- Scenario: Send SOAP messages over JMS by using the DataPower WebSphere JMS front side handler
 - SOAP or JMS web service implementation can read messages from a JMS queue and put response onto another queue



© Copyright IBM Corporation 2014

Figure 15-9. WebSphere JMS support

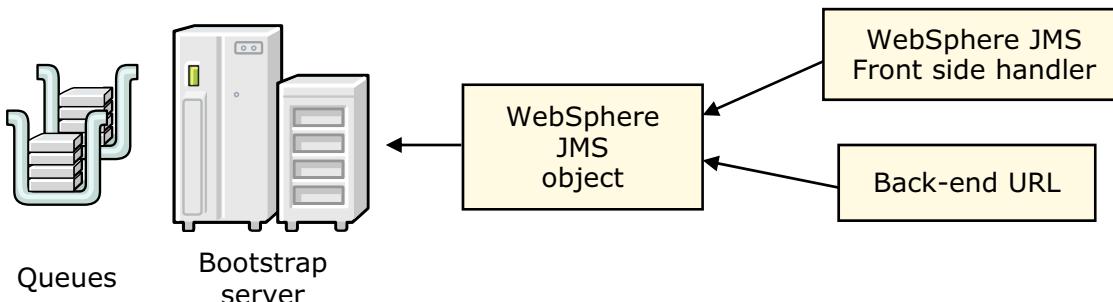
WE601 / ZE6011.1

Notes:

You must create and configure both a service integration bus and the JMS resources for a JMS client to invoke a SOAP/JMS web service that is running on WebSphere Application Server V7.

DataPower and WebSphere JMS interaction

- The **WebSphere JMS** object represents the connection to the bootstrap server on WebSphere Application Server
 - Remote WebSphere JMS applications are unable to communicate directly with a messaging engine on a bus. The bootstrap server enables this communication.
- The **WebSphere JMS Front Side Handler** uses the WebSphere JMS object to communicate with specific queues.
- The **Backend URL** also uses the WebSphere JMS object to manage communication to a back-end JMS service.



© Copyright IBM Corporation 2014

Figure 15-10. DataPower and WebSphere JMS interaction

WE601 / ZE6011.1

Notes:

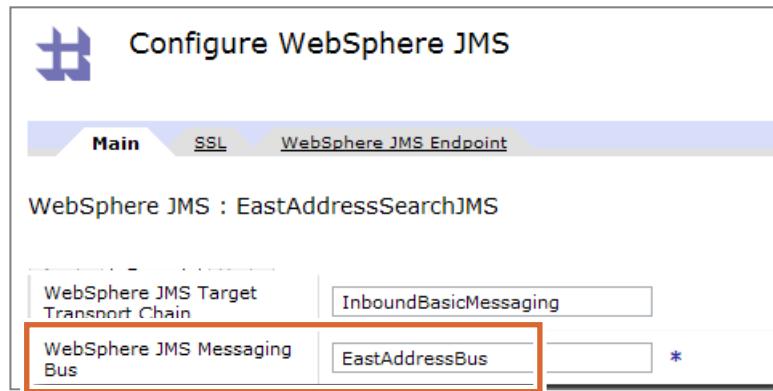
The default WebSphere JMS port is 7276.

The WebSphere JMS FSH and the back-end URL refer to the WebSphere JMS object, and they also define which queues on that object are being accessed.



WebSphere JMS object: Main (1 of 2): Messaging bus

- The WebSphere JMS server object requires the name of the **WebSphere JMS Messaging Bus**
- The **Transport Chain** is predefined on the application server



© Copyright IBM Corporation 2014

Figure 15-11. WebSphere JMS object: Main (1 of 2): Messaging bus

WE601 / ZE6011.1

Notes:

The WebSphere JMS messaging bus name is specified in WebSphere Application Server.

The options for the **WebSphere JMS Target Transport Chain** are:

- InboundBasicMessaging** (the default): specifies the predefined `InboundBasicMessaging` transport chain (JFAP-TCP/IP)
- InboundHTTPMessaging**: specifies the predefined `InboundHTTPMessaging` transport chain (tunnels JFAP by using HTTP wrappers)
- InboundHTTPSMessaging**: specifies the predefined `InboundHTTPSMessaging` transport chain (tunnels JFAP by using HTTPS wrappers)
- InboundSecureMessaging**: specifies the predefined `InboundSecureMessaging` transport chain (JFAP-SSL-TCP/IP)

There are more specifications on the tab; they are covered on the next slide.

WebSphere JMS object: Main (2 of 2): Optional settings

- User Name:** Account name that is used to access the server
- Default Message Type:** Byte or Text, is useful when the message type cannot be determined from the headers
- Automatic Retry:** Attempts to reestablish a lost connection

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
User Name	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
Transactional	<input type="radio"/> on <input checked="" type="radio"/> off
Memory Threshold	268435456 bytes
Maximum Message Size	1048576 bytes
Default Message Type	Byte <input type="button" value="▼"/>
Total Connection Limit	64
Maximum number of Sessions per Connection	100
Automatic Retry	<input checked="" type="radio"/> on <input type="radio"/> off
Retry Interval	1 seconds
Enable JMS-Specific Logging	<input type="radio"/> on <input checked="" type="radio"/> off

© Copyright IBM Corporation 2014

Figure 15-12. WebSphere JMS object: Main (2 of 2): Optional settings

WE601 / ZE6011.1

Notes:

Transactional: Enables (**on**) or disables (**off**) transaction-based processing, in which messages are acknowledged only after the transaction succeeds. Transaction-based processing is disabled by default.

Memory Threshold: Specifies the maximum memory allocation for pending messages. Enter an integer (within the range 1048576 - 1073741824) that specifies the maximum memory (in bytes) that is allocated for pending messages. By default, the maximum memory allocation is set at 268,435,456 bytes.

Maximum Message Size: Specifies the maximum message size that the WebSphere Application Server JMS object can support. Enter an integer (with the range 0 - 1073741824) that specifies the maximum message size in bytes. By default, the maximum message size is set at 1048576 bytes. You can use the special value **0** to disable the enforcement of maximum message sizes.

Default Message Type:

- Byte:** The message payload is accessed as a Java byte array.
- Text:** The message payload is accessed as a Java string value. This message type is suitable for SOAP and XML input.

- **Enable JMS-Specific Logging:** Enables or disables an expanded JMS logging facility. Disabled is the default state.



WebSphere JMS object: WebSphere JMS Endpoint

- The **WebSphere JMS Endpoint** in the WebSphere JMS object is used to connect to a bootstrap server on WebSphere Application Server

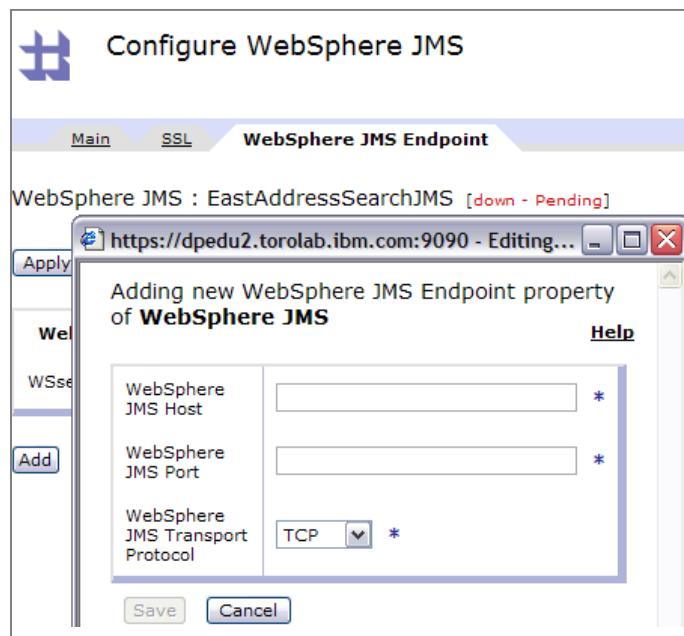


Figure 15-13. WebSphere JMS object: WebSphere JMS Endpoint

© Copyright IBM Corporation 2014

WE601 / ZE6011.1

Notes:

The default WebSphere JMS port is 7276.

WebSphere JMS Transport Protocol: Selects the predefined transport chain that the WebSphere bootstrap server supports and is used for information exchange between the WebSphere JMS object and the bootstrap server. The choices are: TCP, SSL, HTTP, and HTTPS.



Communicating to WebSphere JMS

- The multi-protocol gateway and web service proxy service support WebSphere JMS as a front side handler and back-end URL
 - Polls messages from a JMS queue on the front-side
 - Puts response on a reply queue
 - Sends messages to a JMS queue on the back-end system
 - Reads response from a reply queue
- Similar to the WebSphere MQ support





© Copyright IBM Corporation 2014

Figure 15-14. Communicating to WebSphere JMS

WE601 / ZE6011.1

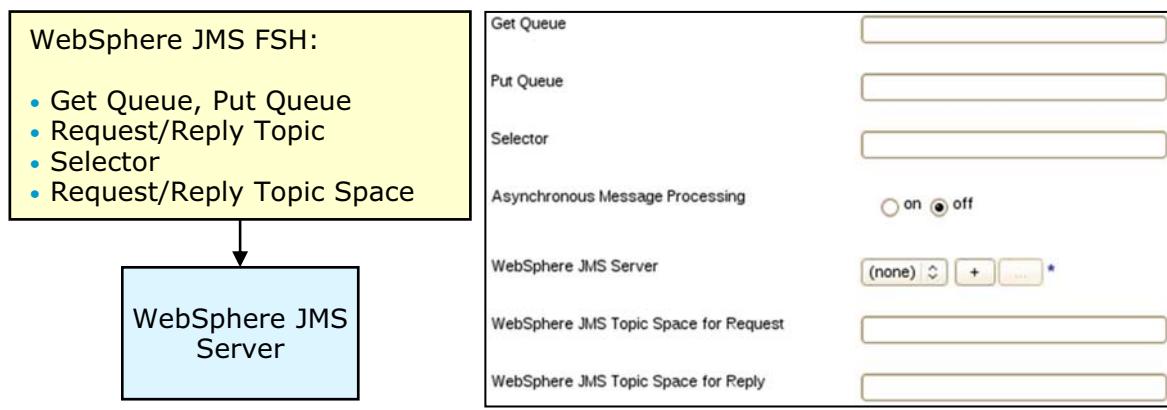
Notes:

A response queue is optional. The WebSphere JMS front side handler also supports one-way messaging.

WebSphere JMS Front Side Handler

The multi-protocol gateway and web service proxy service use the WebSphere JMS Front Side Handler (FSH) object to support JMS messaging

- Two messaging models use the Get and Put Queues:
 - Point-to-point messaging: Queues
 - Publish/subscribe messaging: Topics
- Select an existing WebSphere JMS server object – it defines how to connect to default messaging provider in WebSphere Application Server V7
- The **Selector** field allows you to filter the Get queue for messages to process
- The Request/Reply Topic space is used to uniquely identify topics in multiple destinations



© Copyright IBM Corporation 2014

Figure 15-15. WebSphere JMS Front Side Handler

WE601 / ZE6011.1

Notes:

The actual queues and topics that are defined in the WebSphere JMS FSH are mapped to destinations on the SIBus messaging engine.

The **Selector** field uses an SQL-like syntax for specifying the filter conditions.

WebSphere Education

IBM

WebSphere JMS back-end URL

- Multi-protocol gateway

Backend URL
dpwasjms://EastAddressSearchJMS/?Re *

MQHelper TibcoEMSHelper WebSphereJMSHelper
IMSConnectHelper

Build a WebSphere JMS URL

Server: EastAddressSearchJMS *
Request Queue: EastAddressQueueReq
Reply Queue: EastAddressQueueResp
Request Topic Space:
Response Topic Space:
Selector:
Timeout:

- Web service proxy

Remote

Protocol	WebSphere JMS	Remote URI
DPWASJMS	EastAddressSearchJMS	?RequestQueue=EastAddressQueueReq

© Copyright IBM Corporation 2014

Figure 15-16. WebSphere JMS back-end URL

WE601 / ZE6011.1

Notes:

The generated JMS URL in this example is: dpwasjms://EastAddressSearchJMS/?RequestQueue=EastAddressQueueReq&ReplyQueue=EastAddressQueueResp

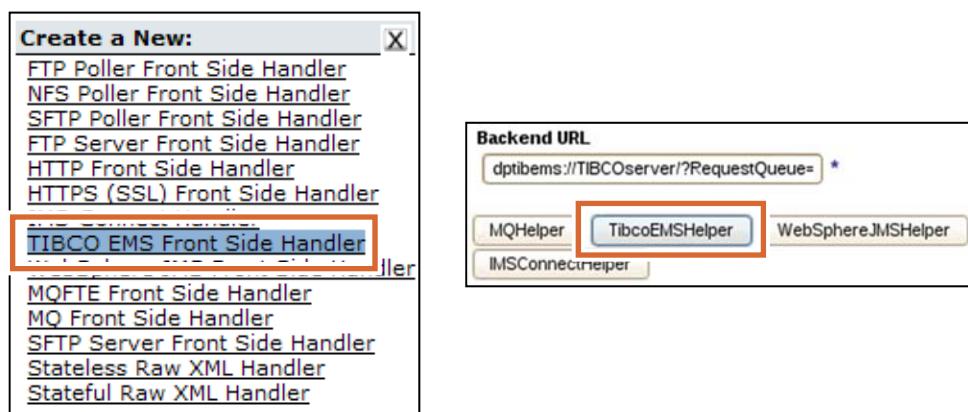
The generated URL is specific to DataPower.

The **Selector** field uses an SQL-like syntax for specifying the filter conditions.



TIBCO EMS support

- DataPower supports JMS messaging with TIBCO EMS
- The TIBCO EMS object represents the connection to the JMS support in TIBCO EMS
 - Similar to the WebSphere JMS object
- Use a TIBCO EMS front side handler and a TIBCO EMS back-end URL
 - Similar to the equivalent WebSphere objects



© Copyright IBM Corporation 2014

Figure 15-17. TIBCO EMS support

WE601 / ZE6011.1

Notes:

JMS support with TIBCO EMS is similar to the support with WebSphere Application Server.

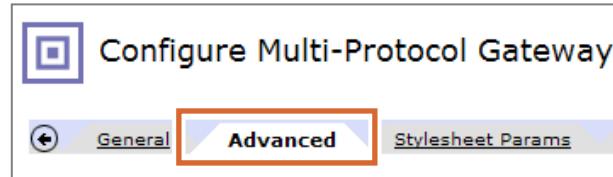


Ordered processing of JMS messages

- Enforces serial processing of JMS messages when the message:
 - Is retrieved from the front side queue and is presented to the request rule
 - Exits the request rule and is sent to the back side queue
 - Is retrieved from the back side queue and presented to the response rule

- The message order is the sequence in which messages are pulled from the front side request queue

- Messages are buffered, if needed, to maintain order
- Messages exiting a response rule might get buffered:
 - Messages sent to the front-end queue are always delivered in the order in which they are pulled from the back side queue



SOAP Schema URL	store:///schemas/soap-envelope.x
Load Balancer Hash Header	
<input type="text"/>	
Message Processing Modes	
<input type="checkbox"/> Request rule in order <input type="checkbox"/> Backend in order <input type="checkbox"/> Response rule in order	
Process Messages Whose Body Is Empty	
<input checked="" type="radio"/> on <input type="radio"/> off	

© Copyright IBM Corporation 2014

Figure 15-18. Ordered processing of JMS messages

WE601 / ZE6011.1

Notes:

Request rule in order: Enforces first-in first-out serial processing of messages for actions in the request rule. The appliance initiates and completes request rule processing for messages in the order in which they were pulled from the front-end request queue. The appliance starts the request rule for the second message in the request queue only after it completes processing the first message. The back-end request queue accepts whatever message arrives first, except in the case where you enforce the back-end to order serial processing. In that case, the appliance buffers messages so that it sends messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Backend in order: Enforces the serial processing of messages that are delivered to the back-end request queue. If necessary, the appliance buffers messages that complete request rule processing out of order. The appliance delivers messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Response rule in order: Enforces serial processing of messages for actions in the response rule. The appliance initiates and completes response rule processing for messages in the order in which they were pulled from the back-end reply queue. The appliance starts the response rule for the second message in the reply queue only after it completes processing the first message. The

appliance always buffers messages so that it sends messages to the front-end reply queue in the same order in which they were pulled from the back-end reply queue.

Ordered messaging applies to WebSphere JMS and TIBCO EMS.

The **web service proxy** has the message processing modes under the **Advanced Proxy Settings** tab.

Unit summary

Having completed this unit, you should be able to:

- Configure a WebSphere JMS front-side handler to send JMS messages to the default messaging provider in WebSphere Application Server V7
- Configure a back-end WebSphere JMS URL to communicate with the default messaging provider in WebSphere Application Server V7
- Describe the components of the service integration bus on WebSphere Application Server V7

© Copyright IBM Corporation 2014

Figure 15-19. Unit summary

WE601 / ZE6011.1

Notes:

Checkpoint questions

1. True or False: A point-to-point model can have only one consumer receive a particular message. In the publish/subscribe model, many consumers can register and receive messages.
2. True or False: The DataPower WebSphere JMS support allows you to send messages to non- WebSphere Application Server JMS providers.
3. Select the three mandatory steps to configure a DataPower WebSphere JMS front side handler:
 - A. Define the queues and destinations
 - B. Define the WebSphere JMS endpoint
 - C. Enter the name of the service integration bus
 - D. Turn on the JMS CloudBurst

© Copyright IBM Corporation 2014

Figure 15-20. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

- 1. True.**
- 2. False.** The DataPower WebSphere JMS works only with the default messaging provider in WebSphere Application Server V7.
- 3. A, B, and C.**

© Copyright IBM Corporation 2014

Figure 15-21. Checkpoint answers

WE601 / ZE6011.1

Notes:

Exercise 12



Configuring a multi-protocol gateway service with WebSphere MQ

© Copyright IBM Corporation 2014

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

5.0 9.0

Figure 15-22. Exercise 12

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Create a WebSphere MQ front-side handler (FSH) that gets messages from a queue and puts responses on a queue
- Send messages from a multi-protocol gateway service to a queue in WebSphere MQ in a fire-and-forget messaging pattern
- Configure transactionality between WebSphere DataPower and WebSphere MQ when errors occur during message processing

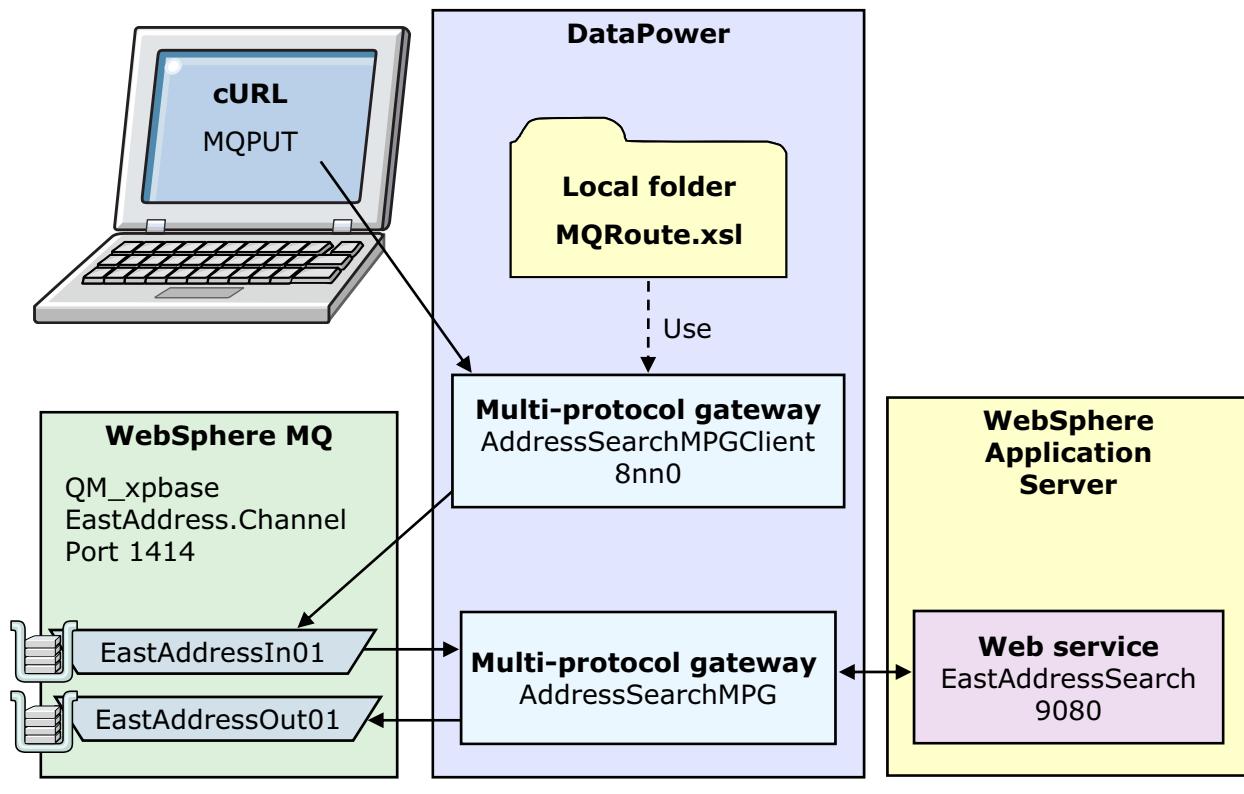
© Copyright IBM Corporation 2014

Figure 15-23. Exercise objectives

WE601 / ZE6011.1

Notes:

Exercise overview



© Copyright IBM Corporation 2014

Figure 15-24. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 16. REST and JSON support for Web 2.0 and Mobile applications

What this unit is about

DataPower services support mobile and desktop clients that communicate by using a REST pattern and JSON structures. If necessary, the appliance can be configured to convert the REST/JSON request into a SOAP request that an existing web service application requires. This unit covers the capabilities that are built into DataPower that support REST and JSON interactions.

What you should be able to do

After completing this unit, you should be able to:

- Use a DataPower appliance to proxy back-end applications for mobile clients
- Describe the purpose of a REST architecture
- Add support to DataPower services for the REST application programming interface (API)
- Describe how to integrate with systems by using RESTful services
- Use the DataPower appliance to proxy a RESTful service

How you will check your progress

- Checkpoint
- Exercise 13: Creating a AAA policy by using LDAP



Unit objectives

After completing this unit, you should be able to:

- Use a DataPower appliance to proxy back-end applications for mobile clients
- Describe the purpose of a REST architecture
- Add support to DataPower services for the REST application programming interface (API)
- Describe how to integrate with systems by using RESTful services
- Use the DataPower appliance to proxy a RESTful service

© Copyright IBM Corporation 2014

Figure 16-1. Unit objectives

WE601 / ZE6011.1

Notes:

Alternatives to SOAP-based web services

- SOAP-based web services work well in certain situations:
 - Provide an interoperable communications platform between computer systems
 - Communicate between different parts of a business process engine and business process management (BPM) solutions
- However, web services are at times too complex for human-facing applications:
 - Web services do not address the presentation layer; extra programming that is needed to support human to computer communication
 - XML-based data structures tend to be more verbose than necessary for simple clients, such as JavaScript applications in a web browser-to-server scenario
- For rich web applications, there is a need for a simpler communications format accessible to every developer skill level

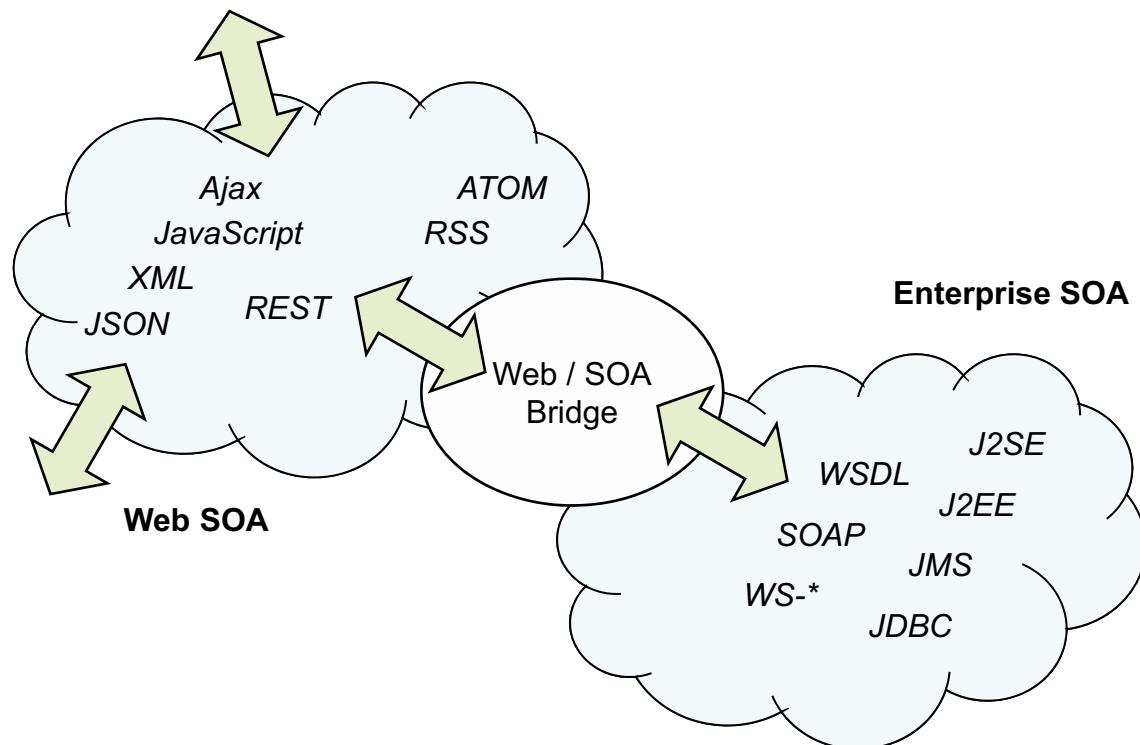
© Copyright IBM Corporation 2014

Figure 16-2. Alternatives to SOAP-based web services

WE601 / ZE6011.1

Notes:

Web SOA (Web 2.0) versus Enterprise SOA



© Copyright IBM Corporation 2014

Figure 16-3. Web SOA (Web 2.0) versus Enterprise SOA

WE601 / ZE6011.1

Notes:

The enterprise SOA and web SOA have different standards, protocols, and techniques. At times, a Web 2.0 client needs to access the enterprise SOA, so some bridging code is necessary. The typical Web 2.0 client platform is a desktop web browser.

Web SOA protocols and standards

- Transport and invocation protocols and techniques
 - Hypertext Transfer Protocol (HTTP/HTTPS)
 - Representational State Transfer (REST)
 - Comet
- Data format standards and techniques
 - Extensible Mark Up Language (XML)
 - Plain-old XML over HTTP (POX/HTTP)
 - JavaScript Object Notation (JSON)
 - JSON-RPC
 - Bayeux
 - Really Simple Syndication (RSS)
 - Atom

© Copyright IBM Corporation 2014

Figure 16-4. Web SOA protocols and standards

WE601 / ZE6011.1

Notes:

Comet is a programming and design technique. It is a model in which a web server can push data to the browser, without the browser explicitly requesting it.

JSON-RPC is a remote procedure call (RPC) protocol that is coded in a JSON format.

Bayeux is a JSON-based protocol for publish/subscribe event management. It uses Ajax and Comet.

RSS is a web feed format for supporting syndications, such as blogs and news. It uses an XML-based structure. There are many versions of RSS, and not all are compatible.

The term “Atom” refers to two related standards. The Atom Syndication Format is an XML structure for web feeds. The Atom Publishing Protocol is a protocol for creating and updating web resources. Atom was developed as an alternative, and improvement to RSS.



Growth of mobile clients

- Different platforms and form factors
 - Smartphones
 - Tablets
- Different types of interfaces
 - Mobile Web
 - Web browser on mobile device
 - Might be the same as the desktop browser page or might be customized for the device size and screen resolution
 - Hybrid/native
 - Mobile application that is written for the specific device, usually by using an SDK
 - Not browser-based
- Mobile clients commonly use REST and JSON
- Desktop web browser interface still needs to be available



© Copyright IBM Corporation 2014

Figure 16-5. Growth of mobile clients

WE601 / ZE6011.1

Notes:

Desktop web browsers are no longer the single target client. Many clients now use smartphones and tablets, in addition to their desktop browsers.

Mobile clients might continue to use a web browser interface, but use it from their device instead of a desktop.

Mobile applications can be developed that use the native capabilities of the device, and look different from the web browser interface. These types of applications are usually written by using a software development kit (SDK).

DataPower as the reverse proxy for Web 2.0 / Mobile clients

- Clients communicate with DataPower as if it is a REST/JSON service
- Target DataPower service sends request to appropriate back-end
 - Converts REST/JSON to XML/web services if needed
 - Applies other mediation to message as needed (AAA, transform, filter)

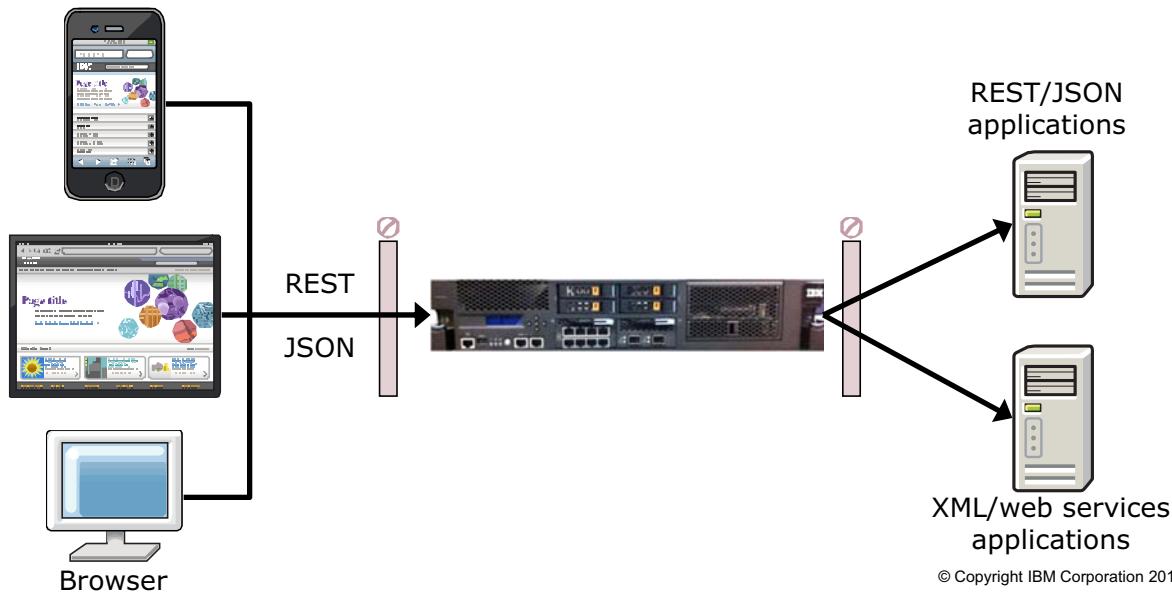


Figure 16-6. DataPower as the reverse proxy for Web 2.0 / Mobile clients

WE601 / ZE6011.1

Notes:

Regardless of whether the client is on a mobile device, or a desktop browser, they can all communicate to the application by using REST and JSON.

The clients see only the DataPower service, and are not aware of the actual back-end application. The back-end applications, regardless of technology, accept traffic only from the appliance. This arrangement is commonly called a “reverse proxy”.

Introduction to REST

- Representational State Transfer (REST) is an architectural style for accessing resources across a network:
 - Application state and functionality are divided into resources
 - Every resource is uniquely addressable with a universal syntax
 - All resources are accessible with a uniform, generic interface
 - A client/server architecture with a pull-based interaction style
 - Each request from the client to the server must contain all necessary information to understand the request
 - REST architectures cannot rely on stored context on the server
- REST is a design pattern, not a standard
- In a Web 2.0 context, REST describes a way to design web applications that:
 - Address resources through URLs
 - Access resources through HTTP methods

© Copyright IBM Corporation 2014

Figure 16-7. Introduction to REST

WE601 / ZE6011.1

Notes:

REST originated from a PhD dissertation from Roy Thomas Fielding called "Architectural Styles and the Design of Network-based Software Architectures." See
<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>



REST style services

- GET
 - Fetch a resource from the server
 - Idempotent
 - No side-effects
- DELETE
 - Remove a resource from the server
 - Idempotent
 - Has side-effect
- PUT
 - Update an existing resource
 - Depending on the defined interface, can also create a resource at a specific URL
 - Idempotent
 - Has side-effect
- POST
 - Create a resource in this collection. Let the server pick the resource's URL
 - Also for invoking a handler, “process this”
 - Non-idempotent
 - Usually has side-effects

© Copyright IBM Corporation 2014

Figure 16-8. REST style services

WE601 / ZE6011.1

Notes:

A REST service is based on well-defined verbs, and well-defined rules about what operations are allowed when using those verbs.

Running an idempotent request more than once yields the same result as would result by running it once. If a client sends a series of idempotent requests, then gets disconnected from the server without getting confirmation that the requests completed. The client can safely retry the series of idempotent requests without worrying that duplicate records are created, or data are deleted that should not be.

For REST in a web environment, these verbs match to the HTTP methods.

Example – Employee processing

- Resources
 - List of Employees
 - Employee
 - Employee salary history
 - Employee performance reviews
- Operations
 - Add employee (Create)
 - Query for information about employee / employees (Read)
 - Modify an employee's information (Update)
 - Remove an employee (Delete)
 - CRUD for salary history
 - Give an employee a raise = update salary history
 - CRUD for performance reviews

© Copyright IBM Corporation 2014

Figure 16-9. Example - Employee processing

WE601 / ZE6011.1

Notes:

Here is an example of simple service for processing employees. You defined the resources that are exposed, and a description of the operations you want to perform on those resources.

Employee REST interface

Resource	URI	Method	Representation	Description	Status Codes *
Employee List	/employees	GET	XML (employee list)	Retrieve list of employees	200
Employee	/employee	POST	XML (employee)	Create new employee	201
"	/employee/{id} or follow href from list	GET	XML (employee)	Retrieve a single employee	200, 404
"	"	PUT	XML (employee)	Update an employee	201, 204, 404
"	"	DELETE	not applicable	Remove an employee	200, 404
Salary History	/employee/salary/{emp-id}	POST	XML (raise)	Give an employee a raise	201, 404
Performance Reviews	/reviews/{emp-id}	GET	XML (review list)	Retrieve an employee's reviews	200, 404

* Because of access control, status codes might include 401 and 403

© Copyright IBM Corporation 2014

Figure 16-10. Employee REST interface

WE601 / ZE6011.1

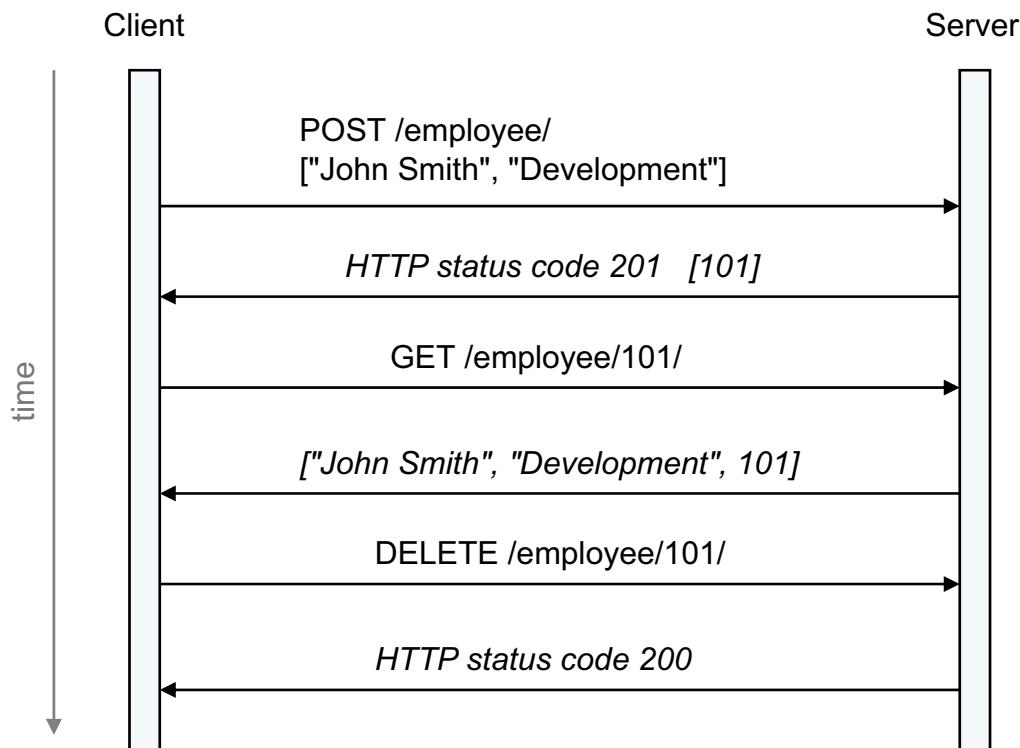
Notes:

Here is your design for the employee processing service. You defined the resources, defined the formats, picked the operations, and defined the status codes.

The typical status codes that you encounter are:

- 200 – OK
- 201 – resource created
- 204 – no content, successful request but no body returned
- 3XX – redirection
- 400 – bad request, malformed syntax of request
- 401 – unauthorized, a WWW-Authenticate header is returned with a challenge dialog
- 403 – forbidden, request is refused by the server
- 404 – not found
- 500 – internal server error

Example: REST interaction



© Copyright IBM Corporation 2014

Figure 16-11. Example: REST interaction

WE601 / ZE6011.1

Notes:

A RESTful interaction shows a client interacting with HTTP and a RESTful URI, and describing intent with the HTTP method.

Example: Add employee REST request explained

POST /employee/ HTTP/1.1

Accept-Language: en-US

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US)

Accept: */*

Cache-Control: max-age=0

Connection: keep-alive

Host: www.example.org

<employee>

 <firstName>**John**</firstName>

 <lastName>**Smith**</lastName>

 <department>**Development**</department>

 <id></id>

</employee>

The URI specifies the resource being accessed. It does not expose the underlying service implementation.

The HTTP method specifies the operation to be performed.

The HTTP request body contains information about the resource. Additional types include binary attachments, such as images.

© Copyright IBM Corporation 2014

Figure 16-12. Example: Add employee REST request explained

WE601 / ZE6011.1

Notes:

Example: Add employee REST response explained

HTTP/1.1 201 OK

Accept-Ranges: bytes

Server: Apache/2.2.3 (Red Hat)

Last-Modified: Wed, 19 Mar 2008 21:51:16 GMT

Expires: Wed, 19 Mar 2008 21:56:12 GMT

Cache-Control: max-age=0, no-cache

Pragma: no-cache

Date: Wed, 19 Mar 2008 21:56:12 GMT

Connection: keep-alive

101

The HTTP status code indicates the success or failure of the operation. The only information transmitted in the HTTP response is the actual data itself: the newly assigned employee ID of 101.

© Copyright IBM Corporation 2014

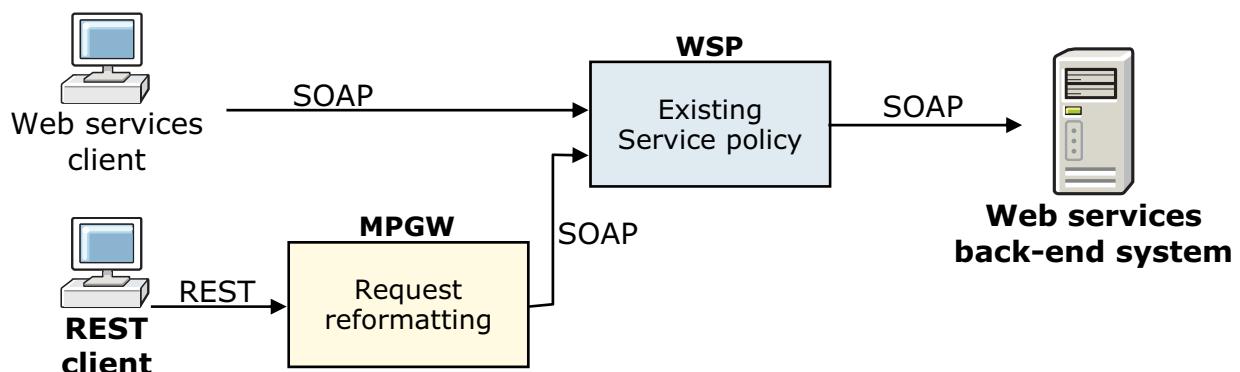
Figure 16-13. Example: Add employee REST response explained

WE601 / ZE6011.1

Notes:

Common DataPower REST patterns: Façade

- Provide a REST interface to an existing web service (web service proxy)
 - Current web service proxy supports a web services back-end system
 - Multi-protocol gateway exposes a REST interface to the client, but converts the REST request to a web services SOAP request



© Copyright IBM Corporation 2014

Figure 16-14. Common DataPower REST patterns: Façade

WE601 / ZE6011.1

Notes:

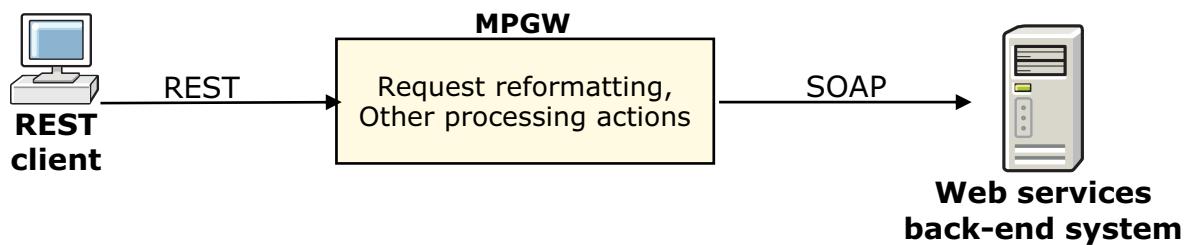
WSP: Web service proxy

MPGW: Multi-protocol gateway

The REST façade service also converts the SOAP response into a REST response.

Common DataPower REST patterns: Bridge

- Bridge between REST client and back-end web services
 - Convert REST request to the needed SOAP request format
 - Can also add other actions as needed (AAA, transforms, and so on)



© Copyright IBM Corporation 2014

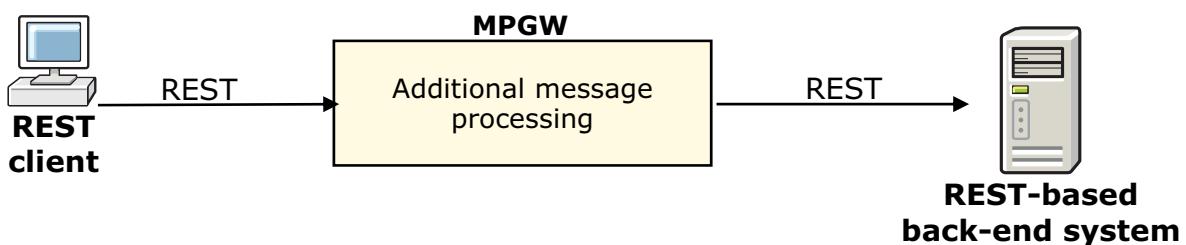
Figure 16-15. Common DataPower REST patterns: Bridge

WE601 / ZE6011.1

Notes:

Common DataPower REST patterns: REST enrichment

- DataPower service adds capabilities beyond what is in REST back-end system:
 - Schema validation
 - XML threat protection
 - Authentication, Authorization, Auditing
 - Service Level Management (SLM)
 - Content-based routing



© Copyright IBM Corporation 2014

Figure 16-16. Common DataPower REST patterns: REST enrichment

WE601 / ZE6011.1

Notes:

The back-end system already has a RESTful interface, but extra processing of the message is required.



Tooling to support REST: service or protocol handler related

- Front side handler support of HTTP method selection
- Request/response type of non-XML/JSON – MPGW/XMLFW
- Process Messages Whose Body Is Empty - MPGW/XMLFW Advanced tab
- JSON threat protection

© Copyright IBM Corporation 2014

Figure 16-17. Tooling to support REST: service or protocol handler related

WE601 / ZE6011.1

Notes:

The first discussion is on the DataPower tooling to support REST that is configured at the service level or the protocol handler level.

MPGW: Multi-protocol gateway service

XMLFW: XML firewall service



Front side handler support of HTTP method selection

- REST design prescribes specific HTTP methods
- The front side handler wizard allows selection of specific HTTP methods
- Note the methods selected by default do not include GET

HTTP Front Side Handler: RESTEastAddressSearch_FSH [up]

Apply Cancel Undo

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

© Copyright IBM Corporation 2014

Figure 16-18. Front side handler support of HTTP method selection

WE601 / ZE6011.1

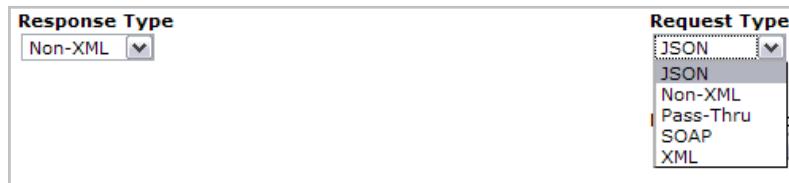
Notes:

The selection of allowed methods has been available for many years, irrespective of the REST model.



JSON request and response types

- REST-based bodies might contain XML or non-XML formatted data
- A popular structure for REST bodies is JSON
 - A data structure that is part of JavaScript
- Multi-protocol gateways and XML firewalls provide some automatic processing of JSON data if specified in the Request or Response Type



© Copyright IBM Corporation 2014

Figure 16-19. JSON request and response types

WE601 / ZE6011.1

Notes:

More information is presented on JSON in this unit.



Process bodyless messages

- REST-based messages might have bodyless requests and responses, but still require a service policy to run
- Multi-protocol gateway – **Advanced** tab
- XML firewall – **Advanced** tab



© Copyright IBM Corporation 2014

Figure 16-20. Process bodyless messages

WE601 / ZE6011.1

Notes:

This option controls whether to force the processing of XML messages when their message body is empty or missing in RESTful web services.

This option applies when the request or response type is XML, JSON, or Non-XML.



JSON threat protection

- Control some aspects of the dimensions of an input JSON message
 - Objects > JSON Processing > JSON Settings
- Referenced from the XML Manager for a service

JSON Settings

(none) + ...

Label-Value pairs		
Maximum label length	256	characters
Maximum value length for strings	8192	characters
Maximum value length for numbers	128	characters
Threat Protection		
Maximum nesting depth	64	levels
Maximum document size	4194304	bytes

© Copyright IBM Corporation 2014

Figure 16-21. JSON threat protection

WE601 / ZE6011.1

Notes:

These settings control the character length of the label, the string value, and the number value. It also sets the maximum nesting depth of elements, and the total message size.

Messages that violate these limits are rejected.

Tooling to support REST: service policy related

- Matching rule – HTTP method (GET, PUT, POST, DELETE, HEAD)
- Method rewrite action – change HTTP method
- Convert query parameters action
- var://service/protocol-method – read/write
- dp:http-request-method() extracts HTTP method from HTTP request
- Extension function to get and set HTTP status code
- JSON request type auto-converted to JSONx
- Convert query parameters action can convert JSON to JSONx
- Jsonx2json.xsl style sheet to transform JSONX to JSON
- Jsonx.xsd to validate JSONx
- Validate action supports JSON schemas
- Transform action adds XQuery/JSONiq processing
- Actions that allow http:// access: Fetch, Results, Results async, Log

© Copyright IBM Corporation 2014

Figure 16-22. Tooling to support REST: service policy related

WE601 / ZE6011.1

Notes:

MPGW: Multi-protocol gateway service

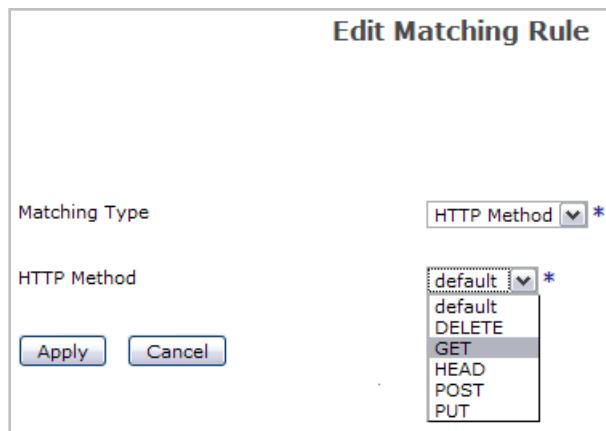
XMLFW: XML firewall service

Fetch, Results, Results Async, and Log actions can make http:// calls, which allows these actions to make REST-based calls.



Matching Rule on HTTP methods

- A processing rule can be set up for each expected HTTP method
- A Match action's Matching Rule can be configured for each of the HTTP methods



© Copyright IBM Corporation 2014

Figure 16-23. Matching Rule on HTTP methods

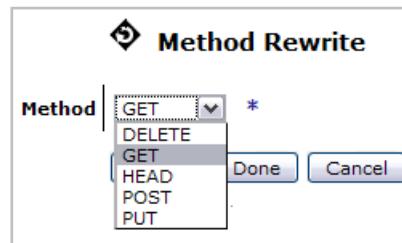
WE601 / ZE6011.1

Notes:

Default indicates either a GET or POST will successfully match.

Changing the HTTP method in the processing rule

- SOAP-based web services use the HTTP POST method, regardless of the nature of the request
- Depending on the nature of the request, a RESTful request uses the associated HTTP method
- When bridging between SOAP web services and REST web services, the HTTP method might need to be changed
- The Method Rewrite action (under Advanced action) lets you specify a specific HTTP method



© Copyright IBM Corporation 2014

Figure 16-24. Changing the HTTP method in the processing rule

WE601 / ZE6011.1

Notes:



Convert Query Params to XML action

- Converts non-XML (HTTP POST, HTML form, or URI parameters) into an equivalent XML message
 - For a service to use this action, the **request type** for that service must be set to **XML**

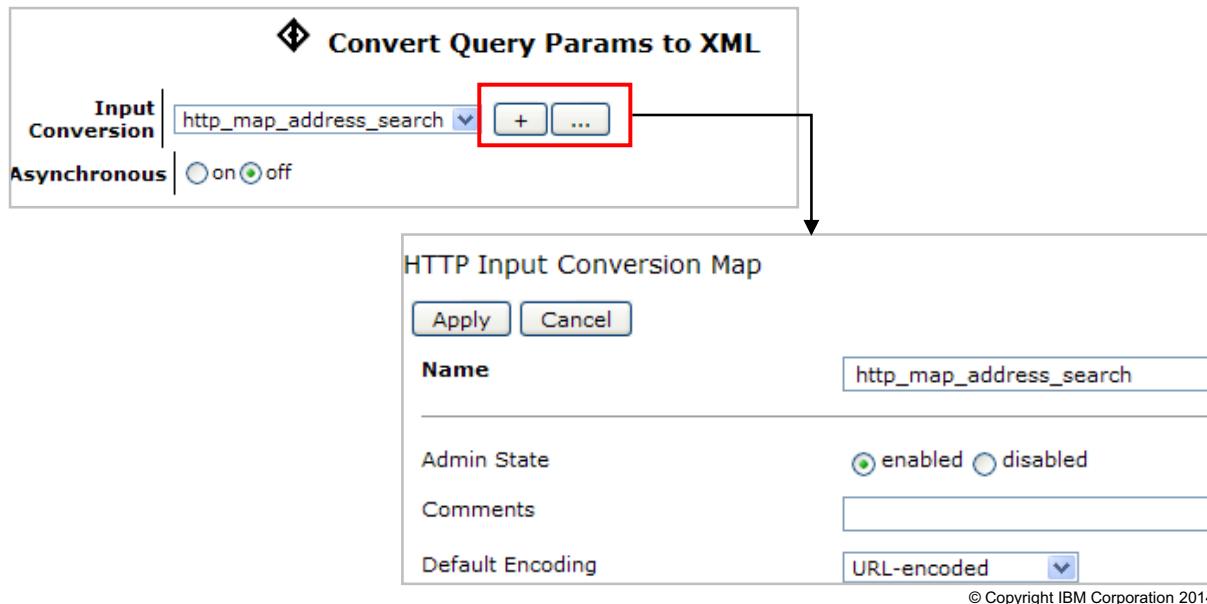


Figure 16-25. Convert Query Params to XML action

WE601 / ZE6011.1

Notes:

This action converts non-XML CGI-encoded input (an HTTP POST of HTML form or URI parameters) into an equivalent XML message.

The choices for the encoding conversions are: Plain, URL-encoded, XML, URL-encoded XML, Base 64, and JSON.

The JSON conversion is covered later.

Convert Query Params to XML action example

- Example cURL command

```
curl -G  
http://dphost:port/EastAddressSearch/people/  
?firstName=Victor&lastName=Collins&title=Mr
```

is converted to the following XML:

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

© Copyright IBM Corporation 2014

Figure 16-26. Convert Query Params to XML action example

WE601 / ZE6011.1

Notes:

The “-G” tells cURL to use an HTTP GET on the request.

Programmatic access to the HTTP method

- In a style sheet, you can retrieve the HTTP method from the request
 - The `dp:http-request-method()` extension function returns a string that contains the HTTP method in the first line of the request
- A service variable, `var://service/protocol-method`, can be used to read or write the HTTP method
 - Set Variable action to set the value
 - `dp:variable('var://service/protocol-method')` or `dp:set-variable('var://service/protocol-method')` in a style sheet

© Copyright IBM Corporation 2014

Figure 16-27. Programmatic access to the HTTP method

WE601 / ZE6011.1

Notes:

`dp:http-request-method()` is a metadata extension function.

Programmatic access to the HTTP status code

- In a style sheet, you can retrieve the HTTP status code from the request
 - The `dp:http-response-header('x-dp-response-code')` extension element returns a string that contains the HTTP status code
 - <xsl:variable name="responseCode" select="dp:http-response-header('x-dp-response-code')"/>
- You can also set it in a similar fashion
 - <dp:set-http-response-header name="'x-dp-response-code'" value="'000' />
 - <dp:set-http-response-header name="'x-dp-response-code'" value="'204' />

© Copyright IBM Corporation 2014

Figure 16-28. Programmatic access to the HTTP status code

WE601 / ZE6011.1

Notes:

Specific HTTP status codes are usually part of the RESTful interface definition.

JSON

- JavaScript Object Notation
- Subset of JavaScript
- Minimal
- Light-weight
- Text-based
- Language Independent
- Easy to parse
- Not a document format
- JSON is a simple, common representation of data that can be used for communication between servers and browser clients, communication between peers, and language independent data interchange.
- <http://json.org>

© Copyright IBM Corporation 2014

Figure 16-29. JSON

WE601 / ZE6011.1

Notes:

XML can be cumbersome in JavaScript to navigate so you move towards using JSON as a structured format. JSON, short for JavaScript Object Notation, is a lightweight computer data interchange format. It is a text-based, human-readable format for representing simple data structures, and associative arrays (called objects). The JSON format is often used for transmitting structured data over a network connection in a process called serialization. Its main application is in Ajax web application programming, where it serves as an alternative to the traditional use of the XML format. JSON is a simple, common representation of data that can be used for communication between servers and browser clients, communication between peers, and language independent data interchange.

JSON data types

"Hello world!\n"

A **string** is a sequence of zero or more Unicode characters.

-1.4719e7

A **number** includes an integer part that can be prefixed with a sign and followed by a fraction or an exponent.

{ "name": "John" }

An **object** is an unordered collection of zero or more name/value pairs.

["a", "b", "c"]

An **array** is an ordered sequence of zero or more values.

true

A **boolean** is a literal value of either **true** or **false**.

null

The keyword **null** represents a null value.

© Copyright IBM Corporation 2014

Figure 16-30. JSON data types

WE601 / ZE6011.1

Notes:

JSON values must be an object, array, number, or string, or one of the three literal names: false, true, null.

JSON version of XML

- XML

```
<Person>
  <details>
    <city>Cleveland</city>
    <state>OH</state>
    <street>3661 Lincoln
                  Ave</street>
    <zipCode>44111</zipCode>
  </details>
  <name>
    <firstName>Sarah</firstName>
    <lastName>Chan</lastName>
    <title>Mrs</title>
  </name>
</Person>
```

- JSON

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

© Copyright IBM Corporation 2014

Figure 16-31. JSON version of XML

WE601 / ZE6011.1

Notes:

On the left of the figure is an XML representation of a Person. On the right of the figure is the JSON equivalent.



JSONx

- Is an XML encoding of a JSON data structure
- Used by several IBM products
- Is an Internet Draft in the IETF
 - tools.ietf.org/html/draft-rsalz-jsonx-00
- The root element is a <json:object> or <json:array> element
- Child elements are elements that are related to the JSON types
 - <json:array>
 - <json:boolean>
 - <json:string>
 - <json:object>
 - <json:number>
 - <json:array>
 - <json:null>

© Copyright IBM Corporation 2014

Figure 16-32. JSONx

WE601 / ZE6011.1

Notes:

IETF: Internet Engineering Task Force

JSONx version of JSON data structure

- JSON

```
{
  "Person": {
    "details": {
      "city": "Cleveland",
      "state": "OH",
      "street": "36 Lincoln Ave",
      "zipcode": 44111
    },
    "name": {
      "firstname": "Sarah",
      "lastname": "Chan",
      "title": "Mrs"
    }
  }
}
```

- JSONx

```
<json:object name="Person">
  <json:object name="details">
    <json:string name="city">Cleveland</json:string>
    <json:string name="state">OH</json:string>
    <json:string name="street">
      36 Lincoln Ave</json:string>
    <json:number name="zipcode">44111</json:number>
  </json:object>
  <json:object name="name">
    <json:string name="firstName">Sarah</json:string>
    <json:string name="lastName">Chan</json:string>
    <json:string name="title">Mrs</json:string>
  </json:object>
</json:object>
```

© Copyright IBM Corporation 2014

Figure 16-33. JSONx version of JSON data structure

WE601 / ZE6011.1

Notes:

Person, details, and name are JSON objects.

City, state, street, firstname, lastname, and title are JSON strings.

Zipcode is a JSON number.

There are related JSONx elements to match the JSON data types.



Handling JSON in the request body

- If Request Type is set as **Non-XML**
 - Use a style sheet in the processing policy to manipulate as needed
 - Can use a Convert Query Parameters action with a JSON input encoding to convert the JSON to JSONx
- If Request Type is set as **JSON**
 - The JSON input is validated as well-formed JSON
 - The service automatically converts the JSON body to JSONx
 - The JSONx version of the body is placed in the `_JSONASJSONX` context, and is available in the processing policy for manipulation
 - The INPUT context is still valid, and contains the original JSON body

© Copyright IBM Corporation 2014

Figure 16-34. Handling JSON in the request body

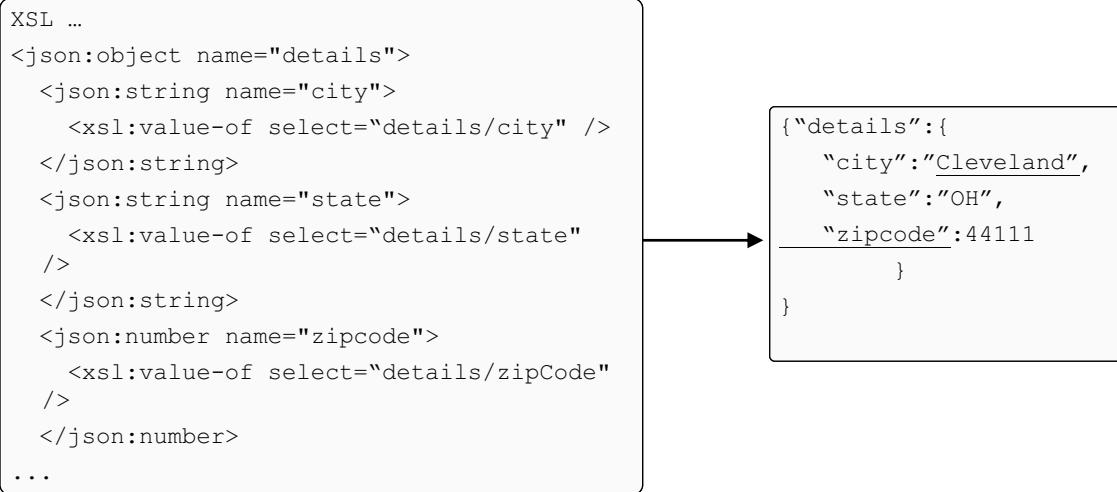
WE601 / ZE6011.1

Notes:

Whether you choose a request type of **Non-XML** or of **JSON** depends on the needs of your service policy.

Converting XML to JSON

- Use a style sheet to build a JSONx data structure
 - Parse the input XML structure to build the wanted JSONx representation



- Use the `store:///jsonx2json.xsl` to convert the JSONx to a JSON data structure
 - This technique performs the reverse of what the JSON request type and `_JSONASJSONX` context does

© Copyright IBM Corporation 2014

Figure 16-35. Converting XML to JSON

WE601 / ZE6011.1

Notes:

Validate action for JSON

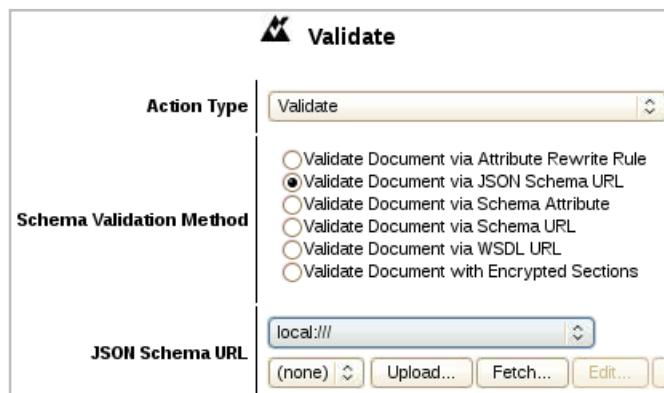
Perform schema-based validation of JSON documents:

- Validate Document by using JSON Schema URL
 - Specifies a schema URL of a JSON schema file

Validates the input document against the specified JSON schema

- Similar to validating an XML document against an XSD or WSDL
- Supports Draft 3 of the IETF specification:

<http://tools.ietf.org/html/draft-zyp-json-schema-03>



© Copyright IBM Corporation 2014

Figure 16-36. Validate action for JSON

WE601 / ZE6011.1

Notes:

The **Validate** action is also used to validate the schema of JSON structures. The JSON schema URL can reference either a local or remote file.

The expected file type for a JSON schema is JSV or JSON.

DataPower version 6.0.0 supports draft 3 of the IETF JSON schema specification. There are a few options in the specification that DataPower version 6.0.0 does not support. For more information, see the product documentation.

Example JSON and a JSON schema

- JSON

```
{
  "Person": {
    "details": {
      "city": "Cleveland",
      "state": "OH",
      "street": "36 Lincoln Ave",
      "zipcode": 44111
    },
    "name": {
      "firstname": "Sarah",
      "lastname": "Chan",
      "title": "Mrs"
    }
  }
}
```

- JSON schema

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "properties": {
    "Person": {
      "type": "object",
      "properties": {
        "details": {
          "type": "object",
          "properties": {
            "city": { "type": "string" },
            "state": { "type": "string" },
            "street": { "type": "string" },
            "zipcode": { "type": "number" }
          }
        },
        "name": {
          "type": "object",
          "properties": {
            "firstname": { "type": "string" },
            "lastname": { "type": "string" },
            "title": { "type": "string" }
          }
        }
      }
    }
  }
}

(and so on)
```

© Copyright IBM Corporation 2014

Figure 16-37. Example JSON and a JSON schema

WE601 / ZE6011.1

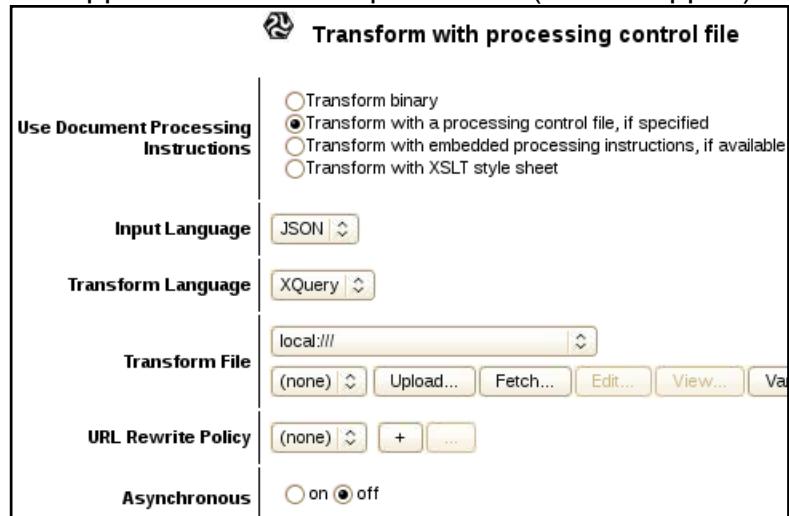
Notes:

The schema does not show constraints like required items, numeric ranges, and more.

Transform action that uses XQuery (JSON and XML)

Use XQuery expressions to manipulate JSON and XML documents

- Transform with a processing control file, if specified
 - Identifies the XQuery transform file that is referenced in the Transform File field
- XQuery is a query language for XML data (like SQL for relational data)
 - DataPower v6.0.0 adds the support for the JSONiq extension (JSON support)
- Input Language:
 - JSON
 - XML
 - XSD
- Transform Language:
 - XQuery



© Copyright IBM Corporation 2014

Figure 16-38. Transform action that uses XQuery (JSON and XML)

WE601 / ZE6011.1

Notes:

This option for the **Transform action** supports XQuery as the transformation language, rather than XSLT.

XQuery is a language that is designed to query XML data, much like SQL is used to query relational data. DataPower v6.0.0 includes the JSONiq extension to XQuery. This extension adds support for JSON to XQuery.

DataPower version 6.0.0 supports XQuery 1.0, and its related specifications. The JSONiq extension support is for 0.4.42.

The **Input Language** indicates whether the input document is JSON or XML. The third option of XSD indicates that the input document is XML, but it also displays another entry field that accepts an XML schema file location. This schema is used to type the data (integer, number, text, for example) for the XQuery processing, but it does **not** perform any validation against the schema. To do that, you must use a Validate action.

The **Transform Language** indicates the language of the transformation file. The only valid option currently is XQuery.

The **URL Rewrite Policy** rewrites external references that are contained within the input document.

XQuery/JSONiq example

- JSON input message

```
[{"firstname": "John", "lastname": "Smith", "order": "20223", "price": 23.95},
 {"firstname": "Alice", "lastname": "Brown", "order": "54321", "price": 199.95},
 {"firstname": "John", "lastname": "Smith", "order": "23420", "price": 104.95},
 {"firstname": "Bob", "lastname": "Green", "order": "90231", "price": 300.00},
 {"firstname": "Scott", "lastname": "Jones", "order": "54321", "price": 199.95},
 {"firstname": "Jim", "lastname": "Lee", "order": "89820", "price": 46.50}]
```

- From the array, return the name of any customers who have an order value of \$100 or more, ordered by last name

```
declare option jsoniq-version "0.4.42";
for $x in jn:members(..)
  where $x("price") >= 100.00
  order by $x("lastname")
  return concat($x("firstname"), ' ', $x("lastname"),
  '
')
```

Output message

Alice Brown
Bob Green
Scott Jones
John Smith

© Copyright IBM Corporation 2014

Figure 16-39. XQuery/JSONiq example

WE601 / ZE6011.1

Notes:

Regarding JSON, Xquery/JSONiq can be used for such operations as:

- Transforming JSON objects into a text report
- Converting JSON objects to XML elements
- Converting XML elements to JSON objects
- Transforming a JSON object into a new JSON object

'
' is an encoded newline character.

Sample service policy: bridging REST and SOAP

REST_GET_REQ	Client to Server								
REST_GET_RESP	Server to Client								

- In the request:
 - Match on HTTP method = GET
 - Convert URI parameters to an XML structure
 - Build the SOAP request
 - Set the HTTP method to POST
 - Set the back-end URL
 - Set the back-end URI
- In the response:
 - Match on all URLs
 - Transform the SOAP response to a JSONx structure
 - Transform the JSONx to JSON by using `jsonx2json.xsl`

© Copyright IBM Corporation 2014

Figure 16-40. Sample service policy: bridging REST and SOAP

WE601 / ZE6011.1

Notes:

The request rule converts the REST request into a SOAP request for the web services back-end system.

The response rule converts the SOAP response into a REST response that includes a JSON structure.



Unit summary

Having completed this unit, you should be able to:

- Use a DataPower appliance to proxy back-end applications for mobile clients
- Describe the purpose of a REST architecture
- Add support to DataPower services for the REST application programming interface (API)
- Describe how to integrate with systems by using RESTful services
- Use the DataPower appliance to proxy a RESTful service

© Copyright IBM Corporation 2014

Figure 16-41. Unit summary

WE601 / ZE6011.1

Notes:



Checkpoint questions

1. True or False: Mobile clients differ from desktop clients in only the capabilities of their web browsers.
2. True or False: REST is an OASIS standard.
3. List three advantages of using a REST API.
4. List the non-XML types supported by the Convert Query Params to XML action.
5. In a service policy, you can change the HTTP method by using the action:
 - a) Convert query parameters
 - b) Header rewrite
 - c) Method rewrite

© Copyright IBM Corporation 2014

Figure 16-42. Checkpoint questions

WE601 / ZE6011.1

Notes:

Write your answers here:

1.

2.



Checkpoint answers

1. **False:** Mobile clients can have mobile applications that are written for the device, and do not use a browser interface.
2. **False.** REST is an architectural style.
3. Three advantages of using a REST API are:
 - a. Easy to secure.
 - b. Easy to navigate.
 - c. Simplicity.
4. The three types of non-XML data supported by the Convert Query Params to XML action are HTTP POST, HTML form, or URI parameters
5. **C.** Method rewrite

© Copyright IBM Corporation 2014

Figure 16-43. Checkpoint answers

WE601 / ZE6011.1

Notes:



Exercise 13



Using DataPower to implement
REST services

© Copyright IBM Corporation 2014

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

9.0

Figure 16-44. Exercise 13

WE601 / ZE6011.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2014

Figure 16-45. Exercise objectives

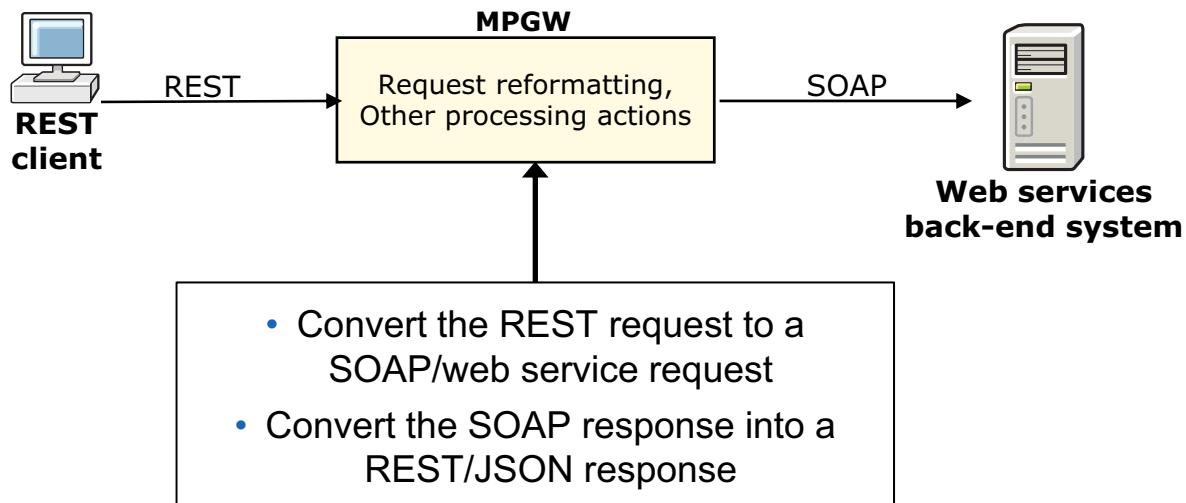
WE601 / ZE6011.1

Notes:

Exercise overview

A cURL request represents the REST client

East Address web service running on the application server



© Copyright IBM Corporation 2014

Figure 16-46. Exercise overview

WE601 / ZE6011.1

Notes:

Unit 17. Course summary

What this unit is about

This unit summarizes the course and provides information for future study.

What you should be able to do

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study



Unit objectives

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

© Copyright IBM Corporation 2014

Figure 17-1. Unit objectives

WE601 / ZE6011.1

Notes:

Course learning objectives

Having completed this course, you should be able to:

- Describe how WebSphere DataPower SOA Appliances are configured
- Create a web service proxy to virtualize web service applications
- Implement web services security
- Create and configure cryptographic objects
- Configure Secure Sockets Layer (SSL) to and from WebSphere DataPower SOA Appliances
- Configure a multi-protocol gateway (MPGW) to handle multiple protocols for a single service
- Configure a service level monitoring (SLM) policy to control message traffic
- Configure support for IBM WebSphere MQ and WebSphere Java Message Service (JMS)
- Use logs and probes to troubleshoot services
- Configure the DataPower resources that are needed to support OAuth 2.0
- Use patterns to define and deploy new services
- Use DataPower resources and options to support REST and JSON-based services
- Handle errors in service policies

© Copyright IBM Corporation 2014

Figure 17-2. Course learning objectives

WE601 / ZE6011.1

Notes:

Course review (1 of 3)

- The XG45/XI52/XI50 blade IBM WebSphere DataPower SOA Appliances secure XML and web service applications through **three main services**:
 - XML firewall
 - Web service proxy
 - Multi-protocol gateway
- The **XML firewall service** secures XML-based applications against XML threats
 - The XML firewall supports message-level security processing actions and content based routing
- Clients can connect to the back-end service through the **multi-protocol gateway**, over a number of different transport and application protocols
 - Protocol handlers are available for HTTP and HTTPS protocols, FTP, raw XML messages, TIBCO EMS, and IBM WebSphere MQ systems
 - Includes all capabilities of XML firewall type

© Copyright IBM Corporation 2014

Figure 17-3. Course review (1 of 3)

WE601 / ZE6011.1

Notes:

Course review (2 of 3)

- The **Web Service Proxy** provides features tailored to web service applications
 - Service-level monitoring and processing policies can be applied at the service definition, interface (portType), and operation level
 - Web service virtualization acts as a single endpoint for a group of operations that different back-end services implement
- Services use the **cryptographic** tools to configure SSL communication, XML encryption, and digital signatures
- The **Encrypt**, **Decrypt**, **Verify**, and **Sign** processing actions provide XML encryption and digital signatures down to the message field level
- The **Authentication**, **Authorization**, and **Auditing** processing action restricts access to resources and monitors XML application traffic
 - A wide range of message-level security specifications are supported, including WS-Security, SAML, XACML, SPNEGO, LTPA, OAuth, Kerberos, and others

© Copyright IBM Corporation 2014

Figure 17-4. Course review (2 of 3)

WE601 / ZE6011.1

Notes:

Course review (3 of 3)

- **WebSphere MQ** is a protocol that is commonly used for DataPower front-end or back-end transport
 - WebSphere MQ header manipulation and transactionality are supported
- DataPower **JMS** support works with WebSphere JMS and TIBCO EMS
- DataPower participates in the **OAuth** framework as a resource server or authorization server. The DataPower OAuth configuration options are:
 - Web token service
 - AAA action options
 - OAuth client profile
 - OAuth client group
- DataPower **Patterns** streamline the creation of services
- **Message monitors** and **service-level monitors** can filter, shape, and throttle traffic through a DataPower SOA appliance

© Copyright IBM Corporation 2014

Figure 17-5. Course review (3 of 3)

WE601 / ZE6011.1

Notes:



Lab exercise solutions

- Solutions are available in the `Solution` subdirectory:

`<lab_files>/Solution`

- Remember to change
 - Port numbers
 - Back-end server (**Network > Interface > DNS Settings > Static Hosts**)
 - Front IP addresses (**Network > Interface > Host Alias**)

© Copyright IBM Corporation 2014

Figure 17-6. Lab exercise solutions

WE601 / ZE6011.1

Notes:



To learn more on the subject

- IBM Training website:
www.ibm.com/training
- DataPower Information Center
 - <http://pic.dhe.ibm.com/infocenter/wsdatap/v6r0m0/index.jsp>
- IBM Redbooks:
 - <http://www.redbooks.ibm.com>
 - Search on “DataPower”
- developerWorks articles:
 - <http://www.ibm.com/developerworks/>
 - Search on “DataPower”
- DataPower SOA Appliance Customer Forum:
 - <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1198>

© Copyright IBM Corporation 2014

Figure 17-7. To learn more on the subject

WE601 / ZE6011.1

Notes:

More DataPower education opportunities

Developer: Advanced

- Available as individual SPVCs
- Covers topics such as:
 - DataPower variables, extensions, and flow control
 - Access control and custom security policies
 - FTP and DataPower
 - SQL and DataPower
 - Enforcing governance
 - And more

System administrator

- Available as a standard course
- Covers topics such as:
 - Initial setup
 - Managing firmware
 - CLI and XML management tools
 - Clustering and failover
 - Troubleshooting
 - Logging
 - And more

Figure 17-8. More DataPower education opportunities

© Copyright IBM Corporation 2014

WE601 / ZE6011.1

Notes:

Always check the IBM website for current education offerings.

An **SPVC** is a self-paced virtual class. It offers the standard classroom material in both visual and audio form. Depending on the course, it might also include hands-on exercises or demonstrations.



Unit summary

Having completed this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

© Copyright IBM Corporation 2014

Figure 17-9. Unit summary

WE601 / ZE6011.1

Notes:

Appendix A. List of abbreviations

A

AAA	Authentication, authorization, and auditing
ACL	Access control list
AES	Advanced Encryption Standard
AMP	Appliance Management Protocol
API	Application programming interface

C

CA	Certificate authority
CBR	Content based routing
CGI	CGI Common Gateway Interface
CHTML	Compact HTML
CLI	Command-line interface
CR	Carriage return
CRL	Certificate revocation list
CSR	Certificate signing request
CSS	Cascading style sheet

D

DAP	Directory Access Protocol
DB	Database
DES	Data Encryption Standard
DH	Diffie-Hellman
DIME	Direct Internet Message Encapsulation
DIT	Directory information tree
DN	Distinguished name
DNS	Dynamic Name Server
DOM	Document Object Model
DOP	Data-oriented programming
DoS	Denial-of-service
DSS	Digital Signature Standard
DTD	Document type definition

E

EDIINT	Electronic Data Interchange-Internet Integration
EMS	Enterprise Messaging System
EON	Edge of Network
ESB	Enterprise service bus
EULA	End-user license agreement
EXSLT	Extensions to Extensible Stylesheet Language Transformation

F

FIFO	First-in first-out
FIPS	Federal Information Processing Standard
FIX	Financial Information Exchange
FO	Formatting object
FTP	File Transfer Protocol

G

GB	Gigabyte
GSS	Generic Security Services
GUI	Graphical user interface

H

HR	Human resources
HREF	Hypertext reference
HSM	Hardware Security Module
HSRP	Hot Standby Router Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL

I

ICAP	Internet Content Adaptation Protocol
ICMP	Internet Control Message Protocol
IDE	integrated development environment
IDEA	International Data Encryption Algorithm
IEEE	Institute of Electrical and Electronics Engineers

IETF	Internet Engineering Task Force
ILD	Intelligent load distribution
IMS	Information Management System
IP	Internet Protocol
IPSec	IP Security
iSCSI	Internet Small Computer Systems Interface

J

JAXP	Java API for XML Processing
JDBC	Java Database Connectivity
JKS	Java Key Store
JMS	Java Message Service
JRE	Java runtime environment
JSON	JavaScript Object Notation

K

KB	Kilobyte
-----------	----------

L

LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LED	Light-emitting diode
LLM	Low Latency Messaging

M

MAC	Message authentication code
MB	Megabyte
MFA	Message filter action
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extensions
MM	Message monitor
MMXDoS	Multiple message XML denial-of-service
MPG	Multi-protocol gateway
MQ	Message Queue
MT	Message type

MTOM Message Transmission Optimization Mechanism

N

NAT Network address translation

NFS Network File System

NG New Generation

NIC Network interface card

NSTISSC National Security Telecommunications and Information Systems Security Committee

NTP Network Time Protocol

O

OASIS Organization for the Advancement of Structured Information Standards

OAuth Open standard for Authorization

OID Object ID

OSI Open Systems Interconnection

OTMA Open Transaction Management Access

P

PCF Processing Control File

PCRE Perl-compatible regular expressions

PDF Portable Document Format

PDP Policy decision point

PED PIN Entry Device

PEP Policy enforcement point

PI Processing instruction

PIN Personal identification number

PKCS Public Key Cryptography Standard

PKI public key infrastructure

PKIX Public Key Infrastructure for X.509 Certificates (IETF)

PMR Program maintenance request

Q

QoS Quality of service

R

RAID	Redundant Array of Independent Disks
RAM	Random access memory
RBM	Role-based management
RDBMS	Relational database management system
RDN	Relative distinguished name
REL	Rights Expression Language
REST	Representational state transfer
RFC	Request for Comments
RSA	Rational Software Architect

S

SAML	Security Assertion Markup Language
SAS	Serial Attached SCSI
SAX	Simple API for XML
SCP	Secure Copy Protocol
SCSI	Small Computer System Interface
SFTP	Secured File Transfer Protocol
SHA1	Secure Hash Algorithm, Version 1
SLA	Service level agreement
SLM	Service level management
SLM	Service level monitoring
SMS	Session management server
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Service-oriented architecture
SOAP	Usage note: SOAP is not an acronym; it is a word in itself (formerly an acronym for Simple Object Access Protocol)
SPNEGO	Simple and Protected GSS-API Negotiation Mechanism
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
SSO	Single sign-on
SUSE	A Linux based operating system

SwA SOAP with Attachments

T

Tcl Tool Control Language (often pronounced as “tickle”)

TCO Total cost of ownership

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TDES Triple Data Encryption Standard

TFIM Tivoli Federated Identity Manager

TIA Telecommunications Industry Association

TIM Tivoli Identity Manager

TLS Transport Layer Security

TTL Time to Live

U

UDDI Universal Description, Discovery, and Integration

UDP User Datagram Protocol

UNIX Uniplexed Information and Computing System

URI Uniform Resource Identifier

URL Uniform Resource Locator

USB Universal Serial Bus

UTC Coordinated Universal Time

V

VIP Virtual IP address

VM Virtual machine

VLAN Virtual local area network

VRPP Virtual Router Redundancy Protocol

W

W3C World Wide Web Consortium

WAF Web application firewall

WML Wireless Markup Language

WS Web services

WSDL Web Services Description Language

WSDM	Web Services Distributed Management
WSRR	WebSphere Service Registry and Repository
WWW	World Wide Web

X

XCF	Cross-system coupling facility
XDoS	XML denial of service
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XMLDS	XML digital signature
XML-PI	XML processing instructions
XPath	XML Path Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

Y**Z**

z/OS	zSeries operating system
-------------	--------------------------

Appendix B. Resource guide

Completing this WebSphere Education course is a great first step in building your WebSphere, CICS, and SOA skills. Beyond this course, IBM offers several resources to keep your WebSphere skills on the cutting edge. Resources available to you range from product documentation to support websites and social media websites.

Training

- **IBM Training website**
 - Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.
 - For more information, see: <http://www.ibm.com/training>
- **IBM Training News**
 - Review or subscribe to updates from IBM and its training partners.
 - For more information, see: <http://bit.ly/IBMTtrainEN>
- **IBM Certification**
 - You can demonstrate to your employer or clients your new WebSphere, CICS, or SOA mastery through achieving IBM Professional Certification. WebSphere certifications are available for developers, administrators, and business analysts.
 - For more information, see: <http://www.ibm.com/certify>
- **Training paths**
 - Find your next course easily with IBM training paths. Training paths provide a visual flow-chart style representation of training for many WebSphere products and roles, including developers and administrators.
 - For more information, see: <http://www.ibm.com/services/learning/sites.wss/us/en?pageType=page&c=a0003096>

Social media links

You can keep in sync with WebSphere Education, including new courses and certifications, course previews, and special offers, by visiting any of the following social media websites.

- **Twitter**
 - Receive short and concise updates from WebSphere Education a few times each week.
 - Follow WebSphere Education at: twitter.com/websphere_edu
- **Facebook:**

- Become a fan of IBM Training on Facebook to keep in sync with the latest news and career trends, and to post questions or comments.
- Find IBM Training at: facebook.com/ibmtraining

- **YouTube:**

- Visit the IBM Training YouTube channel to learn about IBM training programs and courses.
- Find IBM Training at: youtube.com/IBMTutorial

Support

- **WebSphere Support portal**

- The WebSphere Support website provides access to a portfolio of support tools. From the WebSphere Support website, you can access several downloads, including troubleshooting utilities, product updates, drivers, and Authorized Program Analysis Reports (APARs). To collaboratively solve issues, the support website is a clearing house of links to online WebSphere communities and forums. The IBM support website is now customizable so you can add and delete portlets to the information most important to the WebSphere products you work with.

- For more information, see: <http://www.ibm.com/software/websphere/support>

- **IBM Support Assistant**

- The IBM Support Assistant is a local serviceability workbench that makes it easier and faster for you to resolve software product issues. It includes a desktop search component that searches multiple IBM and non-IBM locations concurrently and returns the results in a single window, all within IBM Support Assistant.
- IBM Support Assistant includes a built-in capability to submit service requests; it automatically collects key problem information and transmits it directly to your IBM support representative.

- For more information, see: <http://www.ibm.com/software/support/isa>

- **WebSphere Education Assistant**

- IBM Education Assistant is a collection of multimedia modules that are designed to help you gain a basic understanding of IBM software products and use them more effectively. The presentations, demonstrations, and tutorials that are part of the IBM Education Assistant are an ideal refresher for what you learned in your WebSphere Education course.
- For more information, see:
<http://www.ibm.com/software/info/education/assistant/>

WebSphere documentation and tips

- **IBM Redbooks**

- The IBM International Technical Support Organization develops and publishes IBM Redbooks publications. IBM Redbooks are downloadable PDF files that describe installation and implementation experiences, typical solution scenarios, and step-by-step “how-to” guidelines for many WebSphere products. Often, Redbooks include sample code and other support materials available as downloads from the site.
 - For more information, see: <http://www.ibm.com/redbooks>
- **IBM documentation and libraries**
 - Information centers and product libraries provide an online interface for finding technical information on a particular product, offering, or product solution. The information centers and libraries include various types of documentation, including white papers, podcasts, webcasts, release notes, evaluation guides, and other resources to help you plan, install, configure, use, tune, monitor, troubleshoot, and maintain WebSphere products. The WebSphere information center and library are located conveniently in the left navigation on WebSphere product web pages.
- **developerWorks**
 - IBM developerWorks is the web-based professional network and technical resource for millions of developers, IT professionals, and students worldwide. IBM developerWorks provides an extensive, easy-to-search technical library to help you get up to speed on the most critical technologies that affect your profession. Among its many resources, developerWorks includes how-to articles, tutorials, skill kits, trial code, demonstrations, and podcasts. In addition to the WebSphere zone, developerWorks also includes content areas for Java, SOA, web services, and XML.
 - For more information, see: <http://www.ibm.com/developerworks>

WebSphere Services

- IBM Software Services for WebSphere are a team of highly skilled consultants with broad architectural knowledge, deep technical skills, expertise on suggested practices, and close ties with IBM research and development labs. The WebSphere Services team offers skills transfer, implementation, migration, architecture, and design services, plus customized workshops. Through a worldwide network of services specialists, IBM Software Service for WebSphere makes it easy for you to design, build, test, and deploy solutions, helping you to become an on-demand business.
- For more information, see:
<http://www.ibm.com/developerworks/websphere/services/>

IBM
®