

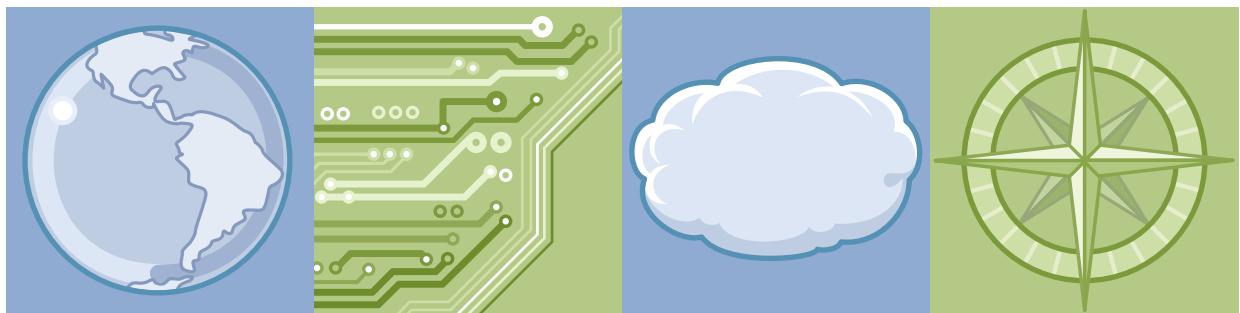


# IBM Training

Student Notebook

**Accelerate, Secure, and Integrate with IBM DataPower V7.1**

Course code WE711 / ZE711 ERC 1.0



WebSphere Education

## 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 in the United States, or other countries, or both:

|            |                 |              |
|------------|-----------------|--------------|
| Approach®  | Bluemix™        | CICS®        |
| DataPower® | DB™             | DB2 Connect™ |
| DB2®       | developerWorks® | Express®     |
| IMS™       | Notes®          | RACF®        |
| Rational®  | Redbooks®       | Tivoli®      |
| WebSphere® | Worklight®      | z/OS®        |
| 400®       |                 |              |

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.

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

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

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.

### April 2015 edition

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

# Contents

|   |              |
|---|--------------|
| <b>Trademarks .....</b>   | <b>xvii</b>  |
| <b>Course description .....</b>                                       | <b>xxi</b>   |
| <b>Agenda .....</b>   | <b>xxiii</b> |
| <b>Unit 1. Quick introduction to developing on DataPower .....</b>    | <b>1-1</b>   |
| Unit objectives .....   | 1-2          |
| DataPower development is done on an appliance .....                   | 1-3          |
| Logging in to the WebGUI .....  | 1-4          |
| Login and development access to the appliance .....                   | 1-5          |
| WebGUI home page .....  | 1-6          |
| WebGUI banner .....   | 1-7          |
| WebGUI navigation bar .....   | 1-8          |
| WebGUI Control Panel: Links to common functions .....                 | 1-9          |
| Navigation bar categories .....                                       | 1-10         |
| Catalog for a multi-protocol gateway (MPGW) .....                     | 1-11         |
| Example service configuration page .....                              | 1-12         |
| The system log .....  | 1-13         |
| Saving configuration changes .....                                    | 1-14         |
| Configuration Checkpoints .....                                       | 1-15         |
| File management .....   | 1-16         |
| File directories for configuration .....                              | 1-17         |
| File directories for security .....                                   | 1-18         |
| File directories for logging .....                                    | 1-19         |
| More file directories .....   | 1-20         |
| Export service and object configurations .....                        | 1-21         |
| Import a configuration .....  | 1-22         |
| System control features in an application domain .....                | 1-23         |
| Globalization: Displaying other languages in WebGUI and the log ..... | 1-24         |
| Enabling languages .....  | 1-25         |
| Getting the WebGUI to display an alternative language .....           | 1-26         |
| Getting an alternative language for the log and messages .....        | 1-27         |
| Unit summary .....  | 1-28         |
| Checkpoint questions .....  | 1-29         |
| Checkpoint answers .....  | 1-30         |
| Exercise 1 .....  | 1-31         |
| Exercise objectives .....   | 1-32         |
| <b>Unit 2. Services overview .....</b>                                | <b>2-1</b>   |
| 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          |

|  |      |
|--|------|
| B2B gateway service .....                | 2-10 |
| Access manager reverse proxy .....       | 2-11 |
| Web application firewall service .....   | 2-12 |
| Other services .....                     | 2-13 |
| Which service type should you use? ..... | 2-15 |
| Unit summary .....                       | 2-16 |
| Checkpoint questions .....               | 2-17 |
| Checkpoint answers .....                 | 2-18 |

**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  |
| Message processing phases .....                        | 3-6  |
| Client-side (Front) Processing Phase .....             | 3-8  |
| Front side access .....                                | 3-9  |
| Service Processing Phase .....                         | 3-10 |
| Service Processing Phase .....                         | 3-11 |
| Match Action .....                                     | 3-12 |
| Server-side (Back) Processing Phase .....              | 3-13 |
| Connection to the back side .....                      | 3-14 |
| DataPower Configuration Architecture .....             | 3-15 |
| Configuring a service policy (processing policy) ..... | 3-16 |
| Policy editor .....                                    | 3-17 |
| Configuring rules within a service policy .....        | 3-18 |
| Processing rules .....                                 | 3-19 |
| Match action .....                                     | 3-20 |
| Processing actions .....                               | 3-21 |
| Processing actions .....                               | 3-22 |
| More processing actions .....                          | 3-24 |
| Multi-step processing rules .....                      | 3-25 |
| Predefined context variables .....                     | 3-26 |
| Validate action for XML .....                          | 3-27 |
| Validate action for JSON .....                         | 3-28 |
| Transform action for XML .....                         | 3-29 |
| Transform action that uses XQuery (JSON and XML) ..... | 3-30 |
| Transform action for binary transformations .....      | 3-31 |
| Filter action .....                                    | 3-32 |
| Filter action: Replay attack .....                     | 3-33 |
| GatewayScript action (1 of 2) .....                    | 3-34 |
| GatewayScript action (2 of 2) .....                    | 3-35 |
| Content-based routing .....                            | 3-36 |
| Route action configuration .....                       | 3-37 |
| Style sheet programming with dynamic routing .....     | 3-38 |
| Results action .....                                   | 3-40 |
| Results asynchronous and multi-way results mode .....  | 3-41 |
| Service settings .....                                 | 3-43 |
| Service types .....                                    | 3-44 |
| URL rewriting .....                                    | 3-45 |
| XML Manager .....                                      | 3-47 |

|   |            |
|---|------------|
| Default XML Manager configuration . . . . .                   | 3-48       |
| XML parser limits . . . . .                                   | 3-49       |
| JSON document limits within the XML manager . . . . .         | 3-51       |
| Exporting a service configuration . . . . .                   | 3-53       |
| Troubleshooting a service configuration . . . . .             | 3-54       |
| Unit summary . . . . .  | 3-55       |
| Checkpoint questions . . . . .                                | 3-56       |
| Checkpoint answers . . . . .                                  | 3-57       |
| Exercise 2 . . . . .  | 3-58       |
| Exercise objectives . . . . .                                 | 3-59       |
| Exercise overview . . . . .                                   | 3-60       |
| <b>Unit 4. Multi-protocol gateway service . . . . .</b>       | <b>4-1</b> |
| Unit objectives . . . . .                                     | 4-2        |
| What is a multi-protocol gateway? . . . . .                   | 4-3        |
| Conceptual architecture of a multi-protocol gateway . . . . . | 4-4        |
| Protocol handlers at a glance (1 of 2) . . . . .              | 4-5        |
| Protocol handlers at a glance (2 of 2) . . . . .              | 4-6        |
| Front side protocol handlers . . . . .                        | 4-7        |
| Static back-end gateway . . . . .                             | 4-8        |
| Dynamic back-end gateway . . . . .                            | 4-9        |
| Multi-protocol gateway and XML firewall compared . . . . .    | 4-10       |
| Multi-protocol gateway editor . . . . .                       | 4-11       |
| Scenario 1: Provide HTTP and HTTPS access . . . . .           | 4-13       |
| Step 1: Configure the back side transport . . . . .           | 4-14       |
| Step 2: Create a document processing rule . . . . .           | 4-15       |
| Step 3: Create the front side handlers . . . . .              | 4-16       |
| Step 4: Configure the front side handler . . . . .            | 4-17       |
| Step 5: Configure the SSL proxy profile . . . . .             | 4-18       |
| Scenario 2: Dynamic back-end service . . . . .                | 4-19       |
| Step 1: Configure the back-end transport . . . . .            | 4-20       |
| Sample service that targets a style sheet . . . . .           | 4-21       |
| Scenario 3: Provide MQ access . . . . .                       | 4-22       |
| Scenario 4: Provide JMS access . . . . .                      | 4-23       |
| Scenario 5: Provide IMS Connect access . . . . .              | 4-24       |
| Scenario 6: Provide a RESTful interface . . . . .             | 4-25       |
| REST support in DataPower . . . . .                           | 4-26       |
| Supported IMS features . . . . .                              | 4-27       |
| IMS Connect support . . . . .                                 | 4-28       |
| WebSocket Proxy (1 of 2) . . . . .                            | 4-29       |
| WebSocket Proxy (2 of 2) . . . . .                            | 4-30       |
| Comparing services . . . . .                                  | 4-31       |
| Unit summary . . . . .  | 4-32       |
| Checkpoint questions . . . . .                                | 4-33       |
| Checkpoint answers . . . . .                                  | 4-34       |
| <b>Unit 5. Problem determination tools . . . . .</b>          | <b>5-1</b> |
| 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       |
| Debugging GatewayScript (1 of 4) .....                  | 5-21       |
| Debugging GatewayScript (2 of 4) .....                  | 5-22       |
| Debugging GatewayScript (3 of 4) .....                  | 5-24       |
| Debugging GatewayScript (4 of 4) .....                  | 5-25       |
| Problem determination with cURL .....                   | 5-26       |
| Communicating with DataPower support .....              | 5-27       |
| Logging basics .....                                    | 5-28       |
| Log targets .....                                       | 5-29       |
| Available log levels .....                              | 5-30       |
| Log target configuration .....                          | 5-31       |
| Log target types .....                                  | 5-32       |
| Event filters .....                                     | 5-33       |
| Object filters .....                                    | 5-34       |
| Event subscriptions .....                               | 5-35       |
| Log action .....  | 5-36       |
| Unit summary .....                                      | 5-37       |
| Checkpoint questions .....                              | 5-38       |
| Checkpoint answers .....                                | 5-39       |
| Exercise 3 .....  | 5-40       |
| Exercise objectives .....                               | 5-41       |
| Exercise overview .....                                 | 5-42       |
| <b>Unit 6. Handling errors in a service policy.....</b> | <b>6-1</b> |
| Unit objectives .....                                   | 6-2        |
| Error handling constructs .....                         | 6-3        |
| Error handling options .....                            | 6-4        |
| Configure an On Error action .....                      | 6-5        |
| Creating an error rule .....                            | 6-6        |
| Configure Transform action in error rule .....          | 6-7        |
| Style sheet programming that use error variables .....  | 6-8        |
| Example custom error style sheet .....                  | 6-9        |
| Error rule versus On Error action .....                 | 6-10       |
| Error Policy .....                                      | 6-11       |
| Typical Error Policy use cases .....                    | 6-13       |
| How the Error Policy works (1 of 3) .....               | 6-14       |
| How the Error Policy works (2 of 3) .....               | 6-15       |
| How the Error Policy works (3 of 3) .....               | 6-17       |

|   |            |
|---|------------|
| Error Policy configuration .....  | 6-18       |
| Error Policy configuration: Multi-Protocol Gateway .....                | 6-19       |
| More Error Policy Processing (1 of 2) .....                             | 6-21       |
| More Error Policy Processing (2 of 2) .....                             | 6-22       |
| Unit summary .....  | 6-23       |
| Checkpoint questions .....  | 6-24       |
| Checkpoint answers .....  | 6-25       |
| Exercise 4 .....  | 6-26       |
| Exercise objectives .....   | 6-27       |
| <b>Unit 7. DataPower cryptographic tools and SSL setup .....</b>        | <b>7-1</b> |
| 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        |
| SSL - crypto object relationships .....                                 | 7-7        |
| Key and certificate objects point to files .....                        | 7-8        |
| Crypto shared secret (symmetric) key .....                              | 7-9        |
| Crypto (asymmetric) key .....   | 7-10       |
| Crypto certificate .....  | 7-11       |
| Crypto identification credential .....                                  | 7-12       |
| Crypto validation credential .....                                      | 7-13       |
| Import and export crypto objects .....                                  | 7-15       |
| Uploading keys .....  | 7-16       |
| Certificates can expire or get revoked .....                            | 7-17       |
| Certificate revocation list (CRL) retrieval .....                       | 7-18       |
| Crypto certificate monitor .....  | 7-19       |
| Hardware Security Module (HSM) .....                                    | 7-20       |
| DataPower support for SSL .....   | 7-21       |
| A crypto profile specifies details of the SSL connection .....          | 7-22       |
| Crypto profile .....  | 7-23       |
| Do not use insecure SSL options in the 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       |
| Recap: SSL - crypto object relationships .....                          | 7-36       |
| User agent .....  | 7-37       |
| What is a “user agent”? .....   | 7-38       |
| Configuring a user agent .....  | 7-39       |
| Create a user agent configuration .....                                 | 7-40       |
| Unit summary .....  | 7-41       |
| Checkpoint questions .....  | 7-42       |

|  |            |
|--|------------|
| Checkpoint answers .....   | 7-43       |
| Exercise 5 .....   | 7-44       |
| Exercise objectives .....  | 7-45       |
| Exercise overview (1 of 2) .....                                       | 7-46       |
| Exercise overview (2 of 2) .....                                       | 7-47       |
| <b>Unit 8. XML and web services security overview .....</b>            | <b>8-1</b> |
| Unit objectives .....  | 8-2        |
| Review of basic security terminology .....                             | 8-3        |
| Web services security .....  | 8-5        |
| Components of WS-Security .....  | 8-6        |
| Specifying security in SOAP messages .....                             | 8-7        |
| Scenario 1: Ensure confidentiality with XML encryption .....           | 8-8        |
| XML encryption and WS-Security .....                                   | 8-9        |
| DataPower support for XML encryption .....                             | 8-10       |
| Encrypt action .....   | 8-11       |
| Decrypt action .....   | 8-13       |
| Field-level encryption and decryption .....                            | 8-14       |
| XPath tool .....   | 8-15       |
| Sample encrypted SOAP message .....                                    | 8-16       |
| Scenario 2: Ensure integrity with XML signatures .....                 | 8-17       |
| DataPower support for XML signature .....                              | 8-19       |
| Sign action .....  | 8-20       |
| Verify action .....  | 8-22       |
| Verify action: Advanced tab .....                                      | 8-23       |
| Field-level message signature and verification .....                   | 8-24       |
| Sample signed SOAP message .....                                       | 8-25       |
| Summary of security and keys .....                                     | 8-26       |
| Unit summary .....   | 8-27       |
| Checkpoint questions .....   | 8-28       |
| Checkpoint answers .....   | 8-29       |
| Exercise 6 .....   | 8-30       |
| Exercise objectives .....  | 8-31       |
| Exercise overview .....  | 8-32       |
| Completed exercise .....   | 8-33       |
| <b>Unit 9. Authentication, authorization, and auditing (AAA) .....</b> | <b>9-1</b> |
| Unit objectives .....  | 9-2        |
| Authentication, authorization, and auditing .....                      | 9-3        |
| Authentication and authorization framework .....                       | 9-4        |
| AAA action and access control policy .....                             | 9-6        |
| How to define an access control policy (1 of 2) .....                  | 9-7        |
| How to define an access control policy (2 of 2) .....                  | 9-8        |
| Access control policy processing .....                                 | 9-9        |
| Scenario 1: Authorize authenticated clients .....                      | 9-10       |
| Scenario 1: Sample SOAP request message .....                          | 9-11       |
| Scenario 1: Identify and authenticate the client .....                 | 9-12       |
| Scenario 1: Authorize access to resources .....                        | 9-14       |
| Scenario 2: Security token conversion .....                            | 9-16       |
| Scenario 2: Sample HTTP request message .....                          | 9-17       |
| Scenario 2: Identify and authenticate the client .....                 | 9-18       |

|  |             |
|--|-------------|
| Scenario 2: Authorize access to resources .....                                | 9-19        |
| Scenario 3: Multiple identity extraction methods .....                         | 9-21        |
| Scenario 3: Identify and authenticate the client .....                         | 9-22        |
| Scenario 3: LDAP details .....   | 9-23        |
| Scenario 3: Authorize access to resources .....                                | 9-24        |
| Internal access control resources .....  | 9-25        |
| AAA XML file .....   | 9-26        |
| Example AAA XML file .....   | 9-27        |
| Lightweight Third Party Authentication .....                                   | 9-28        |
| External access control resource .....   | 9-29        |
| Lightweight Directory Access Protocol .....                                    | 9-30        |
| Security Assertion Markup Language .....                                       | 9-31        |
| Types of SAML assertions .....   | 9-32        |
| Scenario 4: Authorize valid SAML assertions .....                              | 9-33        |
| Scenario 4: SAML authentication statement (1 of 2) .....                       | 9-34        |
| Scenario 4: SAML authentication statement (2 of 2) .....                       | 9-35        |
| Scenario 4: SAML attribute statement .....                                     | 9-36        |
| Scenario 4: Identify and authenticate the client .....                         | 9-37        |
| Scenario 4: Authorize access to resources .....                                | 9-38        |
| Scenario 4: Match SAML attributes .....  | 9-39        |
| Access control policy by SAML information .....                                | 9-40        |
| Unit summary .....   | 9-41        |
| Checkpoint questions .....   | 9-42        |
| Checkpoint answers .....   | 9-43        |
| Exercise 7 .....   | 9-44        |
| Exercise objectives .....  | 9-45        |
| Exercise overview .....  | 9-46        |
| <br>   |             |
| <b>Unit 10. REST and JSON support for Web 2.0 and mobile applications.....</b> | <b>10-1</b> |
| Unit objectives .....  | 10-2        |
| Alternatives to SOAP-based web services .....                                  | 10-3        |
| Web SOA (Web 2.0) versus Enterprise SOA .....                                  | 10-4        |
| Web SOA protocols and standards .....  | 10-5        |
| Growth of mobile clients .....   | 10-6        |
| DataPower as the reverse proxy for Web 2.0 / Mobile clients .....              | 10-7        |
| Securing the reverse proxy with ISAM for Mobile .....                          | 10-8        |
| Introduction to REST .....   | 10-9        |
| REST style services .....  | 10-10       |
| Example: Employee processing .....   | 10-11       |
| Employee REST interface .....  | 10-12       |
| Example: REST interaction .....  | 10-13       |
| Example: Add employee REST request explained .....                             | 10-14       |
| Example: Add employee REST response explained .....                            | 10-15       |
| Common DataPower REST patterns: Facade .....                                   | 10-16       |
| Common DataPower REST patterns: Bridge .....                                   | 10-17       |
| Common DataPower REST patterns: REST enrichment .....                          | 10-18       |
| Tools to support REST: Service or protocol handler related .....               | 10-19       |
| Front side handler support of HTTP method selection .....                      | 10-20       |
| JSON request and response types .....  | 10-21       |
| Process bodyless messages .....  | 10-22       |
| JSON threat protection .....   | 10-23       |

|   |          |
|---|----------|
| Tools to support REST: Service policy related .....                               | 10-24    |
| Matching Rule on HTTP methods .....   | 10-25    |
| Changing the HTTP method in the processing rule .....                             | 10-26    |
| Convert Query Params to XML action .....  | 10-27    |
| Convert Query Params to XML action example .....                                  | 10-28    |
| Programmatic access to the HTTP method .....                                      | 10-29    |
| Programmatic access to the HTTP status code .....                                 | 10-30    |
| JSON .....  | 10-31    |
| JSON data types .....   | 10-32    |
| JSON version of XML .....   | 10-33    |
| JSONx .....   | 10-34    |
| JSONx version of JSON data structure .....  | 10-35    |
| Handling JSON in the request body .....   | 10-36    |
| Converting XML to JSON .....  | 10-37    |
| Validate action for JSON .....  | 10-38    |
| Example JSON and a JSON schema .....  | 10-39    |
| Transform action that uses XQuery (JSON and XML) .....                            | 10-40    |
| XQuery/JSONiq example .....   | 10-42    |
| GatewayScript action that uses JavaScript .....                                   | 10-43    |
| Sample GatewayScript: JSON input to SOAP output .....                             | 10-44    |
| Bridging REST and SOAP: Sample service policy #1 .....                            | 10-45    |
| Bridging REST and SOAP: Sample service policy #2 .....                            | 10-46    |
| Unit summary .....  | 10-47    |
| Checkpoint questions .....  | 10-48    |
| Checkpoint answers .....  | 10-49    |
| Exercise 8 .....  | 10-50    |
| Exercise objectives .....   | 10-51    |
| Exercise overview .....   | 10-52    |
| <br><b>Unit 11. OAuth overview and DataPower implementation .....</b>             | <br>11-1 |
| Unit objectives .....   | 11-2     |
| What is OAuth? .....  | 11-3     |
| Delegated authorization example .....   | 11-4     |
| Example: Allow third-party access to social account .....                         | 11-5     |
| Example: Third-party access to online photo album .....                           | 11-6     |
| Before Oauth: Sharing user passwords .....  | 11-7     |
| OAuth Step 1: Resource owner requests access .....                                | 11-8     |
| OAuth Step 2: OAuth client redirection to owner .....                             | 11-9     |
| OAuth Step 3: Authenticate owner with authorization server .....                  | 11-10    |
| OAuth Step 4: Ask resource owner to grant access to resources .....               | 11-11    |
| OAuth Step 5: Resource owner grants client access to resources .....              | 11-12    |
| OAuth Step 6: Authorization server sends authorization grant code to client ..... | 11-13    |
| OAuth Step 7: Client requests access token from authorization server .....        | 11-14    |
| OAuth Step 8: Authorization server sends authorization token to client .....      | 11-15    |
| OAuth Step 9: OAuth client sends access token to resource server .....            | 11-16    |
| OAuth Step 10: Resource server grants access to OAuth client .....                | 11-17    |
| OAuth 2.0 protocol defines roles .....  | 11-18    |
| OAuth 2.0 roles: Common implementation .....                                      | 11-19    |
| OAuth 2.0 roles in the DataPower world (1 of 2) .....                             | 11-20    |
| OAuth 2.0 roles in the DataPower world (2 of 2) .....                             | 11-21    |
| Sample three-legged scenario in DataPower (1 of 4) .....                          | 11-22    |

|   |             |
|---|-------------|
| Sample three-legged scenario in DataPower (2 of 4) . . . . .  | 11-23       |
| Sample three-legged scenario in DataPower (3 of 4) . . . . .  | 11-24       |
| Sample three-legged scenario in DataPower (4 of 4) . . . . .  | 11-25       |
| Authorization request . . . . .                               | 11-26       |
| Authorization response . . . . .                              | 11-27       |
| Access token request . . . . .                                | 11-28       |
| Access token response . . . . .                               | 11-29       |
| Resource request . . . . .                                    | 11-30       |
| OAuth client and the OAuth Client Profile object . . . . .    | 11-31       |
| DataPower OAuth objects: OAuth Client (1 of 4) . . . . .      | 11-32       |
| DataPower OAuth objects: OAuth Client (2 of 4) . . . . .      | 11-34       |
| DataPower OAuth objects: OAuth Client (3 of 4) . . . . .      | 11-35       |
| DataPower OAuth objects: OAuth Client (4 of 4) . . . . .      | 11-36       |
| DataPower OAuth objects: OAuth Client Group . . . . .         | 11-37       |
| DataPower OAuth objects: Web Token Service . . . . .          | 11-38       |
| AAA policy: Key to OAuth behavior . . . . .                   | 11-39       |
| AAA policy for the web token service . . . . .                | 11-40       |
| AAA policy for the resource server (1 of 2) . . . . .         | 11-41       |
| AAA policy for the resource server (2 of 2) . . . . .         | 11-42       |
| Unit summary . . . . .  | 11-43       |
| Checkpoint questions . . . . .                                | 11-44       |
| Checkpoint answers . . . . .                                  | 11-45       |
| Exercise 9 . . . . .  | 11-46       |
| Exercise objectives . . . . .                                 | 11-47       |
| Exercise overview: User interaction . . . . .                 | 11-48       |
| Exercise overview: OAuth interaction (1 of 4) . . . . .       | 11-49       |
| Exercise overview: OAuth interaction (2 of 4) . . . . .       | 11-50       |
| Exercise overview: OAuth interaction (3 of 4) . . . . .       | 11-51       |
| Exercise overview: OAuth interaction (4 of 4) . . . . .       | 11-52       |
| <b>Unit 12. DataPower caching . . . . .</b>                   | <b>12-1</b> |
| Unit objectives . . . . .                                     | 12-2        |
| Caching in DataPower . . . . .                                | 12-3        |
| Document caching: On and off the appliance . . . . .          | 12-4        |
| Document and style sheet caching in the XML manager . . . . . | 12-6        |
| Document cache policy (1 of 2) . . . . .                      | 12-7        |
| Document cache policy (2 of 2) . . . . .                      | 12-8        |
| Document cache programming interface . . . . .                | 12-9        |
| Document cache providers . . . . .                            | 12-10       |
| XC10 integration overview . . . . .                           | 12-11       |
| XC10 integration in document cache logic flow . . . . .       | 12-12       |
| XC10 integration URL Opener user interface . . . . .          | 12-13       |
| IBM WebSphere eXtreme Scale (1 of 2) . . . . .                | 12-14       |
| IBM WebSphere eXtreme Scale (2 of 2) . . . . .                | 12-15       |
| Unit summary . . . . .  | 12-16       |
| Checkpoint questions . . . . .                                | 12-17       |
| Checkpoint answers . . . . .                                  | 12-18       |
| Exercise 10 . . . . .   | 12-19       |
| Exercise objectives . . . . .                                 | 12-20       |

|  |             |
|--|-------------|
| <b>Unit 13. Integrating with IBM MQ.....</b>               | <b>13-1</b> |
| Unit objectives .....                                      | 13-2        |
| IBM MQ fundamentals .....                                  | 13-3        |
| IBM MQ message .....                                       | 13-4        |
| Transactions .....   | 13-5        |
| DataPower support for IBM MQ .....                         | 13-6        |
| Provide IBM MQ access .....                                | 13-7        |
| WebSphere MQ queue manager overview .....                  | 13-8        |
| WebSphere MQ queue manager overview: Main tab .....        | 13-9        |
| WebSphere MQ queue manager overview: Connections tab ..... | 13-10       |
| WebSphere MQ queue manager overview: CCSI tab .....        | 13-11       |
| WebSphere MQ queue manager overview: MQCSP tab .....       | 13-12       |
| Step 1: Create a WebSphere MQ queue manager (1 of 3) ..... | 13-13       |
| Step 1: Create a WebSphere MQ queue manager (2 of 3) ..... | 13-14       |
| Step 1: Create a WebSphere MQ queue manager (3 of 3) ..... | 13-16       |
| Step 1: Use SSL in mutual authentication mode .....        | 13-17       |
| Step 2: Add an MQ front side handler .....                 | 13-18       |
| Step 3: Configure an IBM MQ back-end transport .....       | 13-19       |
| Publish/subscribe: MQ front side handler support .....     | 13-20       |
| Publish/subscribe: IBM MQ back-end transport support ..... | 13-21       |
| Message properties .....                                   | 13-22       |
| Ordered processing of IBM MQ messages .....                | 13-23       |
| Controlling backout of WebSphere MQ messages .....         | 13-25       |
| Decision tree for the backout settings .....               | 13-26       |
| MQ Header action in service policy .....                   | 13-27       |
| Typical uses of an MQ Header action .....                  | 13-28       |
| Transactions and IBM MQ .....                              | 13-29       |
| IBM MQ front side transactions .....                       | 13-30       |
| IBM MQ back side transactions .....                        | 13-31       |
| IBM MQ DataPower URL .....                                 | 13-32       |
| MQ Queue Manager Group object .....                        | 13-33       |
| Unit summary .....   | 13-34       |
| Checkpoint questions .....                                 | 13-35       |
| Checkpoint answers .....                                   | 13-36       |
| Exercise 11 .....  | 13-37       |
| Exercise objectives .....                                  | 13-38       |
| Exercise overview .....                                    | 13-39       |
| Exercise overview 1 of 2 .....                             | 13-40       |
| Exercise overview 2 of 2 .....                             | 13-41       |
| <b>Unit 14. Web service proxy service.....</b>             | <b>14-1</b> |
| Unit objectives .....                                      | 14-2        |
| Web service proxy overview .....                           | 14-3        |
| Conceptual architecture of a web service proxy .....       | 14-4        |
| Web service virtualization .....                           | 14-5        |
| Web service proxy benefits .....                           | 14-6        |
| Web service proxy configuration tabs (1 of 2) .....        | 14-7        |
| Web service proxy configuration tabs (2 of 2) .....        | 14-8        |
| Web service proxy basic configuration steps .....          | 14-9        |
| Step 1: Obtain WSDL document .....                         | 14-10       |
| WSDL structure .....                                       | 14-11       |

|   |             |
|---|-------------|
| Step 2: Creating a web service proxy . . . . .            | 14-12       |
| An alternative: Web service proxy object editor . . . . . | 14-13       |
| Step 3: Add WSDL document to web service proxy . . . . .  | 14-14       |
| Step 4: Configure WSDL endpoint . . . . .                 | 14-15       |
| Step 5: Configure local endpoint handler . . . . .        | 14-17       |
| Step 6: Add the WSDL to the service . . . . .             | 14-18       |
| Initial WSDL completed . . . . .                          | 14-19       |
| Other ways to get a WSDL . . . . .                        | 14-20       |
| View WSDL services . . . . .                              | 14-21       |
| Retrieve the client WSDL from the service . . . . .       | 14-22       |
| Modifying the location in the client WSDL . . . . .       | 14-23       |
| Step 7: Configuring web service proxy policy . . . . .    | 14-24       |
| Configure web service proxy policy rule . . . . .         | 14-25       |
| Adding a rule . . . . .                                   | 14-26       |
| Default validation (user policies) . . . . .              | 14-27       |
| Create reusable rule . . . . .                            | 14-28       |
| Advanced web service proxy configuration . . . . .        | 14-29       |
| WS-Policy . . . . .                                       | 14-30       |
| Conformance policy . . . . .                              | 14-31       |
| Conformance policy object . . . . .                       | 14-32       |
| Service priority . . . . .                                | 14-33       |
| Proxy settings (1 of 4) . . . . .                         | 14-34       |
| Proxy settings (2 of 4) . . . . .                         | 14-35       |
| Proxy settings (3 of 4) . . . . .                         | 14-37       |
| Proxy settings (4 of 4) . . . . .                         | 14-38       |
| Web service proxy: SLM Policy tab . . . . .               | 14-39       |
| WSDL cache policy . . . . .                               | 14-40       |
| Troubleshooting a web service proxy . . . . .             | 14-41       |
| Unit summary . . . . .                                    | 14-42       |
| Checkpoint questions . . . . .                            | 14-43       |
| Checkpoint answers . . . . .                              | 14-44       |
| Exercise 12 . . . . .                                     | 14-45       |
| Exercise objectives . . . . .                             | 14-46       |
| Exercise overview . . . . .                               | 14-47       |
| <b>Unit 15. Service level monitoring . . . . .</b>        | <b>15-1</b> |
| Unit objectives . . . . .                                 | 15-2        |
| Service level monitoring (SLM) in DataPower . . . . .     | 15-3        |
| The pieces of SLM . . . . .                               | 15-4        |
| Approaches to define SLM policies . . . . .               | 15-5        |
| Approach 1: Add an SLM action to a request rule . . . . . | 15-6        |
| The pieces of SLM . . . . .                               | 15-7        |
| The SLM credential class . . . . .                        | 15-8        |
| The SLM resource class . . . . .                          | 15-10       |
| SLM resource class example . . . . .                      | 15-11       |
| The SLM schedule . . . . .                                | 15-12       |
| The SLM action . . . . .                                  | 15-13       |
| SLM statement (1 of 2) . . . . .                          | 15-14       |
| SLM statement (2 of 2) . . . . .                          | 15-15       |
| SLM policy: Main tab . . . . .                            | 15-17       |
| SLM policy: Statements tab . . . . .                      | 15-18       |

|   |                 |
|---|-----------------|
| Getting SLM statements into the Statement list .....                      | 15-19           |
| Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2) ..... | 15-20           |
| Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2) ..... | 15-21           |
| SLM Policy tab: Graphs .....  | 15-22           |
| SLM action granularity .....  | 15-23           |
| Unit summary .....  | 15-24           |
| Checkpoint questions .....  | 15-25           |
| Checkpoint answers .....  | 15-26           |
| Exercise 13 .....   | 15-27           |
| Exercise objectives .....   | 15-28           |
| Exercise overview .....   | 15-29           |
| <br><b>Unit 16. Patterns for service configuration .....</b>              | <br><b>16-1</b> |
| Unit objectives .....   | 16-2            |
| What is a pattern? .....  | 16-3            |
| Creating a pattern .....  | 16-4            |
| Deploying a pattern .....   | 16-5            |
| The Blueprint Console .....   | 16-6            |
| Getting to the Blueprint Console .....                                    | 16-7            |
| The Blueprint Console .....   | 16-8            |
| The Blueprint Console: Menu bar .....                                     | 16-9            |
| The Blueprint Console: Getting Started tab .....                          | 16-10           |
| The Blueprint Console: Services tab .....                                 | 16-11           |
| The Blueprint Console: Patterns tab .....                                 | 16-12           |
| The pattern toolbar .....   | 16-13           |
| The Blueprint Console: Manage tab .....                                   | 16-14           |
| Steps to generate a service from a pattern .....                          | 16-15           |
| Deployment: Selecting a pattern .....                                     | 16-16           |
| Deployment: Filling out the wizard .....                                  | 16-17           |
| Points of variability .....   | 16-18           |
| Unit summary .....  | 16-19           |
| Checkpoint questions .....  | 16-20           |
| Checkpoint answers .....  | 16-21           |
| Exercise 14 .....   | 16-22           |
| Exercise objectives .....   | 16-23           |
| Exercise overview .....   | 16-24           |
| <br><b>Unit 17. Course summary .....</b>                                  | <br><b>17-1</b> |
| Unit objectives .....   | 17-2            |
| Course learning objectives .....  | 17-3            |
| Course learning objectives .....  | 17-4            |
| Course review (1 of 3) .....  | 17-5            |
| Course review (2 of 3) .....  | 17-6            |
| Course review (3 of 3) .....  | 17-7            |
| Lab exercise solutions .....  | 17-8            |
| To learn more on the subject .....  | 17-9            |
| More DataPower education opportunities .....                              | 17-10           |
| Unit summary .....  | 17-11           |

|  |            |
|--|------------|
| <b>Appendix A. List of abbreviations .....</b> | <b>A-1</b> |
| <b>Appendix B. Resource guide.....</b>         | <b>B-1</b> |



# Trademarks

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

IBM, 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®  | Bluemix™        | CICS®        |
| DataPower® | DB™             | DB2 Connect™ |
| DB2®       | developerWorks® | Express®     |
| IMS™       | Notes®          | RACF®        |
| Rational®  | Redbooks®       | Tivoli®      |
| WebSphere® | Worklight®      | z/OS®        |
| 400®       |                 |              |

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.

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

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

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 V7.1

**Duration:** 5 days

### Purpose

This course teaches you the fundamental skills that are required to configure, implement, and troubleshoot services that are developed on the IBM DataPower Gateway (IDG) appliances with firmware version 7.1.x. The concepts in this course are also beneficial to system administrators of the DataPower appliance, although there is a separate course for administrators.

### Audience

This course is designed for integration developers who configure service policies on IBM DataPower Gateway appliances.

### Prerequisites

Before taking this course, you should successfully complete course VW700, *Technical Introduction to IBM WebSphere DataPower Gateway Appliances V7* and VW710, *What's New in DataPower V7.1*. You should also be familiar with:

- Security-based concepts and protocols
- XML-related technologies such as XML schema, XPath, and XSLT
- JavaScript programming
- Web service fundamentals and the web services security specifications
- REST-based services

### Objectives

After completing this course, you should be able to:

- Describe how DataPower 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 DataPower appliances

- Configure a multi-protocol gateway (MPGW) to handle multiple protocols from a single service
- Configure a service level monitoring (SLM) policy to control message traffic
- Configure support for IBM MQ
- 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
- Configure message transformation and routing by using style sheets (XSL) and GatewayScripts
- Handle errors in service policies

# Agenda

## Day 1

- Course introduction
- Unit 1. Quick introduction to developing on DataPower
- Exercise 1. First exposure to the DataPower developer environment
- Unit 2. Services overview
- Unit 3. Structure of a service
- Exercise 2. Creating a BookingService gateway
- Unit 4. Multi-protocol gateway service

## Day 2

- Unit 5. Problem determination tools
- Exercise 3. Enhancing the BookingService 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
- Unit 8. XML and web services security overview

## Day 3

- Exercise 6. Web service encryption and digital signatures
- Unit 9. Authentication, authorization, and auditing (AAA)
- Exercise 7. Web services authentication and authorization
- Unit 10. REST and JSON support for Web 2.0 and mobile applications
- Exercise 8. Using DataPower to implement REST services

## Day 4

- Unit 11. OAuth overview and DataPower implementation
- Exercise 9. Defining a three-legged OAuth scenario that uses DataPower services
- Unit 12. DataPower caching
- Exercise 10. Configuring a response cache
- Unit 13. Integrating with IBM MQ
- Exercise 11. Configuring a multi-protocol gateway service with IBM MQ

## **Day 5**

- Unit 14. Web service proxy service
- Exercise 12. Configuring a web service proxy
- Unit 15. Service level monitoring
- Exercise 13. Implementing a service level monitor in a web service proxy
- Unit 16. Patterns for service configuration
- Exercise 14. Using a DataPower pattern with the Blueprint console
- Unit 17. Course summary

# Unit 1. Quick introduction to developing on DataPower

## What this unit is about

This unit introduces the developer environment for DataPower Gateway appliances. It presents the WebGUI home page as the entry point for DataPower development, and provides a high-level view of the common pages for service development.

## What you should be able to do

After completing this unit, you should be able to:

- Log in to the WebGUI
- Navigate around the WebGUI interface
- Identify the primary functions of the menus on the navigation bar
- Start the creation of a DataPower service
- Identify the typical areas of a service configuration page
- Save configuration definitions in memory and on the file system
- List the file directories that are commonly used for development
- Support any non-English languages that are enabled on the appliance

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## Unit objectives

After completing this unit, you should be able to:

- Log in to the WebGUI
- Navigate around the WebGUI interface
- Identify the primary functions of the menus on the navigation bar
- Start the creation of a DataPower service
- Identify the typical areas of a service configuration page
- Save configuration definitions in memory and on the file system
- List the file directories that are commonly used for development
- Support any non-English languages that are enabled on the appliance

© Copyright IBM Corporation 2015

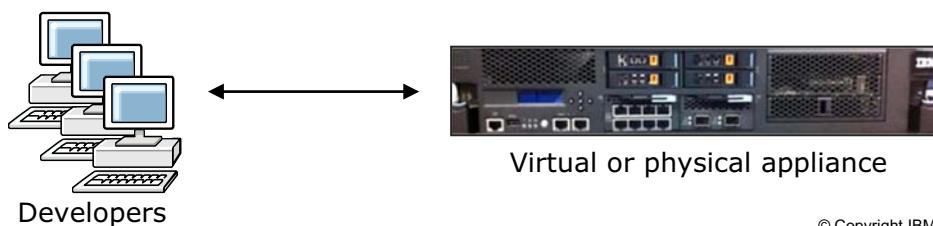
Figure 1-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## DataPower development is done on an appliance

- To configure services, you must be connected to an appliance:
  - XG45, XI52, XB62, IDG physical appliance (multi-developer)
  - XG45, XI52, IDG Virtual Edition for Nonproduction environments (multi-developer)
  - XG45, XI52, IDG Virtual Edition for Developers (single developer)
- Multiple DataPower interfaces available for development:
  - WebGUI web application (standard interface that is used for development)
  - Command-line interface (CLI, typically used for appliance administration, not development)
  - XML Management Interface (XMI, typically used for service migration, configuration management, appliance administration, not development)



© Copyright IBM Corporation 2015

Figure 1-2. DataPower development is done on an appliance

WE711 / ZE7111.0

### Notes:

“IDG” is “IBM DataPower Gateway”, the new appliance series that was announced for V7. It is also referred to as the “9006” appliance.

Although the command-line interface (CLI) and the SOAP management (SOMA) aspect of the XML management interface (XMI) can be used for development, the primary way to develop resources on the appliance is the WebGUI web interface.

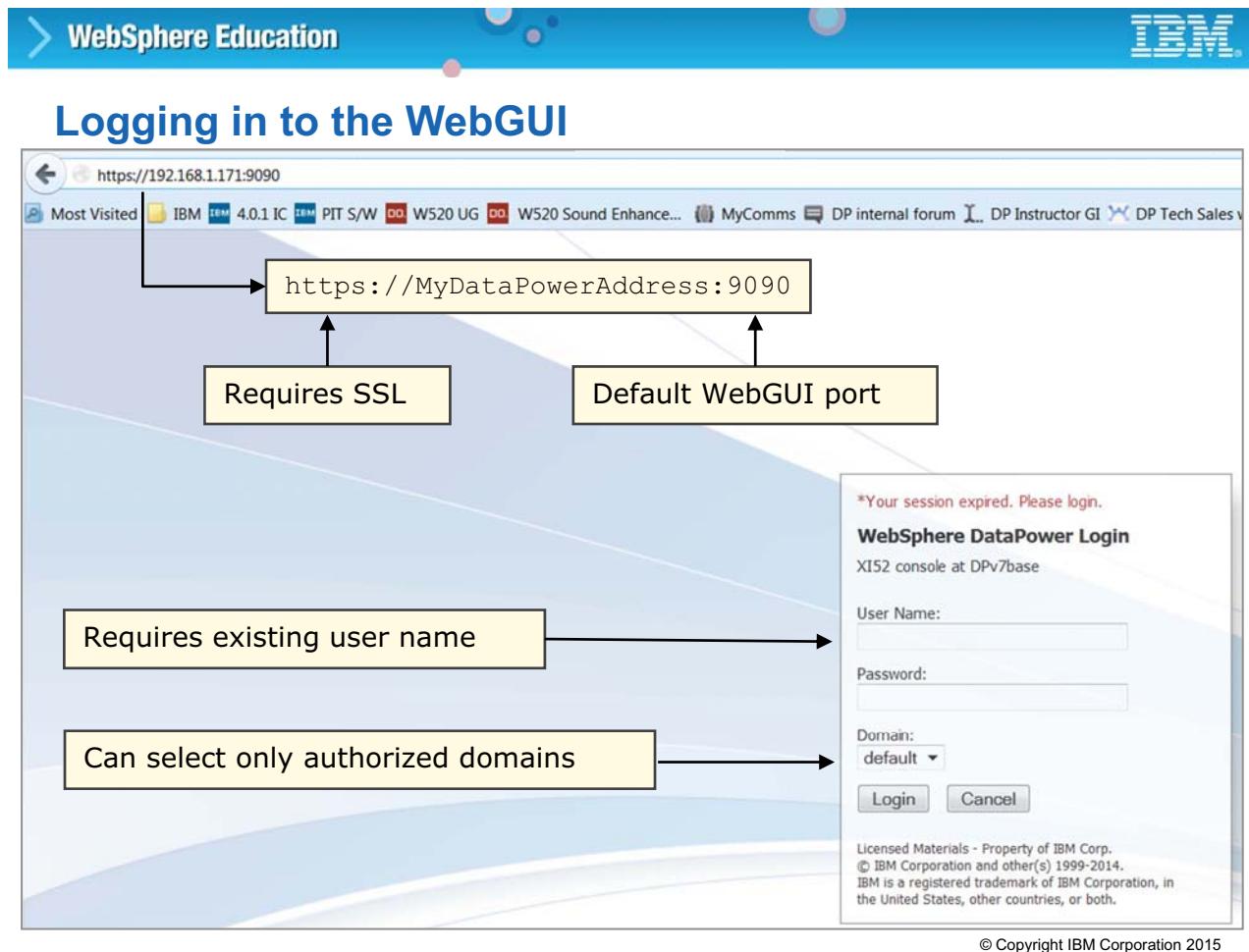


Figure 1-3. Logging in to the WebGUI

WE711 / ZE7111.0

### Notes:

The WebGUI access to the appliance uses SSL, so the protocol is “https.” The default port that the WebGUI is active on is 9090, although the appliance administrator can change it.

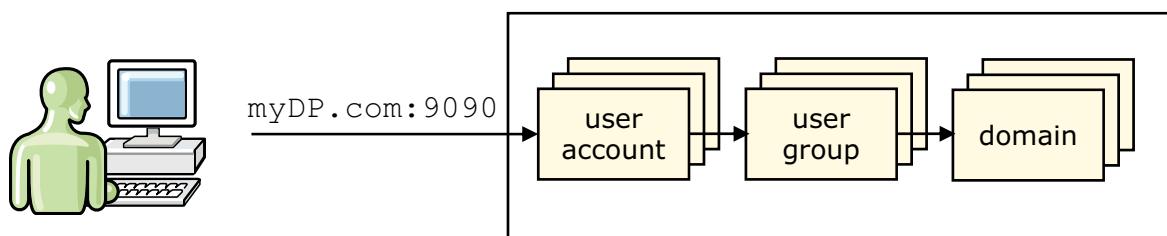
To log in to the appliance, you must use a predefined user name and password. In DataPower administrative pages, the “user name” is the name of a “user account” object.

Although all defined domains are visible under “Domain,” the user can connect only to domains to which the user is authorized.

## Login and development access to the appliance

An administrator provides the user account, password, and one or more domains to work in

- User account is assigned to a single user group
- User group is a definition of permissions, which includes which domains can be accessed
- Domain: “Sandbox” in which you develop application services
  - Predefined **default** domain should be used for administration activities only



“Administration” defines all of these resources

© Copyright IBM Corporation 2015

Figure 1-4. Login and development access to the appliance

WE711 / ZE7111.0

### Notes:

A user group can have multiple sets of permissions, so multiple application domains can be accessible.

Creating user accounts, user groups, and domains are covered in detail in the Administration course.

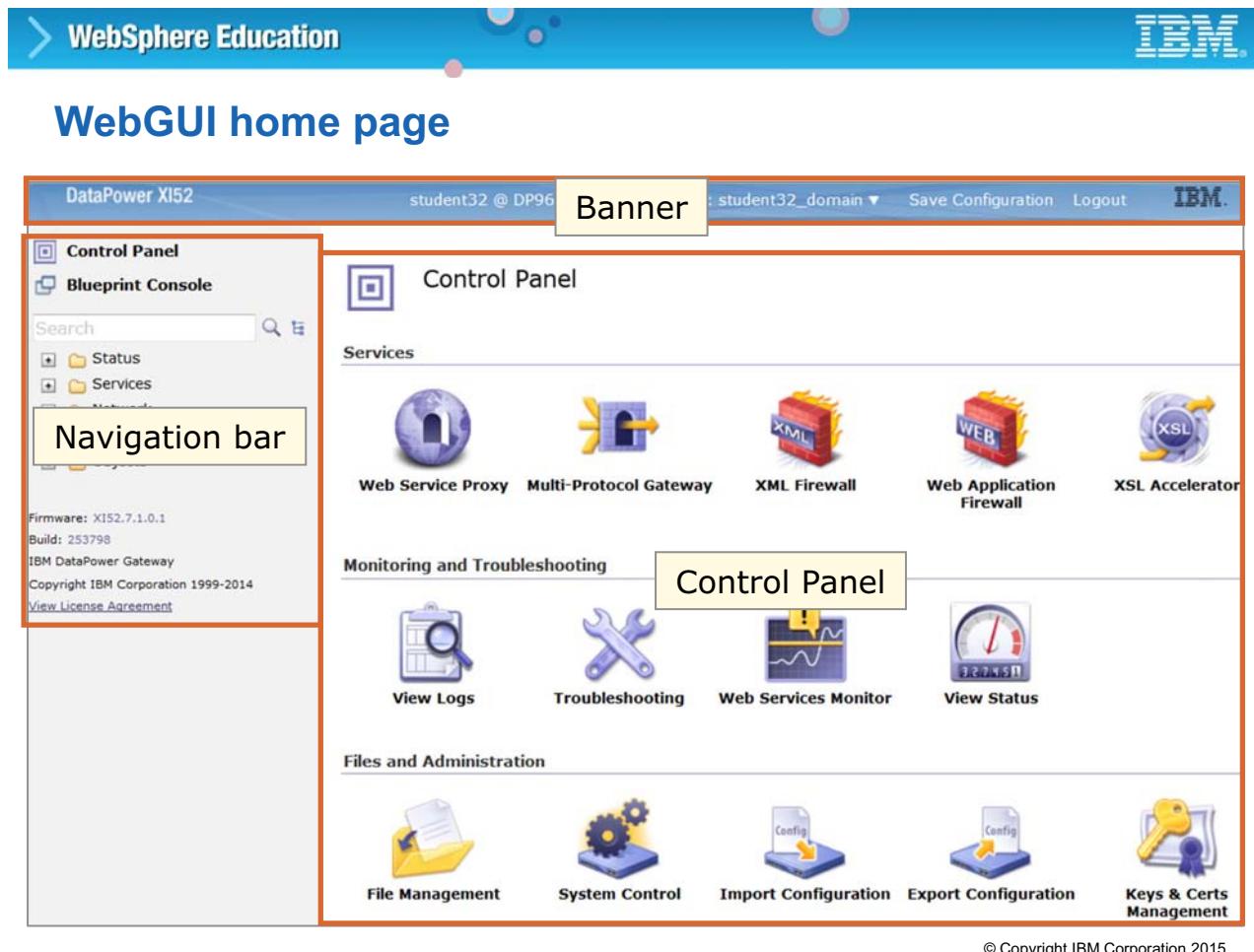


Figure 1-5. WebGUI home page

WE711 / ZE7111.0

## Notes:

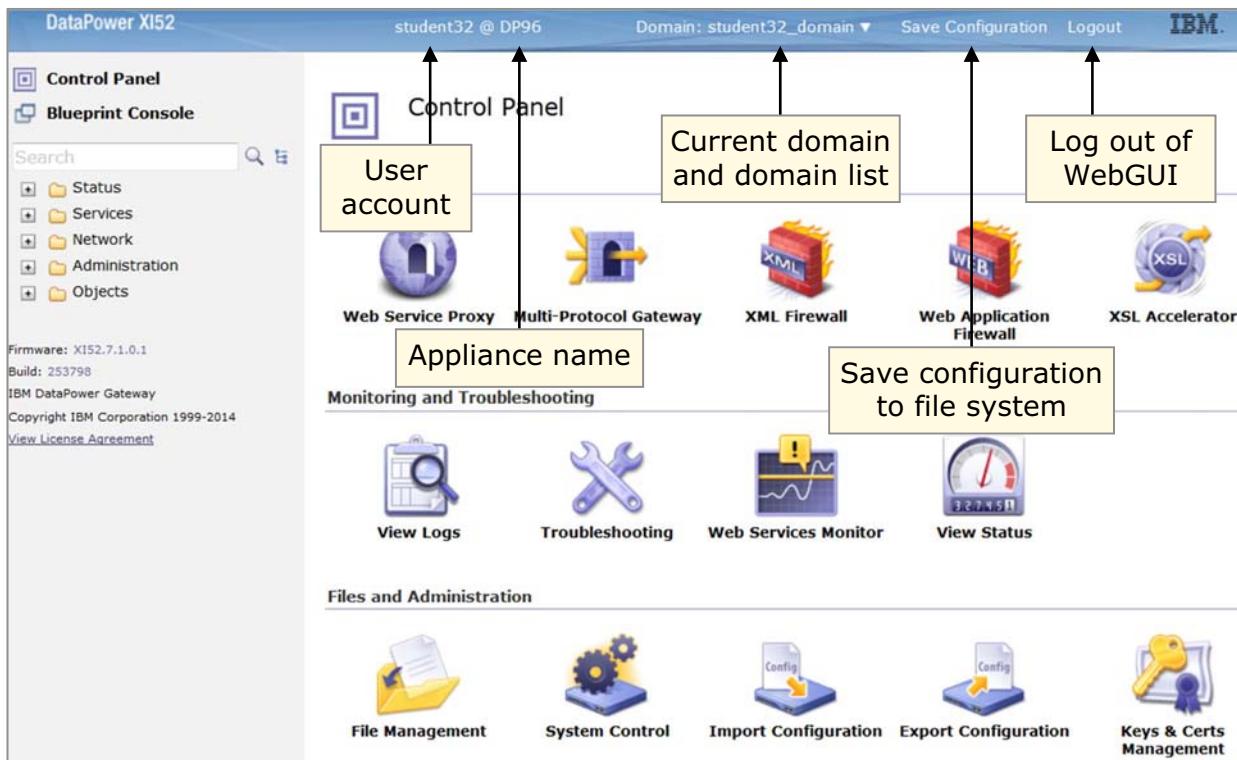
When login is completed, the WebGUI home page is displayed.

The home page has three areas: banner, navigation bar, and Control Panel.

When using a specific function, the work area for that function replaces the Control Panel area.



## WebGUI banner



© Copyright IBM Corporation 2015

Figure 1-6. WebGUI banner

WE711 / ZE7111.0

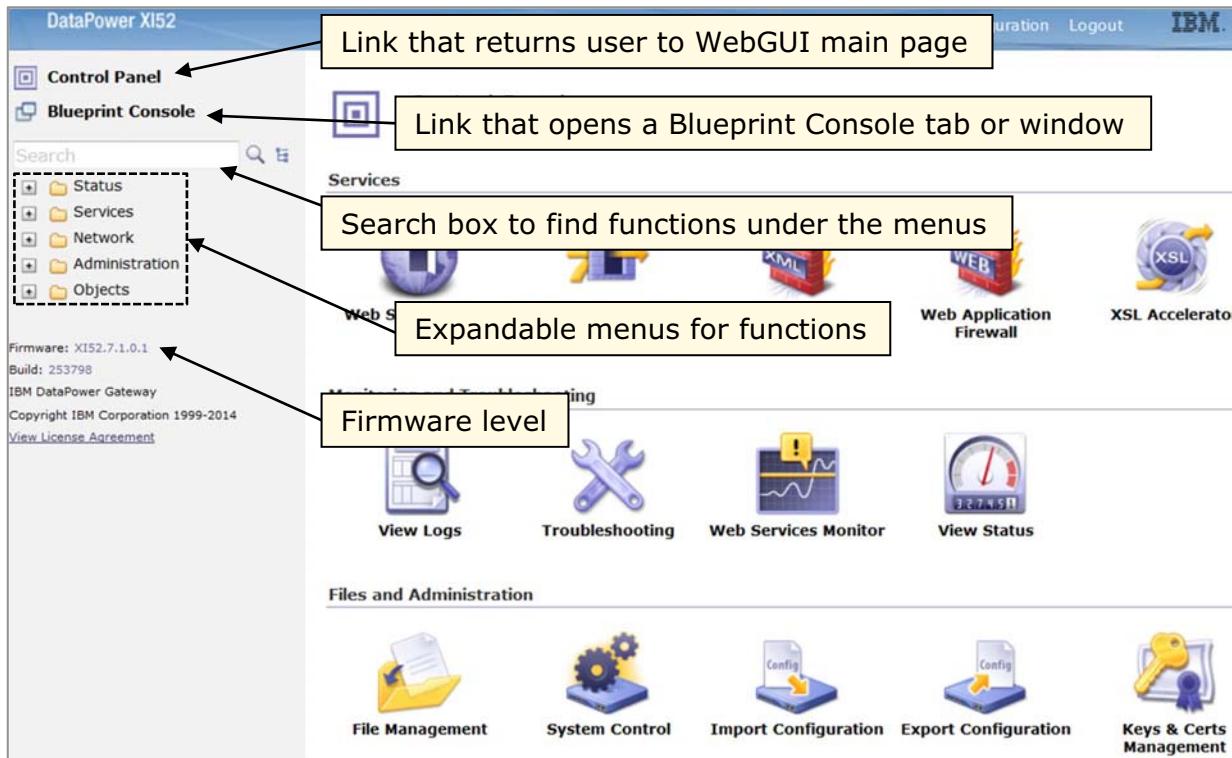
### Notes:

The appliance administrator specifies the appliance name.

The user can switch to other authorized domains from the **Domain** menu.



## WebGUI navigation bar



© Copyright IBM Corporation 2015

Figure 1-7. WebGUI navigation bar

WE711 / ZE7111.0

### Notes:

The Blueprint Console is covered in a later unit.

As text is entered into the search box, candidate functions appear as a list beneath the entry field. As more letters are entered, the list adjusts to accommodate the newly entered text.



## WebGUI Control Panel: Links to common functions

The screenshot shows the DataPower XI52 WebGUI Control Panel. At the top, it displays the user 'student32 @ DP96' and the domain 'student32\_domain'. There are links for 'Save Configuration' and 'Logout'. The IBM logo is in the top right corner.

The left sidebar contains a navigation bar with 'Control Panel' (selected) and 'Blueprint Console'. Below this is a search field and a tree view with categories: Status, Services, Network, Administration, and Objects. A note at the bottom left states: 'Firmware: XI52.7.1.0.1 Build: 253798 IBM DataPower Gateway Copyright IBM Corporation 1999-2014 View License Agreement'.

The main content area is organized into several sections:

- Control Panel:** Shows a link to 'Control Panel' (highlighted with a red box).
- Services:** Shows icons for 'Web Service Proxy', 'Multi-Protocol Gateway', 'XML Firewall', 'Web Application Firewall', and 'XSL Accelerator'.
- Monitoring and Troubleshooting:** Shows icons for 'View Logs', 'Troubleshooting', 'Web Services Monitor', and 'View Status' (highlighted with a red box).
- Files and Administration:** Shows icons for 'File Management', 'System Control', 'Import Configuration', 'Export Configuration', and 'Keys & Certs Management' (highlighted with a red box).

At the bottom right, there is a copyright notice: '© Copyright IBM Corporation 2015'.

Figure 1-8. WebGUI Control Panel: Links to common functions

WE711 / ZE7111.0

### Notes:

The Control Panel allows quick access to common development and administration functions.

The Services section allows you to create or modify the primary DataPower services.

The Monitoring and Troubleshooting section provides a view of the appliance system log, troubleshooting functions, and status.

The Files and Administration section provides links to file management, system control, importing and exporting, and cryptographic keys and certificates on the appliance.

Most of the 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

A screenshot of the DataPower V7.1 navigation bar. It includes a "Control Panel" button, a "Blueprint Console" button, and a "Search" field containing a dropdown menu with options: Status, Services, Network, Administration, and Objects.

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

© Copyright IBM Corporation 2015

Figure 1-9. Navigation bar categories

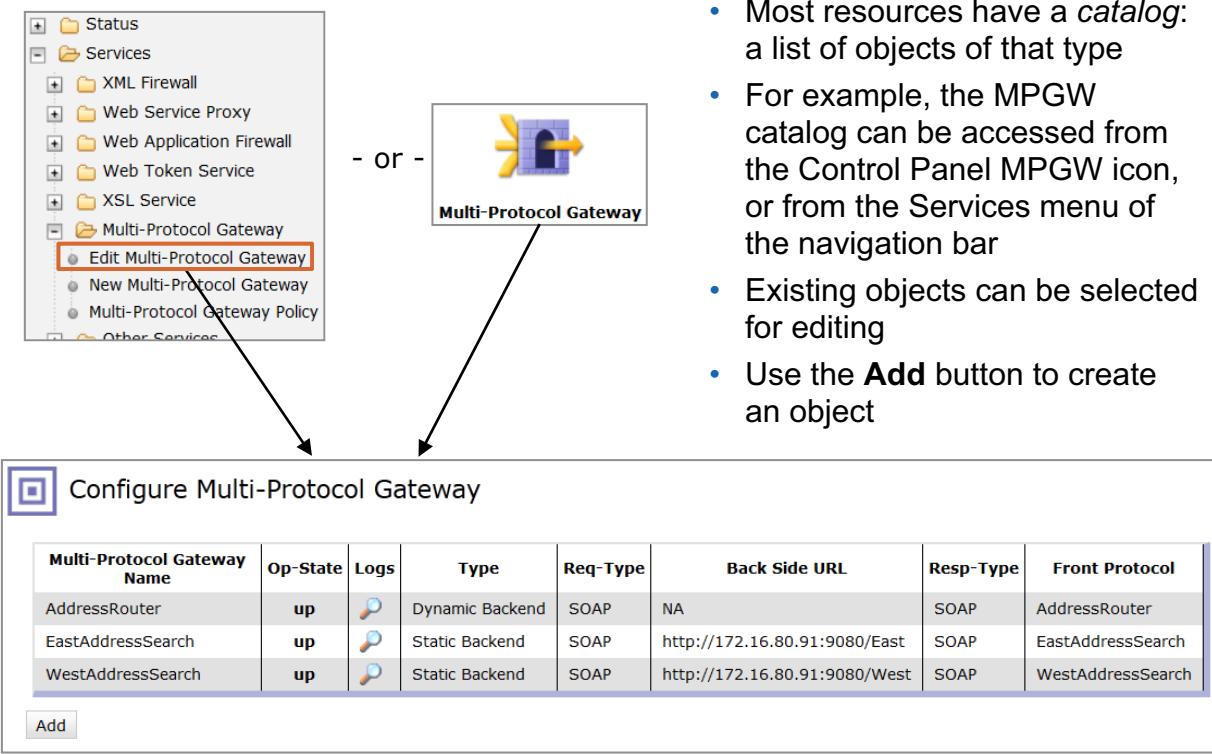
WE711 / ZE7111.0

### Notes:

The main menus in the navigation bar enable many development functions. The most popular development menus are **Status**, **Services**, and **Objects**.

Most of the objects that are created as part of a service, including the service itself, are also individually available under **Objects**.

## Catalog for a multi-protocol gateway (MPGW)



© Copyright IBM Corporation 2015

Figure 1-10. Catalog for a multi-protocol gateway (MPGW)

WE711 / ZE7111.0

### Notes:



## Example service configuration page

- Typical areas in a service configuration page (covered in detail later)

The screenshot shows the DataPower XI52 Control Panel with the following details:

- Control Panel:** Shows 'Control Panel' and 'Blueprint Console'.
- General Configuration:** Contains fields for 'Multi-Protocol Gateway Name' (BookingServiceProxy), 'Summary' (proxy), and 'Type' (dynamic-backends).
- XML Manager:** Shows 'default' selected in a dropdown.
- Multi-Protocol Gateway Policy:** Shows 'BookingServicePolicy' selected.
- URL Rewrite Policy:** Shows '(none)' selected.
- Back side settings:** Shows 'Default Backend URL' set to 'http://dp\_internal\_ip:9080/Booking'.
- Front side settings:** Shows 'Front Side Protocol' with options: 'HTTP\_12321 (HTTP Front Side Handler)', 'HTTPS\_12322 (HTTPS Front Side Handler)', and 'BookingServiceMQfsh (MQ Front Side Handler) [down]'.

Figure 1-11. Example service configuration page

WE711 / ZE7111.0

### Notes:

This slide shows areas that are common for the configuration of a service.

- Navigation bar remains visible and available
- Clicking the object type typically returns you to the catalog for that object type
- Multiple tabs that are object-dependent
- Apply, cancel, or delete the object
- Object-specific links
- Service policy configuration (programming-like configuration)
- Information to connect to the back side application
- Front side interface to access the service

More detailed presentation of the multi-protocol gateway configuration is covered in a later unit.



## The system log

- The default system log captures messages that the firmware and services emit
  - Useful during development, debugging, and production
  - Custom logs can be created
- The system and any custom logs are “push-down”; the newest entry is on the top
- The logs have different ways to sort and filter the entries

System Log

[Refresh Log](#) Target: **default-log** Filter: (none) (none)

current time: 13:33:32 on 2012-08-28

| time            | category      | level  | tid      | direction | client       | msgid      | message   | Show last  |
|-----------------|---------------|--------|----------|-----------|--------------|------------|---|------------|
| Tue Aug 28 2012 |               |        |          |           |              |            |   | 50 100 all |
| 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 2015

Figure 1-12. The system log

WE711 / ZE7111.0

### 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. More log targets can be defined and customized.

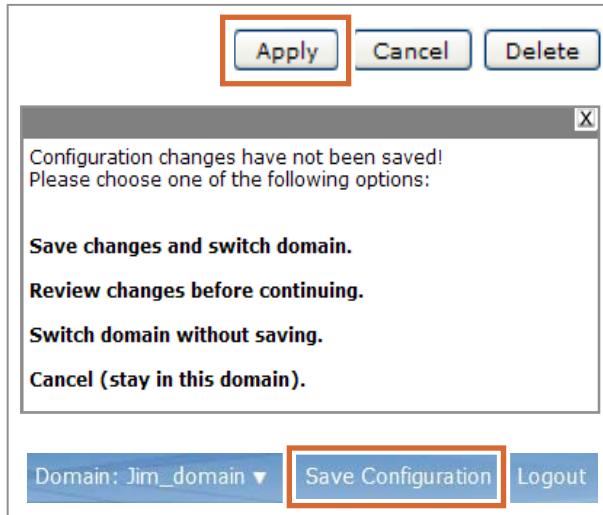
The most recent log entries are shown at the top of the system log.

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

Logging is covered in more detail in a later unit.



## Saving configuration changes



- 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 Configuration** on the upper-right corner of the web page to commit changes to the file system

© Copyright IBM Corporation 2015

Figure 1-13. Saving configuration changes

WE711 / ZE7111.0

### Notes:

Clicking **Apply** commits configuration changes that are made in the current WebGUI page. However, such changes are stored in temporary memory. You must click **Save Configuration** on the upper-right corner of the WebGUI interface to commit changes to permanent storage (file system). 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.



## 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 Configuration**
  - 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 the 'Configuration Checkpoints' page. At the top, there is a 'Refresh List' button and a table with columns 'Name', 'Time', and 'Actions'. One row in the table is shown, labeled 'Checkpoint1' with a timestamp of '2012-10-03 22:12:35 GMT'. Under the 'Actions' column for this row are three buttons: 'Rollback', 'Remove', and 'Compare'. Below the table is a section titled 'Create a new Configuration Checkpoint' with a 'Checkpoint Name:' input field and a 'Save Checkpoint' button.

| Name        | Time                    | Actions                                      |
|-------------|-------------------------|--|
| Checkpoint1 | 2012-10-03 22:12:35 GMT | <b>Rollback</b> <b>Remove</b> <b>Compare</b> |

**Create a new Configuration Checkpoint**

Checkpoint Name:  Save Checkpoint

© Copyright IBM Corporation 2015

Figure 1-14. Configuration Checkpoints

WE711 / ZE7111.0

### 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).

**Control Panel**

- Status
- Services
- Network
- Administration**
  - Main
  - File Management**
  - System Control

**File management**

**File Management**

Available Space: 14283 MBytes (encrypted), 4064 MBytes (temporary)

| Name           | Action              |
|----------------|---------------------|
| cert:          | Actions...          |
| chkpoints:     | Actions...          |
| <b>config:</b> | <b>4</b> Actions... |
| export:        | Actions...          |
| image:         | Actions...          |
| local:         | Actions...          |
| logstore:      | Actions...          |
| logtemp:       | Actions...          |
| pubcert:       | Actions...          |
| sharedcert:    | Actions...          |
| store:         | Actions...          |
| tasktemplates: | Actions...          |
| temporary:     | Actions...          |

© Copyright IBM Corporation 2015

Figure 1-15. File management

WE711 / ZE7111.0

## 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, others are shared across all domains, and a few are specific to the default domain.
4. Actions against a directory are initiated from the **Action** column. Actions against selected files are initiated by using buttons.

## File directories for configuration

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

© Copyright IBM Corporation 2015

Figure 1-16. File directories for configuration

WE711 / ZE7111.0

### Notes:

Directories that are commonly accessed during development are in bold.

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  |
|--------------------|---|--|
| <b>cert:</b>       | Per application domain; not shared                      | Location to store private keys and digital certificates <ul style="list-style-type: none"> <li>• System automatically encrypts all files in this store</li> <li>• After being added, files cannot be copied or modified</li> <li>• You can delete digital certificates and private keys</li> </ul> |
| <b>sharedcert:</b> | <i>System-wide</i> ; shared between application domains | Stores digital certificates to be shared with business partners <ul style="list-style-type: none"> <li>• System automatically encrypts all files in this store</li> </ul>  |
| <b>pubcert:</b>    | <i>System-wide</i> ; 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"> <li>• System automatically encrypts all files in this store</li> <li>• Files cannot be modified, but they can be copied</li> </ul>                       |

© Copyright IBM Corporation 2015

Figure 1-17. File directories for security

WE711 / ZE7111.0

### Notes:

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



## File directories for logging

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

© Copyright IBM Corporation 2015

Figure 1-18. File directories for logging

WE711 / ZE7111.0

### 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   |
|-----------------------|---------------------------------------|---|
| <b>audit:</b>         | default domain                        | Stores the audit log<br>Available from the CLI in the default domain only   |
| <b>checkpoints:</b>   | Per application domain;<br>not shared | Contains the checkpoint configuration files   |
| <b>dpcert:</b>        | default domain                        | Encrypted directory that contains files that the appliance uses for processing<br>Available from CLI in the default domain only |
| <b>image:</b>         | default domain                        | Contains the primary and rollback firmware  |
| <b>tasktemplates:</b> | default domain                        | XSL files that the WebGUI uses  |

© Copyright IBM Corporation 2015

Figure 1-19. More file directories

WE711 / ZE7111.0

### Notes:

- 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 2015

Figure 1-20. Export service and object configurations

WE711 / ZE7111.0

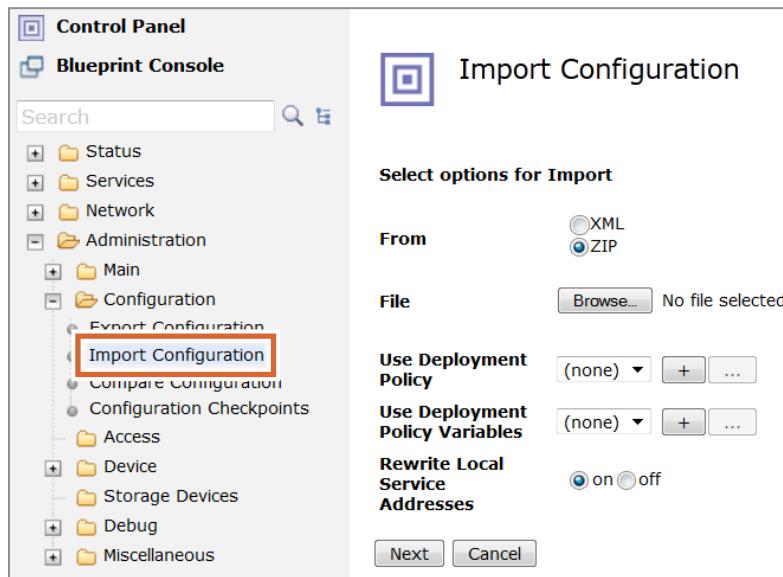
### Notes:

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.



## Import a 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
  - Can import a range of resources, from individual objects to multiple services



© Copyright IBM Corporation 2015

Figure 1-21. Import a configuration

WE711 / ZE7111.0

### Notes:

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

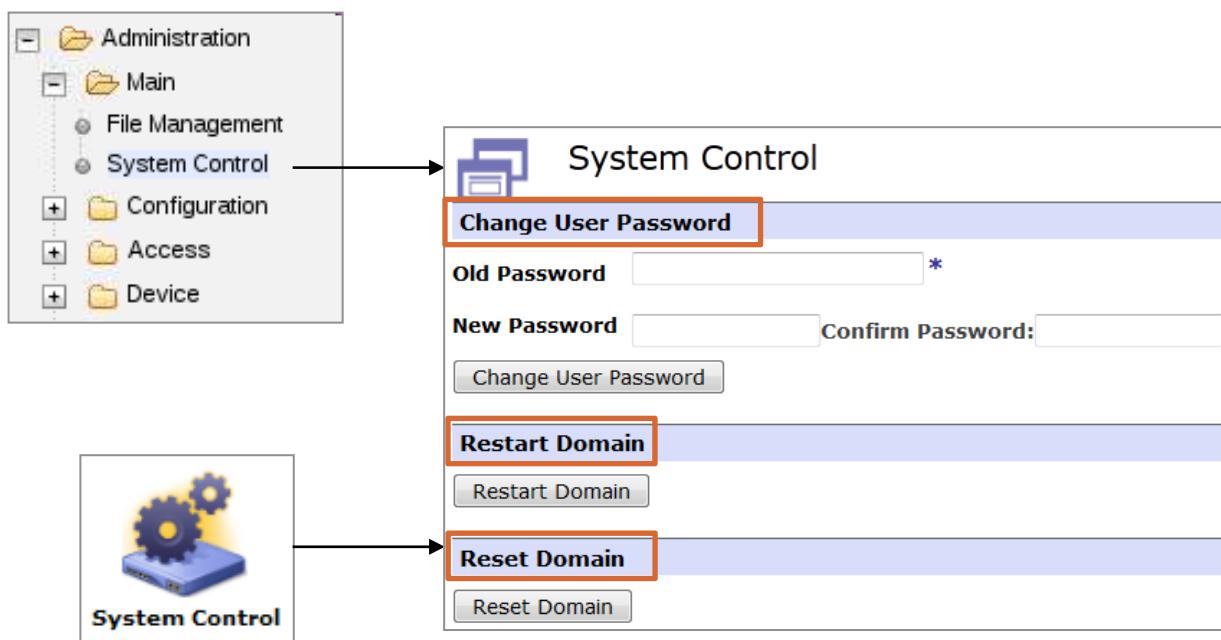
A deployment policy allows an imported configuration to be preprocessed, and certain properties to be modified.

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.



## System control features in an application domain



© Copyright IBM Corporation 2015

Figure 1-22. System control features in an application domain

WE711 / ZE7111.0

### Notes:

The System Control page in an application domain has only three functions. In the default domain, an administrator can use many more functions.

System Control can be accessed by clicking **Control Panel > Administration > Main > System Control**, or the **System Control** icon on the Control Panel.

You can change your own password.

**Restart Domain** restarts the domain from its last persisted configuration. This function is the configuration that is saved when **Save Configuration** is clicked.

**Reset Domain** is destructive. It deletes all of the objects that were created within the domain, including any services, and restarts the empty domain. The only resources that are retained are files in the `local:` directory.



## Globalization: Displaying other languages in WebGUI and the log

- Supported languages:
  - German
  - English
  - Spanish
  - French
  - Italian
  - Japanese
  - Korean
  - Brazilian Portuguese
  - Russian
  - Simplified Chinese
  - Traditional Chinese
- Language files are contained within the firmware
  - No language packs required

© Copyright IBM Corporation 2015

Figure 1-23. Globalization: Displaying other languages in WebGUI and the log

WE711 / ZE7111.0

### Notes:

The DataPower WebGUI and the system log and messages can be displayed in languages other than English.



## 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 default language
- Click **Administration > Device > Language**
  - Visible in default domain only
  - Set by administrator

Configure Language

Configuration successfully saved.

[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          |
| ru    | saved    | down     |      | disabled             | Russian             |
| zh_CN | saved    | down     |      | disabled             | Simplified Chinese  |
| zh_TW | saved    | down     |      | disabled             | Traditional Chinese |

© Copyright IBM Corporation 2015

Figure 1-24. Enabling languages

WE711 / ZE7111.0

### 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
- Example: German as the primary language in the browser

A screenshot of a web-based login form titled "WebSphere DataPower-Anmeldung". The page is in German. It contains fields for "Benutzername" (Username) and "Kennwort" (Password), a dropdown menu for "Domäne" (Domain) set to "default", and two buttons at the bottom: "Anmeldung" (Login) and "Abbrechen" (Cancel). Below the form, there is a copyright notice: "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." A red error message at the top left says: "\*Unvollständige Anmeldung. Bitte versuchen Sie es erneut." (Incomplete registration. Please try again.)

© Copyright IBM Corporation 2015

Figure 1-25. Getting the WebGUI to display an alternative language

WE711 / ZE7111.0

### Notes:

The language must be enabled in DataPower first.

## Getting an alternative language for the log and messages

- Click **Administration > Device > System Settings > System Locale**
  - Set by administrator in default domain
- 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

The screenshot shows two main windows. On the left, a modal dialog titled 'System locale' lists various language options: en (English), de (German), en (English) (selected), es (Spanish), fr (French), it (Italian), ja (Japanese), ko (Korean), pt\_BR (Brazilian Portuguese), ru (Russian), zh\_CN (Simplified Chinese), and zh\_TW (Traditional Chinese). On the right, a log viewer table displays four log entries:

| direction | client        | msgid      | message   |
|-----------|---------------|------------|---|
| 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              |

At the bottom right of the log viewer, there is a link: Show last 50 100 all.

© Copyright IBM Corporation 2015

Figure 1-26. Getting an alternative language for the log and messages

WE711 / ZE7111.0

### Notes:



## Unit summary

Having completed this unit, you should be able to:

- Log in to the WebGUI
- Navigate around the WebGUI interface
- Identify the primary functions of the menus on the navigation bar
- Start the creation of a DataPower service
- Identify the typical areas of a service configuration page
- Save configuration definitions in memory and on the file system
- List the file directories that are commonly used for development
- Support any non-English languages that are enabled on the appliance

© Copyright IBM Corporation 2015

Figure 1-27. Unit summary

WE711 / ZE7111.0

### 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. True or False: A user can access the WebGUI by using **http** or **https**, depending on how the administrator configures it.
3. Which directories are important to a developer and specific to an application domain?
  - A. **cert:**
  - B. **export:**
  - C. **image:**
  - D. **local:**
  - E. **sharedcert:**
  - F. **store:**

© Copyright IBM Corporation 2015

Figure 1-28. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here.

- 1.
- 2.
- 3.



## Checkpoint answers

- 1. True.**
- 2. False.** Access is always over **https**.
- 3. A. cert:, B. export:, and D. local:.**  
The **sharedcert:** and **store:** directories are shared.  
The **image:** directory is in the default domain only.

© Copyright IBM Corporation 2015

Figure 1-29. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 1



First exposure to the DataPower developer environment

© Copyright IBM Corporation 2015

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

10.1

Figure 1-30. Exercise 1

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Log in to the WebGUI
- Use the navigation bar
- Use an object catalog
- Import a service
- Edit a multi-protocol gateway
- Review the actions in a policy editor
- Test a service from a browser and a cURL command

© Copyright IBM Corporation 2015

Figure 1-31. Exercise objectives

WE711 / ZE7111.0

### Notes:

# Unit 2. Services overview

## What this unit is about

This unit describes the service types that are supported on the DataPower appliance. You examine, at a high level, what a service is and what it can communicate with. You also review the characteristics of each service type, and examine 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 DataPower appliance
- Describe the similarities and differences in the features that each DataPower service supports

## How you will check your progress

- Checkpoint

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)

## Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2015

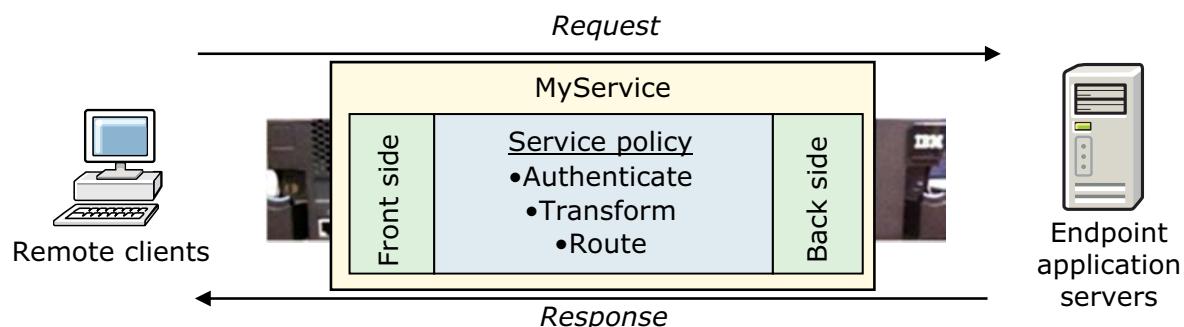
Figure 2-1. Unit objectives

WE711 / ZE7111.0

### 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

© Copyright IBM Corporation 2015

Figure 2-2. Services in a DataPower appliance

WE711 / ZE7111.0

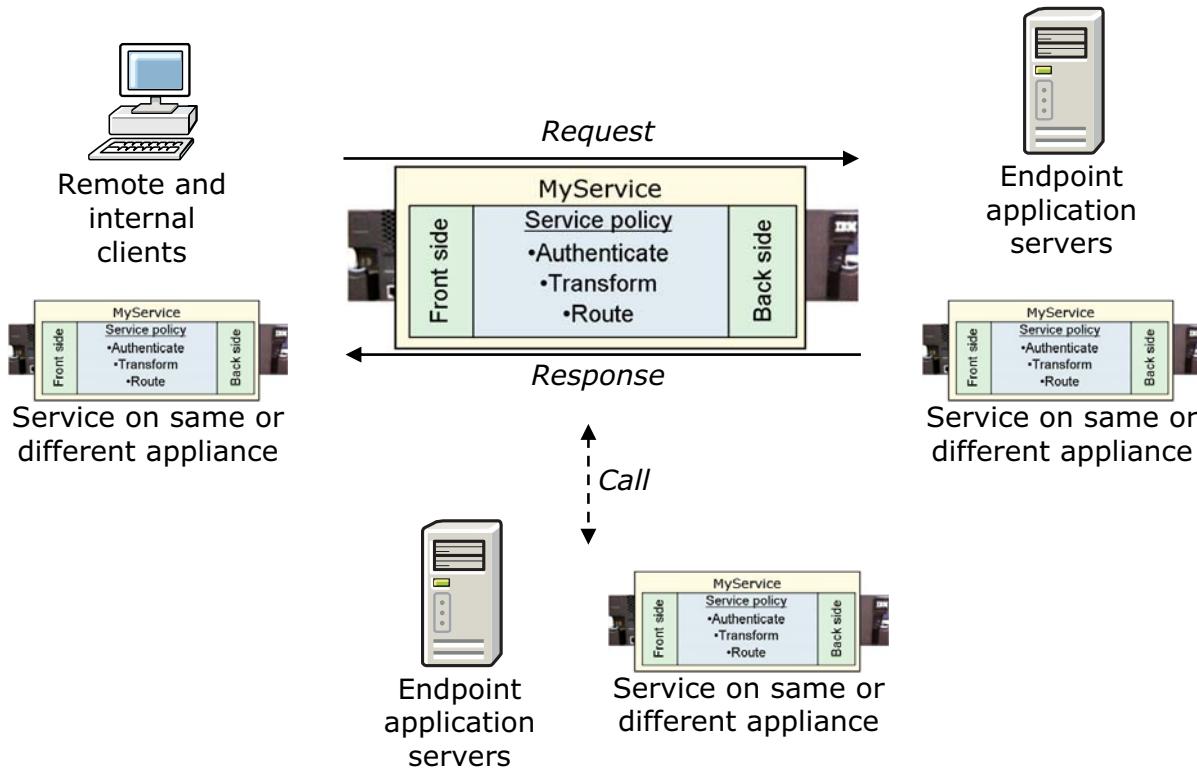
### Notes:

“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.

## Front sides and back sides, and sideways



© Copyright IBM Corporation 2015

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

WE711 / ZE7111.0

### Notes:

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

While executing a service policy, the service can call other services on the appliance or 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
- B2B Gateway
  - Supports specialized B2B message traffic between partners
- Access Manager Reverse Proxy
  - Secure web access to unified web servers
- Web application firewall (WAFW)
  - Secures and offloads processing from web-based applications
  - Threat mediation, AAA, and web-based validation

© Copyright IBM Corporation 2015

Figure 2-4. Services available on the DataPower appliance

WE711 / ZE7111.0

### Notes:

AAA is 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.

The logo for WebSphere Education, featuring the text "WebSphere Education" next to a blue arrow icon.

## 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 2015

Figure 2-5. XSL proxy service

WE711 / ZE7111.0

### 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, IBM DataPower Gateway



XML Firewall

© Copyright IBM Corporation 2015

Figure 2-6. XML firewall service

WE711 / ZE7111.0

### Notes:

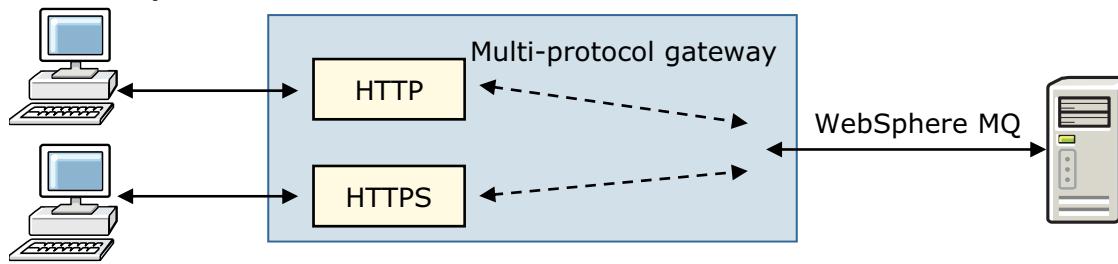
The features that are listed for the XML firewall are not exhaustive. The XML firewall also supports the same features that are mentioned previously for the XSL proxy.

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 actions that are associated with them and 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, IBM DataPower Gateway



© Copyright IBM Corporation 2015

Figure 2-7. Multi-protocol gateway service

WE711 / ZE7111.0

### 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.

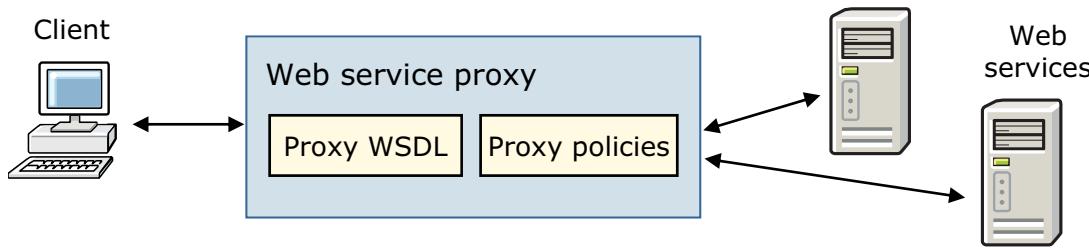
The supported protocols are HTTP, HTTPS, FTP, SFTP, NFS, raw XML, WebSphere MQ, WebSphere MQ File Transfer Edition (MQFTE), TIBCO EMS, WebSphere JMS, and IMS Connect.

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

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

## 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, IBM DataPower Gateway



© Copyright IBM Corporation 2015

Figure 2-8. Web service proxy service

WE711 / ZE7111.0

### 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 a DataPower variable within the service can be used to specify the same effect.

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.

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

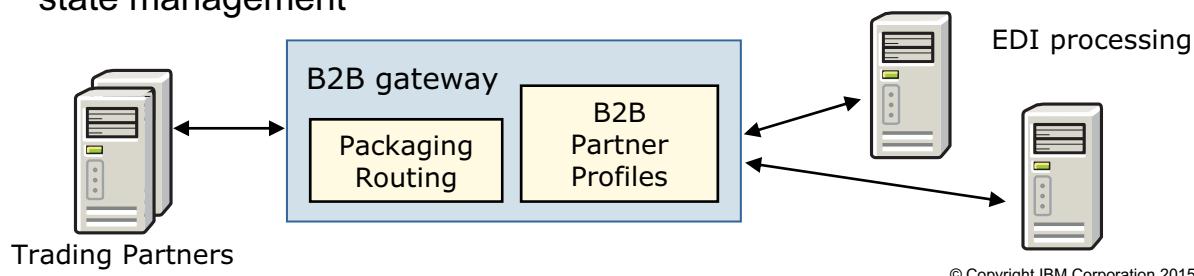
WebSphere Service Registry and Repository can “push” changes to a web service proxy.

## B2B gateway service

- Supports common B2B protocols
  - AS1, AS2, AS3, ebMS
- Handles different message body contents
  - XML, EDI ANSI X12, EDIFACT, Binary (non-XML, non-EDI)
- AS2 and AS3 packaging/unpackaging
- EDI, XML, and binary payload routing
- Works with B2B Partner Profiles that supports multiple destinations
- Non-repudiation of origin and receipt of all AS2 and AS3 messages
- Encrypted on-appliance document storage, metadata storage, B2B state management



B2B Gateway Service



© Copyright IBM Corporation 2015

Figure 2-9. B2B gateway service

WE711 / ZE7111.0

### Notes:

The DataPower B2B support adds additional protocol handlers:

- Applicability Statement 1 (AS1), AS2, AS3, ebXML Message Service (ebMS)

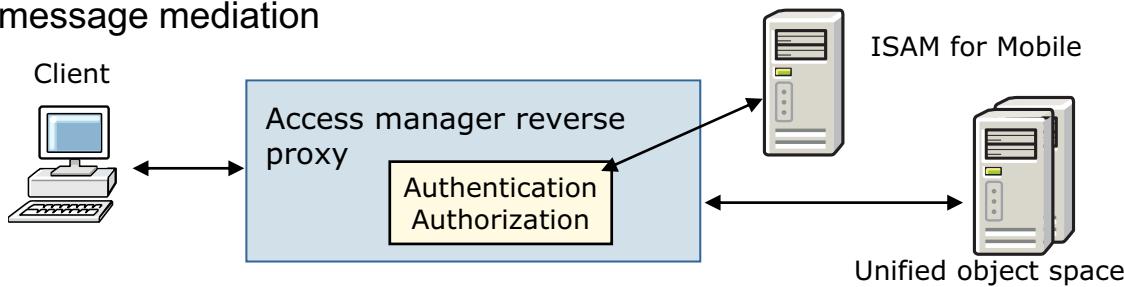
The B2B gateway service supports the following standards for message bodies:

- XML
- Electronic Data Interchange (EDI) ANSI X12
- Electronic Data Interchange for Administration, Commerce, and Transport (EDIFACT)
- Binary that is not XML or EDI.

This service is available on the XB62 and the IDG.

## Access manager reverse proxy

- Secures HTTP/HTTPS access to protected web resources
- Provides a unified object space of protected resources through a “junction” technology
- Integrates with IBM Security Access Manager (ISAM) for Web
  - Authentication, authorization, session management
- Integrates with ISAM for Mobile for advance access management use cases
  - One time password, multi-factor authentication, context-based access
- Can “chain” to other DataPower services so they can provide further message mediation



© Copyright IBM Corporation 2015

Figure 2-10. Access manager reverse proxy

WE711 / ZE7111.0

### Notes:

A junction combines the web space of the web server with the web space of the Access Manager Reverse Proxy. This results in a unified view of the entire web object space.

ISAM for Web and ISAM for Mobile provide their own optional reverse proxy web server, which is known as WebSEAL. The Access manager reverse proxy in DataPower provides the same functions that webSEAL does, but within DataPower itself.

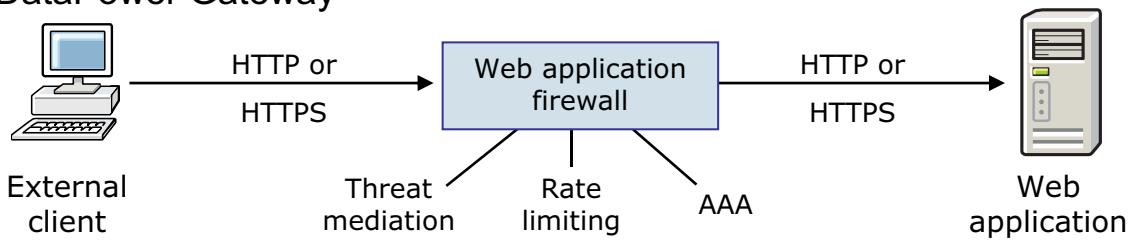
The resources that are supported and protected include URLs, URL-based regular expressions, CGI programs, HTML files, Java servlets, and Java class files.

If the incoming message needs further mediation, such as a transformation, the access manager reverse proxy can pass the message to another DataPower service like an MPGW to do that.

This service is available on the IDG only.

## 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, IBM DataPower Gateway



© Copyright IBM Corporation 2015

Figure 2-11. Web application firewall service

WE711 / ZE7111.0

### 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
- Four 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
  - Cloud Gateway Service: Creates a Cloud Gateway service, which can be used with Bluemix Cloud Integration

© Copyright IBM Corporation 2015

Figure 2-12. Other services

WE711 / ZE7111.0

### Notes:

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

The 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.

A Cloud Gateway service can be used with Bluemix Cloud Integration Services to secure traffic from cloud-based Bluemix applications to enterprise applications or data sources.

Each Cloud Gateway service must have one or more enterprise applications defined. When a cloud-based client establishes a connection to a Cloud Gateway service, the client identifies which enterprise application to connect to using its service name.

For more information, see the IBM Bluemix home page: <https://ace.ng.bluemix.net/>



## Which service type should you use?

- If you are WSDL and web services focused, choose the **web service proxy**
  - Present a single virtual WSDL to the clients that is composed of multiple WSDLs on the back end
  - Require different processing for the individual operations in the WSDLs
- If you are processing B2B messages with your trading partners, choose the **B2B gateway**
- If you require sophisticated authorization of clients by ISAM for protected web resources, select the **access manager reverse proxy**
- For general message processing/routing/authorization, select the **multi-protocol gateway**

© Copyright IBM Corporation 2015

Figure 2-13. Which service type should you use?

WE711 / ZE7111.0

### Notes:

The remaining service types are for more-specialized and less-frequent requirements. Sometimes these primary service types might call services of other types for utility functions.

## Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2015

---

Figure 2-14. Unit summary

WE711 / ZE7111.0

### 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 is the best choice 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 is the best choice 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 2015

Figure 2-15. Checkpoint questions

WE711 / ZE7111.0

### 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.** This service type 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 2015

Figure 2-16. Checkpoint answers

WE711 / ZE7111.0

### Notes:

# Unit 3. Structure of a service

## What this unit is about

Customers purchase DataPower appliances to provide application-related solutions. The key component that DataPower developers configure is a DataPower service. In this unit, you learn about the components that comprise a DataPower service, and the relationships between them. You learn about the front-side access, the back-side connection to the application server, and some of the service-wide settings. You also learn how to construct the service policy that 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 that 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
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## 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 that 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 2015

---

Figure 3-1. Unit objectives

WE711 / ZE7111.0

### 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

| Name  | Status | op-state | admin-state | Detail | logs |
|---|--------|----------|-------------|--------|------|
| Cloud Gateway Service                                 |        |          |             |        |      |
| HTTP Service  |        |          |             |        |      |
| License Agent   |        |          |             |        |      |
| Multi-Protocol Gateway                                |        |          |             |        |      |
| AddressRouter [Multi-Protocol Gateway]                | Saved  | up       | enabled     |        |      |
| CryptoMPGW [Multi-Protocol Gateway]                   | Saved  | up       | enabled     |        |      |
| EastAddressSearch [Multi-Protocol Gateway]            | Saved  | up       | enabled     |        |      |
| EastAddressSearchFromPattern [Multi-Protocol Gateway] | Saved  | up       | enabled     |        |      |
| EastAddressSearchMQclient [Multi-Protocol Gateway]    | Saved  | up       | enabled     |        |      |
| LocaleEnforcementServer [Multi-Protocol Gateway]      | Saved  | up       | enabled     |        |      |
| LocaleRetrievalMPGW [Multi-Protocol Gateway]          | Saved  | up       | enabled     |        |      |
| REST_EastAddressSearch [Multi-Protocol Gateway]       | Saved  | up       | enabled     |        |      |
| WestAddressSearch [Multi-Protocol Gateway]            | Saved  | up       | enabled     |        |      |
| SSL Proxy Service                                     |        |          |             |        |      |

© Copyright IBM Corporation 2015

Figure 3-2. Object-based configuration

WE711 / ZE7111.0

### 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 (MPG and others) have wizards

|                          |
|--------------------------|
| Objects                  |
| Network Settings         |
| Protocol Handlers        |
| Service Configuration    |
| XML Processing           |
| JSON Processing          |
| Web Services             |
| Policy Configuration     |
| Web Applications         |
| Monitoring               |
| Crypto Configuration     |
| Device Management        |
| Access Settings          |
| Configuration Management |
| Logging Configuration    |
| System Settings          |
| z/OS Configurations      |
| Secure Cloud Connector   |

The screenshot shows a configuration interface for a Multi-Protocol Gateway. At the top, there is a blue square icon followed by the text "Configure Multi-Protocol Gateway". Below this is a table with the following data:

| Multi-Protocol Gateway Name | Op-State | Logs | Type           | Req- |
|-----------------------------|----------|------|----------------|------|
| ArrivalToOps                | up       |      | Static Backend | XML  |
| BaggageServiceProxy         | up       |      | Static Backend | JSO  |

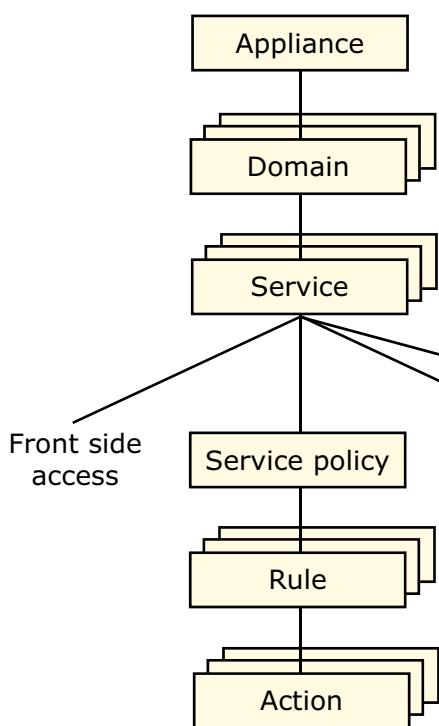
© Copyright IBM Corporation 2015

Figure 3-3. Approach to configuring objects

WE711 / ZE7111.0

## 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**, and **Results**
  - Custom XSLT is always available by using the **Transform** action

© Copyright IBM Corporation 2015

Figure 3-4. Basic architectural model

WE711 / ZE7111.0

### 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.

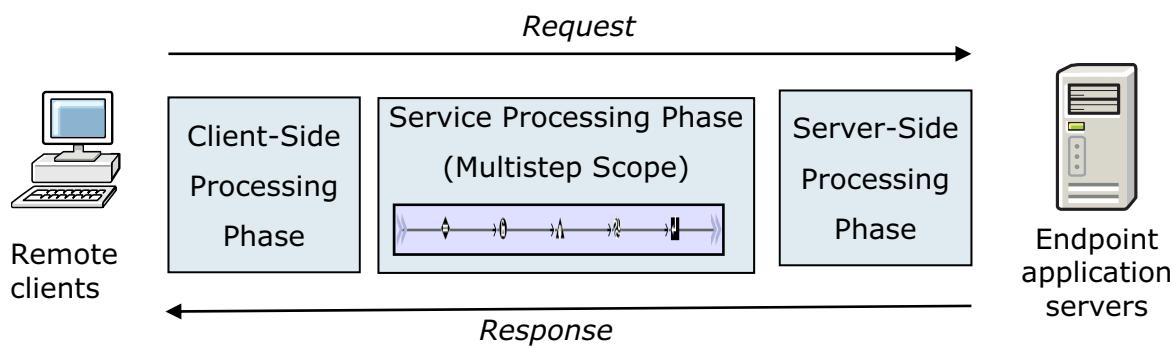
For a service, the configuration is divided between the front side access, connection to the back side, general service settings, and the service policy.

## Message processing phases

Each message passes through three phases:

1. Client side, front side: System throttle, listen for IP address or port, ACL, SSL, attachment processing, URL rewrite, HTTP header injection and suppression, and monitors
2. Service, service policy: Service traffic type (SOAP, XML, preprocessed, unprocessed), XML Manager, SOAP validation
3. Server side, back side: Streaming, URI propagation, user agent, SSL, load balancer, HTTP options

Service settings control client-side and server-side behavior



© Copyright IBM Corporation 2015

Figure 3-5. Message processing phases

WE711 / ZE7111.0

### Notes:

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

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 is 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.

## Client-side (Front) Processing Phase

- During this phase, the received message is directed to the service object that is configured for the IP address and port combination on which the message was received. After the service object (such as a Multi-protocol Gateway or XML Firewall) receives the message, a significant amount of processing of the message occurs.
- For example:
  - If SSL is configured for the service, SSL negotiation and decryption of the data stream occurs.
  - SOAP envelope validation.
  - Protocol-specific actions such as HTTP header suppression or injection.
  - Inspection for known XML threats.

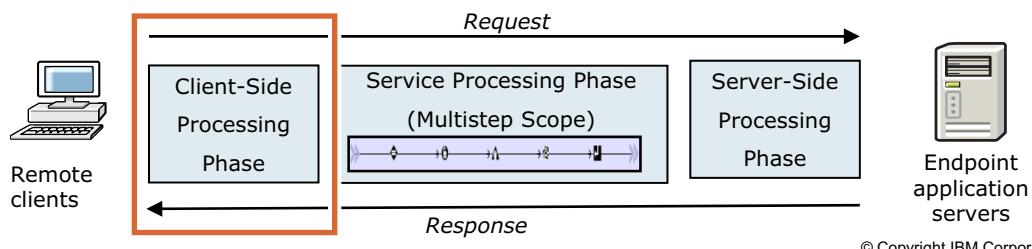


Figure 3-6. Client-side (Front) Processing Phase

WE711 / ZE7111.0

### Notes:



## 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

|  |
|--|
| <b>Front End</b>   |
| <b>Local address</b>   |
| <input type="text" value="dp_public_ip"/>                                  |
| <b>Port Number</b>   |
| <input type="text" value="6161"/> *  |
| <b>Reverse (Server) Crypto Profile</b>                                     |
| (none) <input type="button" value="+"/> <input type="button" value="..."/> |

|   |
|---|
| <b>Front side settings</b>                |
| <b>Front Side Protocol</b>                |
| MySSLSFH (HTTPS (SSL) Front Side Handler) |

|                               |
|-------------------------------|
| <b>Local</b>                  |
| <b>Local Endpoint Handler</b> |
| AddressSearchFSH              |

|                                   |                                  |
|-----------------------------------|----------------------------------|
| <b>Create a New:</b>              | <input type="button" value="X"/> |
| MEIG AS2 Proxy Front Side Handler |                                  |
| 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 Front Side Handler          |                                  |
| IMS Callout Front Side Handler    |                                  |
| IMS Connect 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 2015

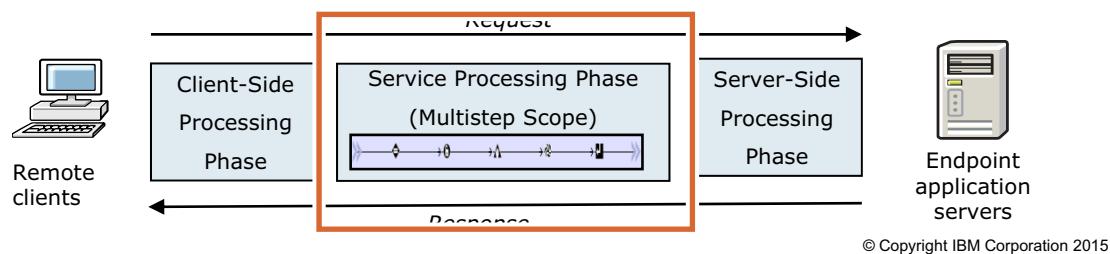
Figure 3-7. Front side access

WE711 / ZE7111.0

### Notes:

## Service Processing Phase

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



© Copyright IBM Corporation 2015

Figure 3-8. Service Processing Phase

WE711 / ZE7111.0

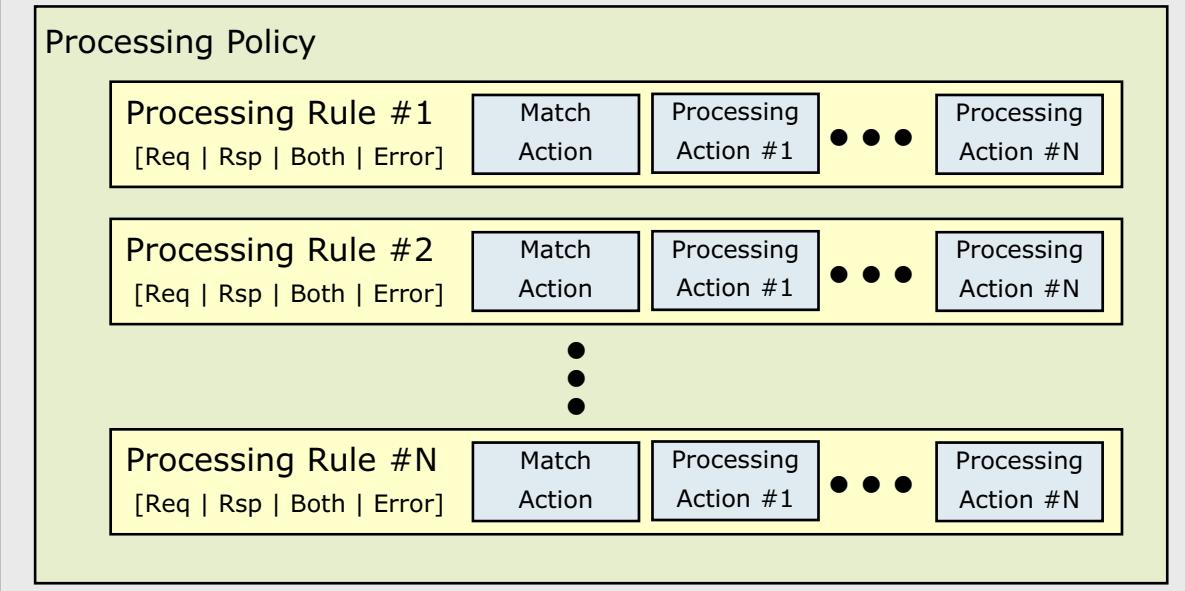
### Notes:

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

## Service Processing Phase

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

### Multi-Protocol Gateway



© Copyright IBM Corporation 2015

Figure 3-9. Service Processing Phase

WE711 / ZE7111.0

### Notes:

Each service that you configure has exactly one *Processing Policy*.

The processing policy defines what should happen when a message arrives from either the client (request), or the server (response).

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



## Match Action

- Evaluate statements by using: AND | OR message arrives from either the client (request), or the server (response).

Match Rule – evaluate statements by using AND | OR

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

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

●  
●  
●

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

© Copyright IBM Corporation 2015

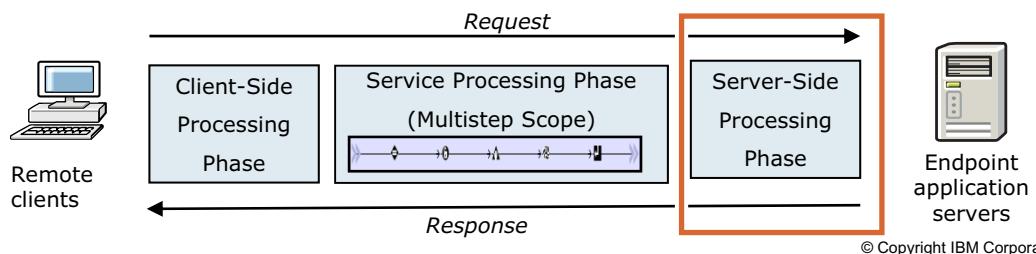
Figure 3-10. Match Action

WE711 / ZE7111.0

### Notes:

## Server-side (Back) Processing Phase

- If the message makes it to this phase, the message is accepted by the client-side phase and processed by the service phase. It's now ready to be sent to the backend server. Before sending though, some additional steps might be required. Those steps might include:
  - Establishing a new SSL connection to the back side server
  - Setting more headers in the request
  - Mediating protocol versions (that is, HTTP 1.1 to HTTP 1.0)
  - Other protocol-related tasks for WebSphere MQ, WebSphere JMS, FTP, NFS, and so on
- After all of the server-side processing is complete, the message is sent to the backend destination.



© Copyright IBM Corporation 2015

Figure 3-11. Server-side (Back) Processing Phase

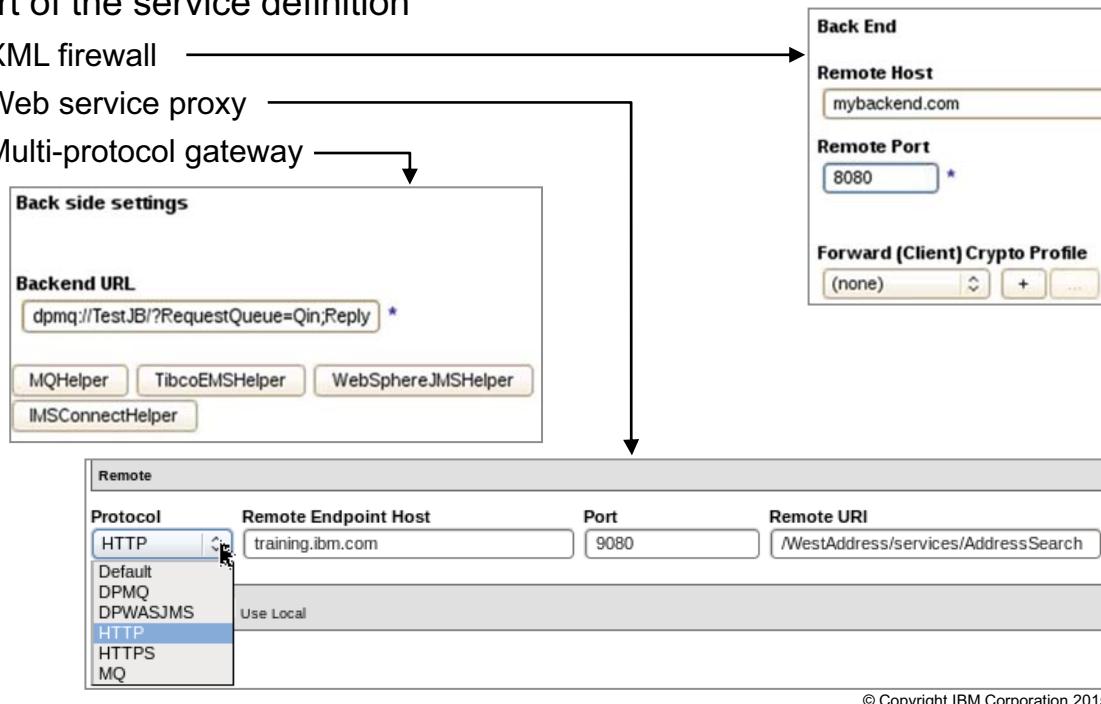
WE711 / ZE7111.0

### 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 2015

Figure 3-12. Connection to the back side

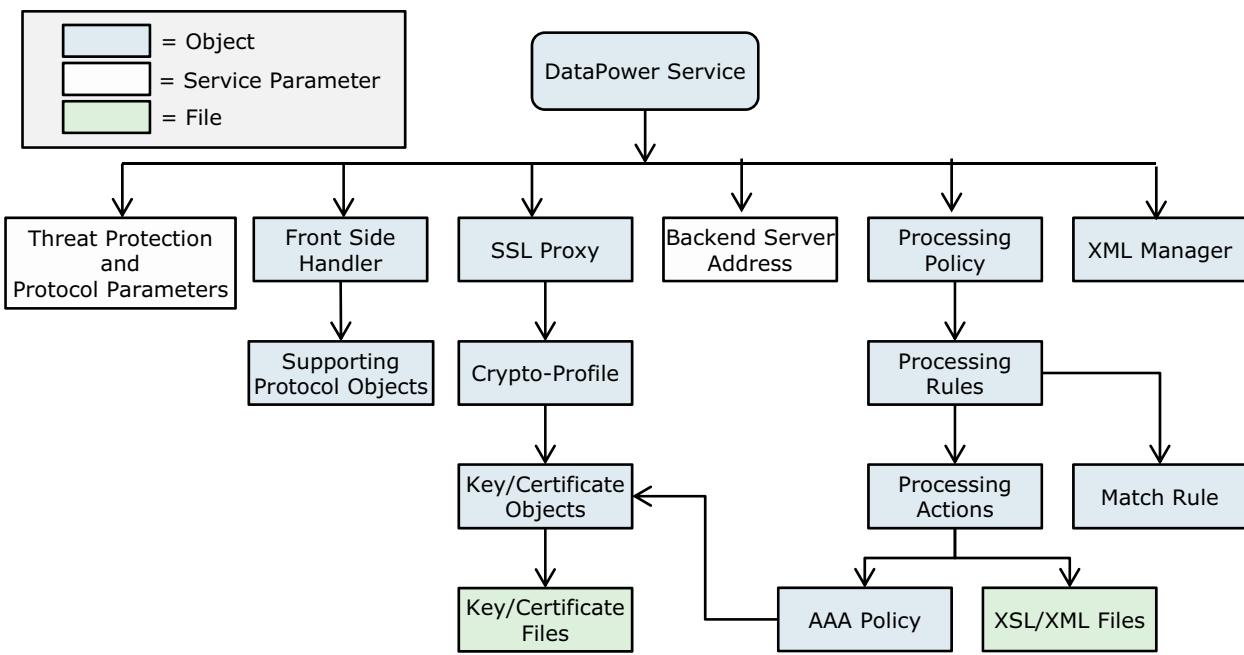
WE711 / ZE7111.0

### 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.

## DataPower Configuration Architecture

- A single DataPower appliance has the ability to host numerous service configurations.



© Copyright IBM Corporation 2015

Figure 3-13. DataPower Configuration Architecture

WE711 / ZE7111.0

### Notes:

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



## 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 is displayed 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 2015

Figure 3-14. Configuring a service policy (processing policy)

WE711 / ZE7111.0

### 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

Rule configuration area

CLIENT → [rule path] → ORIGIN SERVER

Configured Rules

| Order | Rule Name           | Direction        | Actions                   |
|-------|---------------------|------------------|---------------------------|
| ↑ ↓   | MyMPGWPolicy_rule_0 | Server to Client | [edit rule] [delete rule] |
| ↑ ↓   | MyMPGWPolicy_rule_1 | Client to Server | [edit rule] [delete rule] |

Configured rules area

© Copyright IBM Corporation 2015

Figure 3-15. Policy editor

WE711 / ZE7111.0

## 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.

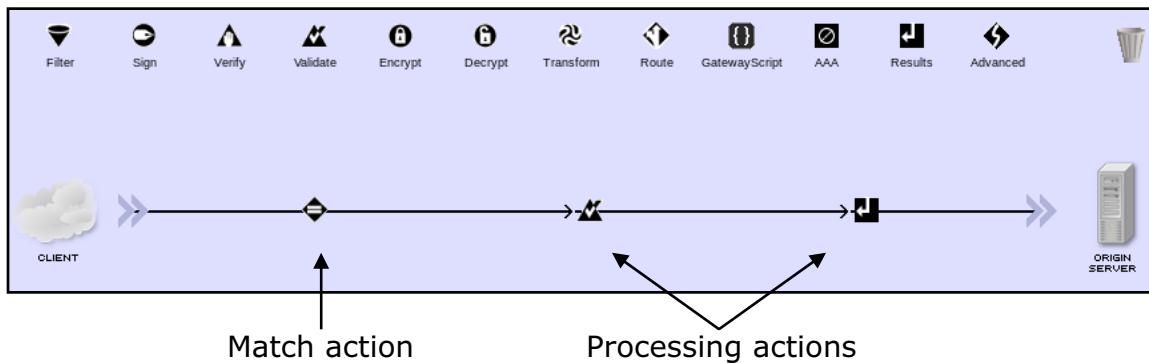
The rule configuration area is where rules are created, named, deleted, and configured. Action icons are dragged onto the rule configuration path.

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 2015

Figure 3-16. Configuring rules within a service policy

WE711 / ZE7111.0

### 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 software interface for managing rules. At the top, there's a header with 'Rule:' and fields for 'Rule Name' (set to 'LDAPTest\_request') and 'Rule Direction' (set to 'Client to Server'). Below this are buttons for 'New Rule' and 'Delete Rule'. 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 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 2015

Figure 3-17. Processing rules

WE711 / ZE7111.0

### 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.

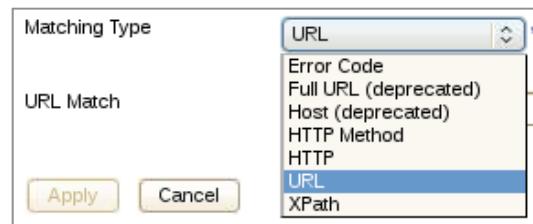


## Match action

- A **Match** action provides 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 2015

Figure 3-18. Match action

WE711 / ZE7111.0

### Notes:

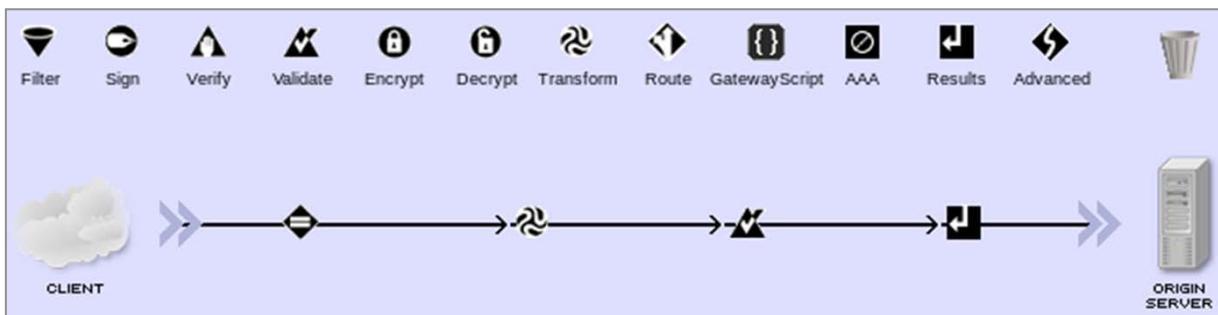
A **Match** action is used to define criteria that are matched against the incoming traffic to determine whether the actions configured in the rule are applicable.

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 2015

Figure 3-19. Processing actions

WE711 / ZE7111.0

### 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.



## 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         |
| Gateway Script | GatewayScript is a JavaScript-based run time for processing                         | GatewayScript |
| AAA            | Starts a AAA policy   | AAA           |
| Results        | Sends a message in specific context to an external destination                      | Results       |

© Copyright IBM Corporation 2015

Figure 3-20. Processing actions

WE711 / ZE7111.0

### Notes:

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

The **AAA** action is presented 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 does 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 (by 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 a service level monitor (SLM) 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 does 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 (that uses 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   |  |
|-------------|---|--|
| Advanced    | A grouping of lesser-used actions   |  Advanced |
| 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 2015

Figure 3-21. More processing actions

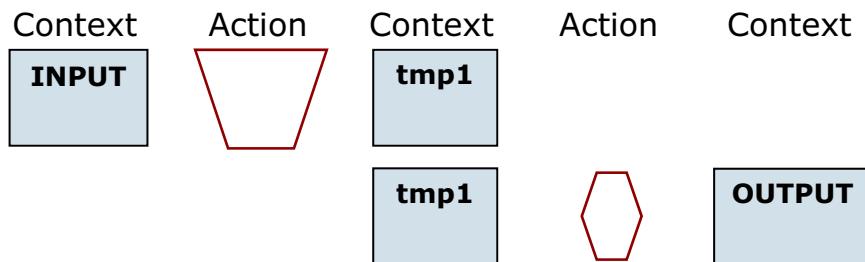
WE711 / ZE7111.0

### 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 2015

Figure 3-22. Multi-step processing rules

WE711 / ZE7111.0

### 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 appliance predefines the `INPUT` and `OUTPUT` context variables to represent the input and output messages.

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



## Predefined 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 2015

Figure 3-23. Predefined context variables

WE711 / ZE7111.0

### 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 with an `xsi:schemaLocation` attribute to locate an XML schema document
- **Validate Document via WSDL URL**
  - Uses an XML schema that is contained in a WSDL document
- **Validate Document with Encrypted Sections**
  - Uses a schema exception map object to validate a document with encrypted parts

Figure 3-24. Validate action for XML

WE711 / ZE7111.0

### Notes:

The **Validate** action is used to validate the schema of XML documents. The schema URL can reference either a local or a 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 4 of the IETF specification:

<http://tools.ietf.org/html/draft-zyp-json-schema-04>

© Copyright IBM Corporation 2015

Figure 3-25. Validate action for JSON

WE711 / ZE7111.0

### 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 2015

Figure 3-26. Transform action for XML

WE711 / ZE7111.0

### 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.



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

© Copyright IBM Corporation 2015

Figure 3-27. Transform action that uses XQuery (JSON and XML)

WE711 / ZE7111.0

### 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 as 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** validate against the schema. For 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 mapping to transform binary-to-XML, XML-to-binary, or binary-to-binary
  - Typically is used to mediate between XML-based clients and COBOL-based mainframe applications
- Transform binary
  - A WebSphere Transformation Extender mapping is used to transform the input document to the output document structure

**Transform binary**

|   |   |
|---|---|
| <b>Use Document Processing Instructions</b> | <input checked="" type="radio"/> Transform binary<br><input type="radio"/> Transform with a processing control file, if specified<br><input type="radio"/> Transform with embedded processing instructions, if available<br><input type="radio"/> Transform with XSLT style sheet |
| <b>Transform File</b>                       | local:/// <input type="button" value="..."/><br>(none) <input type="button" value="..."/> Upload... Fetch...<br>Var Builder   |
| <b>WTX Map file</b>                         | local:/// <input type="button" value="..."/><br>(none) <input type="button" value="..."/> Upload... Fetch... Edit... View... Var  |
| <b>WTX Map Mode</b>                         | DPA <input type="button" value="..."/>  |

- **WTX Map file**
  - File that contains the WebSphere Transformation Extender transformation instructions
- **WTX Map Mode**
  - Format of the WebSphere Transformation Extender mapping file

© Copyright IBM Corporation 2015

Figure 3-28. Transform action for binary transformations

WE711 / ZE7111.0

### Notes:

This version of the **Transform action** uses a WebSphere Transformation Extension 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 2015

Figure 3-29. Filter action

WE711 / ZE7111.0

### 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 'Filter' section, 'Action Type' is set to 'Filter' and 'Filter Method' is set to 'Replay Filter'. The 'Transform File' section contains 'store:///replay-filter.xsl' and 'Stylesheet Summary: Check for replay attacks'. Under 'Asynchronous', 'on' is selected. In the 'Replay Filter Type' section, 'WS-Addressing Message ID' is chosen. The 'Replay duration' is set to '600 sec'. There are 'Save' buttons for each configuration section.

- 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 2015

Figure 3-30. Filter action: Replay attack

WE711 / ZE7111.0

### 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 user name 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 user name 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.



## GatewayScript action (1 of 2)

- GatewayScript is a JavaScript-based run time for processing mobile, web, and API workloads
- Focuses on the “developer” experience, with familiar and friendly constructs and APIs
- Performance
  - Compiler technology and native execution
  - Built on intellectual capital and expertise from 10+ years of securing and optimizing XSLT parsing and compiler technology
  - Ahead of time compilation with caching, not single threaded
- Secure
  - Transaction isolation
  - Code injection protection
  - Short lived execution
  - Small footprint

© Copyright IBM Corporation 2015

Figure 3-31. GatewayScript action (1 of 2)

WE711 / ZE7111.0

### Notes:

Why use JavaScript?

- JavaScript is a widely used scripting language for large computer network environment.
- JavaScript is fast moving and community-driven, both client-side and server-side, and now gateway also.
- A gateway run time that is based on JavaScript simplifies configuration for developers and provides an easier development paradigm for mobile, web, and API.



## GatewayScript action (2 of 2)

- Easily manipulate JSON and binary data to transform payloads or create gateway functionality
- New Processing Policy Action
  - Transformation style processing policy action
  - Access to gateway functions through APIs

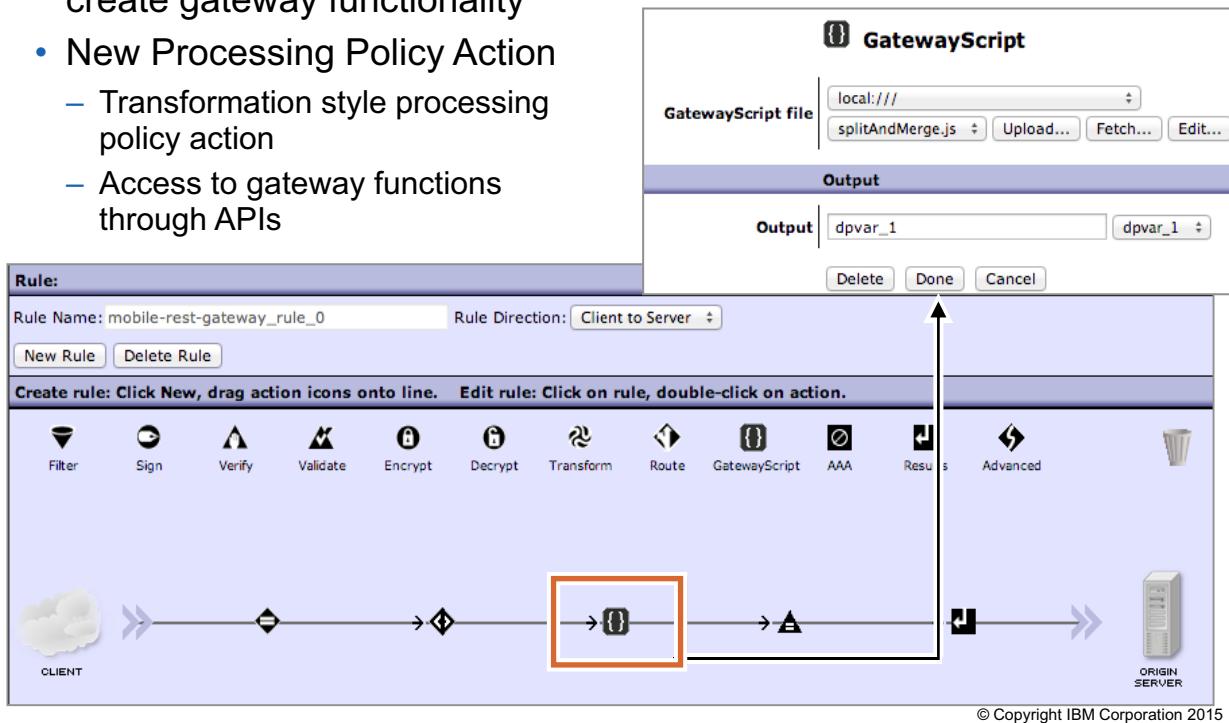


Figure 3-32. GatewayScript action (2 of 2)

WE711 / ZE7111.0

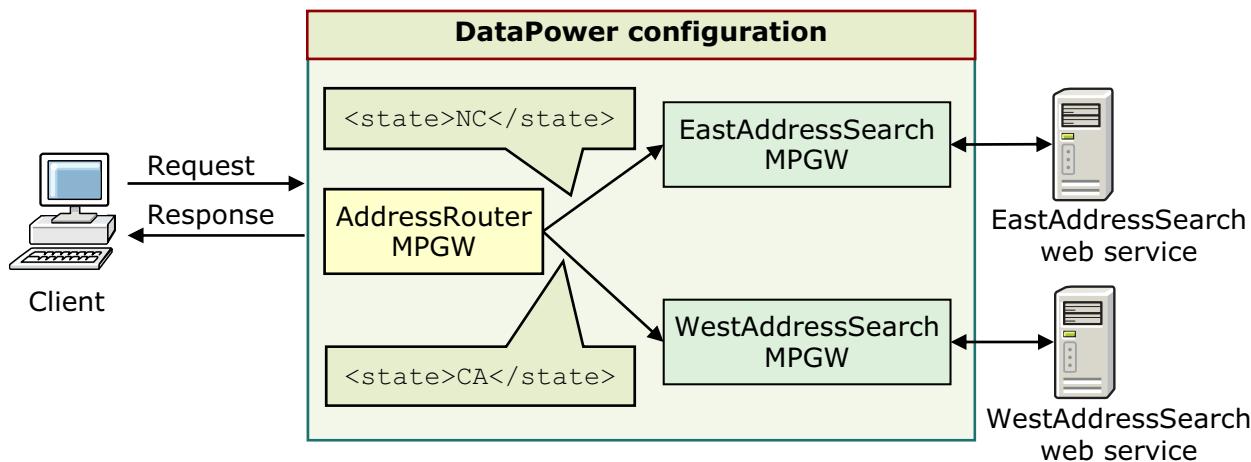
### Notes:

JSONiq or XQuery is not a general-purpose programming language; it is used for selectively finding information from payloads and transforming them.

JavaScript provides you the ability to apply control logic and grant access to gateway API functionality.

## 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 2015

Figure 3-33. Content-based routing

WE711 / ZE7111.0

### 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 2015

Figure 3-34. Route action configuration

WE711 / ZE7111.0

### Notes:

The XPath Routing Map is used 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 2015

Figure 3-35. Style sheet programming with dynamic routing

WE711 / ZE7111.0

### 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)` function is a common extension function that is used to extract an HTTP header from a message.

Here is an 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 `<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 2015

Figure 3-36. Results action

WE711 / ZE7111.0

### Notes:

The **Results** action is typically the last action in a rule because 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

| <b>Results Asynchronous</b> |  |
|-----------------------------|--|
| <b>Destination</b>          | <input type="text" value="http://"/> <input checked="" type="checkbox"/> |
| <b>Number of Retries</b>    | <input type="text" value="0"/>   |
| <b>Retry Interval</b>       | <input type="text" value="1000"/> msec                                   |

- 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

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

© Copyright IBM Corporation 2015

Figure 3-37. Results asynchronous and multi-way results mode

WE711 / ZE7111.0

### 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
  - And more
- Varies by service type

© Copyright IBM Corporation 2015

Figure 3-38. Service settings

WE711 / ZE7111.0

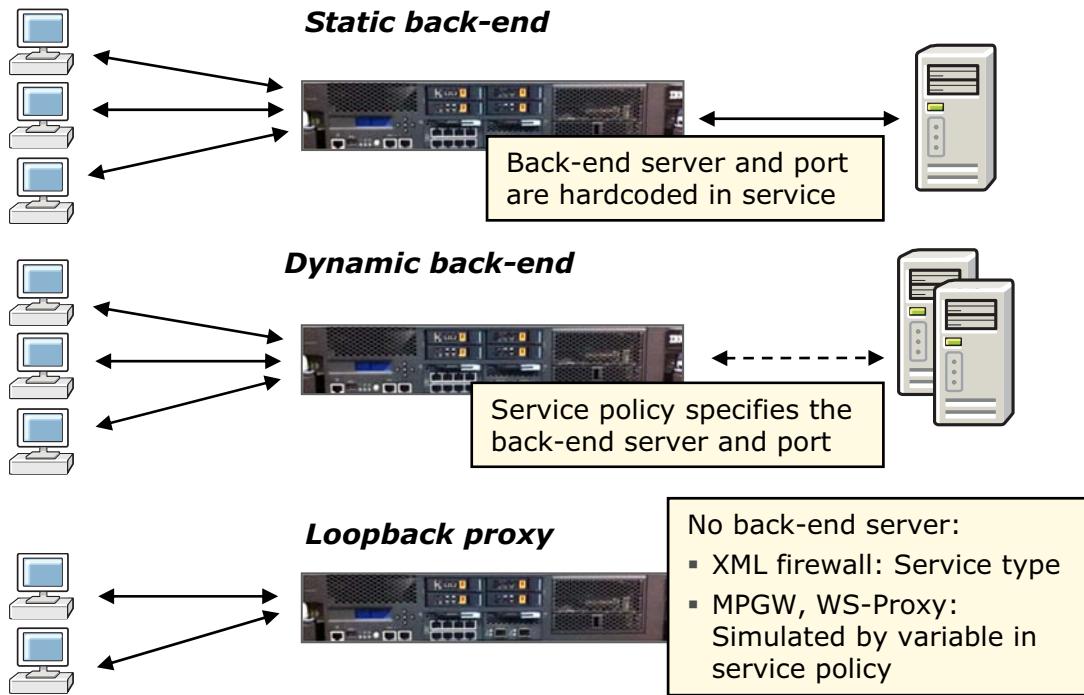
### Notes:

Traffic monitors are covered in another unit.

## Service types

### Remote clients

### Application servers



© Copyright IBM Corporation 2015

Figure 3-39. Service types

WE711 / ZE7111.0

### Notes:

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

The dynamic back end forwards traffic that is based on 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  | 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 |
| <input type="button" value="Save"/> <input type="button" value="Cancel"/> |   |

© Copyright IBM Corporation 2015

Figure 3-40. URL rewriting

WE711 / ZE7111.0

### 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 from 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 2015

Figure 3-41. XML Manager

WE711 / ZE7111.0

### 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.



## 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 checked="" type="radio"/> on <input 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 2015

Figure 3-42. Default XML Manager configuration

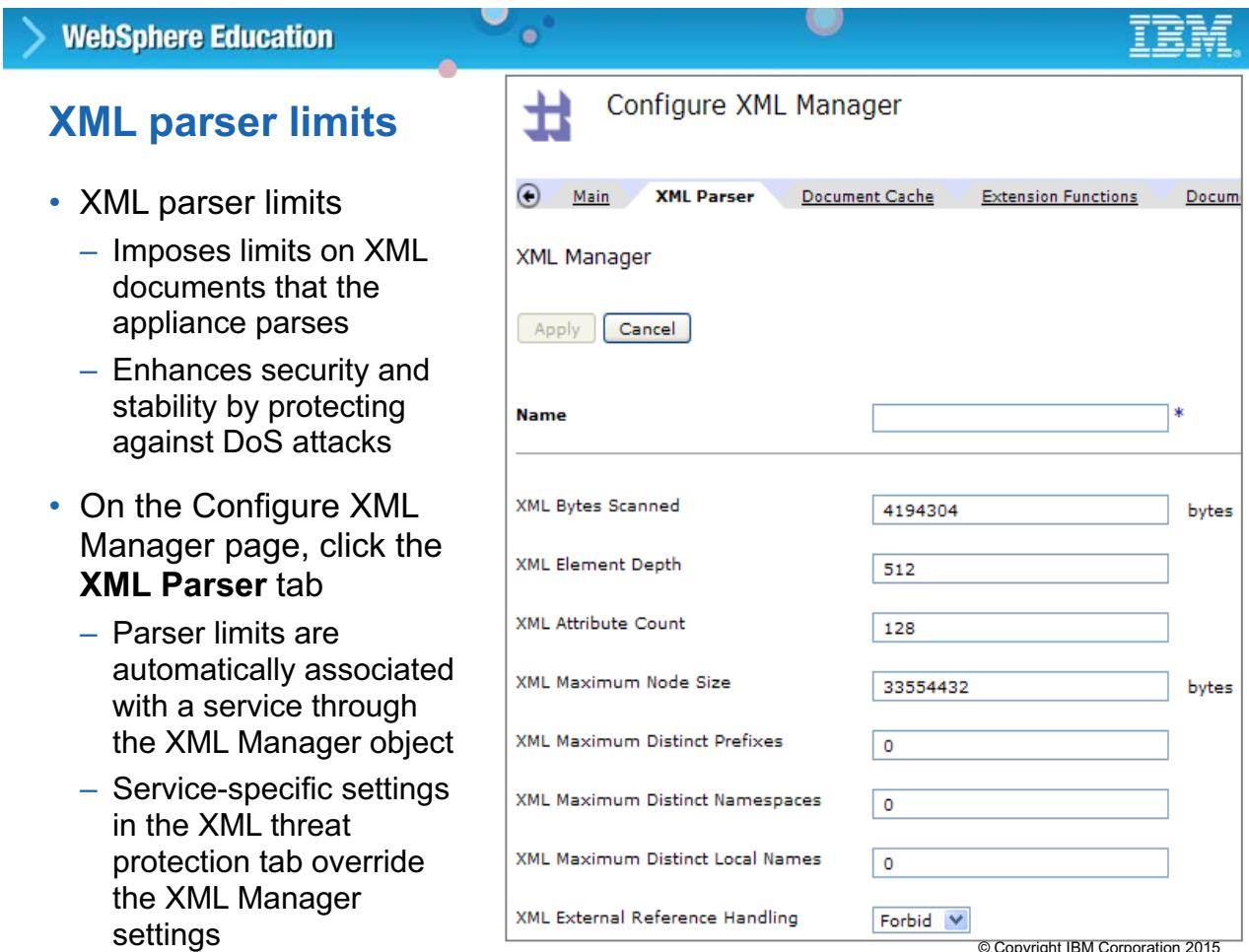
WE711 / ZE7111.0

### 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 other options.



The screenshot shows the 'Configure XML Manager' page in the WebSphere Education interface. The top navigation bar includes 'WebSphere Education' and the IBM logo. The main content area has a title 'XML parser limits' and a list of steps:

- XML parser limits
  - Imposes limits on XML documents that the appliance parses
  - Enhances security and stability by protecting against DoS attacks
- On the Configure XML Manager page, click the **XML Parser** tab
  - Parser limits are automatically associated with a service through the XML Manager object
  - Service-specific settings in the XML threat protection tab override the XML Manager settings

The 'XML Parser' tab is selected in the navigation bar. The configuration form contains the following fields:

| Name                             | Value    | Type     |
|----------------------------------|----------|----------|
| 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 form, there is a copyright notice: © Copyright IBM Corporation 2015.

Figure 3-43. XML parser limits

WE711 / ZE7111.0

## Notes:

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

“DoS” is “denial of service.”

XML Parser limits are as follows:

- **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 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.
- **XML External Reference Handling:** To allow references in DTD to URLs outside the appliance.



## JSON document limits within the XML manager

JSON Settings

**Name**

**General**

Administrative State  enabled  disabled

Comments

**Label-Value pairs**

|                                  |                                   |            |
|----------------------------------|-----------------------------------|------------|
| Maximum label length             | <input type="text" value="256"/>  | characters |
| Maximum value length for strings | <input type="text" value="8192"/> | characters |
| Maximum value length for numbers | <input type="text" value="128"/>  | characters |

**Threat Protection**

|                       |                                      |        |
|-----------------------|--------------------------------------|--------|
| Maximum nesting depth | <input type="text" value="64"/>      | levels |
| Maximum document size | <input type="text" value="4194304"/> | bytes  |

- 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

© Copyright IBM Corporation 2015

Figure 3-44. JSON document limits within the XML manager

WE711 / ZE7111.0

### 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 are as follows:

- **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. Notice 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.

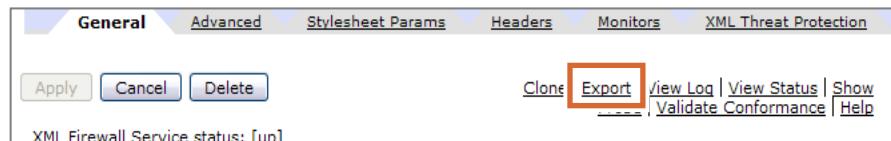
If no JSON Settings object is associated with a service's XML manager, the default values are in effect.

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 2015

Figure 3-45. Exporting a service configuration

WE711 / ZE7111.0

### 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 also an Import Configuration.

 WebSphere Education 

## 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:  Filter:

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 2015

Figure 3-46. Troubleshooting a service configuration

WE711 / ZE7111.0

### 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 that 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 2015

Figure 3-47. Unit summary

WE711 / ZE7111.0

### 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 2015

Figure 3-48. Checkpoint questions

WE711 / ZE7111.0

### 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 2015

Figure 3-49. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 2



Creating a BookingService gateway

© Copyright IBM Corporation 2015

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

10.1

Figure 3-50. Exercise 2

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Create a multi-protocol gateway
- Test the message flow by using the SoapUI graphical interface tool

© Copyright IBM Corporation 2015

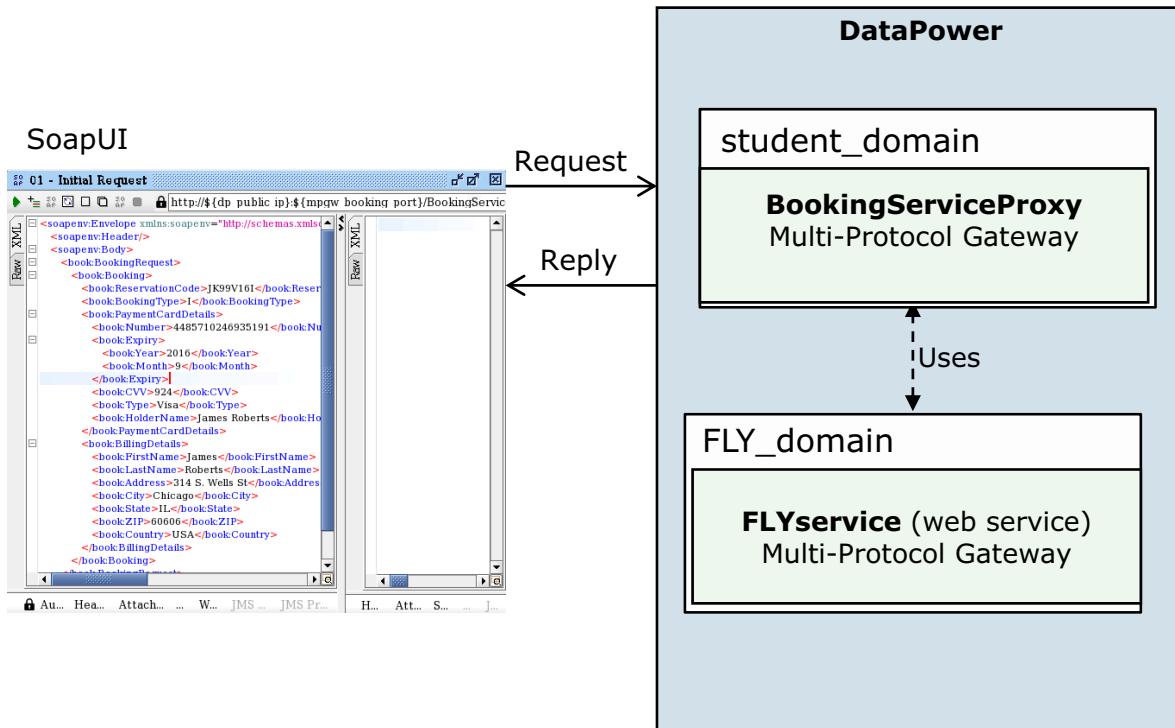
Figure 3-51. Exercise objectives

WE711 / ZE7111.0

### Notes:



## Exercise overview



© Copyright IBM Corporation 2015

Figure 3-52. Exercise overview

WE711 / ZE7111.0

### Notes:

# Unit 4. Multi-protocol gateway service

## What this unit is about

This unit describes the features of the multi-protocol gateway in the DataPower Gateway 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 connection to a dynamic back-end by use of a processing rule to select a back-end service at run time

## How you will check your progress

- Checkpoint

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)

## 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 connection to a dynamic back-end by use of a processing rule to select a back-end service at run time

© Copyright IBM Corporation 2015

Figure 4-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## What is a multi-protocol gateway?



A multi-protocol gateway (MPGW) 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 2015

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

WE711 / ZE7111.0

### Notes:

## Conceptual architecture of a multi-protocol gateway

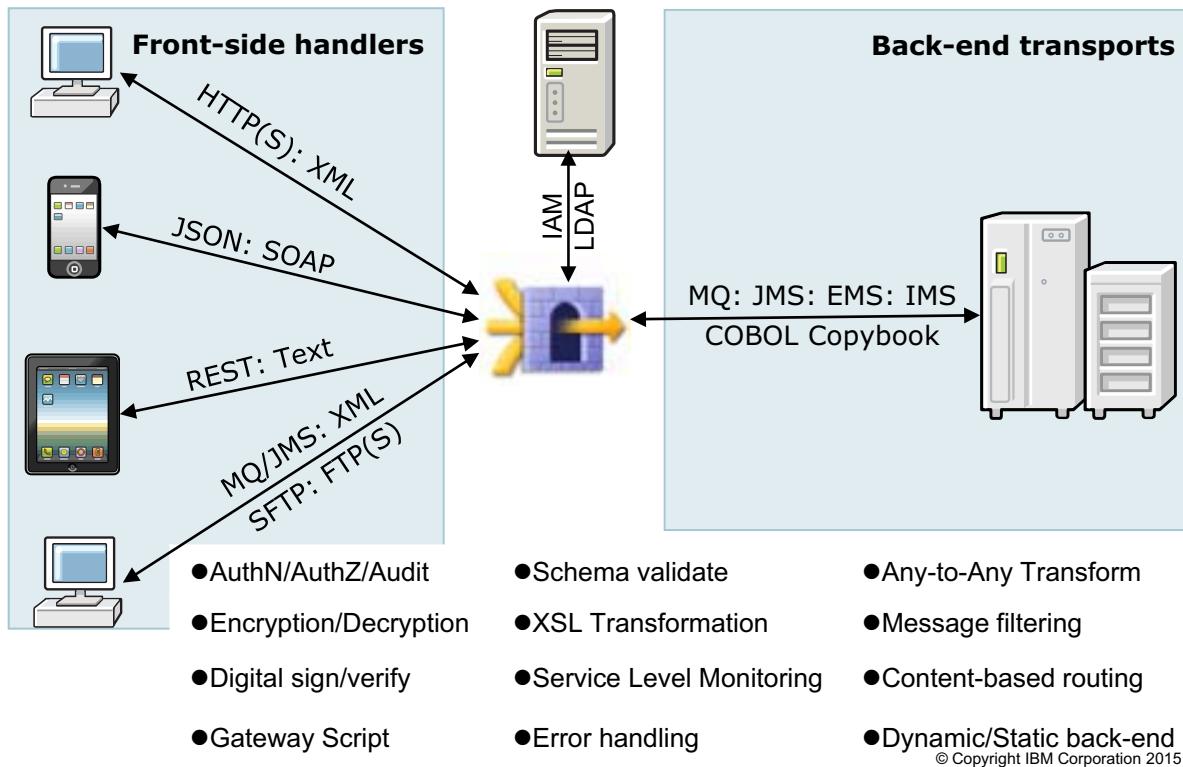


Figure 4-3. Conceptual architecture of a multi-protocol gateway

WE711 / ZE7111.0

### Notes:

The Multi-Protocol Gateway service builds on the XML Firewall's XML and security functionality by adding support for multiple protocols. In addition to HTTP and HTTPS, the Multi-Protocol Gateway supports MQ, JMS, Tibco EMS, FTP(S), SFTP, NFS, and IMS. All of these protocols can be mixed and matched as necessary. For example, messages received over HTTPS can easily be routed to MQ or JMS.

## Protocol handlers at a glance (1 of 2)

| Handlers          | Description   |
|-------------------|---|
| HTTP              | <ul style="list-style-type: none"> <li>Supports GET and POST operations</li> <li>The POST operations payload might contain XML, SOAP, DIME, SOAP with attachments, or MTOM</li> </ul> |
| 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  |
| MQ                | Places and retrieves messages on GET and PUT queues from an MQ system   |
| TIBCO EMS         | Supports the TIBCO Enterprise Message Service product   |

© Copyright IBM Corporation 2015

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

WE711 / ZE7111.0

### Notes:

MTOM is 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 of MQ messaging, review the MQ white paper, “The continuing benefits of commercial messaging.” This paper can 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  |
| JMS                         | Processes JMS messages received from Application Server                                  |
| IMS Connect,<br>IMS Callout | Accepts incoming IMS protocol requests and can initiate IMS connections on the back side |

© Copyright IBM Corporation 2015

Figure 4-5. Protocol handlers at a glance (2 of 2)

WE711 / ZE7111.0

### 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
  - MQ
  - 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 MQ protocol handler connects to an MQ queue manager and the associated PUT and GET queues
- Each JMS and TIBCO EMS handler connects to a JMS server and the associated GET and PUT queues

© Copyright IBM Corporation 2015

Figure 4-6. Front side protocol handlers

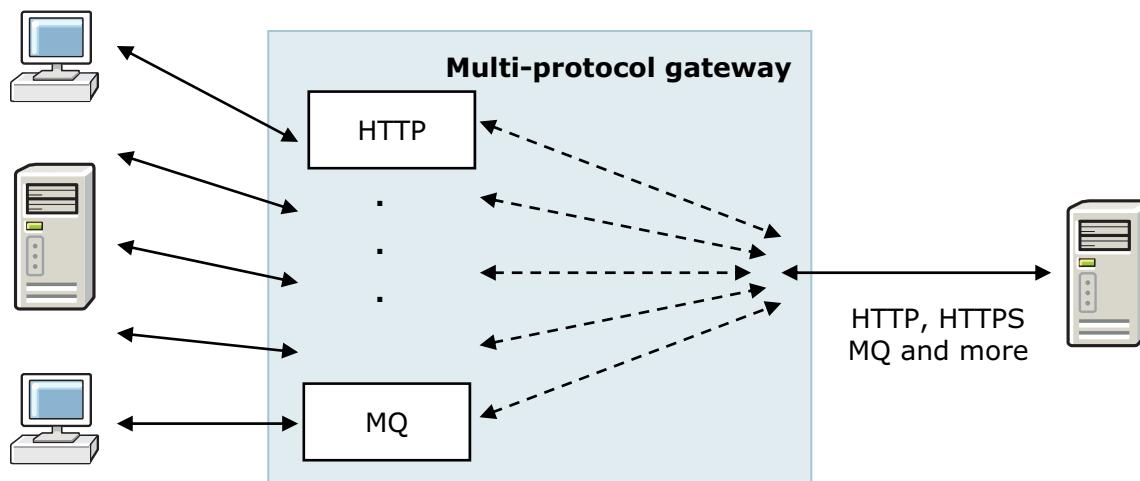
WE711 / ZE7111.0

### Notes:

Some protocol handlers appear only when you have the appropriate license on the appliance: for example, MQ, 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, MQ, or TIBCO EMS)



© Copyright IBM Corporation 2015

Figure 4-7. Static back-end gateway

WE711 / ZE7111.0

### Notes:

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

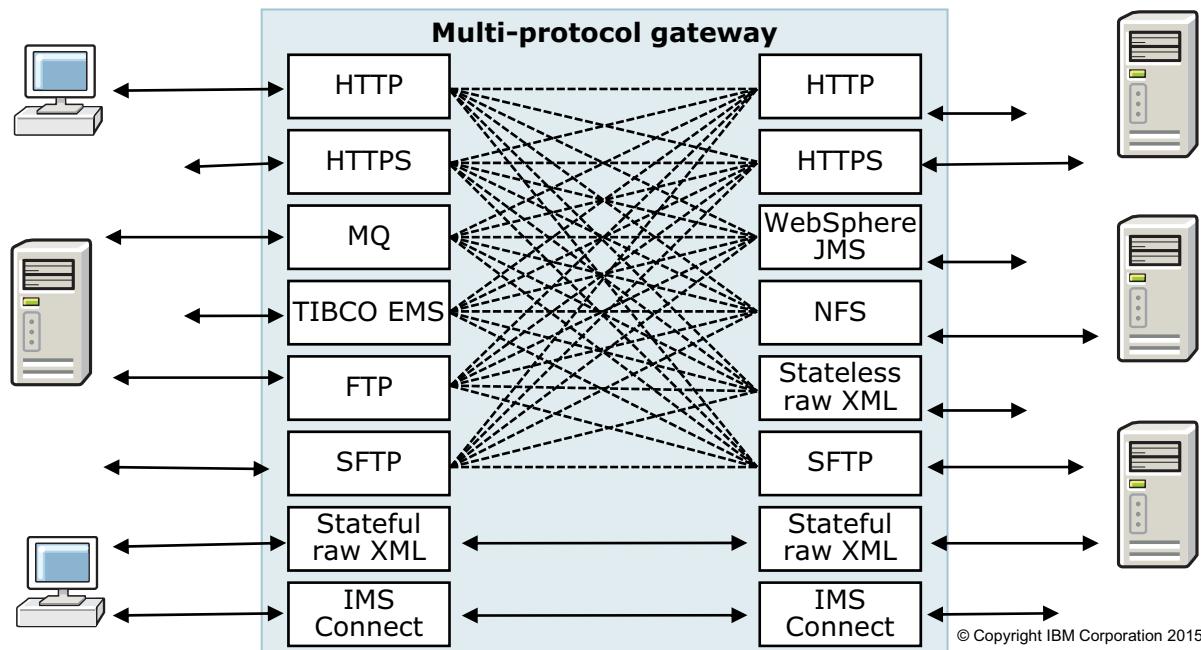


Figure 4-8. Dynamic back-end gateway

WE711 / ZE7111.0

### Notes:

The route action or a url-open within a style sheet specifies 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 2015

Figure 4-9. Multi-protocol gateway and XML firewall compared

WE711 / ZE7111.0

### 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 you had a scenario with a more realistic environment, the services would be implemented as multi-protocol gateways.

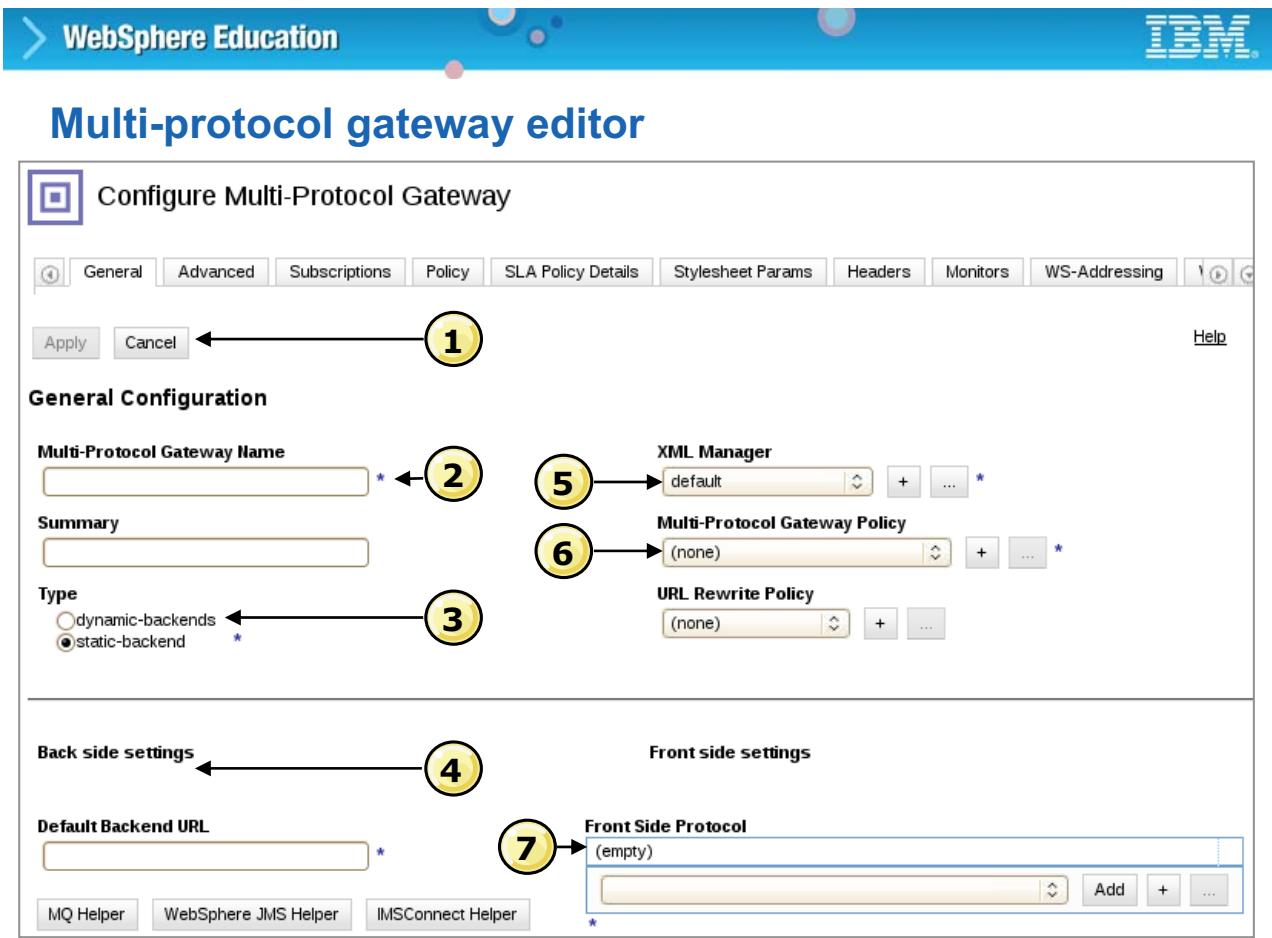


Figure 4-10. Multi-protocol gateway editor

WE711 / ZE7111.0

## 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, see the XML firewall presentation.

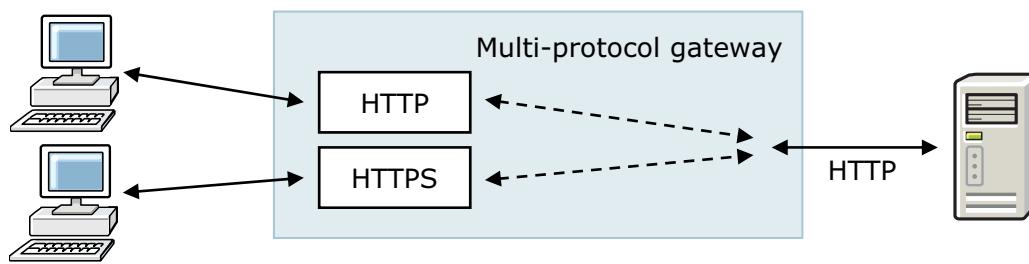
1. Remember to click **Apply** to commit changes that are made in the editor.
2. Specify a name and a description for the multi-protocol gateway.
3. Specify whether the back-end service URL is defined at design time (static back end) or defined at execution (dynamic back end). Keep in mind that the left side of the editor covers **Gateway to back-end** settings, while the right side covers **Client to gateway** settings.
4. For a static back end, enter the endpoint address for the back-end service.
5. 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.
6. 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.

7. 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 2015

Figure 4-11. Scenario 1: Provide HTTP and HTTPS access

WE711 / ZE7111.0

### Notes:

In this scenario, the client can access the back-end service over a regular HTTP connection or a secure HTTPS connection. The DataPower 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 DataPower 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 \*

**Summary**  
East Address Search MPG

**Type**  
 dynamic-backend  
 static-backend \*

**Back side settings**

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

MQHelper TibcoEMSHelper WebSphereJMSHelper  
IMSConnectHelper

**User Agent settings**

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

**SSL Client Crypto Profile**  
(none) + ...

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 2015

Figure 4-12. Step 1: Configure the back side transport

WE711 / ZE7111.0

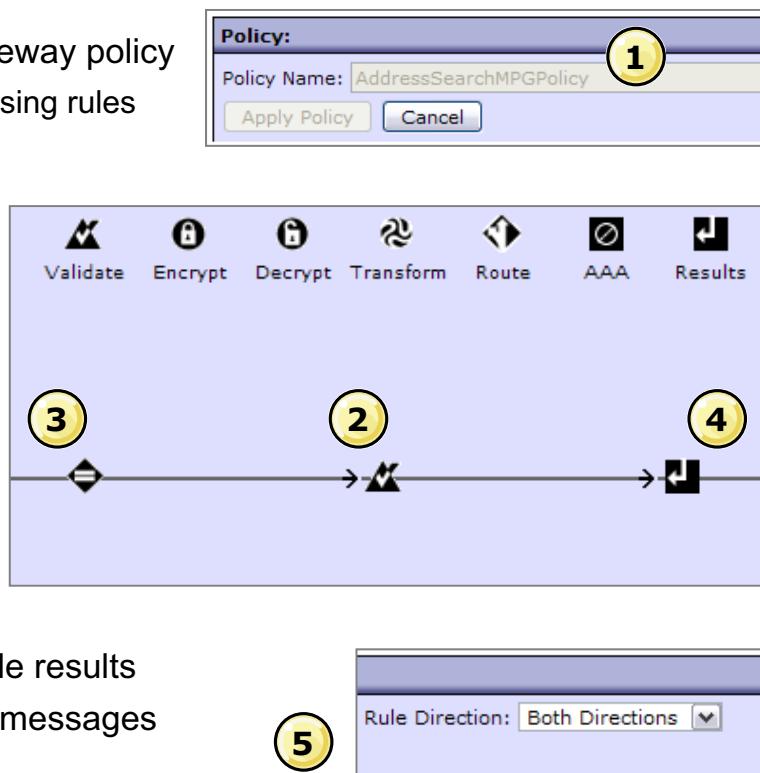
### Notes:

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
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



© Copyright IBM Corporation 2015

Figure 4-13. Step 2: Create a document processing rule

WE711 / ZE7111.0

### 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.

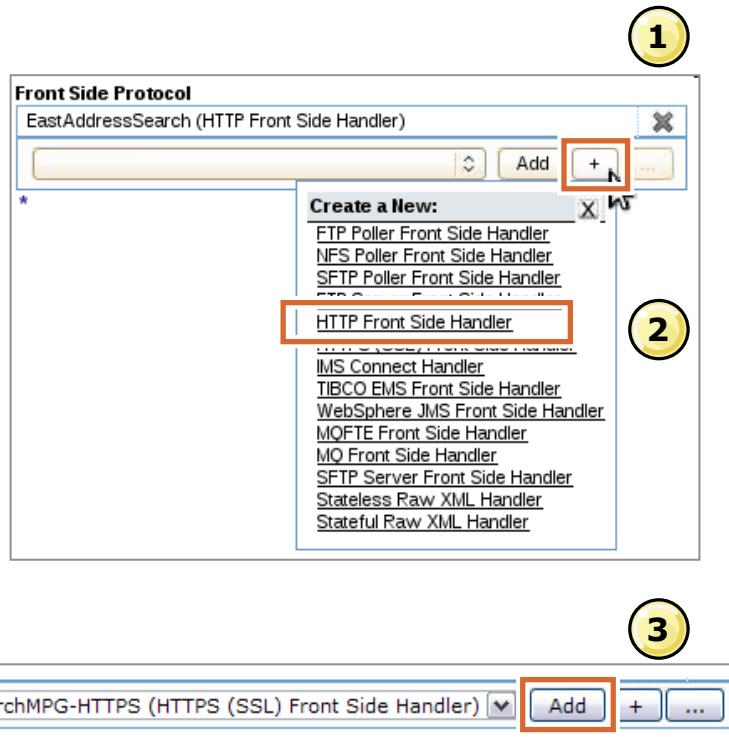
The **Match** action must be the first action on any processing rule. The **Validate** action appears *after* the match rule.

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 an explanation of 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



© Copyright IBM Corporation 2015

Figure 4-14. Step 3: Create the front side handlers

WE711 / ZE7111.0

### 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 **Administrative State** to **enabled**
2. Set the **Local IP Address** to **dp\_public\_ip** to listen to requests 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]

**1**  enabled  disabled

**2** Local IP Address: dp\_public\_ip

**3** Port Number: 6957

**4** HTTP Version to Client: HTTP 1.1

**5** Allowed Methods and Versions:  HTTP 1.0  HTTP 1.1  POST method

**6** SSL Proxy: AddressSSL (none)

© Copyright IBM Corporation 2015

Figure 4-15. Step 4: Configure the front side handler

WE711 / ZE7111.0

### 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 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 **Reverse (Server) Crypto Profile** with the certificate-key pair with which to secure the connection
4. Determine the settings for caching SSL sessions to clients

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

© Copyright IBM Corporation 2015

Figure 4-16. Step 5: Configure the SSL proxy profile

WE711 / ZE7111.0

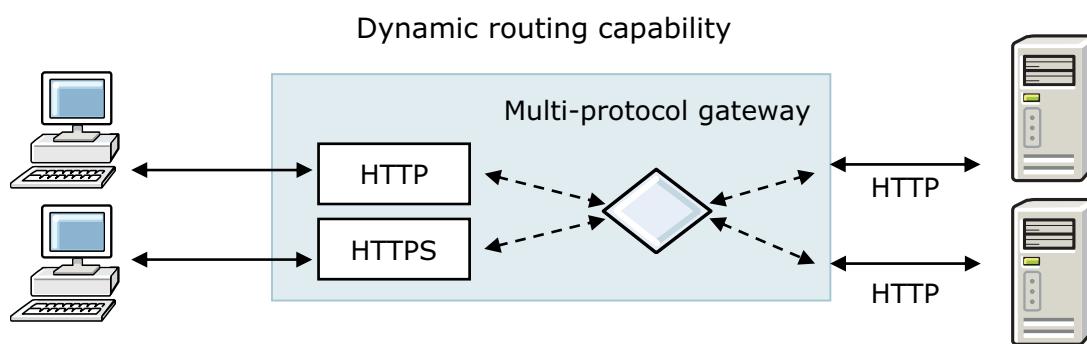
### Notes:

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 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 2015

Figure 4-17. Scenario 2: Dynamic back-end service

WE711 / ZE7111.0

### Notes:

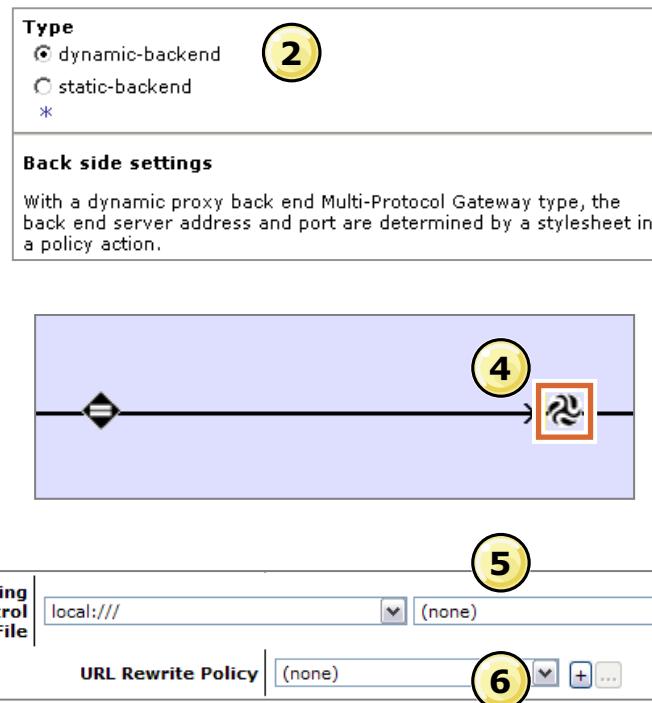
The dynamic back-end service allows one endpoint on the DataPower appliance to represent a single service, which is composed of different operations from different back-end services.

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.

WebSphere Education 

## 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



© Copyright IBM Corporation 2015

Figure 4-18. Step 1: Configure the back-end transport

WE711 / ZE7111.0

### 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 2015

Figure 4-19. Sample service that targets a style sheet

WE711 / ZE7111.0

### Notes:

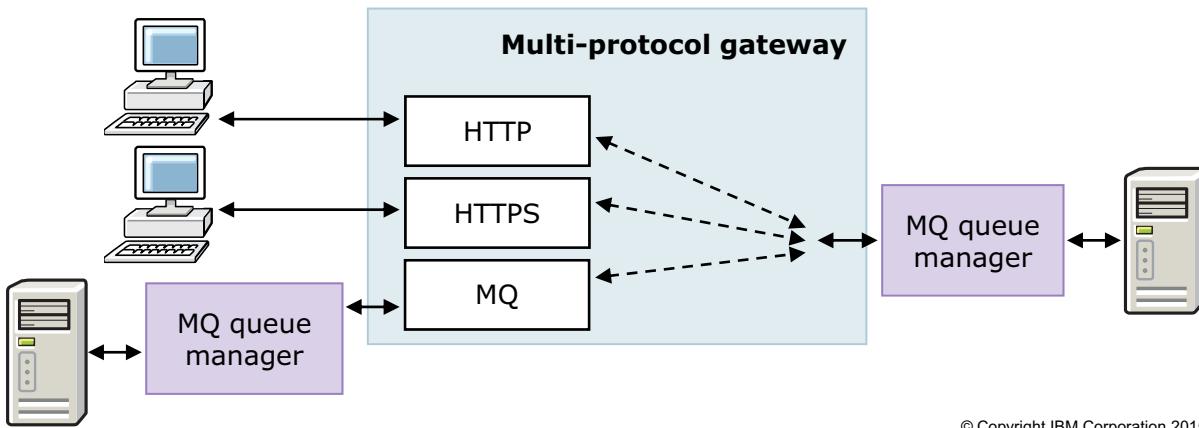
The `<dp:set-target>` built-in element defines the IP address (or host name) and the port for a particular back-end server. Other attributes are available to set up an SSL connection to the back-end service.

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 MQ access

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



© Copyright IBM Corporation 2015

Figure 4-20. Scenario 3: Provide MQ access

WE711 / ZE7111.0

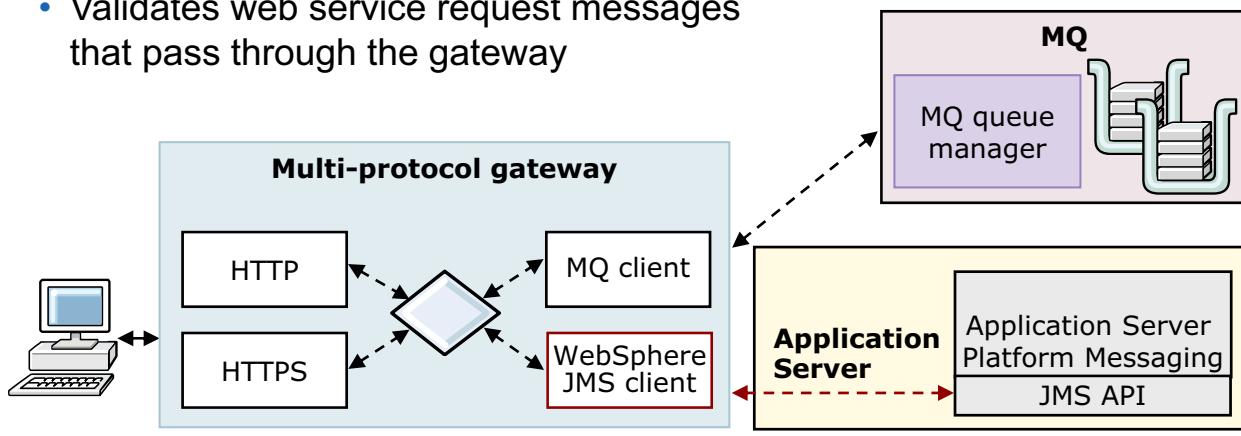
### Notes:

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

## Scenario 4: Provide JMS access

Modify the multi-protocol gateway so that it:

- Also uses the JMS API to forward requests to Application Server platform messaging system
  - Request and response messages reside in queues
  - Back-end Java Platform, Enterprise Edition web services poll queues to obtain messages
- Validates web service request messages that pass through the gateway



© Copyright IBM Corporation 2015

Figure 4-21. Scenario 4: Provide JMS access

WE711 / ZE7111.0

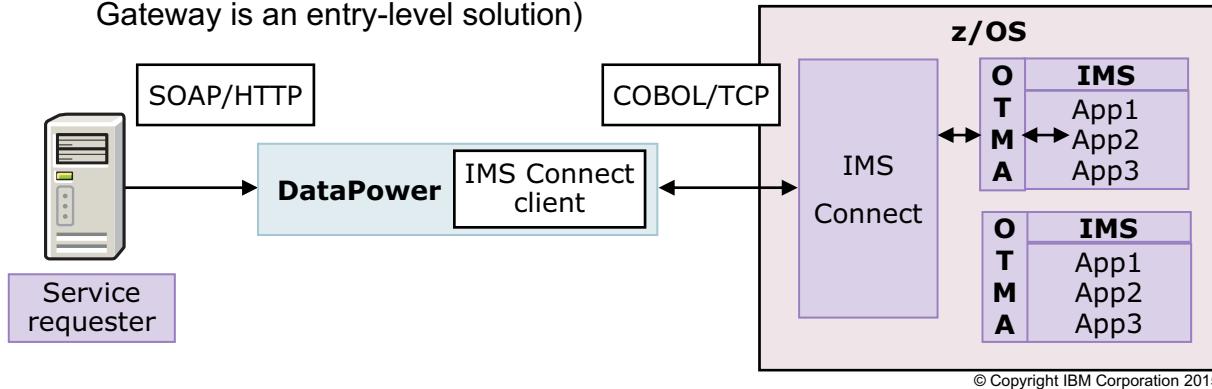
### Notes:

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

The 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 MQ requirement of WS-enablement of IMS
  - IMS has few alternatives (IMS SOAP Gateway is an entry-level solution)



© Copyright IBM Corporation 2015

Figure 4-22. Scenario 5: Provide IMS Connect access

WE711 / ZE7111.0

### Notes:

OTMA is Open Transaction Management Access.

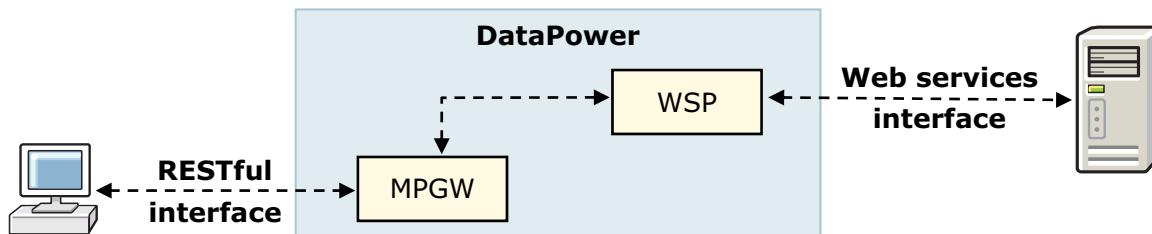
The IMS OTMA facility is a transaction-based connectionless client/server protocol that runs on IMS Version 5.1 or later. It functions as an interface for host-based communications servers that access IMS TM applications through the z/OS Cross Systems Coupling Facility (XCF).

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 2015

Figure 4-23. Scenario 6: Provide a RESTful interface

WE711 / ZE7111.0

### Notes:

For a presentation of REST and DataPower, see: *Implementing REST services with DataPower Appliances* at: [http://www.ibm.com/developerworks/websphere/techjournal/0903\\_peterson/0903\\_peterson.html](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 2015

Figure 4-24. REST support in DataPower

WE711 / ZE7111.0

### Notes:

**Process Messages whose body is empty**: 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 a 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.

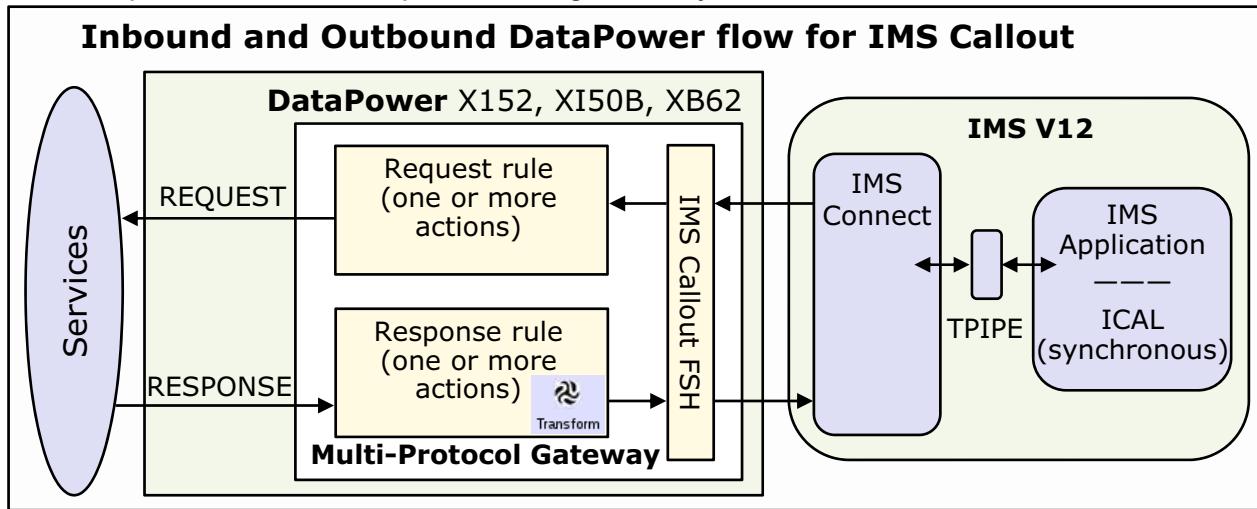
**Method Rewrite by Set Variable** action: Another way to rewrite the HTTP method.

**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: Another request and response type of JSON is added.

## Supported IMS features

- IMS support includes:
  - IMS Connect, IMS Synchronous Callout, and 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 create, retrieve, update, and delete operations, against any IMS DB



© Copyright IBM Corporation 2015

Figure 4-25. Supported IMS features

WE711 / ZE7111.0

### 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 MPGW, 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 that runs 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 MPGW policy, and sends it out to the back-end service. In a similar manner, the response is flown back and processed through the MPGW. 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 is sent back to the client after IMS processes the message

### DataPower support matrix

| Support type                                | DataPower models that support V6 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 2015

Figure 4-26. IMS Connect support

WE711 / ZE7111.0

### 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.

## WebSocket Proxy (1 of 2)

- WebSocket is a bidirectional frame-based protocol for enabling real-time communication over supporting HTTP or HTTPS infrastructure
  - Designed to enable real-time applications such as: Messaging over the WEB, Chat Applications, Video Applications, Notifications, and other applications
- Use DataPower to secure, route, shape, and load-balance initial WebSocket connection establishment



© Copyright IBM Corporation 2015

Figure 4-27. WebSocket Proxy (1 of 2)

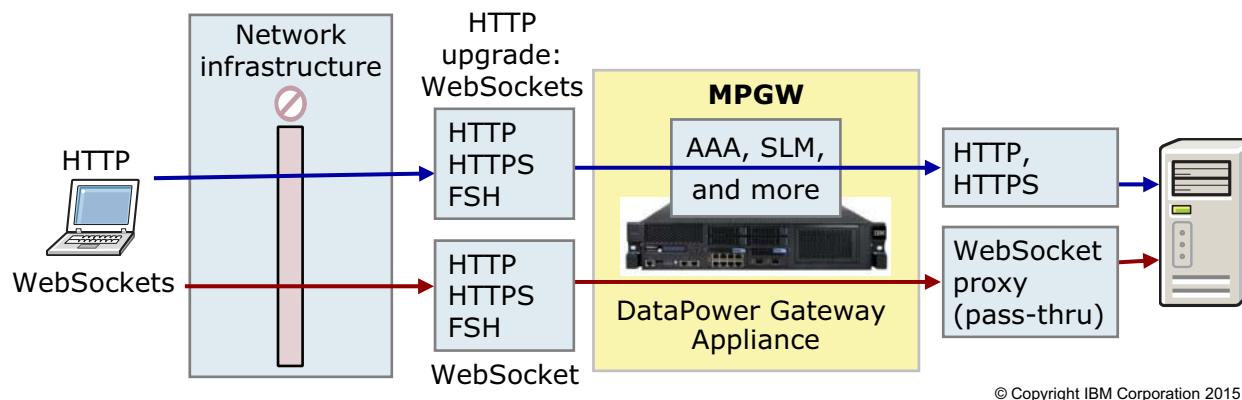
WE711 / ZE7111.0

### Notes:

This slide shows full-duplex, bidirectional, and low-latency communication for web and mobile applications.

## WebSocket Proxy (2 of 2)

- Apply DataPower policy actions until and including WebSocket upgrade request over HTTP or HTTPS
  - After upgrade request is accepted, DataPower simply proxies the client and server communication
- Example: Chat applications that use WebSockets require client authentication and connection throttling
  - Use DataPower AAA to authenticate and authorize client credentials and SLM to enforce connection concurrency



© Copyright IBM Corporation 2015

Figure 4-28. WebSocket Proxy (2 of 2)

WE711 / ZE7111.0

### Notes:

This slide shows full-duplex, bidirectional, and low-latency communication for web and mobile applications.



## 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 2015

Figure 4-29. Comparing services

WE711 / ZE7111.0

### 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 connection to a dynamic back-end by use of a processing rule to select a back-end service at run time

© Copyright IBM Corporation 2015

---

Figure 4-30. Unit summary

WE711 / ZE7111.0

### 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 (MPGW) service 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"> <li>1. Multi-protocol gateway</li> <li>2. Web service proxy</li> </ol> | <ol style="list-style-type: none"> <li>A. WSDL</li> <li>B. Service Registry and Repository concepts</li> <li>C. Multiple front side handlers</li> <li>D. Easy service level rule configuration</li> <li>E. MQ integration</li> </ol> |

© Copyright IBM Corporation 2015

Figure 4-31. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

- 1.
- 2.
3. (1)  
(2)



## Checkpoint answers

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

© Copyright IBM Corporation 2015

Figure 4-32. Checkpoint answers

WE711 / ZE7111.0

### Notes:

# Unit 5. Problem determination tools

## What this unit is about

This unit describes the troubleshooting tools that are available for debugging problems on the DataPower appliance. Several tools are available for various problems, ranging from low-level networking tools to probes that aid in debugging service policies. The logging utilities are available for capturing information that the DataPower objects generate.

## 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 DataPower appliance
- Configure a multi-step probe to examine detailed information about actions within rules
- List the problem determination tools that are available on the DataPower appliance

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)

## Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2015

---

Figure 5-1. Unit objectives

WE711 / ZE7111.0

### 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 2015

Figure 5-2. Common problem determination tools

WE711 / ZE7111.0

### 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  | 13,052 | Mbytes |
| Total Encrypted Space | 14,896 | Mbytes |
| Free Temporary Space  | 466    | Mbytes |
| Total Temporary Space | 512    | Mbytes |
| Free Internal Space   | 971    | Mbytes |
| Total Internal Space  | 1,024  | 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 2015

Figure 5-3. Appliance status information

WE711 / ZE7111.0

### Notes:

It is a good 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



Troubleshooting

© Copyright IBM Corporation 2015

Figure 5-4. Troubleshooting

WE711 / ZE7111.0

### 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.

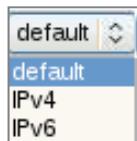
During the production phase, first check the system usage for load and work lists and then check the object status for objects that are changed to the down state. Finally, check the default system log.

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**



**Networking**

| <b>Ping Remote</b>                         |         | <b>TCP Connection Test</b>                         |   |
|--|---------|--|---|
| Remote Host                                | *       | Remote Host  | * |
| Use IP version                             | default | Remote Port  | * |
| <input type="button" value="Ping Remote"/> |         | <input type="button" value="TCP Connection Test"/> |   |

© Copyright IBM Corporation 2015

Figure 5-5. Troubleshooting: Networking

WE711 / ZE7111.0

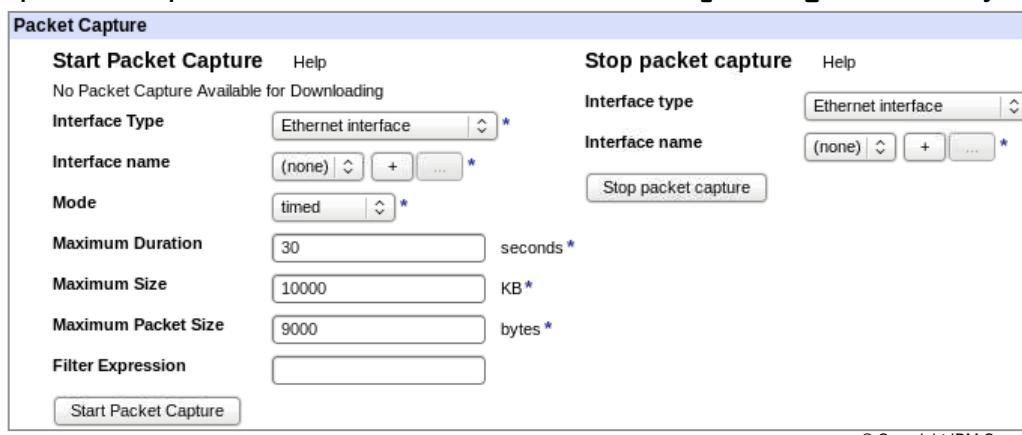
### Notes:

Ping Remote allows DataPower to ping a host system. Using ping confirms that there is network connectivity to the host IP address that 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 whether a service is running remotely or not. 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 2015

Figure 5-6. Troubleshooting: Packet capture

WE711 / ZE7111.0

### 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

The screenshot displays the 'Logging' configuration interface for DataPower V7.1. It is divided into two main sections: 'Set Log Level' on the left and 'Generate Log Event' on the right.

**Set Log Level:**

- View System Logs**: A link to view system logs.
- Log Level**: A dropdown menu set to 'debug' with an asterisk (\*) indicating it is required. Below it is a note: 'Set Log Level'.
- Enable Internal Logging**: A radio button group where 'off' is selected.
- Enable RBM Debug logging**: A radio button group where 'off' is selected.
- Global IP Address Log Filter**: An empty text input field.

**Generate Log Event:**

- Log Category**: A dropdown menu set to '(none)' with a '+' button and a '...' button.
- Log Level**: A dropdown menu set to 'notice' with an asterisk (\*) indicating it is required.
- Log Message**: An empty text input field.
- Event Code**: An empty text input field with a 'Select Code' button and a 'Help' button.
- Generate Log Event**: A button to generate the log event.

© Copyright IBM Corporation 2015

Figure 5-7. Troubleshooting: Logging

WE711 / ZE7111.0

### Notes:

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

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

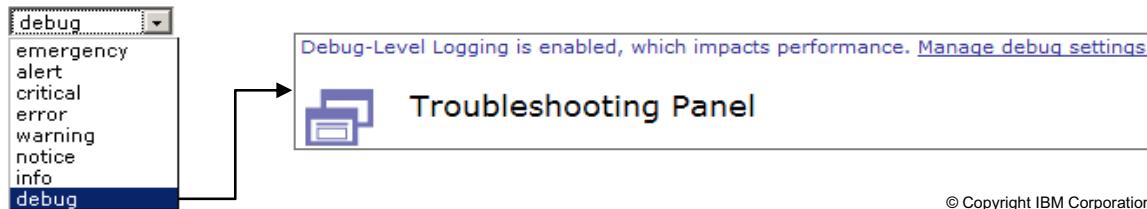


Figure 5-8. Troubleshooting: System log

WE711 / ZE7111.0

### 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.

Here are log levels 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)

**System Log**

Category      Log level

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

current time: 13:33:32 on 2012-08-28

| time            | category      | level  | tid      | direction | client       | msgid      | message   | Show last  |
|-----------------|---------------|--------|----------|-----------|--------------|------------|---|------------|
| Tue Aug 28 2012 |               |        |          |           |              |            |   | 50 100 all |
| 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 2015

Figure 5-9. Filtering system log

WE711 / ZE7111.0

### Notes:

The system log is defined as a log target. A log target receives log entries that DataPower objects generate. Each domain always has a log target that is called **default-log** to represent the default system log. More 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 2015

Figure 5-10. Troubleshooting: Generate Log Event

WE711 / ZE7111.0

### 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 the 'Reporting' section of the WebSphere Education interface. On the left, under 'Generate Error Report', it says 'No Error Report Available for Viewing' and has a 'Generate Error Report' button. On the right, under 'Send Error Report', there are four fields: 'SMTP Server' (with an asterisk), 'Location' (with an asterisk), 'E-mail Address' (with an asterisk), and 'Email Sender Address'. Below these fields is a 'Send Error Report' button. At the bottom right of the window, it says '© Copyright IBM Corporation 2015'.

Figure 5-11. Troubleshooting: Reporting

WE711 / ZE7111.0

### 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



© Copyright IBM Corporation 2015

Figure 5-12. Troubleshooting: Advanced

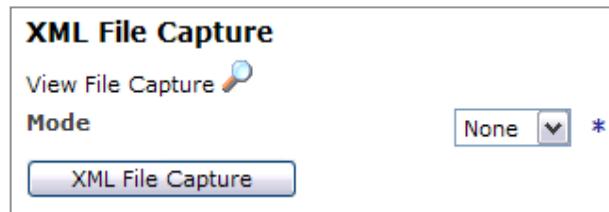
WE711 / ZE7111.0

### 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 2015

Figure 5-13. Troubleshooting: XML File Capture

WE711 / ZE7111.0

### Notes:

XML File Capture 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**

- **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 2015

Figure 5-14. Troubleshooting: Send a test message

WE711 / ZE7111.0

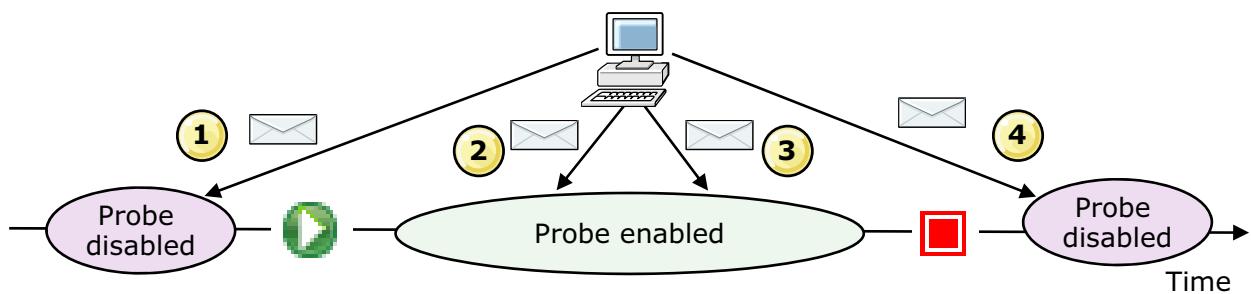
## Notes:

### 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 2015

Figure 5-15. Troubleshooting: Multi-step probe

WE711 / ZE7111.0

### 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, context, 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

Configure Multi-Protocol Gateway

|                       |                        |                        |   |          |               |                      |     |
|-----------------------|------------------------|------------------------|---|----------|---------------|----------------------|-----|
| General               | <u>Advanced</u>        | Stylesheet Params      | Headers   | Monitors | WS-Addressing | WS-ReliableMessaging | XMI |
| <a href="#">Apply</a> | <a href="#">Cancel</a> | <a href="#">Delete</a> | <a href="#">Export</a>   <a href="#">View Log</a>   <a href="#">View Status</a>   <a href="#" style="border: 2px solid orange; padding: 2px;">Show Probe</a>   <a href="#">Validate Conformance</a> |          |               |                      |     |

© Copyright IBM Corporation 2015

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

WE711 / ZE7111.0

### Notes:

Probes are enabled for the following services:

- XSL proxy and XSL coprocessor
- B2B gateway
- XML firewall
- Multi-protocol gateway
- Web service proxy
- Web token service
- Web application firewall



## Multi-step probe window

- Enable the probe in the multi-step probe window
  - Send 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                |
|------|----------|----------|--|--|---------------------|
| [+]  | 25506257 | request  | http://172.16.78.44:6957/EastAddressSearch | http://WSserver:9080/EastAddressSearch | EastAddressRequest  |
| [+]  | 25510545 | request  | http://172.16.78.44:6957/EastAddressSearch | http://WSserver:9080/EastAddressSearch | EastAddressRequest  |
| [+]  | 25510545 | response | http://172.16.78.44:6957/EastAddressSearch | http://WSserver:9080/EastAddressSearch | EastAddressResponse |

© Copyright IBM Corporation 2015

Figure 5-17. Multi-step probe window

WE711 / ZE7111.0

### 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.

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

Restarting the appliance disables all probes.

**Input Context '1' of Step 0**

Step 1: AAA Action:Input=INPUT, ActionDebug=off, Output=dpvar\_8, NamedInOutLocationType=default, AAA=BookingServiceAAAPolicy, Transactional=off, SOAPValidation=body, SQLSourceType=static, Asynchronous=off, ResultsMode=first-available, RetryCount=0, RetryInterval=1000, MultipleOutputs=off, IteratorType=XPATH, Timeout=0, MethodRewriteType=GET, MethodType=POST, MethodType2=POST

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

**Content of context 'INPUT':**

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:book="http://www.ibm.com/datapower/FLY/BookingService/"
>
<soapenv:Header />
<soapenv:Body>
  <book:BookingReq>
    <book:Booking>
```

- 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, context, global, and service variables
  - Actions that the processing rule executes

© Copyright IBM Corporation 2015

Figure 5-18. Multi-step probe content

WE711 / ZE7111.0

## Notes:

1. The row of actions across the top show what executed in the rule. 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. When you click a particular magnifying glass, the contents of the rest of the page changes to the state at that point in the processing. The square brackets around the magnifying glass indicate which one is selected. You can also click **Next** and **Previous** to view the message step-by-step as it is executed from the processing rule.
2. The default tab that is displayed is the **Content** tab. The tab renders the message contents if it can.
3. Other tabs are available to show more state that is associated with the message processing at the selected point in the rule.

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



## Debugging GatewayScript (1 of 4)

- To activate the GatewayScript debugging, two conditions must be met:
  - Debugging must be enabled in the GatewayScript action
  - The script that is invoked in the GatewayScript Action must contain a "debugger;" statement

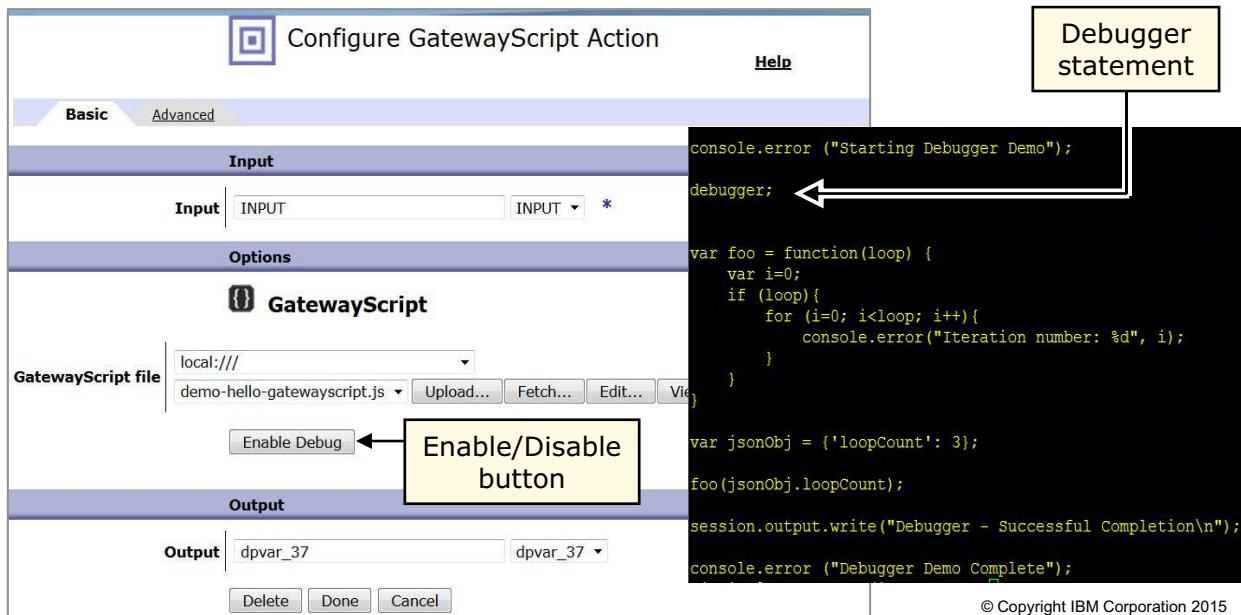


Figure 5-19. Debugging GatewayScript (1 of 4)

WE711 / ZE7111.0

### Notes:

To activate the GatewayScript debugger, two conditions must be met. The first condition requires the GatewayScript debugging to be enabled. GatewayScript debugging is enabled by clicking **Enable Debug** in the configuration screen of the GatewayScript action. The enable-disable button is highlighted in the image on the left side of this slide.

The **Debug** button does not persist during a domain or appliance reboot. Therefore, if the button was enabled, and the appliance is rebooted, the button is in a disabled state after the reboot.

The second condition that must be met requires the syntax of the GatewayScript code to contain the debugger statement. An example is represented in the image that is on the right side of the screen.



## Debugging GatewayScript (2 of 4)

- The flow of a transaction is paused indefinitely
  - The GatewayScript processing breaks at the “`debugger;`” line
  - A maximum of 10 debug sessions are in progress at any time
  - Use “`show debug-actions`” (in config mode) to find available sessions to debug

```

dpblade36-CLI - A maximum of 10 debug sessions will be allowed at any time.

xi50[2459-GatewayScript] (config)# show debug-actions

Session ID Transaction ID      Service Name      File Location
Remote Address   In Use Remote User User Location    Elapsed Time
-----          -----          -----          -----
-----          -----          -----          -----
85             63553           GatewayScript-Loopback local:///demo-debugger.js
127.0.0.1       No              00:06:43

xi50[2459-GatewayScript] (config)#
  
```

© Copyright IBM Corporation 2015

Figure 5-20. Debugging GatewayScript (2 of 4)

WE711 / ZE7111.0

### Notes:

This screen capture image is an example of what you would see and how you would figure out how to begin the debugger. There is status with the name *debug-actions*. When debugging is enabled, and a debugger statement exists in the GatewayScript script, a “`show debug-actions`” message shows the debug requests.

The GatewayScript execution breaks at the debugger statement. You might have up to 10 debug sessions in progress at one time. The scope of the maximum debug sessions is per appliance (not per domain).

- The session ID is used to identify which debug session you want to work with.
- The transaction ID is the ID of the transaction.
- The service name is the name of the service.
- The file location is the actual location of the script file that is being paused.
- The remote address is the address of the client.

**In Use** represents whether someone else is debugging the session. Currently, joint debugging is not allowed, so if **In Use** is set to Yes, you cannot debug this session.

If **In Use** is Yes, then the following fields contain data that represents the user currently debugging the session:

- User: The user currently debugging the session
- User location: IP address of the user
- Elapsed time: The amount of time that the transaction remains the debugger

## Debugging GatewayScript (3 of 4)

- Enter the CLI debugger – GDB-like interface
    - Must be in config mode in the domain where the action executed
    - `debug-action <session ID>`: Enter the CLI debugger until the script completes

```
xi50[2459-GatewayScript](config)# debug-action 85
 3:// Show how to invoke the CLI debugger. To use the
 4:// debugger, you need to enable debug either for the
 5:// specific action, or for the service (will include
 6:// all actions running in that service). And you must
 7:// have a "debugger;" statement in your code. The
 8:// "debugger;" statement serves as the initial breakpoint.
 9:
10:console.error ("Starting Debugger Demo");
11:
=>12:debugger; // Initial break point. Only has an affect
13:           // if debugging is enabled on action or service.
14:
15:var foo = function(loop) {
16:    var i=0;
17:    if (loop){
18:        for (i=0; i<loop; i++){
19:            console.error("Iteration number: %d", i);
20:        }
21:    }
22:}
(debug)
```

© Copyright IBM Corporation 2015

Figure 5-21. Debugging GatewayScript (3 of 4)

WE711 / ZE7111.0

## **Notes:**

The GatewayScript debugger is similar to the GNU Project debugger (GDB), which shows what is going on *inside* another program while it is running.

The *debug-action* must be executed from within the domain that is being debugged, and from within configuration mode (use the CLI `co` command).

The previous slide showed a debug session ID of 85. This image shows how you enter a debugging session, by executing a debug-action and the session ID: debug-action 85

What you are going to see, as represented on the image, is that the debugger shows a listing of the code around the debug statement. The debug listing includes line numbers and an arrow => pointing to the debug statement.

In the debugger, many commands can be executed, such as step-into, step-over, and other commands. The debugger commands are listed on the next slide.

## Debugging GatewayScript (4 of 4)

### Debugging commands:

- List source code
  - list(l) [number of lines]
- Breakpoints
  - break (b) <line | script.js:line | function()>
  - delete (d) <identifier | all>
  - info break (ib)
- Print variable values
  - print (p) <variable>
- Explore stack trace
  - backtrace (bt)
- Program execution control
  - continue (c)
  - next (n) [count]
  - step (n) [count]
  - out (o) [count]
  - quit (q)

© Copyright IBM Corporation 2015

Figure 5-22. Debugging GatewayScript (4 of 4)

WE711 / ZE7111.0

### Notes:

This slide includes a list of some of the GatewayScript debugging commands that the debugger supports.

For more detailed information, see the “GatewayScript debugger commands” section in the DataPower Knowledge Center.

## 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** option 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 2015

Figure 5-23. Problem determination with cURL

WE711 / ZE7111.0

### Notes:

The **-v** verbose flag produces much information in the output. It allows the user to see all of the client and 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 2015

Figure 5-24. Communicating with DataPower support

WE711 / ZE7111.0

### 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, HTTPS)
  - 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 2015

Figure 5-25. Logging basics

WE711 / ZE7111.0

### Notes:

Log files can be encrypted or signed for more security.

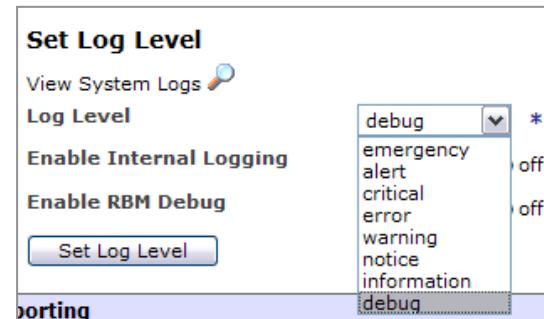
Objects that generate log messages have different priorities. These messages range from verbose debugging to infrequent critical or emergency level messages.



## 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



© Copyright IBM Corporation 2015

Figure 5-26. Log targets

WE711 / ZE7111.0

### 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 generate events

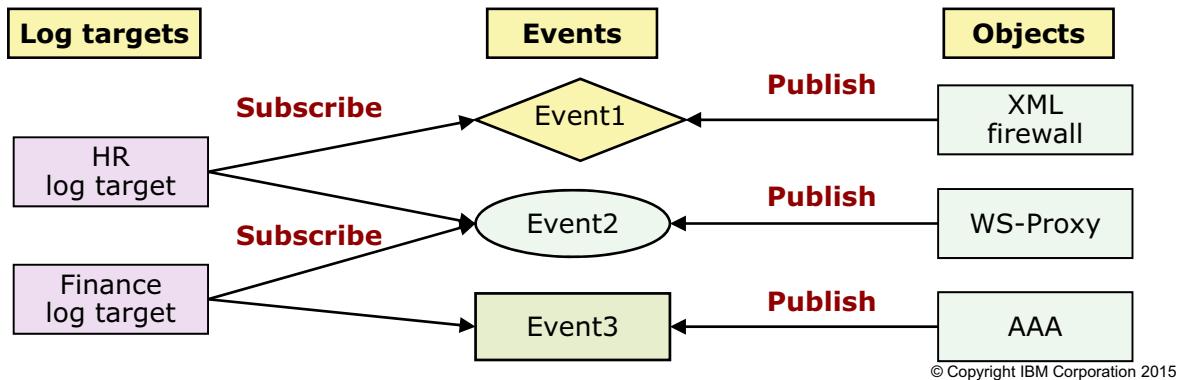


Figure 5-27. Available log levels

WE711 / ZE7111.0

### Notes:

The diagram in the slide shows 2 log targets: HR and Finance log targets. These log targets subscribe to certain types of events that are generated or published from objects on the DataPower appliance.

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



## 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

Comments:

Target Type: Cache

Log Format: XML

Timestamp Format: syslog

Feedback Detection: off

Identical Event Detection: off

Buttons: 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, xslt, and more
  - Categories have a priority level
- Log targets are required to subscribe to at least one event category

© Copyright IBM Corporation 2015

Figure 5-28. Log target configuration

WE711 / ZE7111.0

### Notes:

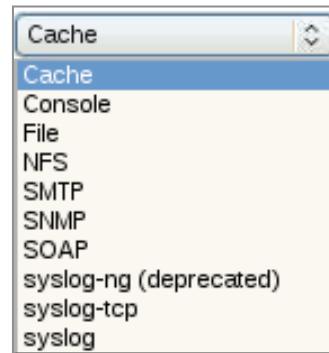
The **Event Subscription** tab is not visible in the screen capture.

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

## 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 on the serial port
- **File**: Writes log entries to a file on the appliance
- **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**: Uses TCP to forward log entries 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 over UDP



© Copyright IBM Corporation 2015

Figure 5-29. Log target types

WE711 / ZE7111.0

### 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.

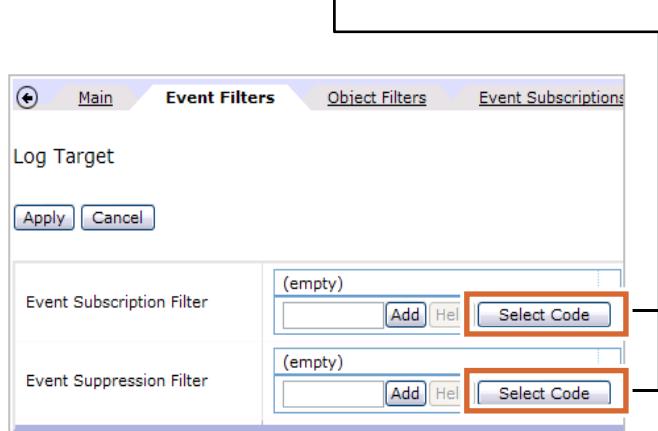
Syslog-NG (New Generation) is being deprecated. Use **syslog-tcp** in place of **syslog-*ng***.



## Event filters

- On the Configure Log Target page, click the **Event Filters** tab
- Event filters create filters for a log target that are 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 2015

Figure 5-30. Event filters

WE711 / ZE7111.0

### Notes:

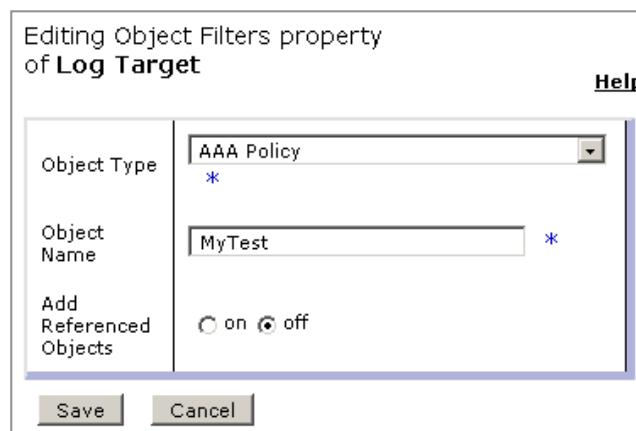
You can subscribe the current log target to particular event code categories. Example event codes include out of memory, failed to install on local port, and other codes.

These event codes are event conditions that are specific to DataPower.



## Object filters

- On the Configure Log Target page, click the **Object Filters** tab
- Object filters allow only those messages that selected objects generate 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 2015

Figure 5-31. Object filters

WE711 / ZE7111.0

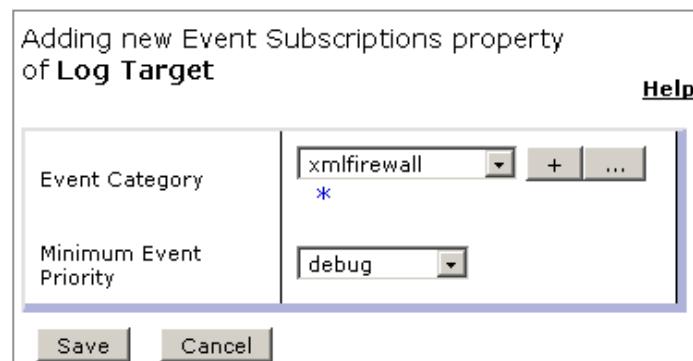
### Notes:

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
  - Another level of filtering



© Copyright IBM Corporation 2015

Figure 5-32. Event subscriptions

WE711 / ZE7111.0

### 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.

**WebSphere Education**

**IBM**

## Log action

The **Log** action sends the contents of the **Input** context to a destination URL

- Is used to log entire message instead of creating a log entry
- Configure:
  - Destination:** Must be a valid URL to either a local file or a remote destination
  - Log Level:** Event category
  - Log Type:** Log priority
  - Method:** HTTP method of POST, PUT, or DELETE

Configure Log Action

Basic Advanced

Input

Input (auto) (auto)

Options

Log

Destination http:// Var Builder

Log Level notice \*

Log Type mpgw + ... \*

Asynchronous on off

Method POST \*

Output

Output OUTPUT

Delete Done Cancel

© Copyright IBM Corporation 2015

Figure 5-33. Log action

WE711 / ZE7111.0

### 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 DataPower appliance
- Configure a multi-step probe to examine detailed information about actions within rules
- List the problem determination tools that are available on the DataPower appliance

© Copyright IBM Corporation 2015

Figure 5-34. Unit summary

WE711 / ZE7111.0

### 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 2015

Figure 5-35. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.



## Checkpoint answers

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

© Copyright IBM Corporation 2015

Figure 5-36. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 3



Enhancing the BookingService gateway

© Copyright IBM Corporation 2015

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

10.1

Figure 5-37. Exercise 3

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Perform advanced configuration of an MPGW
- Configure a document processing policy with more actions
- Test the MPGW policy by using the graphical SoapUI tool
- Perform basic debugging by using the system log

© Copyright IBM Corporation 2015

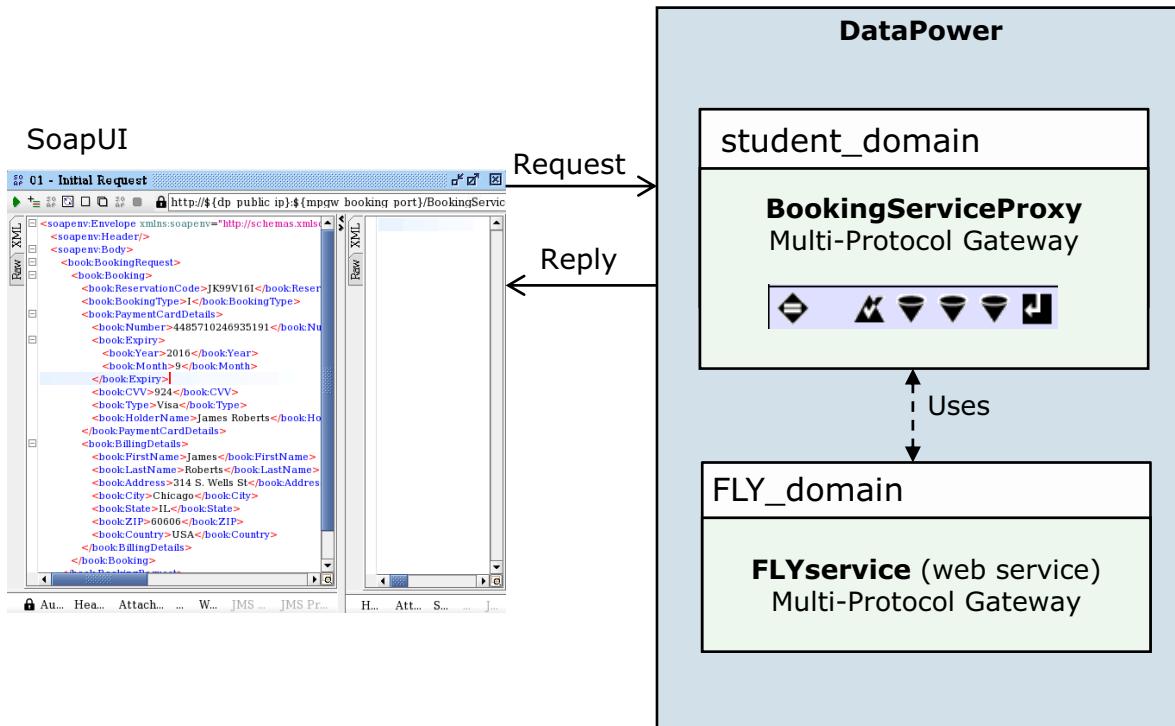
Figure 5-38. Exercise objectives

WE711 / ZE7111.0

### Notes:



## Exercise overview



© Copyright IBM Corporation 2015

Figure 5-39. Exercise overview

WE711 / ZE7111.0

### Notes:

# Unit 6. Handling errors in a service policy

## What this unit is about

Errors might occur when a service processes messages. The developers of services need to plan for error handling within those services. In this unit, you learn how to use the On Error action, the error rule, and the MPGW's error policy to control error handling.

## What you should be able to do

After completing this unit, you should be able to:

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

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2015

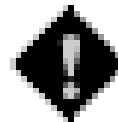
Figure 6-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## Error handling constructs

Default error handling procedure is to cancel the current document processing rule and log an error message



Three methods for handling errors:

- **Error policy**
  - Assign an error policy to the MPGW service.
  - The error policy defines the actions to take against errors in an HTTP or HTTPS flow that no precedent error handler handles.
- **On Error action**
  - Used to 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 2015

Figure 6-2. Error handling constructs

WE711 / ZE7111.0

### Notes:

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

## Error handling options

Processing policies might have On Error Actions and Error Rules.

### Multi-Protocol Gateway

#### Processing Policy

Processing Rule #1  
[Req | Rsp | Both | Error]

Match Action

On Error Action #1

• • •

Processing Action #N

Processing Rule #2  
[Req | Rsp | Both | Error]

Match Action

Processing Action #1

• • •

On Error Action #N

•  
•  
•

Processing Rule #N  
[Req | Rsp | Both | Error]

Match Action

Processing Action #1

• • •

Processing Action #N

© Copyright IBM Corporation 2015

Figure 6-3. Error handling options

WE711 / ZE7111.0

### Notes:

Error rules are executed only when an error occurs during processing.

**Error Code:** A match template that matches against specific error codes that might be raised by previously executed processing rules.

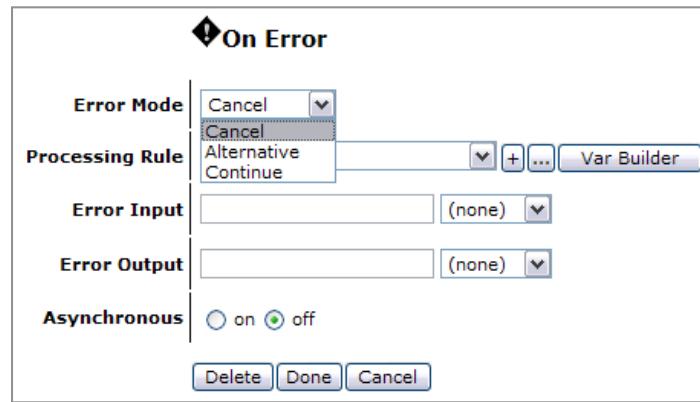
## Configure an On Error action

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 rules 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



© Copyright IBM Corporation 2015

Figure 6-4. Configure an On Error action

WE711 / ZE7111.0

### Notes:

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

1. Drag the **Advanced** icon to the rule configuration path.
2. Double-click the **Advanced** icon.
3. On the Configure Action page, select **On Error** and click **Next**.
4. 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).

5. Use the context **OUTPUT** in the **Error Output** field to return the error message to the client.

**WebSphere Education**

**Creating an error rule**

The screenshot shows the IBM DataPower configuration interface. At the top, there's a header with the WebSphere Education logo and the IBM logo. Below the header, the title "Creating an error rule" is displayed. The main area is divided into several sections:

- Rule:** A form with "Rule Name: Test\_rule\_1" and "Rule Direction: Error" (highlighted with a red box).
- Action Icons:** A row of icons for Filter, Sign, Verify, Validate, Encrypt, Decrypt, Transform, Route, AAA, Results, and Advanced.
- Flow Diagram:** A diagram showing a flow from an "ORIGIN SERVER" icon to a "DATAPOWER" device icon, with various action icons placed along the path.
- Create Reusable Rule:** A button to create reusable rules.
- Configured Rules:** A table listing configured rules:
 

| Order | Rule Name     | Direction        | Actions |
|-------|---------------|------------------|---------|
| 1     | Test_request  | Client to Server | ↓ ↘     |
| 2     | Test_response | Server to Client | ↑ ↗     |
| 3     | Test_rule_1   | Error            | ← ↖ ↙ ↘ |
- Information Box:** A yellow box containing text about error rules and their actions.

**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 to the client
  - Use the **Log** action to log entire message
  - Use the **Transform** action to build custom error messages

© Copyright IBM Corporation 2015

Figure 6-5. Creating an error rule

WE711 / ZE7111.0

## 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 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:/// custom-error.xsl

URL Rewrite Policy: (none)

Asynchronous: on off

**Output**

Output: OUTPUT OUTPUT

Buttons: Delete Done Cancel

© Copyright IBM Corporation 2015

Figure 6-6. Configure Transform action in error rule

WE711 / ZE7111.0

### 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 2015

Figure 6-7. Style sheet programming that use error variables

WE711 / ZE7111.0

### 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 clicking the **Service Variables** tab.

A log target can gather messages that use the `dp:type` attribute in the `<xsl:message>` tag, 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 2015

Figure 6-8. Example custom error style sheet

WE711 / ZE7111.0

### 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 2015

Figure 6-9. Error rule versus On Error action

WE711 / ZE7111.0

### 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 or non-XML client applications in a simpler manner
  - Fallback for an error that is not successfully handled with any precedent error handlers (such as multistep error rule)

© Copyright IBM Corporation 2015

Figure 6-10. Error Policy

WE711 / ZE7111.0

### 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 any precedent error handler did not successfully handle, the new Error Policy is executed to generate the final error response.

The reasons for the new feature are described as follows.

Before Release 6.0.0, the MPGW service tended 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 an 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. You can use the new Error Policy when there is no error handler

(such as multistep error rule) or the precedent error handler fails. By using the new Error Policy, you can have a *fallback* to generate the error response (such as an HTML, a plaintext, 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 logic needs to involve few error actions
  - Implements 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 2015

Figure 6-11. Typical Error Policy use cases

WE711 / ZE7111.0

### 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 a 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 logic (including many actions that are involved in the multistep error rule) to generate the error response. For example, in a circumstance when you need to respond with an HTTP URL redirection without a complex error rule configuration, then the Error Policy is a convenient and effective way for this purpose.
- You 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 fails.



## 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** objects
  - In a Multi-Protocol Gateway service configuration, specify the property **Error Policy** with the Multi-Protocol Gateway Error Policy

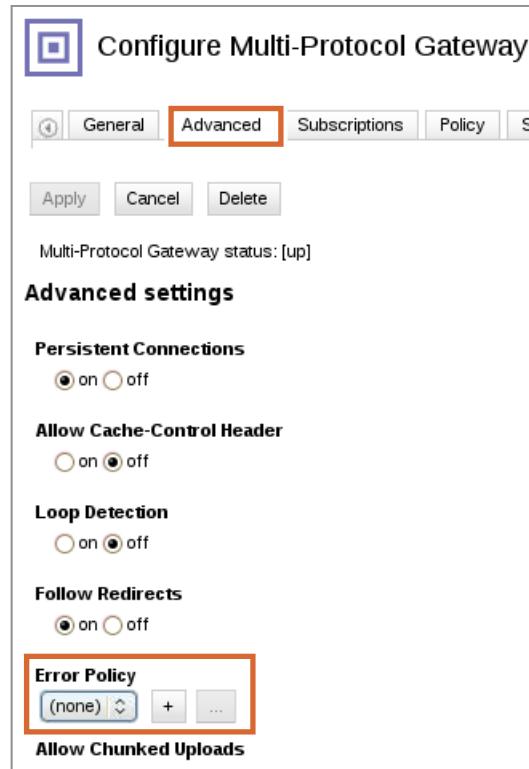


Figure 6-12. How the Error Policy works (1 of 3)

WE711 / ZE7111.0

### Notes:

The next three slides explain the following concepts: the required configurations to enable the new Error Policy, the preconditions 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 this enhancement introduces. You 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 fails.

## How the Error Policy works (2 of 3)

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

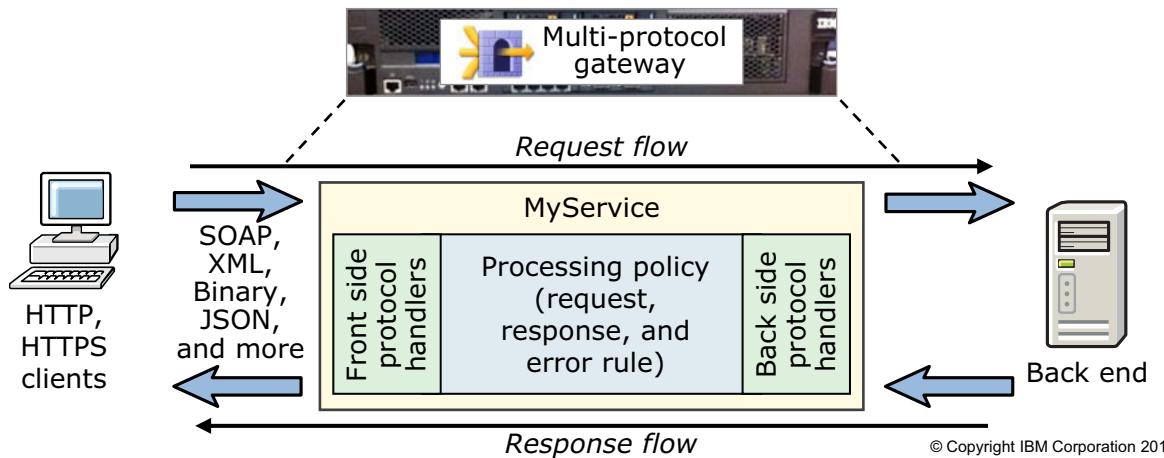


Figure 6-13. How the Error Policy works (2 of 3)

WE711 / ZE7111.0

### Notes:

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

- The MPGW transaction is initiated from an HTTP or HTTPS request that represents a web gateway flow. The back-end system can be any type.
- An error is occurring in one of the following areas:
  - The front side (for example, header parsing failure in the front side)
  - Request processing (for example, the Encrypt action in a request rule fails)
  - Back-end server (for example, failure to establish a connection to the back-end server)
  - 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, any precedent error handler does not handle the error. Under these conditions, the situation of “unhandled error” occurs, which requires a fallback.

For example, you have the error rule that the On-Error action invokes, and it fails. Then, the multistep processing rule matching selects the error rule to process, and the matched error rule also fails. Then, it is 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-to-one
  - The Error Action that is associated with the first-matched matching rule is invoked and returns its output to the front side HTTP or HTTPS client

Note: If no matching rule is satisfied 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 must not 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 must not be invoked if Padding Oracle Protection is enabled

© Copyright IBM Corporation 2015

Figure 6-14. How the Error Policy works (3 of 3)

WE711 / ZE7111.0

### 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 or HTTPS client.

If an error occurs during the Error action execution – for example, it fails 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 message is obscured.

Configure Multi-Protocol Gateway Error Policy

Main

Multi-Protocol Gateway Error Policy

Name: demo-errorpolicy \*

Administrative State: enabled

Comments:

Policy Maps

| Matching Rule | Error Action    |
|---------------|-----------------|
| XML           | redirect2websrv |
| HTML          | LocalHTMLRender |
| ALL           | DefaultResponse |

Matching Rule: \* Error Action: Add

Configure Multi-Protocol Gateway

General Advanced Subscriptions Policy S

Apply Cancel Delete

Multi-Protocol Gateway status: [up]

**Advanced settings**

Persistent Connections  
on off

Allow Cache-Control Header  
on off

Loop Detection  
on off

Follow Redirects  
on off

**Error Policy**  
(none) + ...

Allow Chunked Uploads  
on off

© Copyright IBM Corporation 2015

Figure 6-15. Error Policy configuration

WE711 / ZE7111.0

## Notes:

Multi-Protocol Gateway Error Policy is a new type of configuration 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 (the Matching Rule evaluates) 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 configuration for the matching strategy; it works as follows:

- For any XML request (matched by Content-Type “\*xml\*”), the associated error map is called if there is an XML match.
- For any HTML request (matched by Content-Type “\*html\*”), the associated error map is called if there is an HTML match.
- If none of the defined matching rule is evaluated to be true, then the associate map is called for all other “non-matching” conditions.



## Error Policy configuration: Multi-Protocol Gateway

Configure Multi-Protocol Gateway Error Action

Main    HTTP Header Injection

Multi-Protocol Gateway Error Action

Name:  \*

Administrative State:  enabled  disabled

Comments:

Mode:  Error Rule  
 Proxy (Remote)  
 Redirect  
 Static (Local)

Local page location: local:///     \*

Response Code:

Reason Phrase:

• 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

© Copyright IBM Corporation 2015

Figure 6-16. Error Policy configuration: Multi-Protocol Gateway

WE711 / ZE7111.0

### Notes:

The Multi-Protocol Gateway Error Action is the other new type of configuration that is 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 choose the processing rule only with rule direction “Error.” In this “error rule”, you can define how the error is handled (such as logging and rewriting the service variables). 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 or HTTPS 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 or HTTPS 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 such as **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.

## More 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)
  - 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 2015

Figure 6-17. More Error Policy Processing (1 of 2)

WE711 / ZE7111.0

### 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 configuration, 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 consider. 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 can use header injection to override the value. For *Rule* mode, you can either set the Content-Type on the rule, use the header injection, or do both to override at the end.



## More 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 available only in Multi-Protocol Gateway services and applies only to HTTP or HTTPS 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 2015

Figure 6-18. More Error Policy Processing (2 of 2)

WE711 / ZE7111.0

### 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 also be used for setting the timeout value for the connection to the remote URL.

The feature is available only for MPGW with HTTP or HTTPS traffic and has no effect on other services and flows.

Before release 6.0.0, the web application firewall had a concept similar to the object "Error Policy." To eliminate the naming confusion against the new "Multi-Protocol Gateway Error Policy," the previously known "Error Policy" was renamed to "Web Application Firewall Error Policy." The change took effect only 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 error policy
- Configure an On Error action in a service policy
- Configure an error rule in a service policy
- Describe how On Error actions, error rules, and error policies are selected during error handling

© Copyright IBM Corporation 2015

Figure 6-19. Unit summary

WE711 / ZE7111.0

### 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 to 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 2015

Figure 6-20. Checkpoint questions

WE711 / ZE7111.0

### 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 2015

Figure 6-21. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 4



Adding error handling to a service

policy

© Copyright IBM Corporation 2015

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

10.1

Figure 6-22. Exercise 4

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Configure an error policy at the MPGW service level
- Configure a service policy with an On Error action
- Configure a service policy with an Error rule

© Copyright IBM Corporation 2015

Figure 6-23. Exercise objectives

WE711 / ZE7111.0

### 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, and how to secure connections by using SSL to and from the DataPower appliance. You also learn how to set the DataPower objects that are used to validate certificates and configure certificate monitoring to ensure that only valid certificates exist on the appliance.

## What you should be able to do

After completing this unit, you should be able to:

- Explain how to use the DataPower tools to generate cryptographic keys
- Create a crypto identification credential object that contains a matching public and private key
- Create a crypto validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up to date
- Configure an SSL proxy profile that accepts an SSL connection request from a client
- Configure an SSL proxy profile that initiates an SSL connection from a DataPower service

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)

## Unit objectives

After completing this unit, you should be able to:

- Explain how to use the DataPower tools to generate cryptographic keys
- Create a crypto identification credential object that contains a matching public and private key
- Create a crypto validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up to date
- Configure an SSL proxy profile that accepts an SSL connection request from a client
- Configure an SSL proxy profile that initiates an SSL connection from a DataPower service

© Copyright IBM Corporation 2015

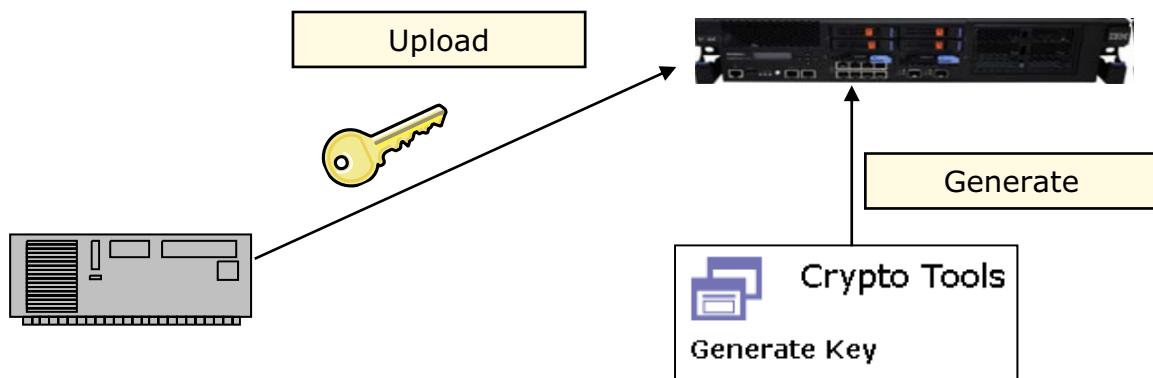
Figure 7-1. Unit objectives

WE711 / ZE7111.0

### 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 2015

Figure 7-2. DataPower crypto tools

WE711 / ZE7111.0

### 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.



## 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**

|                                     |   |
|-------------------------------------|---|
| <b>LDAP (reverse) Order of RDNs</b> | <input type="radio"/> on <input checked="" type="radio"/> off |
| <b>Country Name (C)</b>             | <input type="text"/>  |
| <b>State or Province (ST)</b>       | <input type="text"/>  |
| <b>Locality (L)</b>                 | <input type="text"/>  |
| <b>Organization (O)</b>             | <input type="text"/>  |
| <b>Organizational Unit (OU)</b>     | <input type="text"/>  |
| <b>Organizational Unit 2 (OU)</b>   | <input type="text"/>  |
| <b>Organizational Unit 3 (OU)</b>   | <input type="text"/>  |
| <b>Organizational Unit 4 (OU)</b>   | <input type="text"/>  |
| <b>Common Name (CN)</b>             | <input type="text"/> *  |
| <b>RSA Key Length</b>               | 1024 bits <input type="button" value="▼"/>                    |
| <b>File Name</b>                    | <input type="text"/>  |
| <b>Validity Period</b>              | <input type="text"/> 365 days                                 |

© Copyright IBM Corporation 2015

Figure 7-3. Generating crypto (asymmetric) keys on-board (1 of 2)

WE711 / ZE7111.0

### 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
  - 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 dialog for generating keys. It includes fields for 'Password' and 'Confirm Pass', 'Password Alias', and several checkboxes for generating different types of certificates and keys. The 'Export Private Key' checkbox is checked ('on'). Other checkboxes for 'Generate Self-Signed Certificate', 'Export Self-Signed Certificate', and 'Generate Key and Certificate Objects' are unchecked ('off'). There are also input fields for 'Object Name' and 'Using Existing Key Object', and a 'Generate Key' button.

© Copyright IBM Corporation 2015

Figure 7-4. Generating crypto (asymmetric) keys on-board (2 of 2)

WE711 / ZE7111.0

### 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.

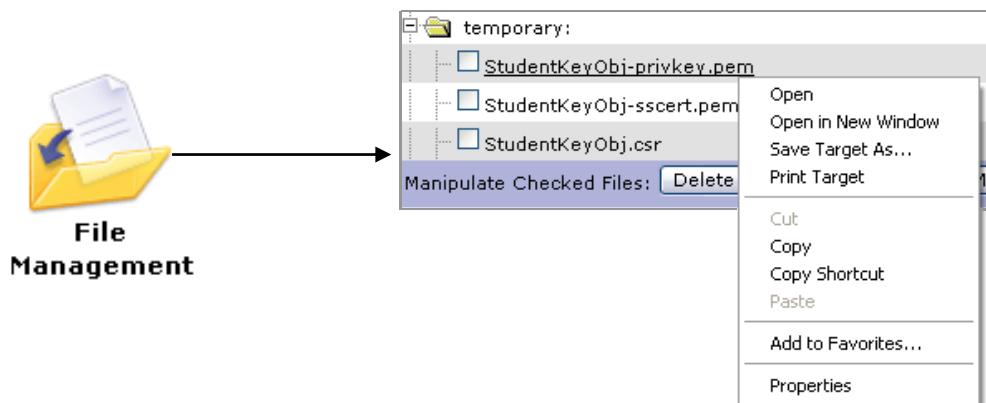
If **Export Self-Signed Certificate** or **Export Private Key** is **off**, then the generated key or certificate is placed in the cert: directory only, where it cannot be edited.

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 2015

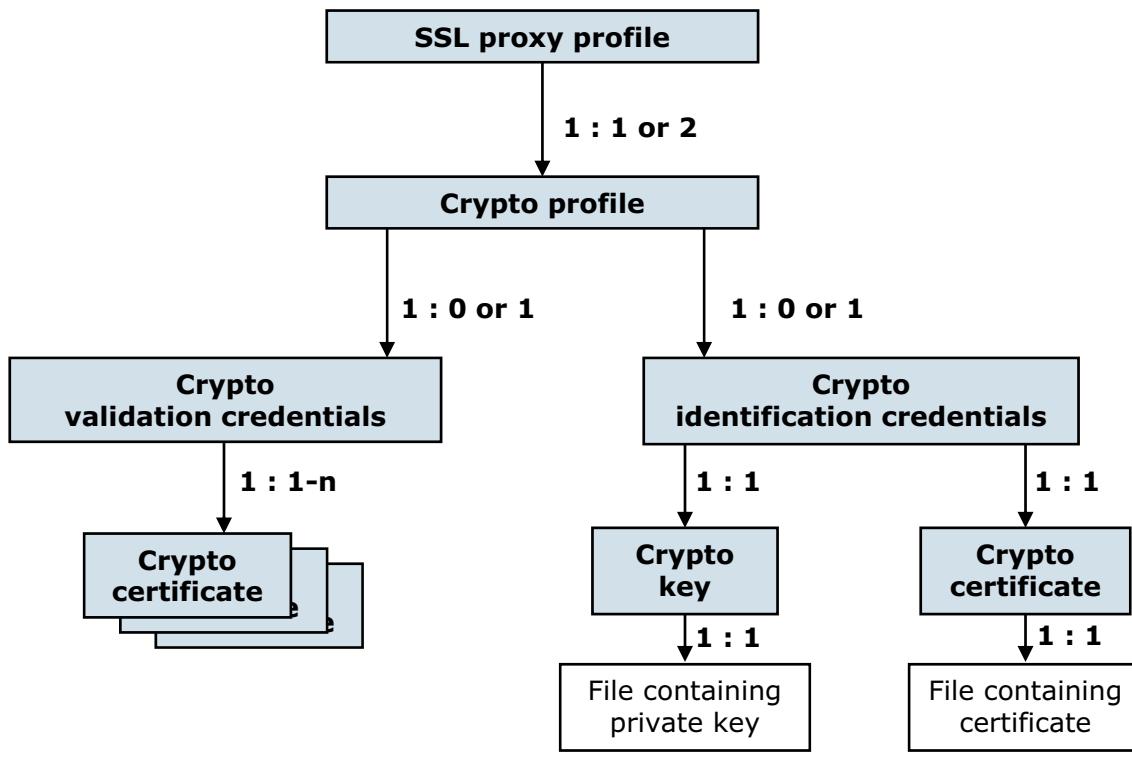
Figure 7-5. Download keys from temporary storage

WE711 / ZE7111.0

### Notes:

The `temporary:` directory is cleared when the appliance shuts down or restarts.

## SSL - crypto object relationships



© Copyright IBM Corporation 2015

Figure 7-6. SSL - crypto object relationships

WE711 / ZE7111.0

### Notes:

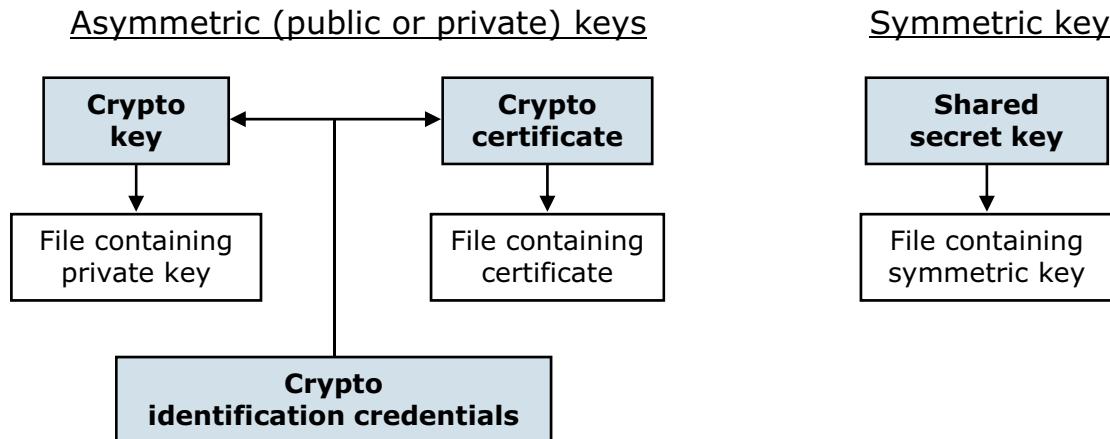
This graphic shows the relationships of the various objects and files that are involved in SSL and other crypto work on the appliance. It also shows the multiplicity of the relationship. For example, an SSL proxy profile object can have 1 or 2 crypto profile objects that are related to it.

The crypto key and crypto certificate are also used in encryption and digital signatures.

The details on the objects are in the following slides.

## 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 2015

Figure 7-7. Key and certificate objects point to files

WE711 / ZE7111.0

### Notes:

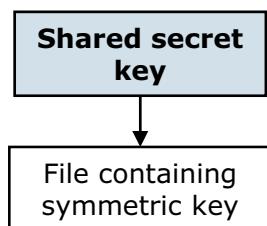
Although the shared secret key is not used in SSL, it is used in OAuth, and infrequently in encryption and signatures.



## 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**
- Use the Crypto Shared Secret Key page to define a secret key object for a symmetric key
  - Provides an extra level of security by providing an indirect reference to the file



**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" value="dpedu.p12"/>                            |   |

© Copyright IBM Corporation 2015

Figure 7-8. Crypto shared secret (symmetric) key

WE711 / ZE7111.0

### Notes:

A secret key is used for symmetric key encryption.

Symmetric keys are used in OAuth.

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.

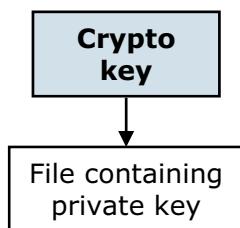
The key file can be uploaded from this page.



## 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**
- Use the Crypto Key page 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



| Crypto Key                           |   |
|--------------------------------------|---|
| <input type="button" value="Apply"/> | <input type="button" value="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                             | *****<br>*****  |
| Password Alias                       | <input type="radio"/> on <input checked="" type="radio"/> off           |

© Copyright IBM Corporation 2015

Figure 7-9. Crypto (asymmetric) key

WE711 / ZE7111.0

### Notes:

A crypto key represents the private key that is used for asymmetric key encryption.

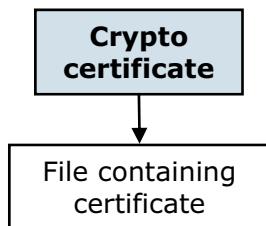
The key file can be uploaded from this page.



## 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                   |   |
|--------------------------------------|---|
| <input type="button" value="Apply"/> | <input type="button" value="Cancel"/>                                   |
| Name                                 | <input type="text"/>  |
| Admin State                          | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| File Name                            | <input type="text"/> 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 2015

Figure 7-10. Crypto certificate

WE711 / ZE7111.0

### 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**

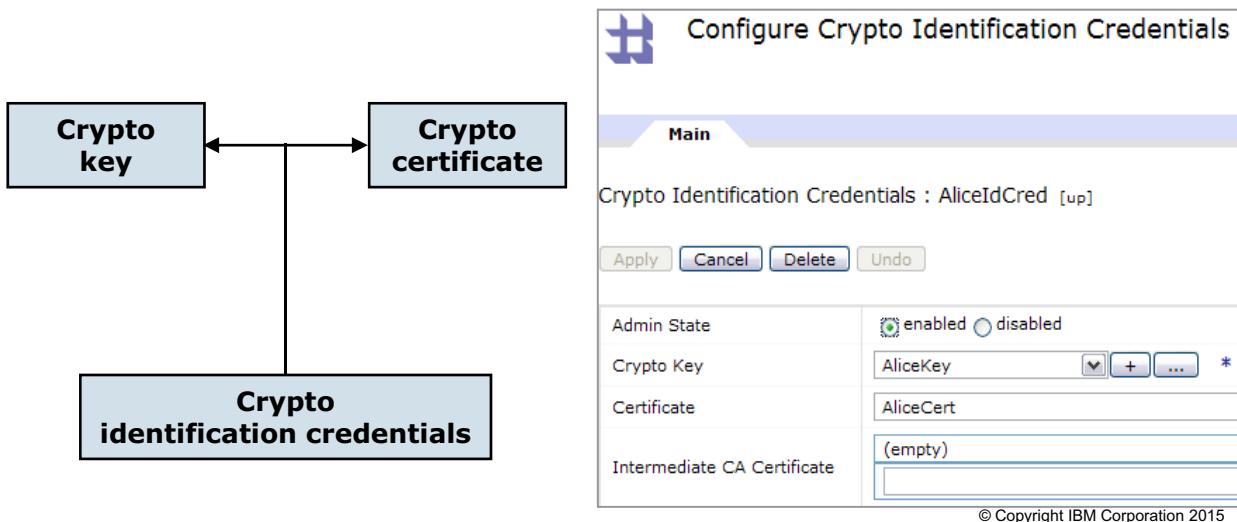


Figure 7-11. Crypto identification credential

WE711 / ZE7111.0

### 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.

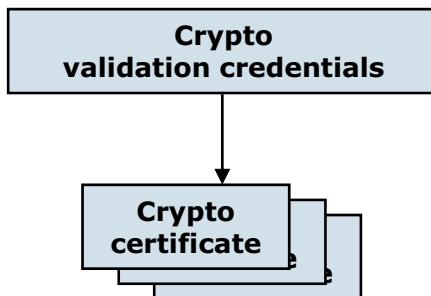
Specify the intermediate certificate authority (CA) certificates, if available, by clicking the **Add**. The process establishes a trust chain that consists of one or more CA certificates.

You can also create a crypto identification credential by clicking **Keys and Certs 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 whether the presenter is who or what 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-
- 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 2015

Figure 7-12. Crypto validation credential

WE711 / ZE7111.0

### 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 is 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' section of the WebSphere Administration interface. The 'Export Crypto Object' tab is selected. The form contains the following fields:
 

- Object Type:** Certificate (selected from a dropdown menu)
- Object Name:** [Input field]
- Output File Name:** [Input field]

 Below the form is a large 'Export Crypto Object' button.

© Copyright IBM Corporation 2015

Figure 7-13. Import and export crypto objects

WE711 / ZE7111.0

### Notes:

This page is accessed from the 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.



## Uploading keys

- From the vertical navigation bar, click **Objects > Crypto Configuration > Crypto Key**
- On the Crypto Key page, click **Upload** to upload the key file

The figure consists of two screenshots. On the left is a screenshot of the 'Crypto Key' configuration page. It has fields for 'Name' (with an asterisk), 'Admin State' (radio buttons for 'enabled' and 'disabled'), 'File Name' (dropdown menu showing 'cert:' and '(none)', with an 'Upload...' button highlighted by a red box), 'Password', 'Confirm Password', and 'Password Alias' (radio buttons for 'on' and 'off'). On the right is a screenshot of an 'Upload File to Directory cert:///'. It has fields for 'File to upload' (with a 'Browse...' button) and 'Save as' (with an asterisk). There is also a checkbox for 'Overwrite Existing File' and 'Upload' and 'Cancel' buttons. An arrow points from the 'Upload...' button on the main page to the 'Upload' button in the dialog box.

© Copyright IBM Corporation 2015

Figure 7-14. Uploading keys

WE711 / ZE7111.0

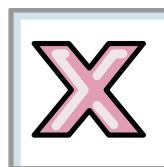
### Notes:

## Certificates can expire or get revoked

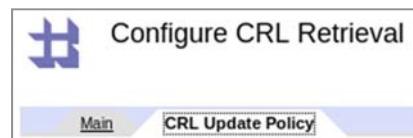


**Valid until  
02-14-2015**

- 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 2015

Figure 7-15. Certificates can expire or get revoked

WE711 / ZE7111.0

### 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="+"/> <input type="button" value="..."/> |
| Refresh Interval                  | 240 minutes *  |
| Cryptographic Profile             | <input type="text"/>   |
| Fetch URL                         | <input type="text"/>   |

© Copyright IBM Corporation 2015

Figure 7-16. Certificate revocation list (CRL) retrieval

WE711 / ZE7111.0

### 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 are displayed to support the protocol.

The **Cryptographic Profile** identifies the crypto profile to use to connect to the CRL issuer when using SSL.



## 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 <small>day *</small>                             |
| Reminder Time                | <input type="text"/> 30 <small>day *</small>                            |
| Log Level                    | <input type="button"/> warning *  |
| Disable Expired Certificates | <input type="radio"/> on <input checked="" type="radio"/> off *         |

© Copyright IBM Corporation 2015

Figure 7-17. Crypto certificate monitor

WE711 / ZE7111.0

### Notes:

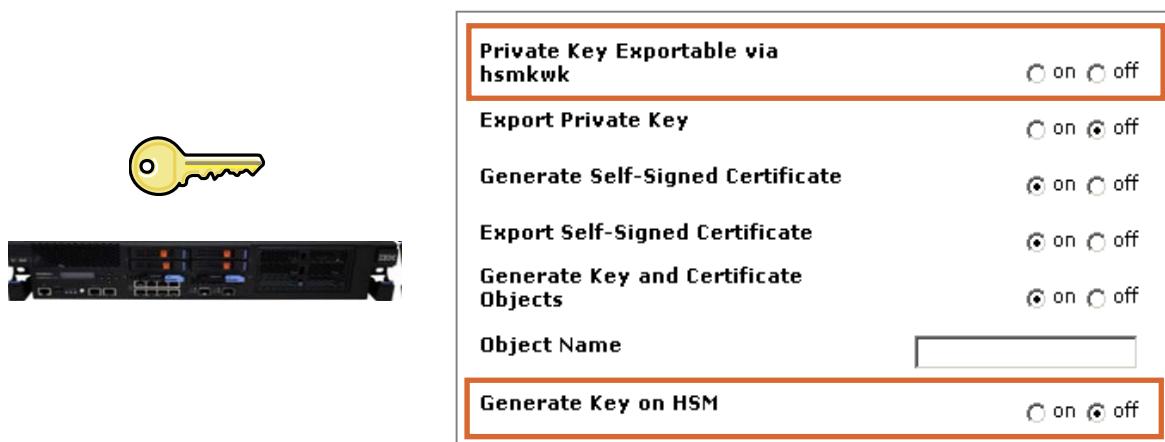
Polling interval specifies the frequency with which certificate expiration dates are checked.

Reminder time is the number of days before the certificate expiration event is written to the log file.

WebSphere Education

## 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 that is installed
- DataPower supports FIPS 140-2 level 2 and level 3 security



© Copyright IBM Corporation 2015

Figure 7-18. Hardware Security Module (HSM)

WE711 / ZE7111.0

### 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).

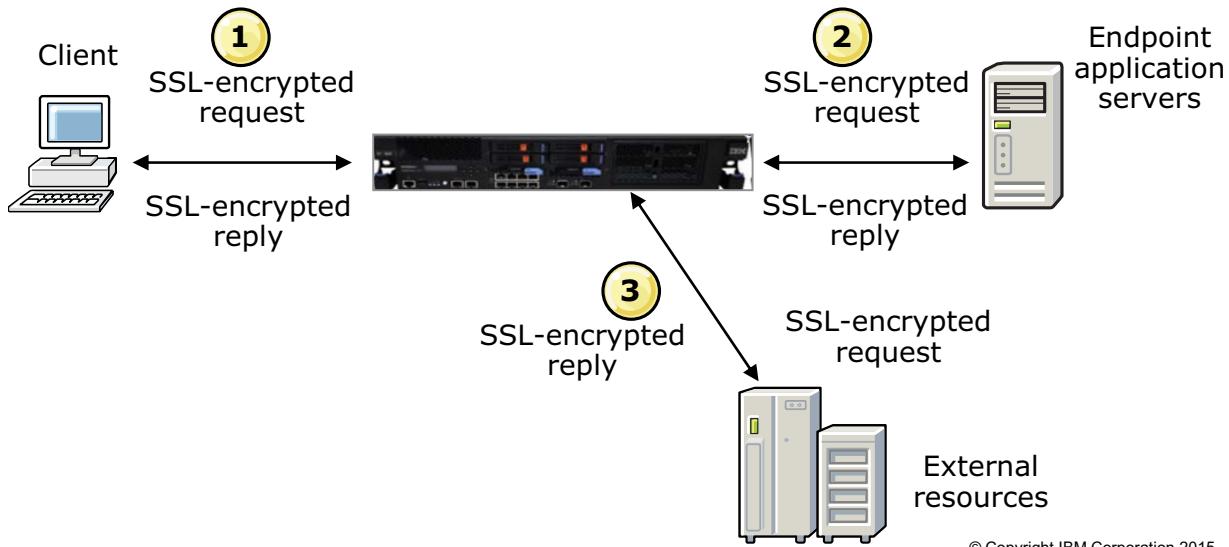
The HSM option is shown only if you have an HSM installed.

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 2015

Figure 7-19. DataPower support for SSL

WE711 / ZE7111.0

### 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 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**

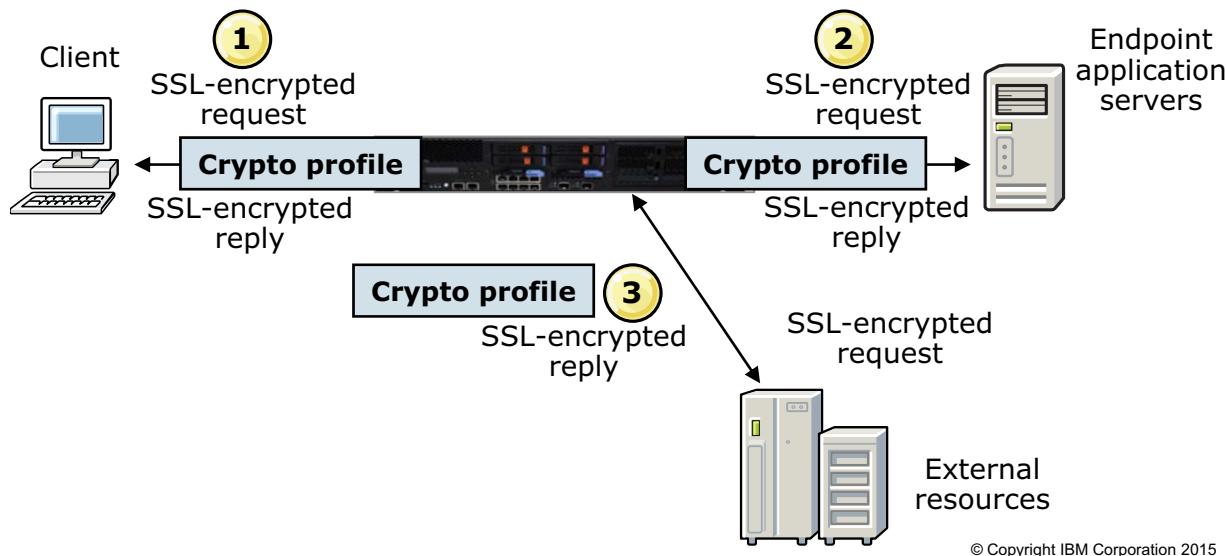


Figure 7-20. A crypto profile specifies details of the SSL connection

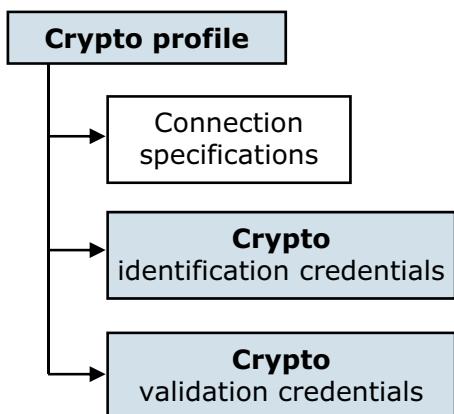
WE711 / ZE7111.0

### 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                       |   |
|--------------------------------------|---|
| <input type="button" value="Apply"/> | <input type="button" value="Cancel"/>   |
| Name                                 | <input type="text" value="StudentServerCP"/> *  |
| Administrative state                 | <input checked="" type="radio"/> enabled <input type="radio"/> disabled   |
| Identification Credentials           | <input type="text" value="StudentIdCred"/> <input type="button" value="+"/> <input type="button" value="..."/>  |
| Validation Credentials               | <input type="text" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>   |
| Ciphers                              | <input type="text" value="HIGH:MEDIUM:!aNULL:!eNULL:@STRENGTH"/>  |
| Options                              | <input checked="" type="checkbox"/> Enable default settings<br><input checked="" type="checkbox"/> Disable SSL version 2<br><input checked="" type="checkbox"/> Disable SSL version 3<br><input type="checkbox"/> Disable TLS version 1.0<br><input type="checkbox"/> Permit insecure SSL renegotiation to a legacy SSL client<br><input type="checkbox"/> Enable compression<br><input type="checkbox"/> Disable TLS version 1.1<br><input type="checkbox"/> Disable TLS version 1.2 |

© Copyright IBM Corporation 2015

Figure 7-21. Crypto profile

WE711 / ZE7111.0

### Notes:

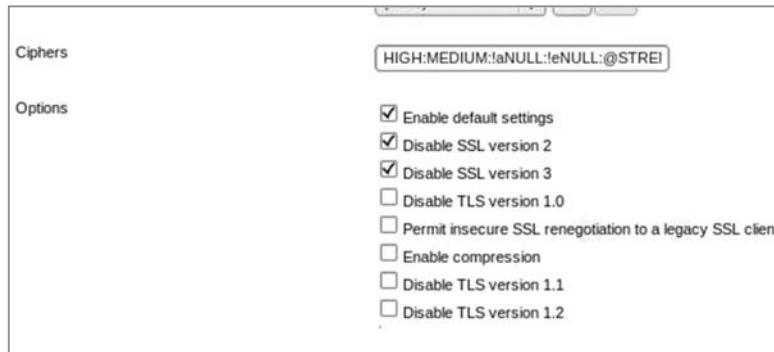
The **Ciphers** field specifies what cipher specifications are supported at the DataPower end of the connection. It is composed of one or more cipher suites.

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).



## Do not use insecure SSL options in the crypto profile

- SSL version 2 and 3 are now considered insecure and are vulnerable
  - In firmware 7.1.0, SSL version 2 and 3 are disabled by default
- Do not enable weak cipher suites
  - Weak or export-level cipher suites are vulnerable. If you change the default cipher suite, avoid the RSA export-level (EXPORT40, EXPORT56) algorithms



- IBM Support Portal for DataPower
  - [https://www.ibm.com/support/entry/portal/product/websphere\\_ibm\\_datapower\\_gateways](https://www.ibm.com/support/entry/portal/product/websphere_ibm_datapower_gateways)

© Copyright IBM Corporation 2015

Figure 7-22. Do not use insecure SSL options in the crypto profile

WE711 / ZE7111.0

### Notes:

IBM Support Portal: Security Bulletin: Vulnerability in SSLv3 affects DataPower (CVE-2014-3566)

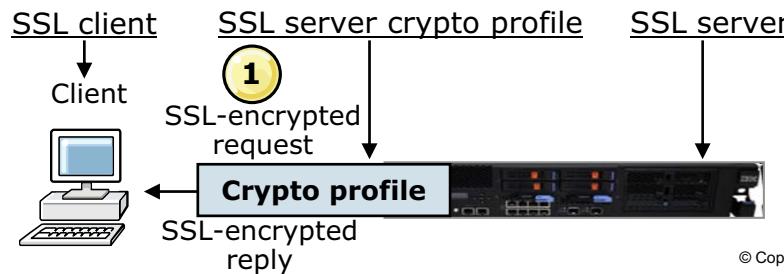
• <http://www.ibm.com/support/docview.wss?uid=swg21687189>

IBM Support Portal: Do not enable weak cipher suites for IBM DataPower Gateway appliances.

• <http://www.ibm.com/support/docview.wss?uid=swg21699042>

## 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**
  - Configures 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 the appliance presents, 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 2015

Figure 7-23. Securing connections from client to appliance

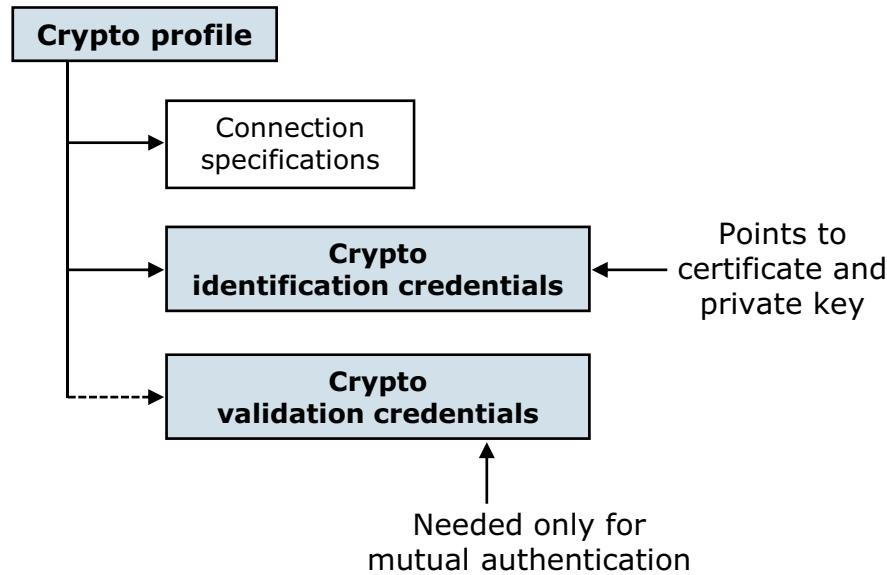
WE711 / ZE7111.0

### 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 2015

Figure 7-24. Step 1: Appliance supplies cryptographic certificate

WE711 / ZE7111.0

### 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 a **Reverse (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 a configuration interface with two main sections: 'Back End' and 'Front End'. In the 'Back End' section, there are fields for 'Remote Host' and 'Remote Port', both marked with a required asterisk (\*). In the 'Front End' section, there are fields for 'Local address' (set to 0.0.0.0) and 'Port Number' (set to 2048), also marked with a required asterisk (\*). Below these sections are two dropdown menus: 'Forward (Client) Crypto Profile' and 'Reverse (Server) Crypto Profile'. The 'Reverse (Server) Crypto Profile' dropdown is highlighted with a red rectangular box.

© Copyright IBM Corporation 2015

Figure 7-25. Step 2: Configuring SSL server crypto profile

WE711 / ZE7111.0

### Notes:

Configure an SSL server crypto profile with cryptographic objects that link to the certificate-key pair:

- On the **Configure XML Firewall** page, you can specify the server SSL crypto profile under **Front End**.



## Create an SSL server crypto profile

**Front End**

|                   |         |              |
|-------------------|---------|--------------|
| Device Address    | 0.0.0.0 | Select Alias |
| Device Port       | 2048 *  |              |
| SSL Server Crypto | (none)  | <b>[+]</b>   |
| Profile Name      |         |              |

**Server Side SSL**

|   |   |
|---|---|
| Server Credentials                            | The server certificate presented to clients making S            |
| Server Private Key                            | cert: (none)  |
| Private Key Password                          | <input type="password"/> Confirm Password                       |
| Use Password Alias                            | <input type="radio"/> on <input checked="" type="radio"/> off   |
| Server Certificate                            | cert: (none)  |
| Server Certificate Password                   | <input type="password"/> Confirm Password                       |
| Use Password Alias                            | <input type="radio"/> on <input checked="" type="radio"/> off   |
| Trusted Clients                               | Trusted certificate(s) presented by clients to the Da           |
| Authenticate/ Validate Certificates           | <input type="radio"/> on <input checked="" type="radio"/> off * |
| Trusted Certificates/ Certificate Authorities | <input type="button" value="Delete"/>                           |

Create a profile from:

- **[+] (new)** on the Configure XML Firewall page
- **Objects > Crypto Configuration > Crypto Profile**

**Crypto Profile**

Apply Cancel

Name:

Administrative state:  enabled  disabled

Identification Credentials: StudentIdCred

Validation Credentials: (none)

Ciphers: HIGH:MEDIUM:!aNULL:!eNULL

Options:

Enable default settings  
 Disable SSL version 2  
 Disable SSL version 3  
 Disable TLS version 1.0  
 Permit insecure SSL renegotiation  
 Enable compression  
 Disable TLS version 1.1  
 Disable TLS version 1.2

© Copyright IBM Corporation 2015

Figure 7-26. Create an SSL server crypto profile

WE711 / ZE7111.0

### 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 the crypto profile supports, which is negotiated during an SSL handshake.

The **Options** field provides options 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. To send a client CA list to clients, SSL servers are not required.

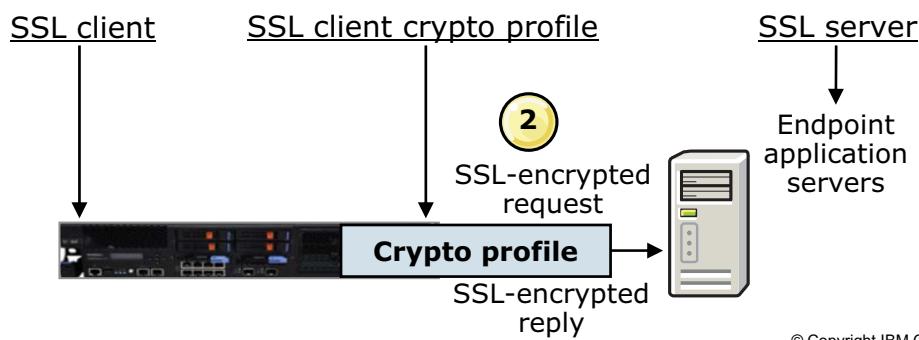
## Securing connection from appliance to external application server

To set up SSL between the appliance and external application server:

- The DataPower appliance validates the certificate that is supplied by the external application server
  - Configure an **SSL client crypto profile** with validation credentials to validate the presented certificate

The application server might request a certificate for the appliance (mutual authentication)

- Appliance can respond with the certificates in the **identification credentials**



© Copyright IBM Corporation 2015

Figure 7-27. Securing connection from appliance to external application server

WE711 / ZE7111.0

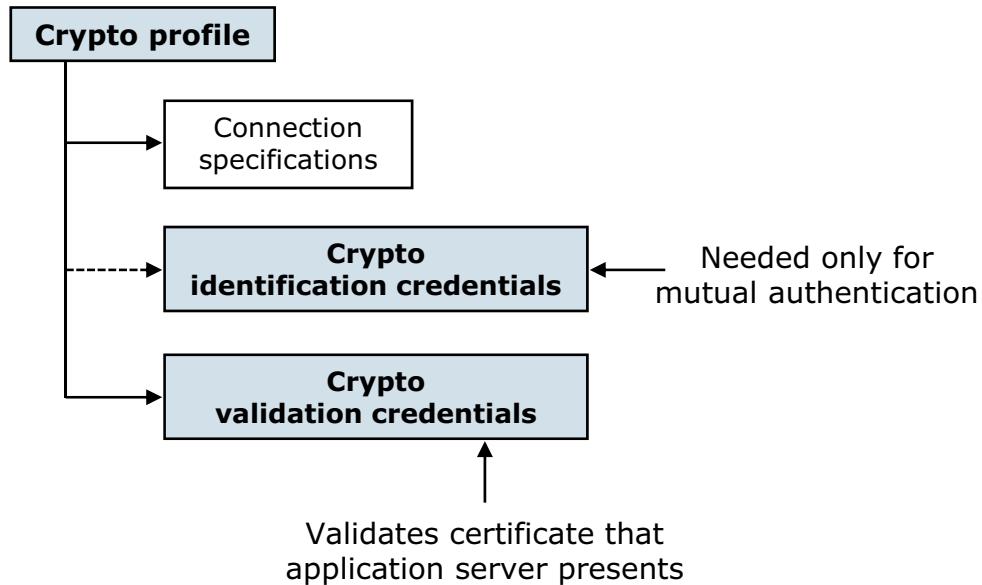
### Notes:

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 2015

Figure 7-28. Step 1: Appliance validates presented certificate

WE711 / ZE7111.0

### 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 a **Forward (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)

|  |                      |  |                                      |
|--|----------------------|--|--------------------------------------|
| <b>Back End</b>  |                      | <b>Front End</b>   |                                      |
| <b>Remote Host</b>   | <input type="text"/> | <b>Local address</b>   | <input type="text" value="0.0.0.0"/> |
| <b>Remote Port</b>   | <input type="text"/> | <b>Port Number</b>   | <input type="text" value="2048"/> *  |
| <b>Forward (Client) Crypto Profile</b><br><input type="button" value="(none)"/> <input type="button"/> <input type="button"/> <input type="button"/> |                      | <b>Reverse (Server) Crypto Profile</b><br><input type="button" value="(none)"/> <input type="button"/> <input type="button"/> <input type="button"/> |                                      |

© Copyright IBM Corporation 2015

Figure 7-29. Step 2: Configuring an SSL client crypto profile

WE711 / ZE7111.0

### 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

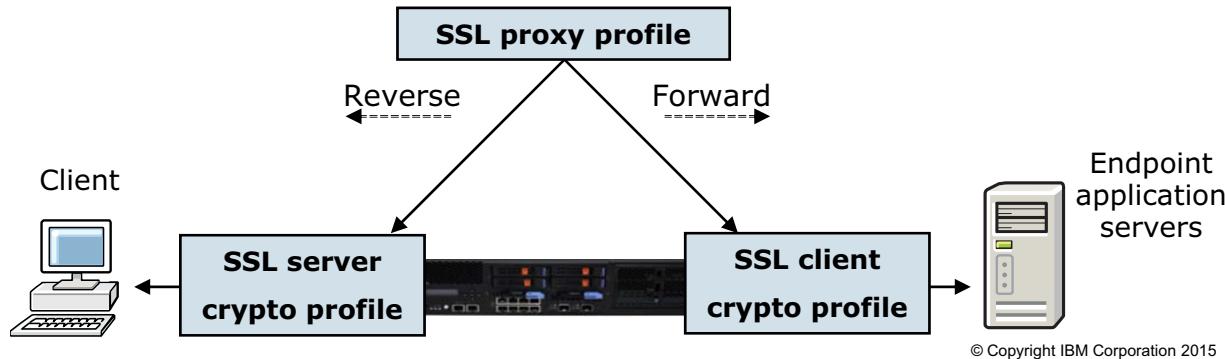


Figure 7-30. The SSL proxy profile

WE711 / ZE7111.0

### 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 supported

SSL Proxy Profile

Apply Cancel

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 2015

Figure 7-31. SSL proxy profile when the appliance is the SSL server

WE711 / ZE7111.0

### Notes:

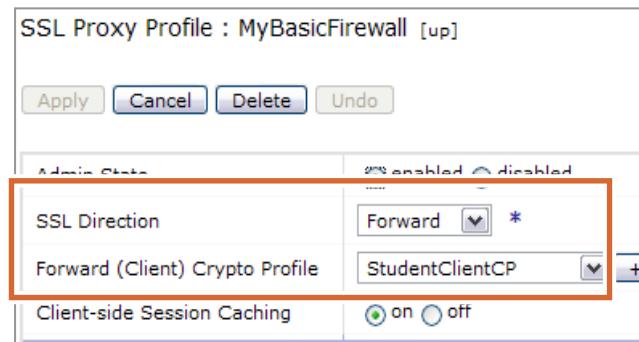
Set **Client Authentication is optional** to control when SSL client authentication is optional. When set to **on**, client authentication is not required. When there is no client certificate, the request does not fail. When set to **off** (Default), the SSL server requires client authentication only when the server crypto profile has an assigned validation credential.

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 credential.



## 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



© Copyright IBM Corporation 2015

Figure 7-32. SSL proxy profile when the appliance is the SSL client

WE711 / ZE7111.0

### 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**

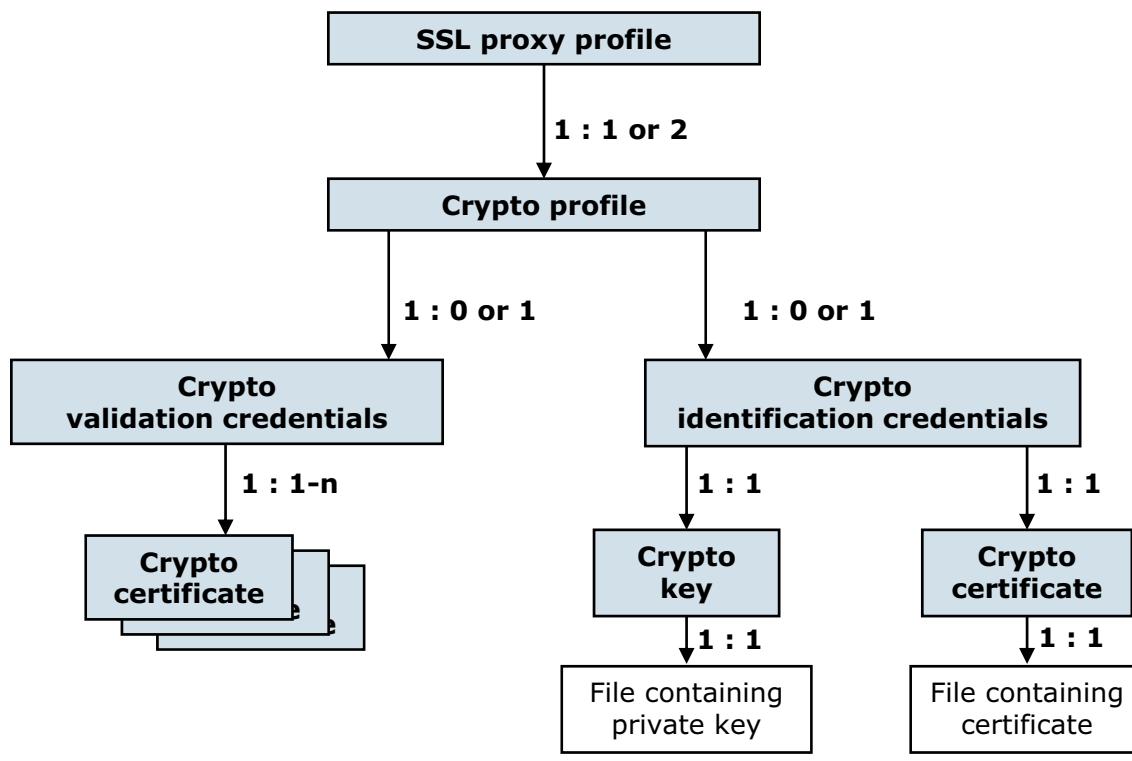
© Copyright IBM Corporation 2015

Figure 7-33. SSL Proxy Profile list

WE711 / ZE7111.0

### Notes:

## Recap: SSL - crypto object relationships



© Copyright IBM Corporation 2015

Figure 7-34. Recap: SSL - crypto object relationships

WE711 / ZE7111.0

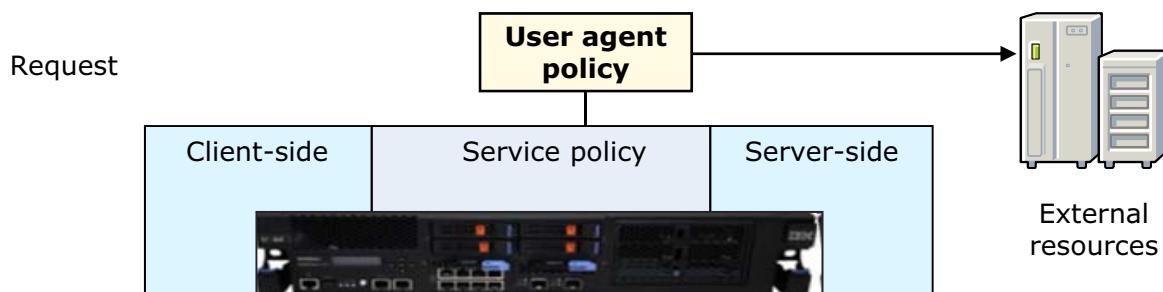
### Notes:

Numerous objects are involved in configuring SSL support. This graphic provides an opportunity to review them.

## User agent

A client that operates on behalf of a service when establishing a connection to a remote server

- Commonly used for remote requests that are 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 2015

Figure 7-35. User agent

WE711 / ZE7111.0

### Notes:

A user agent uses one or more policies to connect to an external server or back side service.



## What is a “user agent”?

- The User Agent can be thought of as a utility object that other higher-level DataPower objects use
- *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 (user ID, password) into requests
  - **Soap-Action Policy:** Inject SOAPAction headers into web services requests
  - **Pubkey-Auth Policy:** Private key that is used to validate a certificate that is presented in a request for authentication
  - **Allow-Compression Policy:** Allow or 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 that are based on URL matching
  - **Chunked Uploads Policy:** Similar to Allow-Compression Policy, is used for HTTP 1.1 chunked uploading
  - **FTP Client Policies:** Is used to match FTP URLs with client policies to control options for the outgoing FTP connection
  - **SFTP Client Policies:** Same as FTP Client Policies but with SFTP

© Copyright IBM Corporation 2015

Figure 7-36. What is a “user agent”?

WE711 / ZE7111.0

### 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

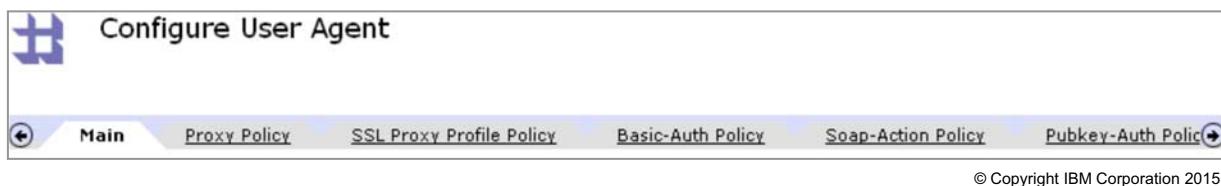


Figure 7-37. Configuring a user agent

WE711 / ZE7111.0

### 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

A screenshot of a software interface titled "SSL Proxy Profile Policy". It displays a table with two columns: "URL Matching Expression" and "SSL proxy profile". A single row is present with the expression "/resource" and the profile "EastAddressSSLProfile". To the right of the table are icons for edit, delete, and add. An "Add" button is located at the bottom right of the table area.

| SSL Proxy Profile Policy |                       |
|--------------------------|-----------------------|
| URL Matching Expression  | SSL proxy profile     |
| /resource                | EastAddressSSLProfile |

Add

© Copyright IBM Corporation 2015

Figure 7-38. Create a user agent configuration

WE711 / ZE7111.0

### 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 DataPower tools to generate cryptographic keys
- Create a crypto identification credential object that contains a matching public and private key
- Create a crypto validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up to date
- Configure an SSL proxy profile that accepts an SSL connection request from a client
- Configure an SSL proxy profile that initiates an SSL connection from a DataPower service

© Copyright IBM Corporation 2015

Figure 7-39. Unit summary

WE711 / ZE7111.0

### 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 2015

Figure 7-40. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

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



## 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 2015

Figure 7-41. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 5



Creating cryptographic objects and  
configuring SSL

© Copyright IBM Corporation 2015

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

10.1

Figure 7-42. Exercise 5

WE711 / ZE7111.0

### Notes:

## Exercise objectives

After completing this exercise, you should be able to:

- Generate crypto keys by using the 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 2015

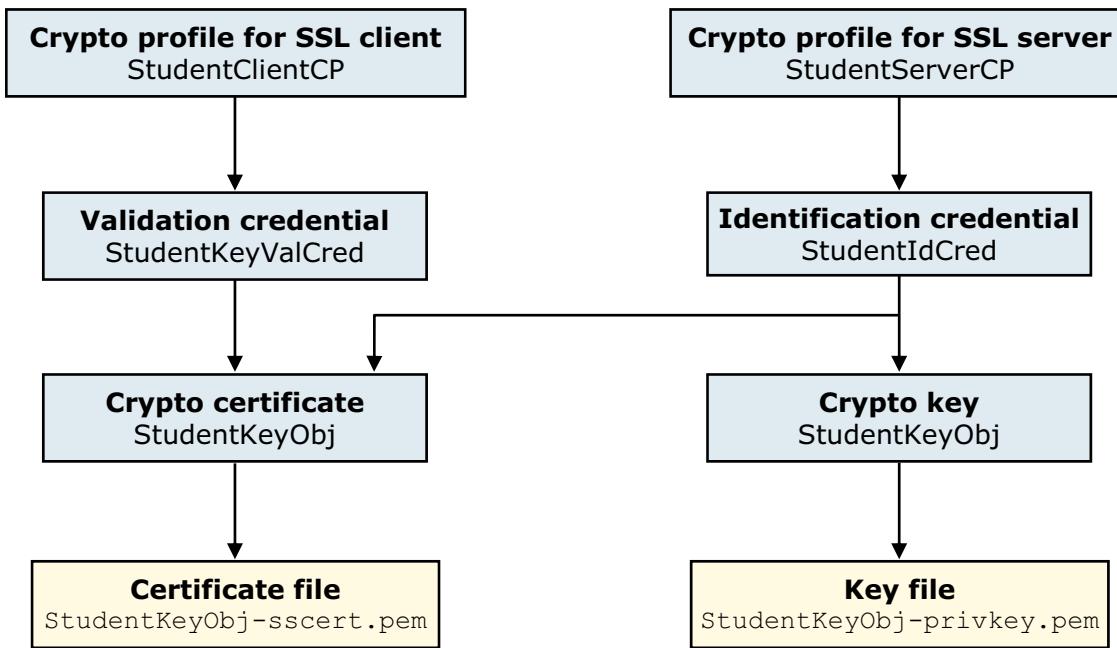
Figure 7-43. Exercise objectives

WE711 / ZE7111.0

### Notes:

## Exercise overview (1 of 2)

- Create the files and objects that are needed to configure the SSL connections for the services



© Copyright IBM Corporation 2015

Figure 7-44. Exercise overview (1 of 2)

WE711 / ZE7111.0

### Notes:

## Exercise overview (2 of 2)

1. Add an HTTPS front side handler that acts as the SSL server
2. Use the HTTPS protocol in the back-end URL to act as the SSL client
3. Test with SoapUI

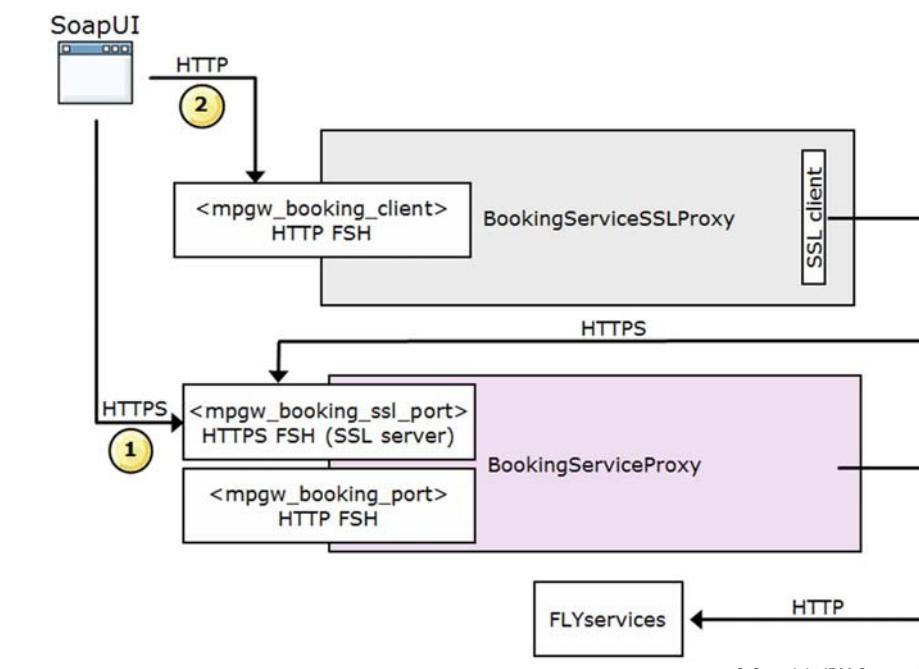


Figure 7-45. Exercise overview (2 of 2)

WE711 / ZE7111.0

### Notes:



# Unit 8. XML and web services security overview

## What this unit is about

This unit describes the features of the web services security specification. This specification uses XML encryption and XML signatures to provide message level security to ensure message confidentiality and integrity. You also learn how to use the DataPower appliance 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
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## 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 2015

---

Figure 8-1. Unit objectives

WE711 / ZE7111.0

### 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 2015

Figure 8-2. Review of basic security terminology

WE711 / ZE7111.0

### 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.

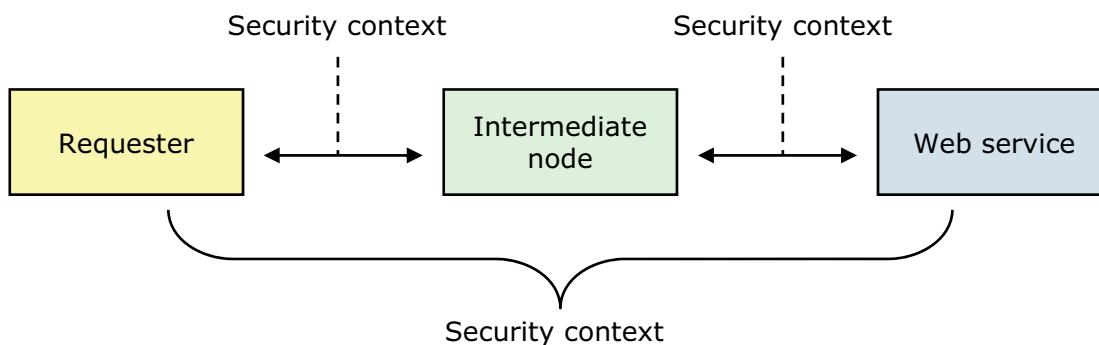
**Confidentiality** ensures that only authorized parties have access to protected resources. The effect of confidentiality is to keep private data or resources secret. This quality is often implemented through the encryption of data, in which only authorized parties have the means of making obscured 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 2015

Figure 8-3. Web services security

WE711 / ZE7111.0

### 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
  - User name 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 2015

Figure 8-4. Components of WS-Security

WE711 / ZE7111.0

### 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 2015

Figure 8-5. Specifying security in SOAP messages

WE711 / ZE7111.0

### 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 2015

Figure 8-6. Scenario 1: Ensure confidentiality with XML encryption

WE711 / ZE7111.0

### 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 2015

Figure 8-7. XML encryption and WS-Security

WE711 / ZE7111.0

### 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 2015

Figure 8-8. DataPower support for XML encryption

WE711 / ZE7111.0

### 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

| <b>Encrypt</b>  |  |
|---|--|
| <b>Envelope Method</b>                                  | <input checked="" type="radio"/> WSSec Encryption<br><input type="radio"/> Standard XML Encryption<br><input type="radio"/> Advanced   |
| <b>Message Type</b>                                     | <input checked="" type="radio"/> SOAP Message<br><input type="radio"/> Raw XML Document<br><input type="radio"/> Selected Elements (Field-Level)<br><input type="radio"/> Advanced |
| <b>Asynchronous</b>                                     | <input type="radio"/> on <input checked="" type="radio"/> off  |
| <b>Message and Attachment Handling</b>                  | Message Only <input type="button" value="..."/> <input type="checkbox"/> Save  |
| <b>Encryption Key Type</b>                              | Use Ephemeral Key Transported by Asymmetric Algorithm <input type="checkbox"/>   |
| <b>Use Dynamically Configured Recipient Certificate</b> | <input type="radio"/> on <input checked="" type="radio"/> off <input type="checkbox"/> Save  |
| <b>One Ephemeral Key</b>                                | <input type="radio"/> on <input checked="" type="radio"/> off <input type="checkbox"/> Save  |
| <b>Recipient Certificate</b>                            | (none) <input type="button" value="..."/> <input type="button" value="+"/> <input type="button" value="..."/> <input type="checkbox"/> Save  |
| <b>WS-Security Version</b>                              | 1.0 <input type="button" value="..."/> <input type="checkbox"/> Save   |

© Copyright IBM Corporation 2015

Figure 8-9. Encrypt action

WE711 / ZE7111.0

### 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:

- The 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. Therefore, it 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 2015

Figure 8-10. Decrypt action

WE711 / ZE7111.0

### Notes:

On the **Advanced** tab, you can override the style sheet that is used to decrypt. The default file that is used is `store:///decrypt.xsl`.



## 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

The screenshot shows the 'Document Crypto Map : Name\_DCM' configuration window. At the top, there are buttons for 'Apply', 'Cancel', 'Undo', and links for 'Export', 'View Log', 'View Status', and 'Help'. Below these are sections for 'Admin State' (enabled), 'Comments', 'Operation' (set to 'Decrypt'), and 'XPath Expression'. The XPath expression is defined as `/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByName']/*[local-name()='EncryptedData']`. There are buttons for 'Add' and 'XPath Tool' at the bottom of the expression input field. A legend below the main window defines 'Message Type' options: 'Entire Message/Document' (radio button), 'Selected Elements (Field-Level)' (radio button, selected), 'Advanced' (radio button), and '\*' (radio button). The 'Document Crypto Map' section shows 'Name\_DCM' with a dropdown menu, a '+' button, and a '...' button, where the '+' button is highlighted with a red box.

Figure 8-11. Field-level encryption and decryption

WE711 / ZE7111.0

### 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

Build XPath Expression from sample XML File

Select sample XML document

URL of Sample XML Document

local:///     \*

Namespace Handling

local  
 prefix  
 uri

Selected XPath Expression

XPath \*

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

© Copyright IBM Corporation 2015

Figure 8-12. XPath tool

WE711 / ZE7111.0

### 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 with specific elements highlighted in red: <xenc:EncryptedKey>, <xenc:EncryptedData>, <EncryptionMethod />, <CipherData>, <CipherValue>, and <q0:findByName>. Brackets on the right side group these elements into two categories: 'Key that is used to encrypt message' (covering the <xenc:EncryptedKey> block) and 'Field-level XML encryption' (covering the <xenc:EncryptedData> block under <q0:findByName>).

© Copyright IBM Corporation 2015

Figure 8-13. Sample encrypted SOAP message

WE711 / ZE7111.0

### 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 assurance that message is not changed
  - Mismatch between decrypted hash (from signature) and computed hash (from cleartext) indicates that the message is modified
  - Signatures provide a strong indication of identity
  - Only the holder of the private key can create a 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 a message digest of the received message
  - Obtain the original message digest by decrypting the signature from the received message by using the certificate that holds the public key
  - Compare the computed message digest against the original message digest

© Copyright IBM Corporation 2015

Figure 8-14. Scenario 2: Ensure integrity with XML signatures

WE711 / ZE7111.0

### 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.

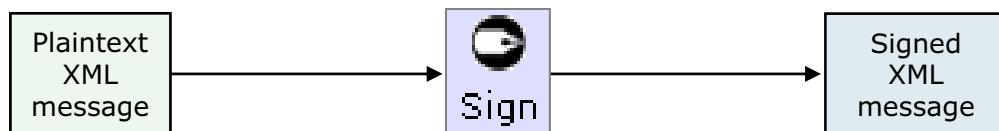
Consider the following example:

```
<transaction-info>
<user-id>jsmith</user-id>
<action>buy</action>
<symbol>IBM</symbol>
</transaction-info>
```

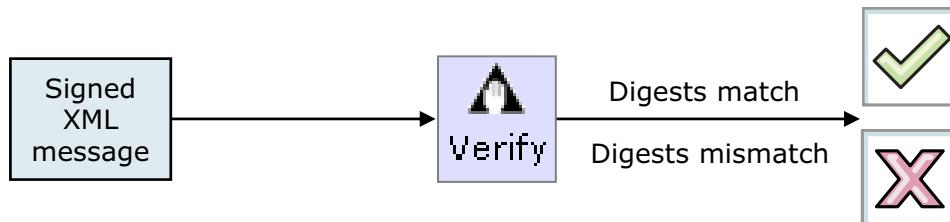
You can sign the value within the symbol element, the entire symbol element (including the value), or a group of elements within transaction-info.

## DataPower support for XML signature

- Apply an 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 2015

Figure 8-15. DataPower support for XML signature

WE711 / ZE7111.0

### 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.



## Sign action

- The **Sign** action signs specific elements or the entire message by using a crypto key object

– **Envelope Method:** Determines placement of the signature in a message

– **Message Type**

– **Key:** Crypto key object that is used to sign a message

– **Certificate:** Crypto certificate object that is associated with the crypto key object

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

© Copyright IBM Corporation 2015

Figure 8-16. Sign action

WE711 / ZE7111.0

### 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** is 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 a passed certificate
- Validation Credential**
  - One or more certificate objects that are used to validate the signer certificate

**Verify**

|   |  |
|---|--|
| <b>Asynchronous</b>   | <input type="radio"/> on <input checked="" type="radio"/> off  |
| <b>Signature Verification Type</b>  | RSA/DSA Signatures <input type="button" value="▼"/> <input type="checkbox"/> Save  |
| <b>Optional Signer Certificate</b>  | <input type="text"/>   |
| <b>Validation Credential</b>  | AliceValidCred <input type="button" value="▼"/> <input type="button" value="+"/> <input type="button" value="..."/> <input checked="" type="checkbox"/> Save |
| <b>Output</b>   |  |
| <b>Output</b>   | <input type="text"/>   |
| <input type="button" value="Delete"/> <input type="button" value="Done"/> <input type="button" value="Cancel"/> |  |

© Copyright IBM Corporation 2015

Figure 8-17. Verify action

WE711 / ZE7111.0

### 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:/// verify.xsl   |
| 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  |

© Copyright IBM Corporation 2015

Figure 8-18. Verify action: Advanced tab

WE711 / ZE7111.0

### 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

The screenshot shows the 'Document Crypto Map' configuration screen. On the left, under 'Message Type', the 'Selected Elements (Field-Level)' option is selected. On the right, the 'XPath Expression' field contains the path '/SOAP-ENV:Envelope/SOAP-ENV:Body/q0:findByName/name/title'. The 'Operation' dropdown is set to 'Sign (WS-Security)'. The 'Admin State' is enabled.

© Copyright IBM Corporation 2015

Figure 8-19. Field-level message signature and verification

WE711 / ZE7111.0

### 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 2015

Figure 8-20. Sample signed SOAP message

WE711 / ZE7111.0

### 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

### Signing: Message integrity

**Receiver knows that only sender can build the signature**

|  |  |
|--|--|
| Sender uses the <b>Sign</b> action and its own private key to add a signature to the message | Receiver uses the sender public key or certificate to <i>verify</i> the signature in the message |
|--|--|

### Encrypting: Message confidentiality

**Sender knows that only receiver can decrypt the message**

|  |   |
|--|---|
| Sender uses the receiver public key or certificate to <i>encrypt</i> a message | Receiver <i>decrypts</i> the message by using its own private key |
|--|---|

Sign a message, then encrypt, for best confidentiality and integrity

© Copyright IBM Corporation 2015

Figure 8-21. Summary of security and keys

WE711 / ZE7111.0

### 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 2015

Figure 8-22. Unit summary

WE711 / ZE7111.0

### 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 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 2015

Figure 8-23. Checkpoint questions

WE711 / ZE7111.0

### 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 2015

Figure 8-24. Checkpoint answers

WE711 / ZE7111.0

### Notes:



## Exercise 6



Web service encryption and digital signatures

© Copyright IBM Corporation 2015

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

10.1

Figure 8-25. Exercise 6

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2015

Figure 8-26. Exercise objectives

WE711 / ZE7111.0

### Notes:

## Exercise overview

1. Create crypto DataPower objects
2. Configure BookingServiceProxy to verify a signed message
3. Test the BookingServiceProxy signature verification by sending a signed message from SoapUI
4. Configure BookingServiceProxy to sign the response message
5. Test that the BookingServiceProxy returns a signed message to SoapUI
6. Configure BookingServiceProxy to decrypt a message
7. Test the BookingServiceProxy message decryption by ending an encrypted message from SoapUI
8. Configure BookingServiceProxy to encrypt the response message
9. Test that the BookingServiceProxy returns an encrypted message to SoapUI

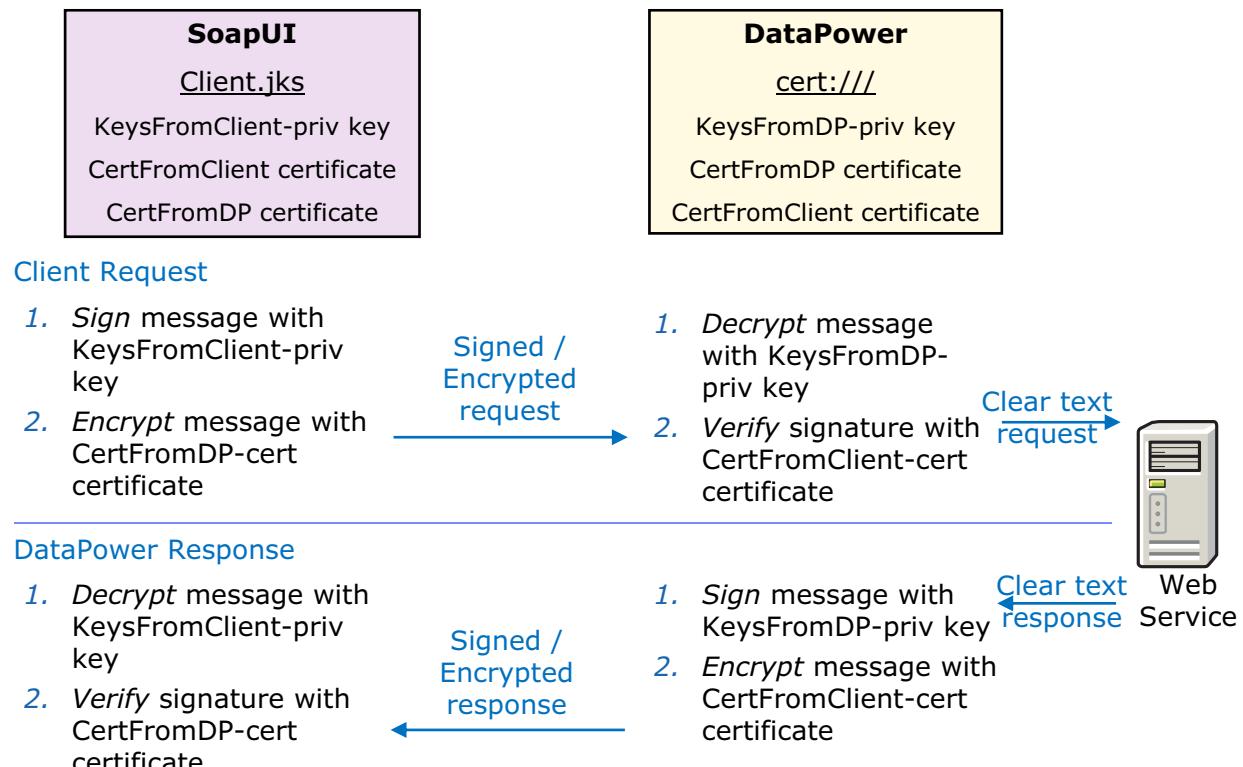
© Copyright IBM Corporation 2015

Figure 8-27. Exercise overview

WE711 / ZE7111.0

### Notes:

## Completed exercise



© Copyright IBM Corporation 2015

Figure 8-28. Completed exercise

WE711 / ZE7111.0

### Notes:



# Unit 9. Authentication, authorization, and auditing (AAA)

## What this unit is about

This unit describes the authentication, authorization, and auditing (AAA) framework within the IBM DataPower Gateway 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 DataPower 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
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## Unit objectives

After completing this unit, you should be able to:

- Describe the AAA framework within the DataPower 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 2015

Figure 9-1. Unit objectives

WE711 / ZE7111.0

### 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 2015

Figure 9-2. Authentication, authorization, and auditing

WE711 / ZE7111.0

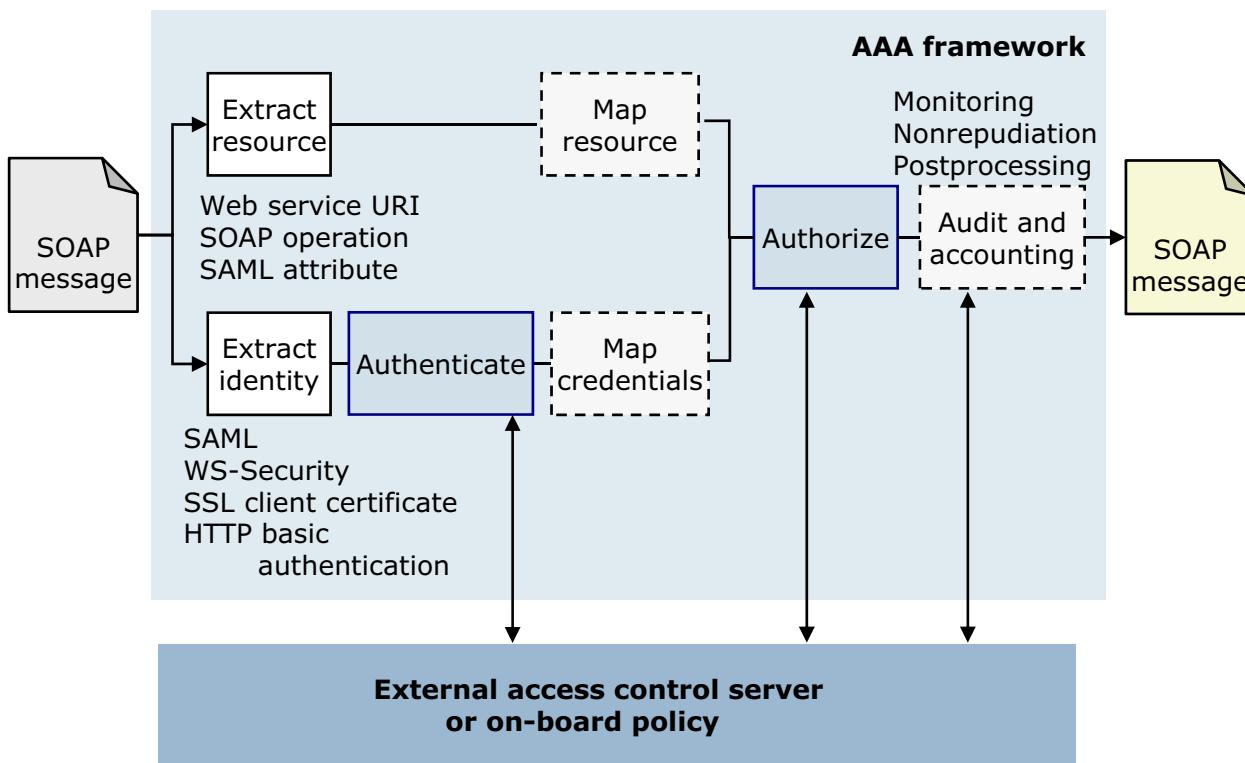
### Notes:

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 2015

Figure 9-3. Authentication and authorization framework

WE711 / ZE7111.0

### 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.

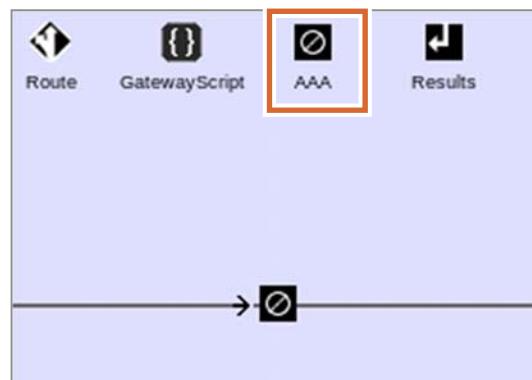
The various tokens that can be generated are:

- 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 2015

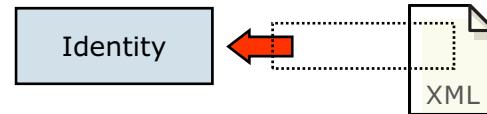
Figure 9-4. AAA action and access control policy

WE711 / ZE7111.0

### Notes:

## How to define an access control policy (1 of 2)

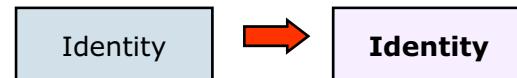
1. Define one or more identity extraction methods



2. Define the authentication method



3. Map authentication credentials (optional)



© Copyright IBM Corporation 2015

Figure 9-5. How to define an access control policy (1 of 2)

WE711 / ZE7111.0

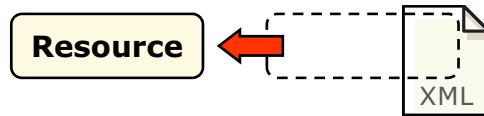
### Notes:

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)

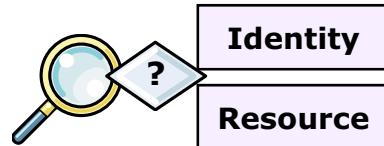
4. Define resource extraction methods



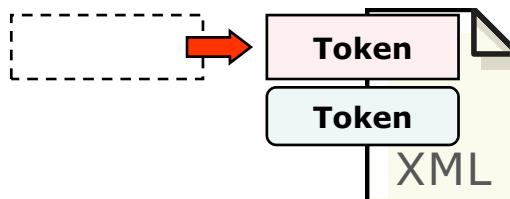
5. Map requested resources (optional)



6. Define the authorization method



7. Specify postprocessing actions (optional)



© Copyright IBM Corporation 2015

Figure 9-6. How to define an access control policy (2 of 2)

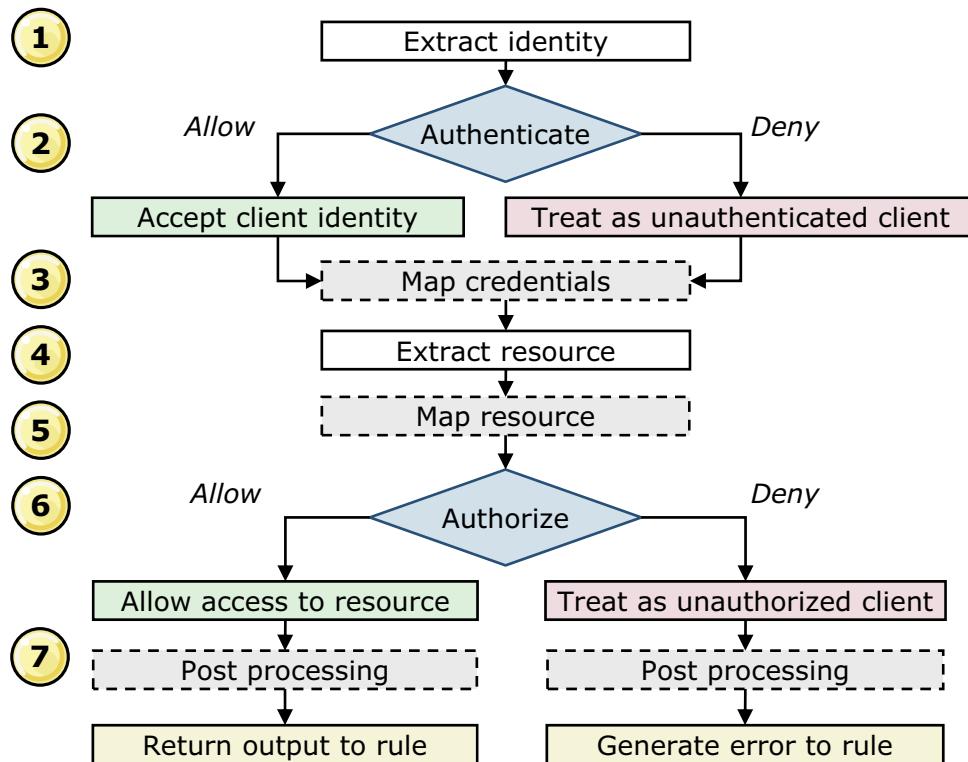
WE711 / ZE7111.0

### 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 2015

Figure 9-7. Access control policy processing

WE711 / ZE7111.0

### 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 2015

Figure 9-8. Scenario 1: Authorize authenticated clients

WE711 / ZE7111.0

### Notes:

In this scenario, the client includes a WS-Security user name 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 user name 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:Username>Alice</wsse:Username>
                <wsse:Password>ond3mand</wsse:Password>
            </wsse:UsernameToken>
        </wsse:Security>
    </soap:Header>
    <soap:Body>
        <q0:retrieveAll />
    </soap:Body>
</soap:Envelope>
```

© Copyright IBM Corporation 2015

Figure 9-9. Scenario 1: Sample SOAP request message

WE711 / ZE7111.0

### Notes:

The WS-Security user name 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. Because 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.

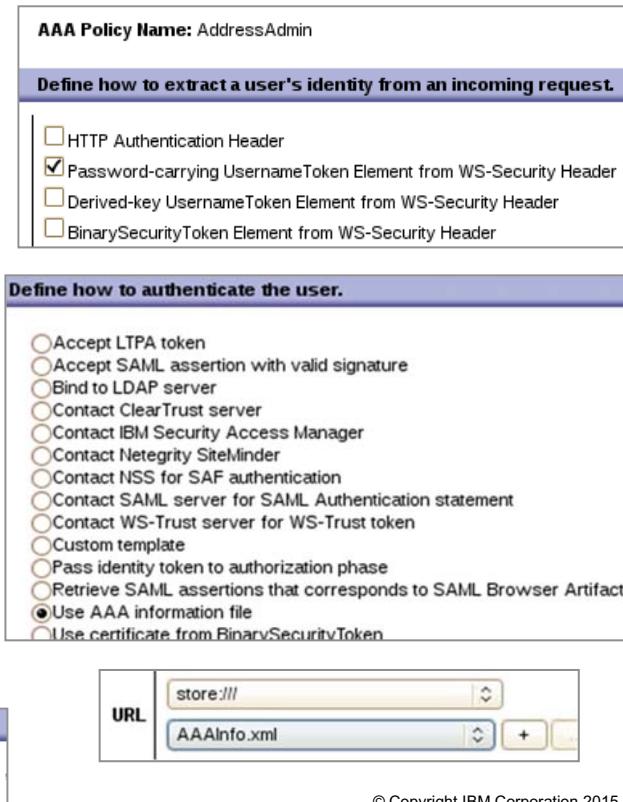
For the sake of brevity, the URI for the **wsse** namespace declaration is truncated. For the URI, see the WS-Security V1.1 specification.

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 and authenticate 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**



The screenshot shows the 'AAA Policy Name: AddressAdmin' configuration screen. It has two main sections:

- 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

**Define how to map credentials.**

|        |      |
|--------|------|
| Method | None |
|--------|------|

**URL**

|             |                   |
|-------------|-------------------|
| store:///   | [button]          |
| AAAInfo.xml | [button] [button] |

© Copyright IBM Corporation 2015

Figure 9-10. Scenario 1: Identify and authenticate the client

WE711 / ZE7111.0

### 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**

|   |   |
|---|---|
| <b>Resource Identification Methods</b>  | <input type="checkbox"/> URL Sent to Back End<br><input type="checkbox"/> URL Sent by Client<br><input type="checkbox"/> URI of Toplevel Element in the Message<br><input checked="" type="checkbox"/> Local Name of Request Element<br><input type="checkbox"/> HTTP Operation (GET/POST)<br><input type="checkbox"/> XPath Expression<br><input type="checkbox"/> Processing Metadata |
| <b>Define how to map resources.</b><br><b>Method</b> <input style="width: 150px;" type="text" value="None"/> *  |   |
| <b>Define how to authorize a request.</b> <ul style="list-style-type: none"> <li><input type="radio"/> AAA information file</li> <li><input checked="" type="radio"/> Allow any authenticated client</li> <li><input type="radio"/> Always allow</li> <li><input type="radio"/> Check membership in LDAP group</li> <li><input type="radio"/> Contact ClearTrust server</li> <li><input type="radio"/> Contact IBM Security Access Manager</li> <li><input type="radio"/> Contact Netegrity SiteMinder</li> <li><input type="radio"/> Contact NSS for SAF authorization</li> <li><input type="radio"/> Contact OAuth STS</li> <li><input type="radio"/> Custom template</li> <li><input type="radio"/> Generate SAML Attribute query</li> <li><input type="radio"/> Generate SAML Authorization query</li> <li><input type="radio"/> Use SAML attributes from authentication</li> <li><input type="radio"/> Use XACML Authorization decision</li> </ul> |   |

© Copyright IBM Corporation 2015

Figure 9-11. Scenario 1: Authorize access to resources

WE711 / ZE7111.0

### 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 2015

Figure 9-12. Scenario 2: Security token conversion

WE711 / ZE7111.0

### 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 2015

Figure 9-13. Scenario 2: Sample HTTP request message

WE711 / ZE7111.0

### 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 an 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 and authenticate the client

1. Create a AAA policy object on the DataPower SOA appliance
2. Extract the client's identity with the **HTTP's 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**

|   |
|---|
| <input checked="" type="checkbox"/> HTTP's Authentication Header<br><input type="checkbox"/> Password-carrying UsernameToken Element from WS-Security Header<br><input type="checkbox"/> Derived-key UsernameToken Element from WS-Security Header<br><input type="checkbox"/> BinarySecurityToken Element from WS-Security Header<br><input type="checkbox"/> WS-SecureConversation Identifier<br><input type="checkbox"/> WS-Trust Base or Supporting Token<br><input type="checkbox"/> Kerberos AP-REQ from WS-Security Header<br><input type="checkbox"/> Kerberos AP-REQ from SPNEGO Token<br><input type="checkbox"/> Subject DN of the SSL Certificate from the Connection Peer<br><input type="checkbox"/> Name from SAML Attribute Assertion<br><input type="checkbox"/> Name from SAML Authentication Assertion |
|---|

|   |
|---|
| <input type="radio"/> Contact SAML server for SAML Authentication statement<br><input type="radio"/> Contact WS-Trust server for WS-Trust token<br><input type="radio"/> Custom template<br><input checked="" type="radio"/> Pass identity token to authorization phase<br><input type="radio"/> Retrieve SAML assertions that corresponds to SAML Browser Artifact<br><input type="radio"/> Use AAA information file<br><input type="radio"/> Use certificate from BinarySecurityToken |
|---|

|                                       |  |
|---------------------------------------|--|
| <b>Define how to map credentials.</b> |  |
| Method                                | <input type="text" value="none"/>  *<br><small>(Required)</small> |

© Copyright IBM Corporation 2015

Figure 9-14. Scenario 2: Identify and authenticate the client

WE711 / ZE7111.0

### 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

|  |   |
|--|---|
| <b>Resource Identification Methods</b>   | <input type="checkbox"/> URL Sent to Back End<br><input type="checkbox"/> URL Sent by Client<br><input type="checkbox"/> URI of TopLevel Element in the Message<br><input checked="" type="checkbox"/> Local Name of Request Element<br><input type="checkbox"/> HTTP Operation (GET/POST)<br><input type="checkbox"/> XPath Expression<br><input type="checkbox"/> Processing Metadata<br><br><i>*</i> |
| <b>Define how to authorize a request.</b>  |   |
| <input type="radio"/> AAA information file<br><input type="radio"/> Allow any authenticated client<br><input checked="" type="radio"/> Always allow<br><input type="radio"/> Check membership in LDAP group<br><input type="radio"/> Contact ClearTrust server |   |
| <b>Choose any post processing.</b>   |   |
| <b>Include Password</b><br><br><b>WS-Security UsernameToken Password Type</b><br><br><b>Actor/Role Identifier</b>  | <input checked="" type="radio"/> on <input type="radio"/> off<br><br><input type="button" value="Digest"/>  |

© Copyright IBM Corporation 2015

Figure 9-15. Scenario 2: Authorize access to resources

WE711 / ZE7111.0

### 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 user name 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).

The added WS-Security Username token for a username of “student”, a password of “web1sphere”, and a password type of “Text” is:

```
<wsse:UsernameToken>
  <wsse:Username>student</wsse:Username>
  <wsse:Password
    Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">websphere</wsse:Password>
  <wsse:Nonce
    EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message"
```

```
-security-1.0#Base64Binary">N2FmMzg3OTEtZDI4OS00MzkzLThmYWUtNzM3MzhkYmRmM2Zh</wsse
:Nonce>
```

```
<wsu:Created
```

```
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd"
```

```
>2015-03-24T21:16:44Z</wsu:Created>
```

```
</wsse:UsernameToken>
```

The same token, with a password type of “Digest” is:

```
<wsse:UsernameToken>
```

```
<wsse:Username>student</wsse:Username>
```

```
<wsse:Password
```

```
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profi
le-1.0#PasswordDigest">JDqa4I7DaWicLrj+ykiSBQT0MFc=</wsse:Password>
```

```
<wsse:Nonce
```

```
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message
-security-1.0#Base64Binary">YzVhNmQ3OTctZmE5Mi00NjdhLWFiYmYtZDUyNDVmNmRjNTEw</wsse
:Nonce>
```

```
<wsu:Created
```

```
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd"
```

```
>2015-03-24T21:25:12Z</wsu:Created>
```

```
</wsse:UsernameToken>
```

In both cases, the optional <Nonce> and <Created> elements are specified. These elements and the password can be used to protect against replay attacks.

## 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 2015

Figure 9-16. Scenario 3: Multiple identity extraction methods

WE711 / ZE7111.0

### 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 and authenticate 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.**

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 Basic or Supporting token

**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

**Define how to map credentials.**

|        |                                   |   |
|--------|-----------------------------------|---|
| Method | <input type="text" value="none"/> | * |
|--------|-----------------------------------|---|

© Copyright IBM Corporation 2015

Figure 9-17. Scenario 3: Identify and authenticate the client

WE711 / ZE7111.0

### Notes:

## Scenario 3: LDAP details

When connecting to LDAP, further details are needed

1. Specify the LDAP server URL
2. Indicate the **LDAP Bind DN** and **LDAP Bind Password** for the LDAP query
3. Use the **LDAP Search Attribute** fields to verify the password digest from a WS-Security Username Token
4. Use the **LDAP Prefix** and **LDAP Suffix** fields to build an LDAP query
  - For example, the extracted identity of **John** would result in a distinguished name of **cn=John,dc=IBM,dc=com**

|                                       |  |
|---------------------------------------|--|
| <b>LDAP Load Balancer Group</b>       | (none) <input type="button" value="+"/> <input type="button" value="..."/> |
| <b>Host</b>                           | <input type="text"/>   |
| <b>Port</b>                           | 389  |
| <b>SSL Proxy Profile</b>              | (none) <input type="button" value="+"/>                                    |
| <b>LDAP Bind DN</b>                   | <input type="text"/>   |
| <b>LDAP Bind Password</b>             | <input type="password"/><br><input type="password"/>                       |
| <b>LDAP Search Attribute</b>          | userPassword   |
| <b>LDAP Version</b>                   | v2 <input type="button" value="..."/>                                      |
| <b>LDAP Search for DN</b>             | <input type="radio"/> on <input checked="" type="radio"/> off              |
| <b>LDAP Prefix</b>                    | <input type="text"/> cn=   |
| <b>LDAP Suffix</b>                    | <input type="text"/> dc=ibm,dc=com   |
| <b>User auxiliary LDAP attributes</b> | <input type="text"/>   |
| <b>LDAP Read Timeout</b>              | 60   |

© Copyright IBM Corporation 2015

Figure 9-18. Scenario 3: LDAP details

WE711 / ZE7111.0

### Notes:

If **Bind to LDAP server** is selected, the page repaints to display entry fields for the LDAP details.

The targeted LDAP server can be a load balancer group that is composed of multiple LDAP servers.

The DataPower to LDAP server connection is usually over an SSL connection. The SSL proxy profile object defines the SSL connection information.



## 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

|  |  |
|--|--|
| <b>Resource Identification Methods</b> | <input type="checkbox"/> URL Sent to Back End<br><input type="checkbox"/> URL Sent by Client<br><input type="checkbox"/> URI of Toplevel Element in the Message<br><input checked="" type="checkbox"/> Local Name of Request Element<br><input type="checkbox"/> HTTP Operation (GET/POST)<br><input type="checkbox"/> XPath Expression<br><input type="checkbox"/> Processing Metadata<br>* |
|--|--|

- 6. Leave the resource mapping method at none**

|                                     |      |   |
|-------------------------------------|------|---|
| <b>Define how to map resources.</b> |      |   |
| Method                              | none | * |

- 7. For the authorization method, allow any request from an authenticated client to proceed**

|   |                                |  |
|---|--------------------------------|--|
| <b>Define how to authorize a request.</b> |                                |  |
| <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 2015

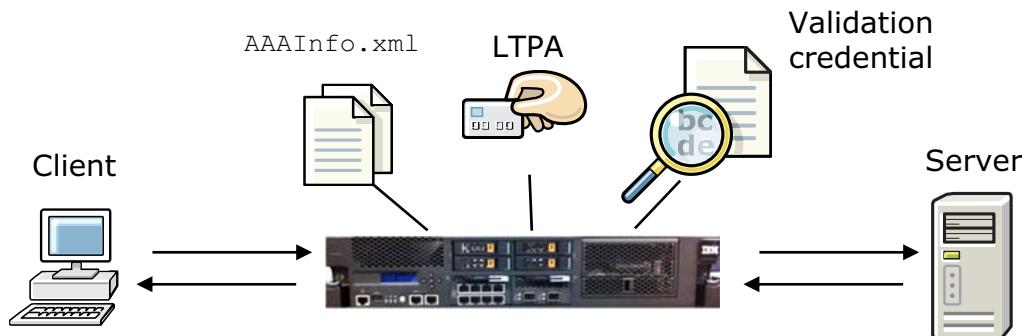
Figure 9-19. Scenario 3: Authorize access to resources

WE711 / ZE7111.0

### 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 2015

Figure 9-20. Internal access control resources

WE711 / ZE7111.0

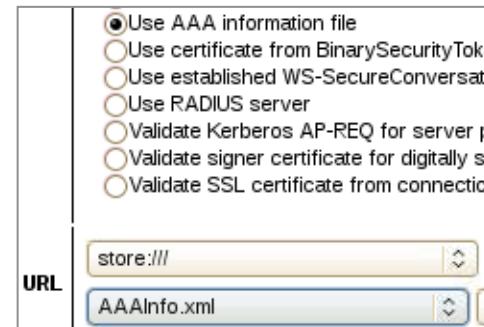
### Notes:

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 2015

Figure 9-21. AAA XML file

WE711 / ZE7111.0

### 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 2015

Figure 9-22. Example AAA XML file

WE711 / ZE7111.0

### 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 2015

Figure 9-23. Lightweight Third Party Authentication

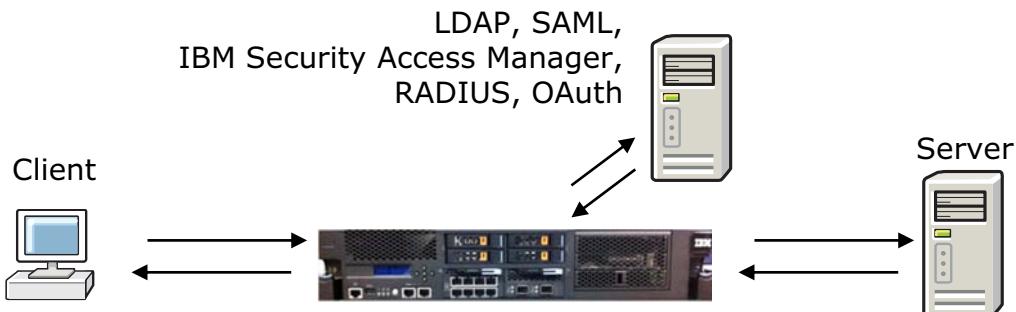
WE711 / ZE7111.0

### 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](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 Security 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 2015

Figure 9-24. External access control resource

WE711 / ZE7111.0

### 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 2015

Figure 9-25. Lightweight Directory Access Protocol

WE711 / ZE7111.0

### 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 2015

Figure 9-26. Security Assertion Markup Language

WE711 / ZE7111.0

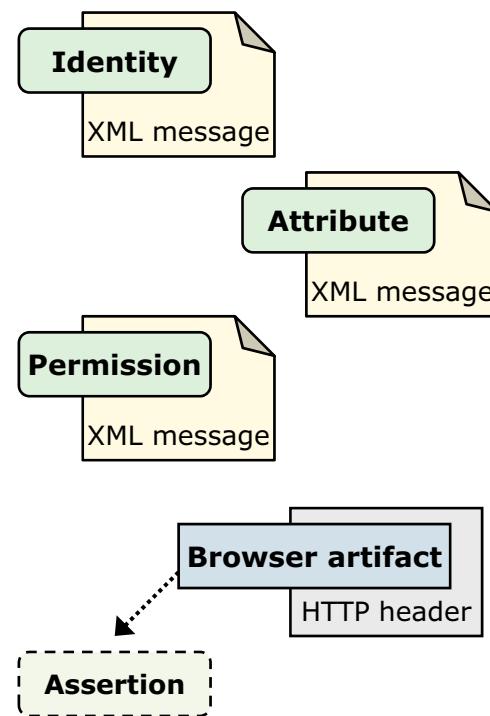
### Notes:

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 another entity verifies
  - **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 2015

Figure 9-27. Types of SAML assertions

WE711 / ZE7111.0

### 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 2015

Figure 9-28. Scenario 4: Authorize valid SAML assertions

WE711 / ZE7111.0

### 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 2015

Figure 9-29. Scenario 4: SAML authentication statement (1 of 2)

WE711 / ZE7111.0

### 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 2015

Figure 9-30. Scenario 4: SAML authentication statement (2 of 2)

WE711 / ZE7111.0

### 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 2015

Figure 9-31. Scenario 4: SAML attribute statement

WE711 / ZE7111.0

### 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 and authenticate 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
  - If blank, certificate validation is skipped
4. Leave the identity mapping method at **None**

| <b>Identification Methods</b>  | <input type="checkbox"/> Name from SAML attribute assertion<br><input checked="" type="checkbox"/> Name from SAML authentication assertion<br><input type="checkbox"/> SAML artifact<br><input type="checkbox"/> Client IP address<br><input type="checkbox"/> Subject DN from certificate in the message's signature |        |   |
|--|---|--------|---|
| <b>Define how to authenticate the user.</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Method</th> <th style="width: 90%;"> <input checked="" type="radio"/> Accept a SAML Assertion with a Valid Signature<br/> <input type="radio"/> Accept an LTPA token<br/> <input type="radio"/> Bind to Specified LDAP Server<br/> <input type="radio"/> Contact a SAML Server for a SAML Authentication             </th> </tr> </thead> </table> |   | Method | <input checked="" type="radio"/> Accept a SAML Assertion with a Valid Signature<br><input type="radio"/> Accept an LTPA token<br><input type="radio"/> Bind to Specified LDAP Server<br><input type="radio"/> Contact a SAML Server for a SAML Authentication |
| Method   | <input checked="" type="radio"/> Accept a SAML Assertion with a Valid Signature<br><input type="radio"/> Accept an LTPA token<br><input type="radio"/> Bind to Specified LDAP Server<br><input type="radio"/> Contact a SAML Server for a SAML Authentication   |        |   |
| <b>SAML Signature Validation Credentials</b> <input style="margin-right: 10px;" type="button" value="pubcert"/> <input style="margin-right: 10px;" type="button" value="+"/> <input type="button" value="..."/>  |   |        |   |

© Copyright IBM Corporation 2015

Figure 9-32. Scenario 4: Identify and authenticate the client

WE711 / ZE7111.0

### Notes:

To verify the signature of the SAML assertion, the access control policy needs the validation credential. If the validation credentials field is blank, then the certificate is not validated. In either case, the signature is verified.

## 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

| Resource Identification Methods | <input type="checkbox"/> URL sent to back end<br><input type="checkbox"/> URL sent by client<br><input type="checkbox"/> URI of toplevel element in the message<br><input checked="" type="checkbox"/> Local name of request element<br><input type="checkbox"/> HTTP operation (GET/POST)<br><input type="checkbox"/> XPath expression<br>* |
|---------------------------------|--|
|---------------------------------|--|

6. For the authorization method, **Use SAML attributes from authentication**

- Set the SAML attribute that matches type as **Any value**

| Define how to authorize a request. |   |
|------------------------------------|---|
| Type                               | <input checked="" type="radio"/> Use SAML attributes from authentication<br><input type="radio"/> Use XACML Authorization decision<br><br><input type="radio"/> All values<br><input type="radio"/> All<br><input type="radio"/> Any value<br><input checked="" type="radio"/> Any<br><input type="radio"/> XPath |

7. Click **SAML Attributes** from the authentication method page

|   |
|---|
| <b>SAML Attributes</b>                                |
| <b>Back</b> <b>Next</b> <b>Advanced</b> <b>Cancel</b> |

© Copyright IBM Corporation 2015

Figure 9-33. Scenario 4: Authorize access to resources

WE711 / ZE7111.0

### 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://ibm.com/datapower/FLY/BookingService"/> |
| Local name  | <input type="text" value="BookingService"/>                              |
| Attribute value   | <input type="text" value="Request"/>                                     |
| <input type="button" value="Submit"/> <input type="button" value="Cancel"/> |  |

**Configure an Access Control Policy** [Help](#)

**SAML Attributes**

| Namespace URI  | Local name                                  | Attribute value                      |
|--|---|--------------------------------------|
| <input type="text" value="http://www.ibm.com/datapower/FLY/BookingService"/> | <input type="text" value="BookingService"/> | <input type="text" value="Request"/> |

© Copyright IBM Corporation 2015

Figure 9-34. Scenario 4: Match SAML attributes

WE711 / ZE7111.0

### 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 2015

Figure 9-35. Access control policy by SAML information

WE711 / ZE7111.0

### 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 DataPower 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 2015

Figure 9-36. Unit summary

WE711 / ZE7111.0

### 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 allow an access control policy to use different methods for authorization than for authentication. 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 2015

Figure 9-37. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.



## Checkpoint answers

**1. True.**

**2. True.**

**3. True.**

© Copyright IBM Corporation 2015

Figure 9-38. Checkpoint answers

WE711 / ZE7111.0

### Notes:



## Exercise 7



Web services authentication and authorization

© Copyright IBM Corporation 2015

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

10.1

Figure 9-39. Exercise 7

WE711 / ZE7111.0

### 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 2015

Figure 9-40. Exercise objectives

WE711 / ZE7111.0

### Notes:



## Exercise overview

- Configure BookingServiceProxy for authentication by using a AAA action that gets the userid/password information from an HTTP authentication header and authenticates with a AAAInfo file
- Send a request from SoapUI that contains an HTTP authentication header and is authenticated with a AAAInfo file
- Configure the AAA policy to use a SAML attribute assertion
- Send a request from SoapUI that contains a message with a SAML attribute assertion

© Copyright IBM Corporation 2015

Figure 9-41. Exercise overview

WE711 / ZE7111.0

### Notes:

# Unit 10. 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
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## 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 2015

Figure 10-1. Unit objectives

WE711 / ZE7111.0

### 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 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 communication format accessible to every developer skill level

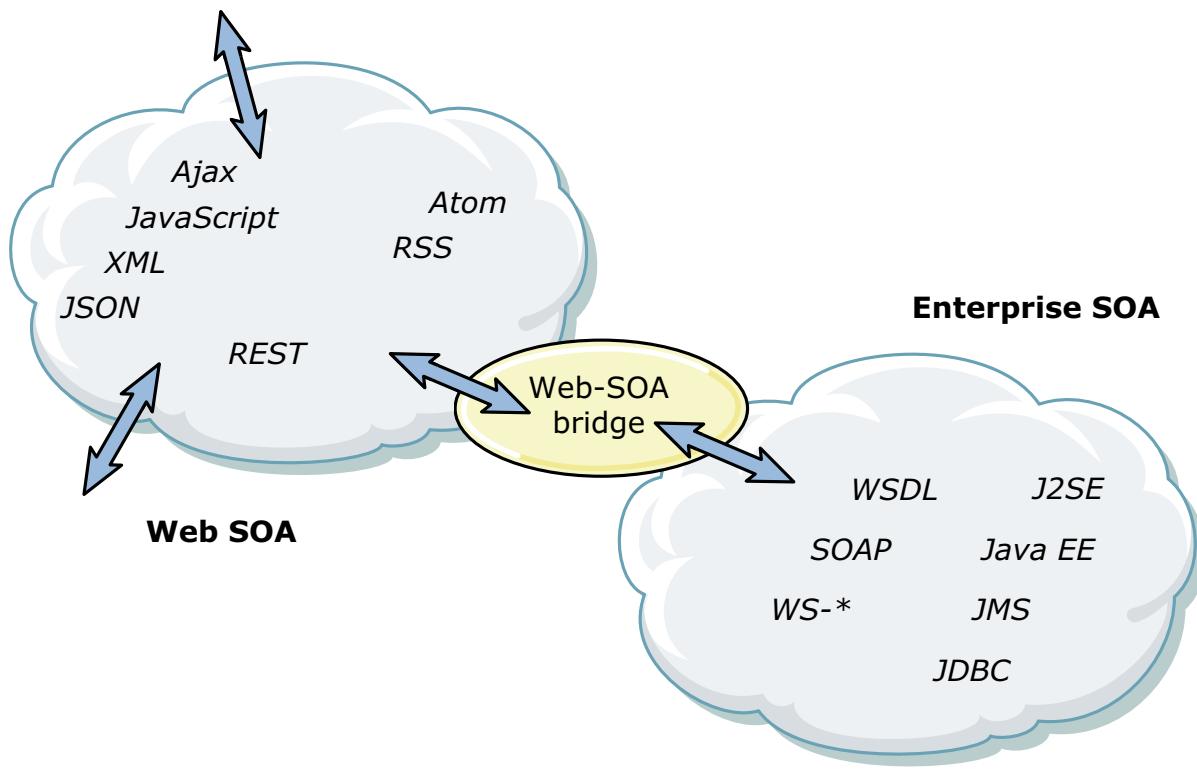
© Copyright IBM Corporation 2015

Figure 10-2. Alternatives to SOAP-based web services

WE711 / ZE7111.0

### Notes:

## Web SOA (Web 2.0) versus Enterprise SOA



© Copyright IBM Corporation 2015

Figure 10-3. Web SOA (Web 2.0) versus Enterprise SOA

WE711 / ZE7111.0

### 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 Markup Language (XML)
  - Plain old XML over HTTP (POX/HTTP)
- JavaScript Object Notation (JSON)
- JSON-RPC
- Bayeux
- Really Simple Syndication (RSS)
- Atom

© Copyright IBM Corporation 2015

Figure 10-4. Web SOA protocols and standards

WE711 / ZE7111.0

### 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 2015

Figure 10-5. Growth of mobile clients

WE711 / ZE7111.0

### Notes:

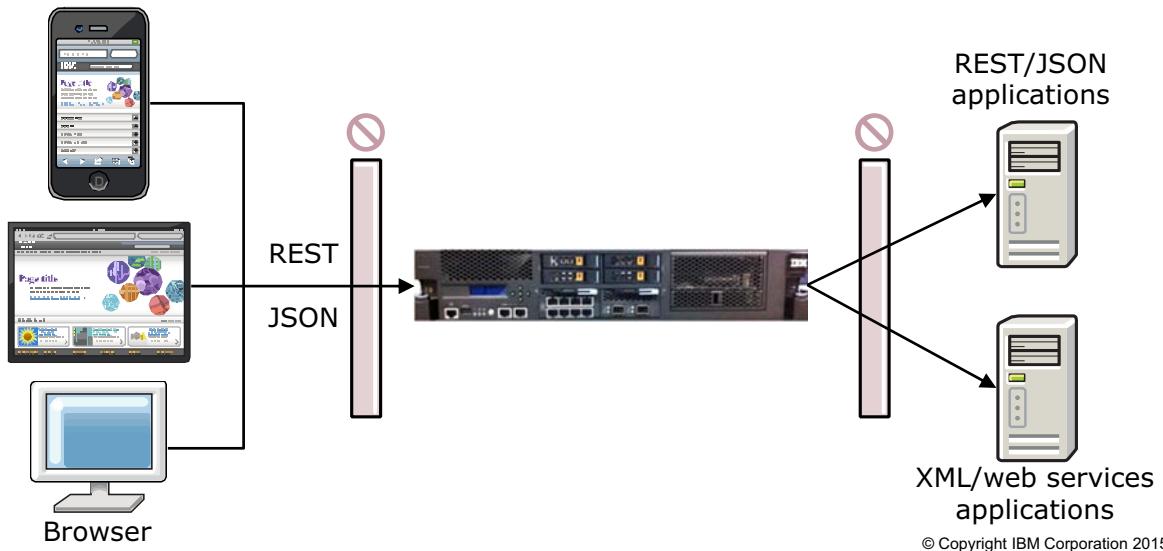
Desktop web browsers are no longer the single target client. Many clients now use smartphones and tablets, in addition to their desktop browsers.

Mobile clients might continue to use a web browser interface, but they 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 or JSON service
- Target DataPower service sends request to appropriate back end
  - Converts REST or JSON to XML or web services if needed
  - Applies other mediation to message as needed (AAA, transform, filter)



© Copyright IBM Corporation 2015

Figure 10-6. DataPower as the reverse proxy for Web 2.0 / Mobile clients

WE711 / ZE7111.0

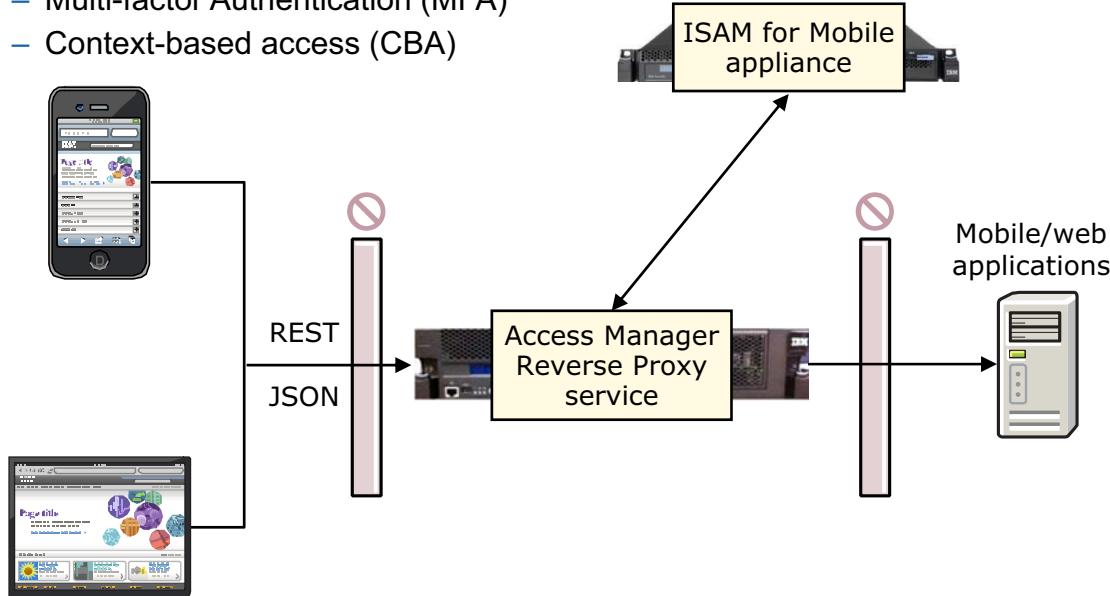
### Notes:

Regardless of whether the clients are 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.”

## Securing the reverse proxy with ISAM for Mobile

- Access Manager Reverse Proxy service can interact with IBM Security Access Manager (ISAM) for Mobile to provide more security:
  - One-Time Password (OTP)
  - Multi-factor Authentication (MFA)
  - Context-based access (CBA)



© Copyright IBM Corporation 2015

Figure 10-7. Securing the reverse proxy with ISAM for Mobile

WE711 / ZE7111.0

### Notes:

By integrating with ISAM for Mobile, the Access Manager Reverse Proxy service can provide more security support beyond forms authentication or HTTP basic authorization. This service and ISAM for Mobile also supports more advanced protection such as One-Time Password (OTP), Multi-factor Authentication (MFA), and Context-based access (CBA).

The Access Manager Reverse Proxy service itself can secure a directory tree-like structure of web resources. Additionally, the Access Manager Reverse Proxy service can chain with a multi-protocol gateway service to offer further mediation of the message contents.

The Access Manager Reverse Proxy service capabilities are similar to the capabilities of WebSEAL. IBM Tivoli Access Manager WebSEAL is a web server that applies fine-grained security policy to the Tivoli Access Manager protected web object space. WebSEAL can provide single sign-on solutions and incorporate back-end web application server resources into its security policy.

The Access Manager Reverse Proxy service requires the ISAM Proxy module to be licensed and installed on the appliance.

## Introduction to REST

Representational State Transfer (REST) is an architectural style for accessing resources across a network:

- Application state and functions 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 2015

Figure 10-8. Introduction to REST

WE711 / ZE7111.0

### Notes:

REST originated from a PhD dissertation from Roy Thomas Fielding called “Architectural Styles and the Design of Network-based Software Architectures.” For more information, 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

- PUT

- Update an existing resource
- Depending on the defined interface, can also create a resource at a specific URL
- Idempotent
- Has side effect

- DELETE

- Remove a resource from the server
- 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 2015

Figure 10-9. REST style services

WE711 / ZE7111.0

### 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 occur by running it once. For example, a client might send a series of idempotent requests and then get disconnected from the server without getting confirmation that the requests completed. If this situation happens, the client can safely try the series of idempotent requests again without worrying that duplicate records are created, or data is deleted that should not be.

For REST in a web environment, these verbs match with 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 or employees (Read)
- Modify an employee's information (Update)
- Remove an employee (Delete)
- Create, read, update, and delete for salary history
  - Give an employee a raise = update salary history
- Create, read, update, and delete for performance reviews

© Copyright IBM Corporation 2015

Figure 10-10. Example: Employee processing

WE711 / ZE7111.0

### 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 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 2015

Figure 10-11. Employee REST interface

WE711 / ZE7111.0

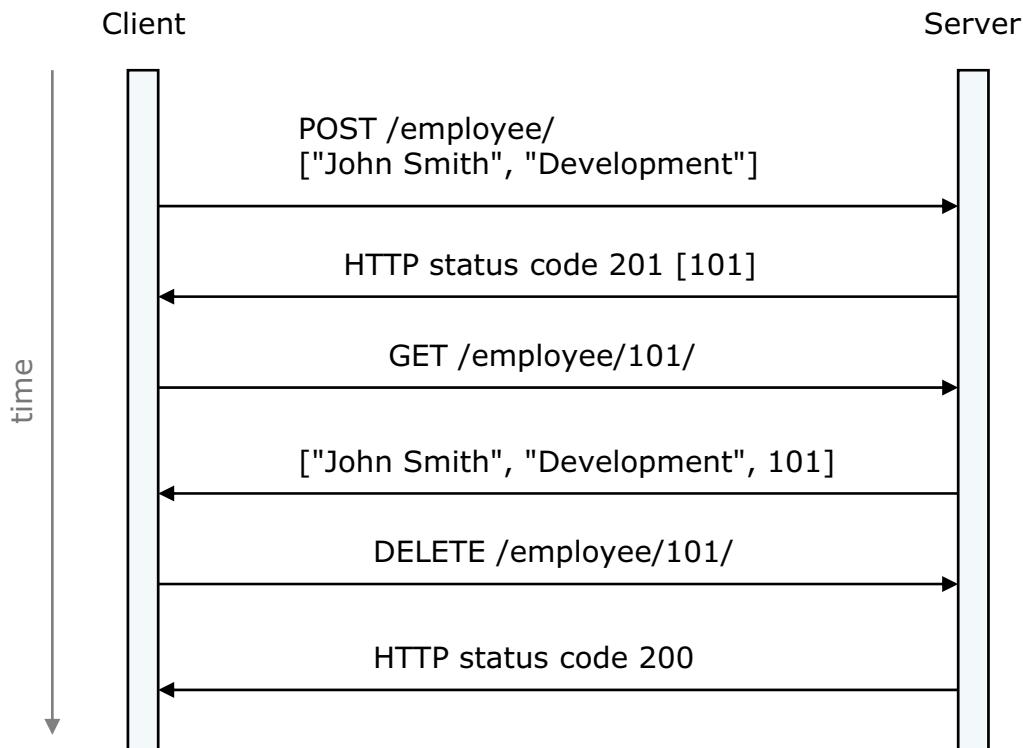
### Notes:

Here is your design for the employee processing service. You defined the resources, defined the formats, chose 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 box
- 403: Forbidden, the server refuses the request
- 404: Not found
- 500: Internal server error

## Example: REST interaction



© Copyright IBM Corporation 2015

Figure 10-12. Example: REST interaction

WE711 / ZE7111.0

### 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

- The URI specifies the resource that is being accessed
- It does not expose the underlying service implementation

```
<employee>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <department>Development</department>
  <id></id>
</employee>
```

The HTTP method specifies the operation to be performed

- The HTTP request body contains information about the resource
- More types include binary attachments, such as images

© Copyright IBM Corporation 2015

Figure 10-13. Example: Add employee REST request explained

WE711 / ZE7111.0

### 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 that is transmitted in the HTTP response is the actual data itself: the newly assigned employee ID of 101

© Copyright IBM Corporation 2015

Figure 10-14. Example: Add employee REST response explained

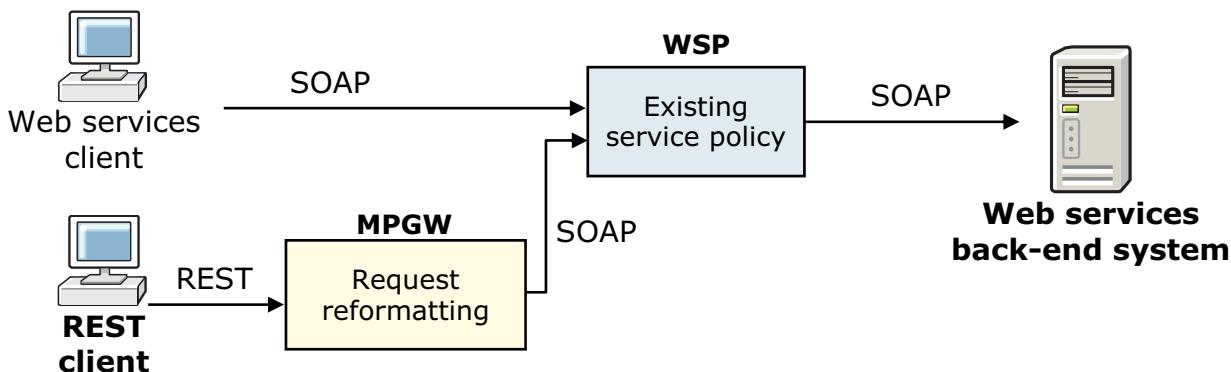
WE711 / ZE7111.0

### Notes:

HTTP status code 201 is defined as “CREATED”.

## Common DataPower REST patterns: Facade

- 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 2015

Figure 10-15. Common DataPower REST patterns: Facade

WE711 / ZE7111.0

### Notes:

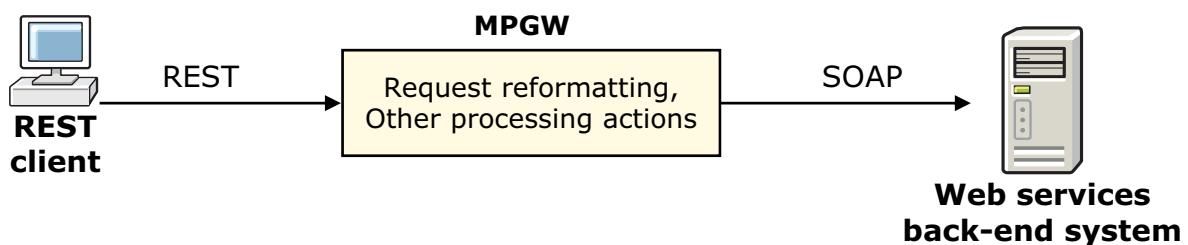
WSP is web service proxy.

MPGW is multi-protocol gateway.

The REST facade 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)



© Copyright IBM Corporation 2015

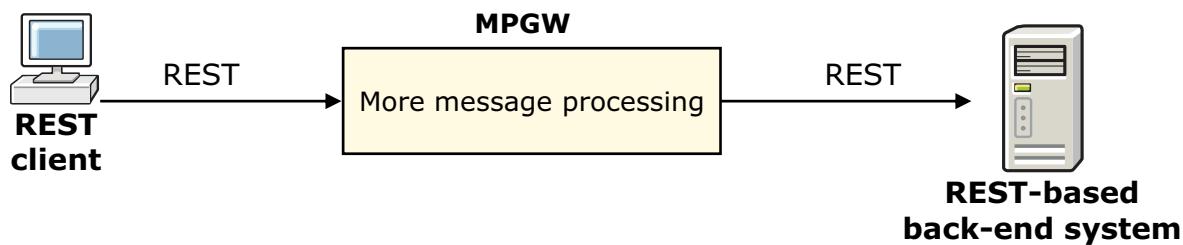
Figure 10-16. Common DataPower REST patterns: Bridge

WE711 / ZE7111.0

### 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 (AAA)
  - Service level management (SLM)
  - Content-based routing



© Copyright IBM Corporation 2015

Figure 10-17. Common DataPower REST patterns: REST enrichment

WE711 / ZE7111.0

### Notes:

The back-end system already has a RESTful interface, but extra processing of the message is required.



## Tools 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 2015

Figure 10-18. Tools to support REST: Service or protocol handler related

WE711 / ZE7111.0

### Notes:

The first topic is on the DataPower tools to support REST that are configured at the service level or the protocol handler level.

MPGW is multi-protocol gateway service.

XMLFW is 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 that are selected by default do not include GET

HTTP Front Side Handler: RESTEastAddressSearch\_FSH [up]

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

© Copyright IBM Corporation 2015

Figure 10-19. Front side handler support of HTTP method selection

WE711 / ZE7111.0

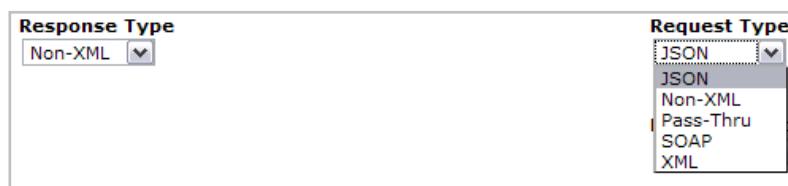
### Notes:

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 common 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 Type or Response Type



© Copyright IBM Corporation 2015

Figure 10-20. JSON request and response types

WE711 / ZE7111.0

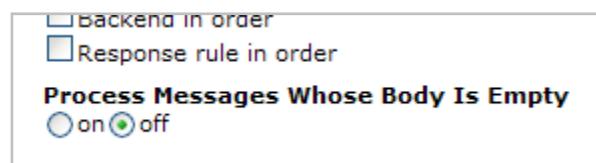
### Notes:

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 2015

Figure 10-21. Process bodyless messages

WE711 / ZE7111.0

### 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 2015

Figure 10-22. JSON threat protection

WE711 / ZE7111.0

### 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.

## Tools 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
- XSL and GatewayScript read/write access to HTTP method and 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
- GatewayScript action to support JavaScript processing
- Actions that allow `http://` access: Fetch, Results, Results async, Log

© Copyright IBM Corporation 2015

Figure 10-23. Tools to support REST: Service policy related

WE711 / ZE7111.0

### Notes:

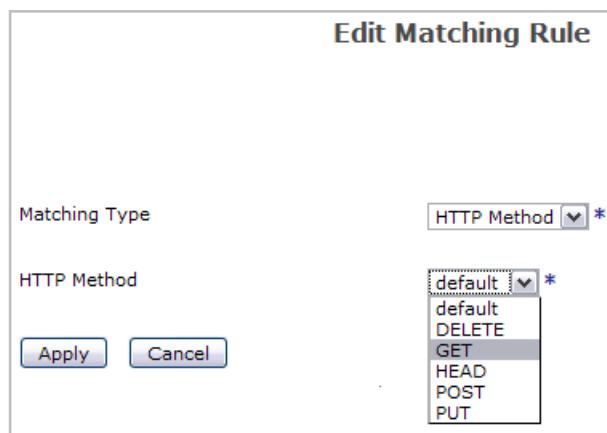
The second topic is on the DataPower tools to support REST that are configured within a service policy.

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 2015

Figure 10-24. Matching Rule on HTTP methods

WE711 / ZE7111.0

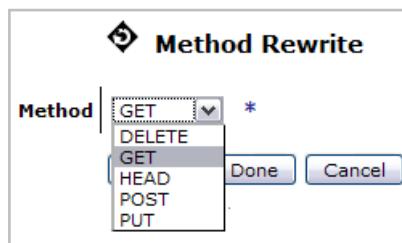
### Notes:

**default** indicates that either a GET or POST successfully matches.



## 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) specifies a specific HTTP method
  - The original REST GET request gets converted to a SOAP POST request



© Copyright IBM Corporation 2015

Figure 10-25. Changing the HTTP method in the processing rule

WE711 / ZE7111.0

### 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**

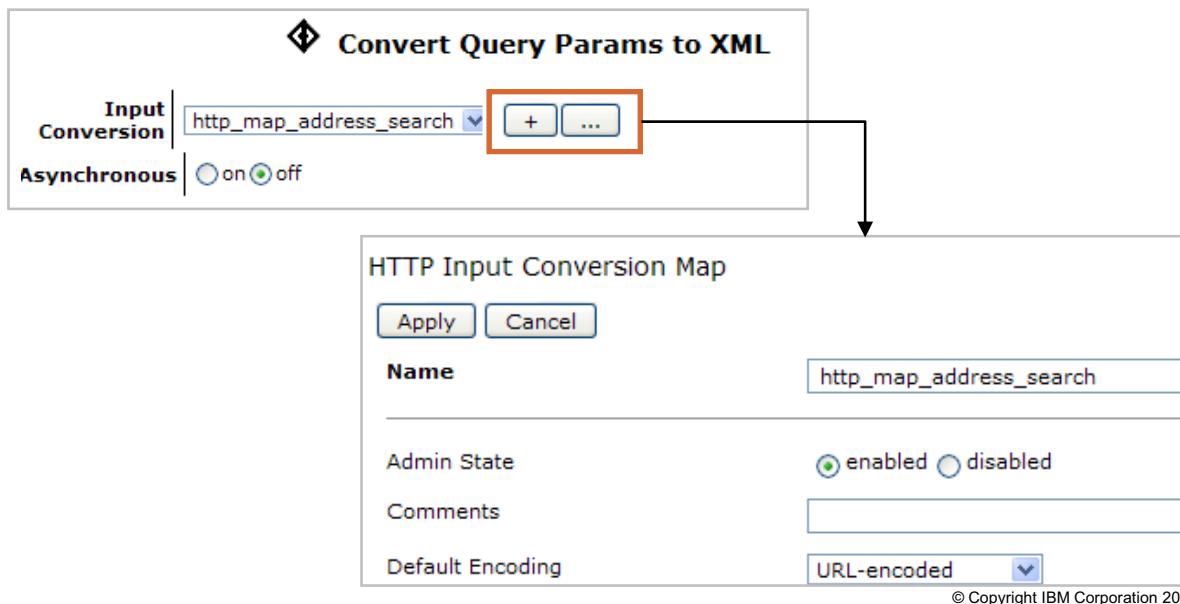


Figure 10-26. Convert Query Params to XML action

WE711 / ZE7111.0

### 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 2015

Figure 10-27. Convert Query Params to XML action example

WE711 / ZE7111.0

### Notes:

The “-G” tells cURL to use an HTTP GET on the request.

## Programmatic access to the HTTP method

- XSL extension function to retrieve which HTTP method is invoked in the HTTP request
  - The `dp:http-request-method()` extension function returns a string that contains the HTTP method
  
- A service variable, `var://service/protocol-method`, can be used to *read* or *write* the HTTP method
  - XSL:
 

```
<xsl:variable name="HTTPmethod"
            select="dp:variable('var://service/protocol-method')"/>
        or
        <dp:set-variable name="'var://service/protocol-method'"
            value="'GET'" />
```
  - GatewayScript:
 

```
var serviceVars = require ('service-metadata');
var HTTPmethod = serviceVars.protocolMethod;
or
serviceVars.protocolMethod = 'GET';
```

© Copyright IBM Corporation 2015

Figure 10-28. Programmatic access to the HTTP method

WE711 / ZE7111.0

### Notes:

`dp:http-request-method()` is a metadata extension function.

The variable that contains the HTTP protocol method has both a slash notation (`var://service/protocol-method`) and a dot notation (`serviceVars.protocolMethod`). In GatewayScript, you can use either one.

Not all variables support the dot notation. For more information, see the product documentation.

## Programmatic access to the HTTP status code

- You can retrieve the HTTP status code from the request
  - XSL: 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')"/>`
  - GatewayScript:  
`var hm = require('header-metadata');`  
`var currentSC = hm.current.statusCode;`
- You can also set it in a similar fashion
  - XSL:  
`<dp:set-http-response-header name="'x-dp-response-code'" value="'204'" />`
  - GatewayScript:  
`hm.response.statusCode = "204";`

© Copyright IBM Corporation 2015

Figure 10-29. Programmatic access to the HTTP status code

WE711 / ZE7111.0

### Notes:

Specific HTTP status codes are usually part of the RESTful interface definition.



## JSON

- JavaScript Object Notation
  - Subset of JavaScript
  - Minimal
  - Lightweight
  - 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
- For more information, see: <http://json.org>

© Copyright IBM Corporation 2015

Figure 10-30. JSON

WE711 / ZE7111.0

### 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 2015

Figure 10-31. JSON data types

WE711 / ZE7111.0

### 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 2015

Figure 10-32. JSON version of XML

WE711 / ZE7111.0

### 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](http://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 2015

Figure 10-33. JSONx

WE711 / ZE7111.0

### Notes:

IETF is 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 2015

Figure 10-34. JSONx version of JSON data structure

WE711 / ZE7111.0

### 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 or GatewayScript 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 2015

Figure 10-35. Handling JSON in the request body

WE711 / ZE7111.0

### 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 required JSONx representation

```
XSL ...
<json:object name="details">
  <json:string name="city">
    <xsl:value-of select="details/city" />
  </json:string>
  <json:string name="state">
    <xsl:value-of select="details/state" />
  </json:string>
  <json:number name="zipcode">
    <xsl:value-of select="details/zipCode" />
  </json:number>
...

```

```
{"details": {
  "city": "Cleveland",
  "state": "OH",
  "zipcode": 44111
}}
```

- 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 2015

Figure 10-36. Converting XML to JSON

WE711 / ZE7111.0

### 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 4 of the IETF specification:  
<http://tools.ietf.org/html/draft-zyp-json-schema-04>

© Copyright IBM Corporation 2015

Figure 10-37. Validate action for JSON

WE711 / ZE7111.0

### Notes:

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

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

DataPower version 7.1.0 supports several JSON schema specifications:

- JSON Schema: core definitions and terminology -  
<http://tools.ietf.org/html/draft-zyp-json-schema-04>
- JSON Schema: interactive and non-interactive validation -  
<http://tools.ietf.org/html/draft-fge-json-schema-validation-00>

## 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-04/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": "Sarah",
            "lastname": "Chan",
            "title": "Mrs"
          }
        }
      }
    }
  }
}

(and so on)
```

© Copyright IBM Corporation 2015

Figure 10-38. Example JSON and a JSON schema

WE711 / ZE7111.0

### Notes:

The schema does not show constraints like required items, numeric ranges, and other constraints.

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 added the support for the JSONiq extension (JSON support)
- **Input Language:**
  - JSON
  - XML
  - XSD
- **Transform Language:**
  - XQuery

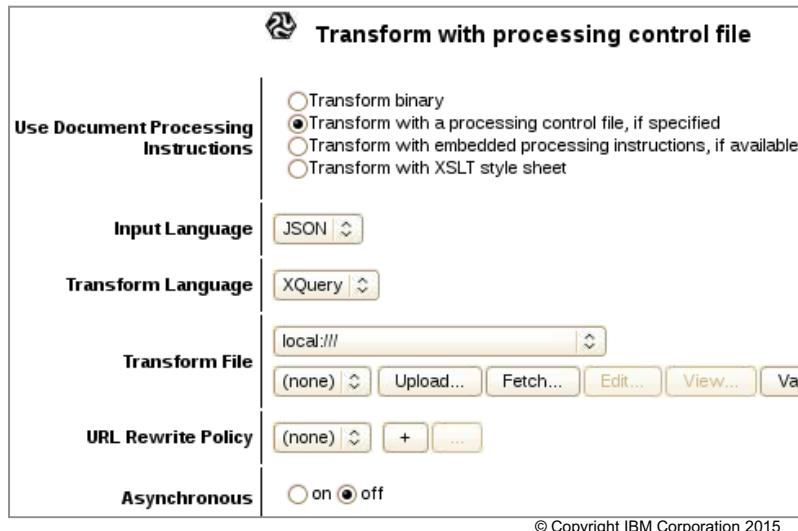


Figure 10-39. Transform action that uses XQuery (JSON and XML)

WE711 / ZE7111.0

### 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 as SQL is used to query relational data. It uses XPath and XML elements, much like a style sheet, but it also supports an SQL-like query function: for, let, where, order by, return (FLWOR). DataPower supplies several extension functions to XQuery to allow manipulation of DataPower variables and protocol headers.

DataPower V6.0.0 included the JSONiq extension to XQuery. This extension added support for JSON to XQuery.

DataPower Version 7.1.0 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* validate 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": 99.95}, {"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 lastname

```
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 2015

Figure 10-40. XQuery/JSONiq example

WE711 / ZE7111.0

### 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.



## GatewayScript action that uses JavaScript

- Provides an encapsulated and secure JavaScript environment
  - ECMAScript 5.1 language specification plus block scoping, strict mode, no `eval()` function or compilation from strings, read-only access to only certain directories in the file system
  - CommonJS Module/1.0 specification
- Access to DataPower variables, contexts, objects similar to XSL
  - Some objects are provided by default, such as the `session` object, some objects need a `require` function, such as `header-metadata`
  - `urlopen.open()` to communicate with other servers, and read access to `local:` and `store:`

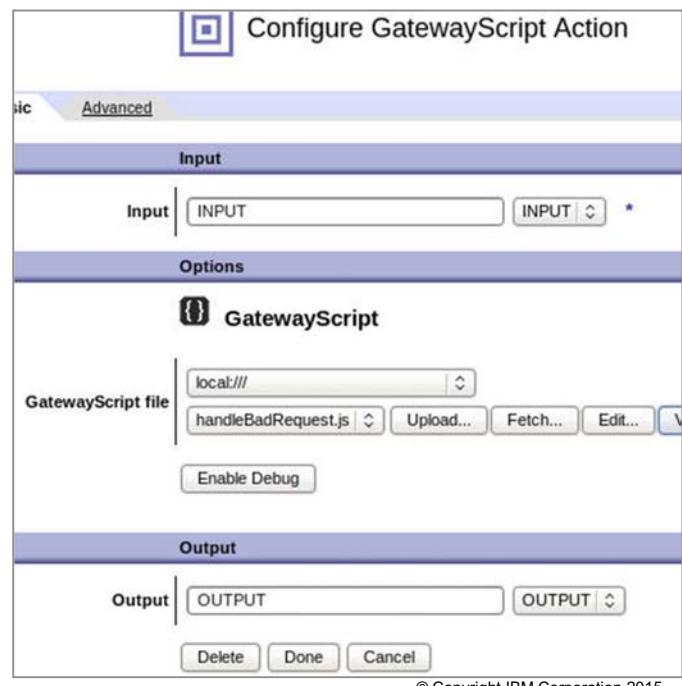


Figure 10-41. GatewayScript action that uses JavaScript

WE711 / ZE7111.0

### Notes:

## Sample GatewayScript: JSON input to SOAP output

```

session.input.readAsJSON(function(error,json) {
    if (error) {session.output.write("oops error " +
        JSON.stringify(error.toString()));
    } else {
        debugger;
        var refNo = json.refNumber;
        var lastName = json.lastName;
        console.info("Request for %s with reference %i", lastName, refNo);
        session.output.write(
            "<soapenv:Envelope
            xmlns:soapenv=\\"http://schemas.xmlsoap.org/soap/envelope/\\" "
            + "xmlns:fly=\\"http://www.ibm.com/datapower/FLY/BaggageService/\\">" +
            + "<soapenv:Header/><soapenv:Body>" +
            + "<fly:BaggageStatusRequest>" +
            + "<fly:refNumber>" + refNo + "</fly:refNumber>" +
            + "<fly:lastName>" + lastName + "</fly:lastName>" +
            + "</fly:BaggageStatusRequest>" +
            + "</soapenv:Body></soapenv:Envelope>" +
        );
    }
});

```

© Copyright IBM Corporation 2015

Figure 10-42. Sample GatewayScript: JSON input to SOAP output

WE711 / ZE7111.0

### Notes:

“session.input” identifies the input context for this action.

The “readAsJSON” method reads the context and places it into an object named “json”. If there is an error on the read operation, an error message is written to the output context.

“debugger” enables the CLI debugger capability.

The input “json” object is read to retrieve the reference number and last name of the passenger. Those values are placed into variables that are used later in the script.

Information on the request is written to the system log at the information level.

The SOAP message is literally constructed. The reference number and last name are retrieved and placed as contents within the message. The SOAP message is written to the output context of the action.

The typical spacing of the code is compressed to fit the slide.

## Bridging REST and SOAP: Sample service policy #1

|               |                  |  |  |  |  |  |  |  |  |
|---------------|------------------|--|--|--|--|--|--|--|--|
| 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 2015

Figure 10-43. Bridging REST and SOAP: Sample service policy #1

WE711 / ZE7111.0

### 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.

This approach is an XML and style sheet focused approach.



## Bridging REST and SOAP: Sample service policy #2

|                                   |                  |  |  |  |  |  |  |
|-----------------------------------|------------------|--|--|--|--|--|--|
| BaggageServicePolicy_FindBag_req  | Client to Server |  |  |  |  |  |  |
| BaggageServicePolicy_FindBag_Resp | Server to Client |  |  |  |  |  |  |

- In the request:
  - Match on HTTP method = GET
  - Convert URI parameters to an XML structure
  - Normalize the XML with an identity transform
  - Set the HTTP method to POST
  - Set the back-end URI
  
- In the response:
  - Use XPath to match on the specific response
  - Use XQuery and JSONiq to convert the SOAP response to a JSON response

© Copyright IBM Corporation 2015

Figure 10-44. Bridging REST and SOAP: Sample service policy #2

WE711 / ZE7111.0

### 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.

This approach is a GatewayScript and XQuery focused approach.



## 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 2015

Figure 10-45. Unit summary

WE711 / ZE7111.0

### Notes:



## Checkpoint questions

1. True or False: Mobile clients differ from desktop clients only in 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 that the Convert Query Params to XML action supports.
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 2015

Figure 10-46. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.
- 5.



## 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:
  - Easy to secure
  - Easy to navigate
  - Simplicity
4. The three types of non-XML data that the Convert Query Params to XML action supports are HTTP POST, HTML form, or URI parameters.
5. **C.** Method rewrite.

© Copyright IBM Corporation 2015

Figure 10-47. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 8



Using DataPower to implement  
REST services

© Copyright IBM Corporation 2015

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

10.1

Figure 10-48. Exercise 8

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Create a service policy to handle JSON and REST requests and responses
- Use a GatewayScript to build a SOAP request from HTTP query parameters or JSON
- Enable and use the CLI debugger
- Define and use style sheet parameters
- Convert a SOAP response to a JSON-formatted data structure by using XQuery/JSONiq

© Copyright IBM Corporation 2015

Figure 10-49. Exercise objectives

WE711 / ZE7111.0

### Notes:



## Exercise overview

- Receive a request that includes the input as JSON data in the HTTP body, and convert it to a standard SOAP request for the back-end web service. Convert the SOAP response to a JSON structure for the client.
- Test with SoapUI
- Add JSON schema validation, and test
- Use the CLI GatewayScript debugger
- Receive a REST GET request and convert it to a standard SOAP request for the back-end web service. Convert the SOAP response to a JSON structure for the client
- Test with SoapUI

© Copyright IBM Corporation 2015

Figure 10-50. Exercise overview

WE711 / ZE7111.0

### Notes:

# Unit 11. OAuth overview and DataPower implementation

## What this unit is about

This unit introduces the OAuth 2.0 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
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)

## 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 2015

---

Figure 11-1. Unit objectives

WE711 / ZE7111.0

### 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 2015

Figure 11-2. What is OAuth?

WE711 / ZE7111.0

### 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 asked and stored 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 this key, with some restrictions:
  - Speed restriction
  - Distance restriction
  - Cannot alter some car functions, such as radio stations
  - Cannot open car storage areas
- When you allow the valet to borrow the valet key, you delegate access to certain features of the car to a third party.

© Copyright IBM Corporation 2015

Figure 11-3. Delegated authorization example

WE711 / ZE7111.0

### Notes:

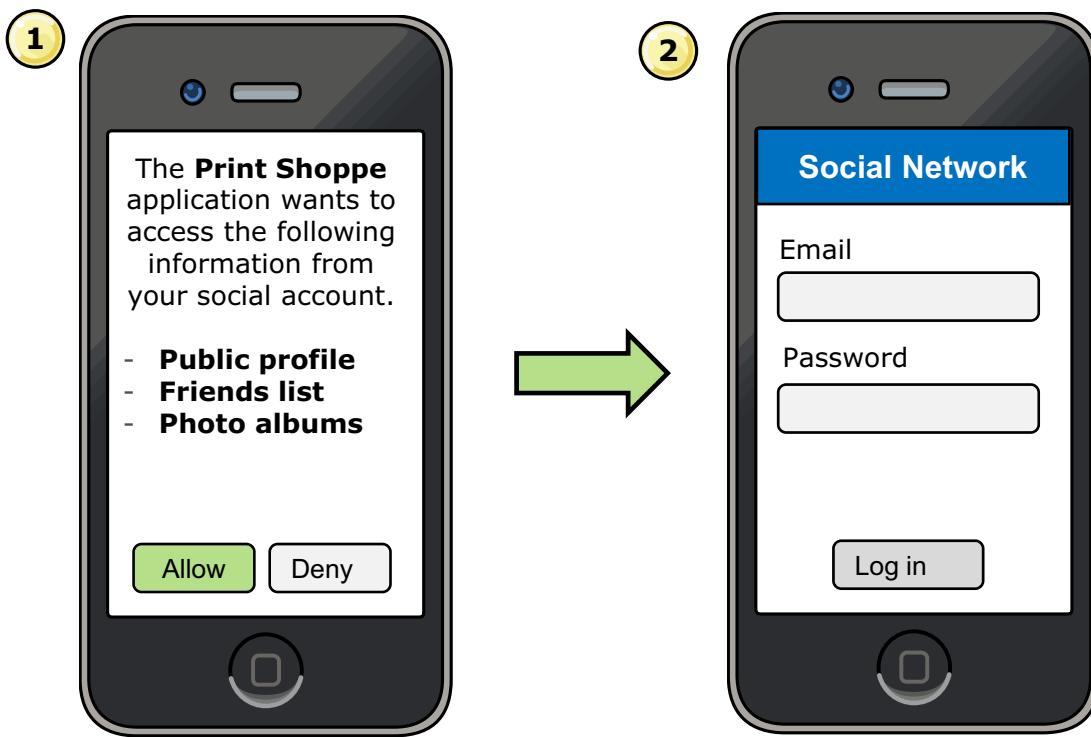
When you purchase a car, you receive a main key and a valet key.

The valet can use your car with this key, with some restrictions. The car cannot exceed a certain speed. The car might not be able to travel as far as with the regular key. The valet cannot change the radio station settings. The valet cannot open car storage areas.

When you allow the valet to borrow the valet key, you delegate access to certain features of the car to a third party.



## Example: Allow third-party access to social account



© Copyright IBM Corporation 2015

Figure 11-4. Example: Allow third-party access to social account

WE711 / ZE7111.0

### 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 an account.

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.



Alice is the **owner** of an online photo album. As a **resource owner**, Alice declares which applications can access her online photo album.



The social network provides an online photo album. This service acts as the **resource server**. It manages access to Alice's photos.



The Print Shoppe, a third-party **client application**, wants to access Alice's photos from an online service.



An **authorization server** verifies the identity of the client that wants to access Alice's photo album. This server issues a token or a code to access the photo album from the resource server.

© Copyright IBM Corporation 2015

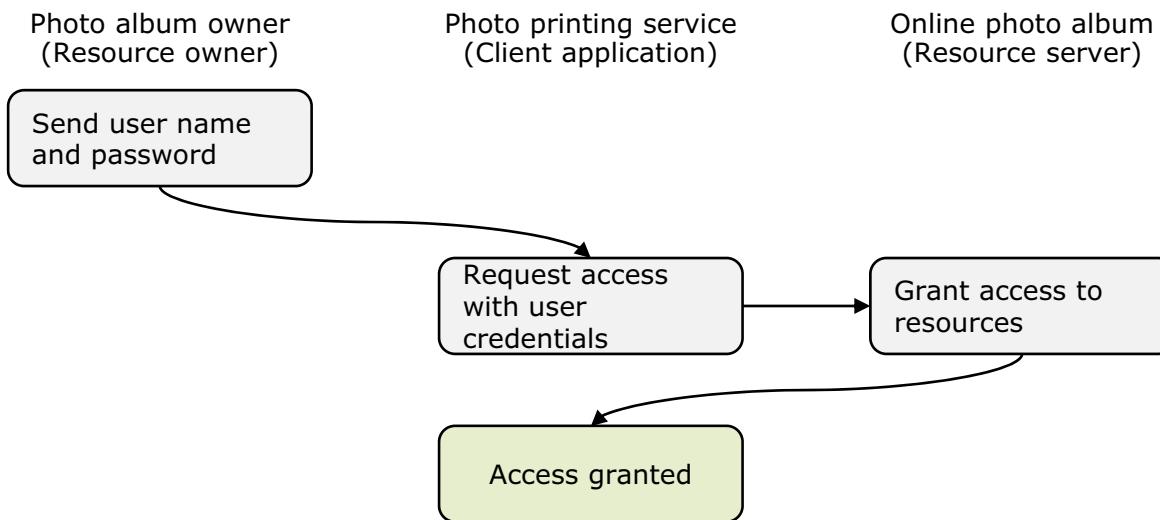
Figure 11-5. Example: Third-party access to online photo album

WE711 / ZE7111.0

### 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**, Alice 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 act on Alice's behalf.

## 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 2015

Figure 11-6. Before OAuth: Sharing user passwords

WE711 / ZE7111.0

### Notes:

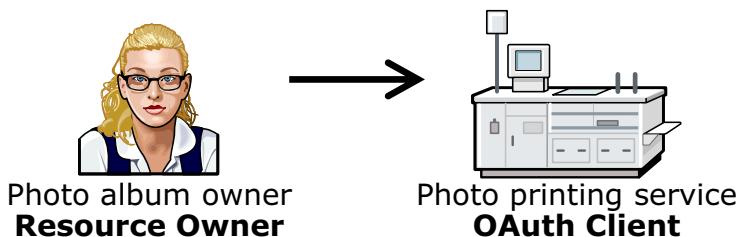
Without OAuth, the user must give their user name and password to the third-party application. In turn, the third-party application sends these credentials while posing as the user. For convenience's sake, the application saves a copy of the user name and password.

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 Step 1: Resource owner requests access

- Step 1: Alice, as the resource owner, requests access to the online photo album to the OAuth client.



© Copyright IBM Corporation 2015

Figure 11-7. OAuth Step 1: Resource owner requests access

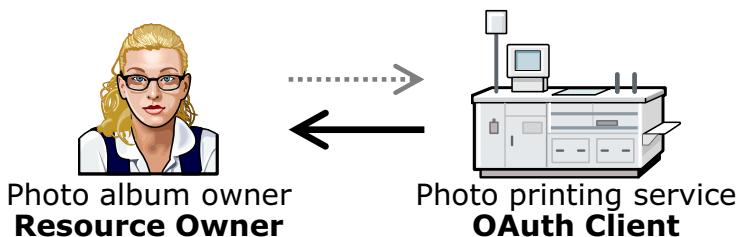
WE711 / ZE7111.0

### Notes:

In this scenario, Alice is the owner of a photo album that is hosted on an online photo service. Alice wants to print a set of photos with a photo printing service. Alice is the resource owner, and the photo printing service is a third-party OAuth client application. Alice starts the process when she selects the "print from my photo album" option in the third-party application.

## OAuth Step 2: OAuth client redirection to owner

- Step 2: The OAuth client sends the resource owner a redirection to the authorization server.



© Copyright IBM Corporation 2015

Figure 11-8. OAuth Step 2: OAuth client redirection to owner

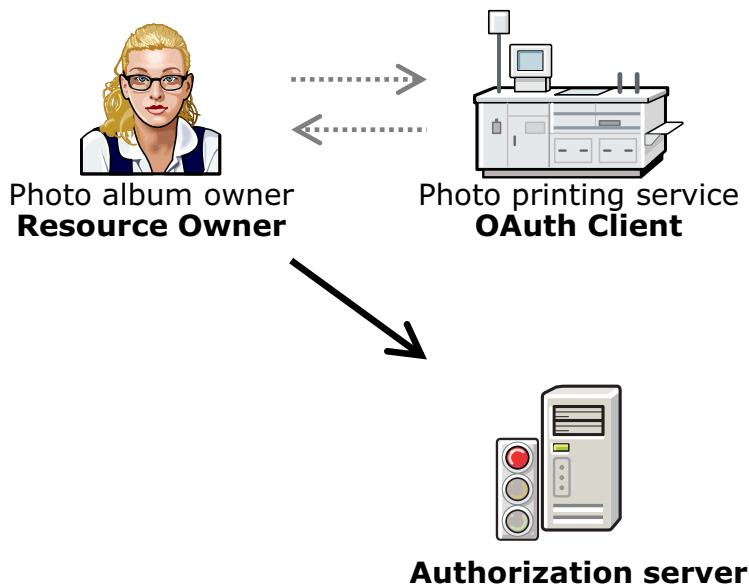
WE711 / ZE7111.0

### Notes:

In the second step, the third-party application requires the resource owner's authorization before it can access her online photo album. Instead of asking Alice directly for her user credentials, the third-party client application redirects Alice's request to an authorization server.

## OAuth Step 3: Authenticate owner with authorization server

- Step 3: The resource owner authenticates against the authorization server.



© Copyright IBM Corporation 2015

Figure 11-9. OAuth Step 3: Authenticate owner with authorization server

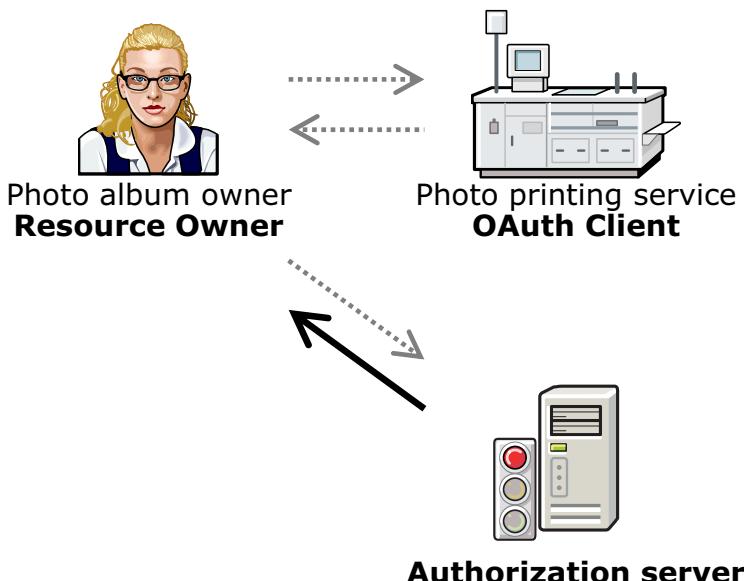
WE711 / ZE7111.0

### Notes:

In the third step, the authorization server asks for Alice's user credentials to verify her identity.

## OAuth Step 4: Ask resource owner to grant access to resources

- Step 4: The authorization server returns a web form to the resource owner to grant access.



© Copyright IBM Corporation 2015

Figure 11-10. OAuth Step 4: Ask resource owner to grant access to resources

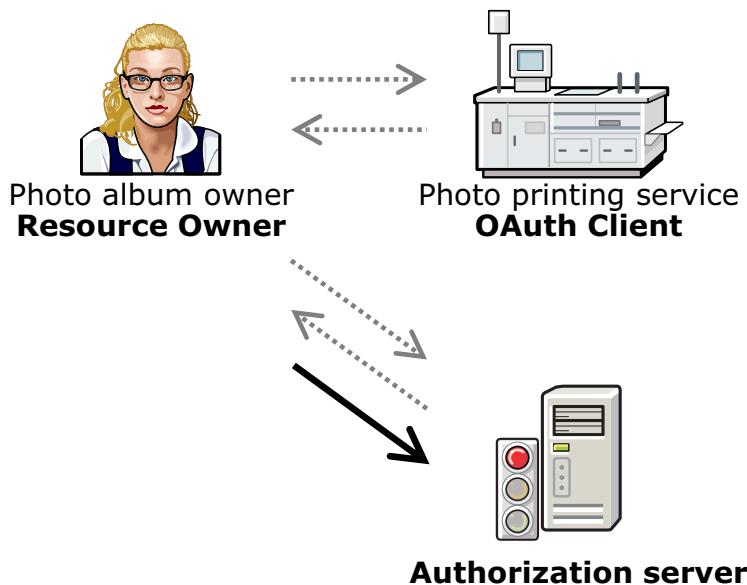
WE711 / ZE7111.0

### Notes:

The authorization server returns a web form to ask Alice whether she grants the OAuth client access to her resources.

## OAuth Step 5: Resource owner grants client access to resources

- Step 5: The resource owner submits the form to allow or to deny access.



© Copyright IBM Corporation 2015

Figure 11-11. OAuth Step 5: Resource owner grants client access to resources

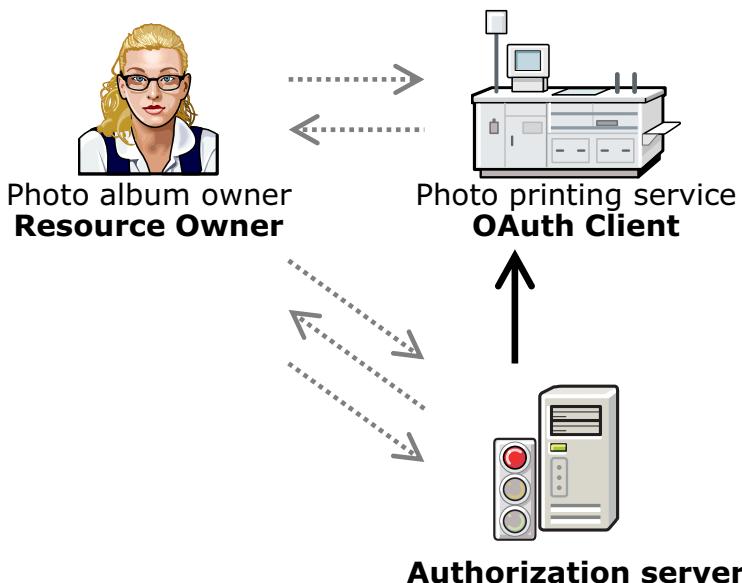
WE711 / ZE7111.0

### Notes:

The resource owner, Alice, submits the web form to allow or deny access to her resources.

## OAuth Step 6: Authorization server sends authorization grant code to client

- Step 6: If the resource owner allows access, the authorization server sends the OAuth client a redirection with the authorization grant code.



© Copyright IBM Corporation 2015

Figure 11-12. OAuth Step 6: Authorization server sends authorization grant code to client

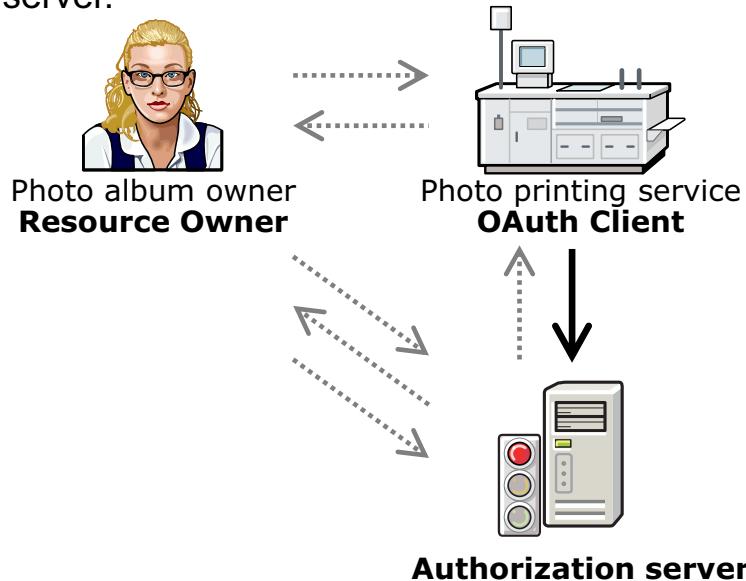
WE711 / ZE7111.0

### Notes:

The authorization server never transmits the resource owner's user name and password to the OAuth client. Instead, the server sends an authorization grant code: a token that allows the OAuth client to access Alice's resources on her behalf.

## OAuth Step 7: Client requests access token from authorization server

- Step 7: To access the resource, the OAuth client sends the authorization grant code and other information to the authorization server.



© Copyright IBM Corporation 2015

Figure 11-13. OAuth Step 7: Client requests access token from authorization server

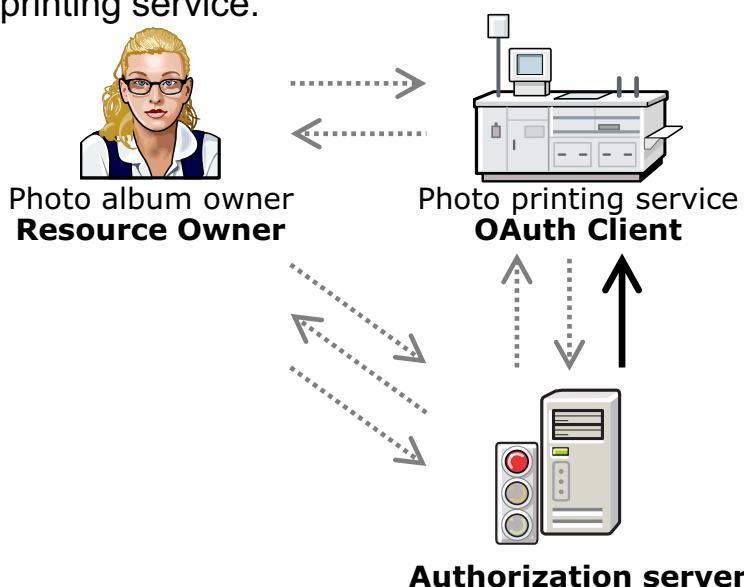
WE711 / ZE7111.0

### Notes:

The OAuth client sends three pieces of information to the authorization server: an authorization grant code, the client ID, and the client secret or client certificate. If the OAuth client is a public client, then it does not send the client secret or certificate.

## OAuth Step 8: Authorization server sends authorization token to client

- Step 8: If the authorization server verifies the grant authorization information, it returns an access token to the OAuth client, the photo printing service.



© Copyright IBM Corporation 2015

Figure 11-14. OAuth Step 8: Authorization server sends authorization token to client

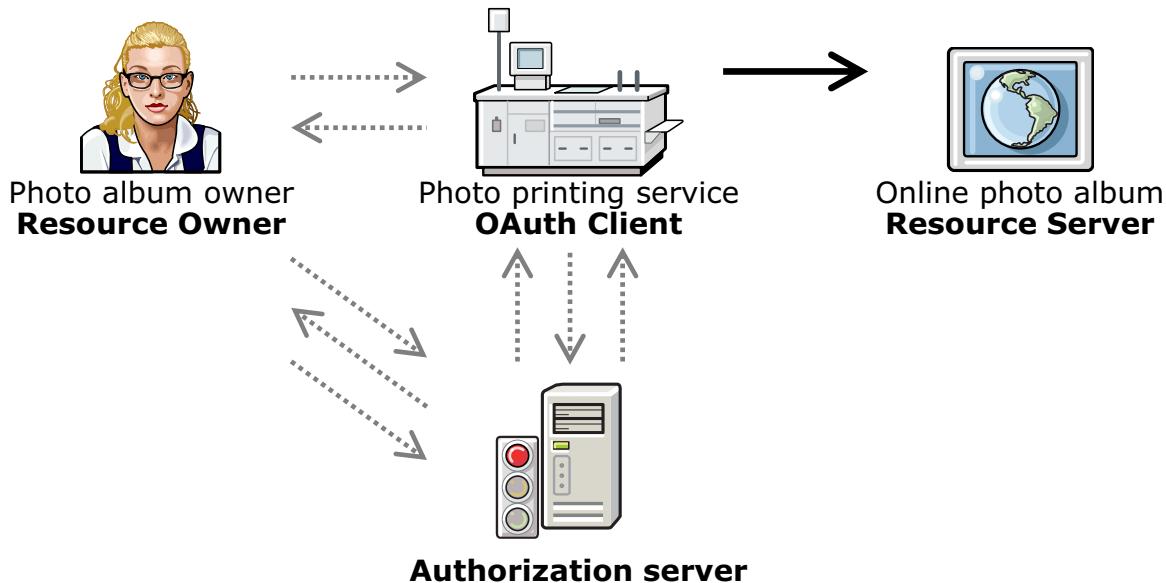
WE711 / ZE7111.0

### Notes:

Optionally, the authorization server can also return a refresh token. After the current access token expires, the OAuth client sends the refresh token to the authorization server to request another access token.

## OAuth Step 9: OAuth client sends access token to resource server

- Step 9: The OAuth client sends the access token to the resource server.



© Copyright IBM Corporation 2015

Figure 11-15. OAuth Step 9: OAuth client sends access token to resource server

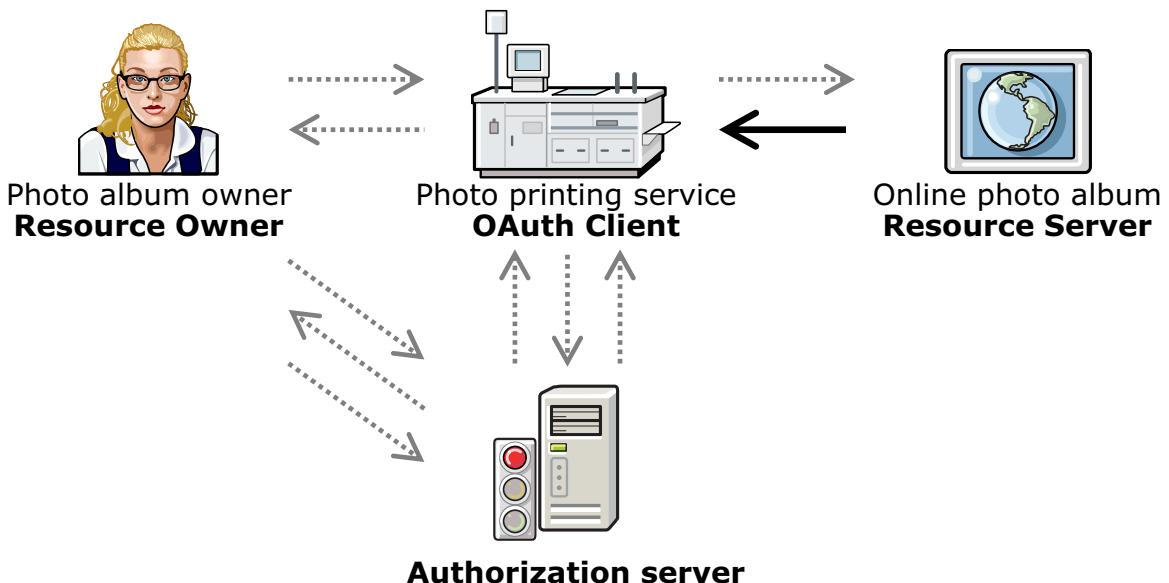
WE711 / ZE7111.0

### Notes:

It is possible that the authorization server and the resource server are the same server.

## OAuth Step 10: Resource server grants access to OAuth client

- Step 10: If the access token is valid for the requested resource, the resource server allows the OAuth client to access the resource.



© Copyright IBM Corporation 2015

Figure 11-16. OAuth Step 10: Resource server grants access to OAuth client

WE711 / ZE7111.0

### Notes:

Optionally, the authorization server can also return a refresh token. After the current access token expires, the OAuth client sends the refresh token to the authorization server to request another access token.



## OAuth 2.0 protocol defines 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 2015

Figure 11-17. OAuth 2.0 protocol defines roles

WE711 / ZE7111.0

### Notes:

These items are the role definitions that are directly from the specification.



## OAuth 2.0 roles: Common implementation

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 and resource server

The server that provides the authorization activities and serves the protected resource

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.

Typically called the three-legged scenario  
Google and Yahoo OAuth APIs, for example, use this approach

© Copyright IBM Corporation 2015

Figure 11-18. OAuth 2.0 roles: Common implementation

WE711 / ZE7111.0

### Notes:

Although some implementations of OAuth combine the roles into what appears to be a single "server," the roles and protocols are still the same.



## 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 profile** 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 2015

Figure 11-19. OAuth 2.0 roles in the DataPower world (1 of 2)

WE711 / ZE7111.0

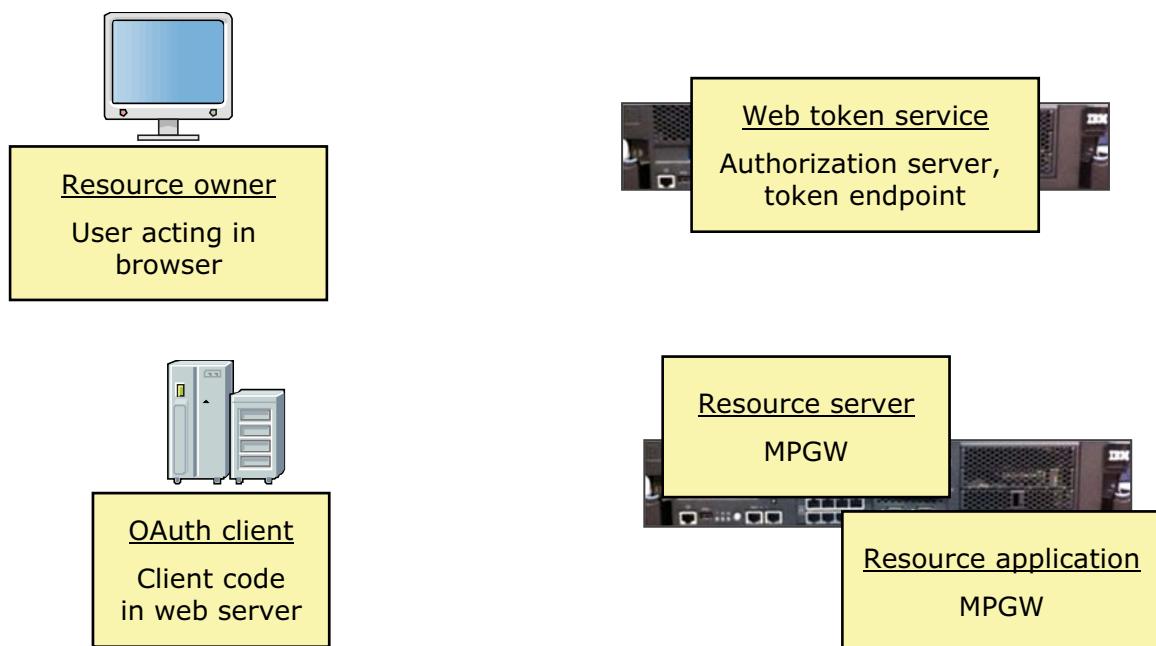
### Notes:

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 2015

Figure 11-20. OAuth 2.0 roles in the DataPower world (2 of 2)

WE711 / ZE7111.0

### Notes:

The resource server is also considered as the enforcement point (EP) of the secured access to the actual back-end application.

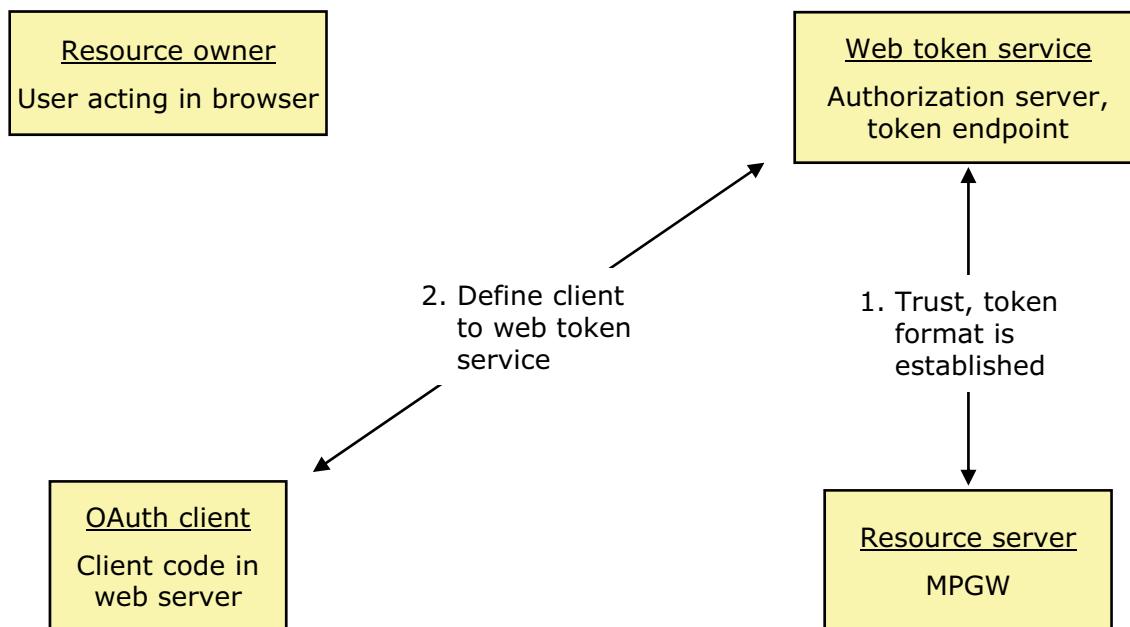
The resource application can be another service on the appliance, or an application service that is running on a server in the trusted domain.

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 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 2015

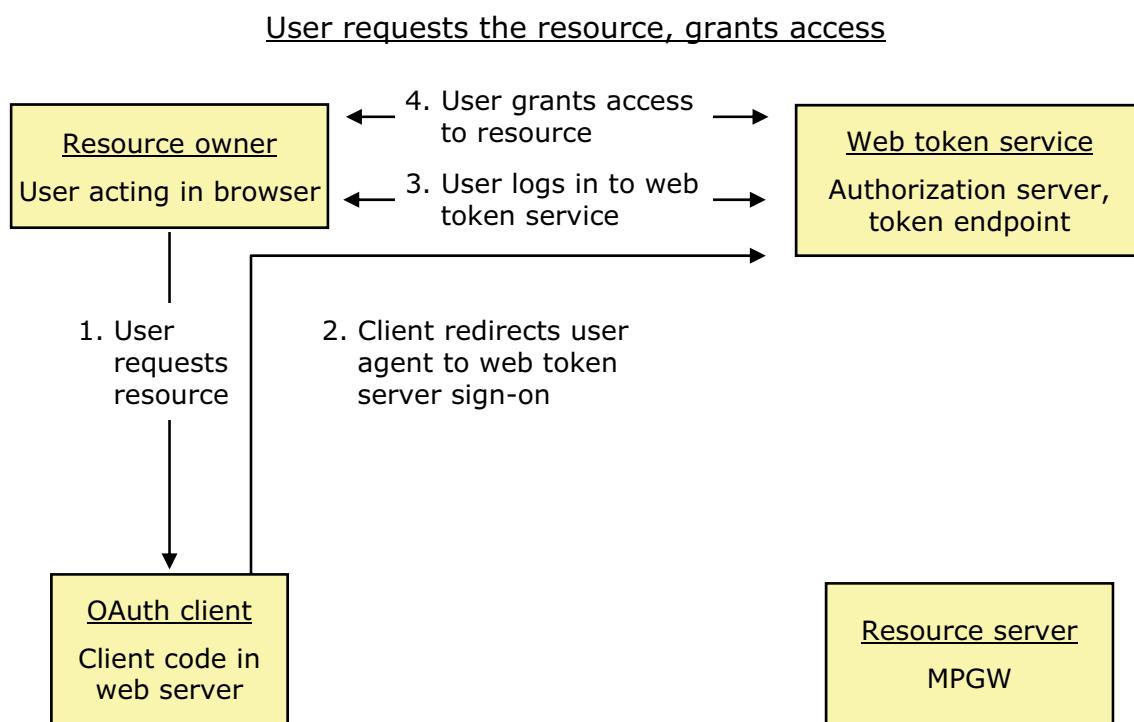
Figure 11-21. Sample three-legged scenario in DataPower (1 of 4)

WE711 / ZE7111.0

### 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 profile object that the web token service and the resource server can access.

## Sample three-legged scenario in DataPower (2 of 4)



© Copyright IBM Corporation 2015

Figure 11-22. Sample three-legged scenario in DataPower (2 of 4)

WE711 / ZE7111.0

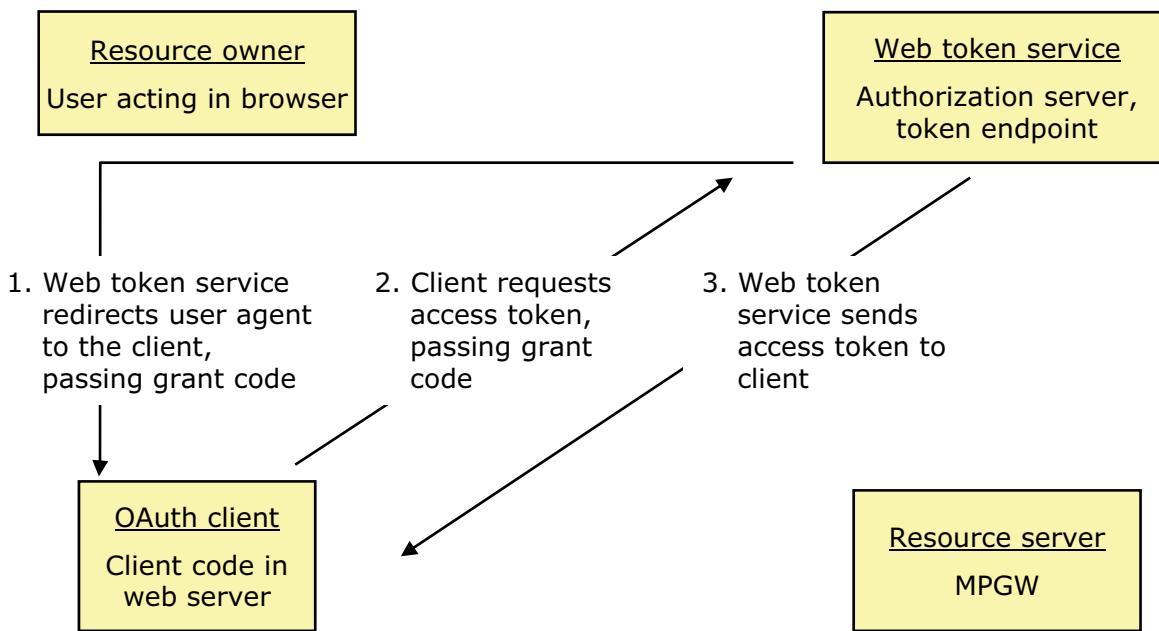
### Notes:

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.

The user requests the resource from the client.

## Sample three-legged scenario in DataPower (3 of 4)

Grant code is used to get access token



© Copyright IBM Corporation 2015

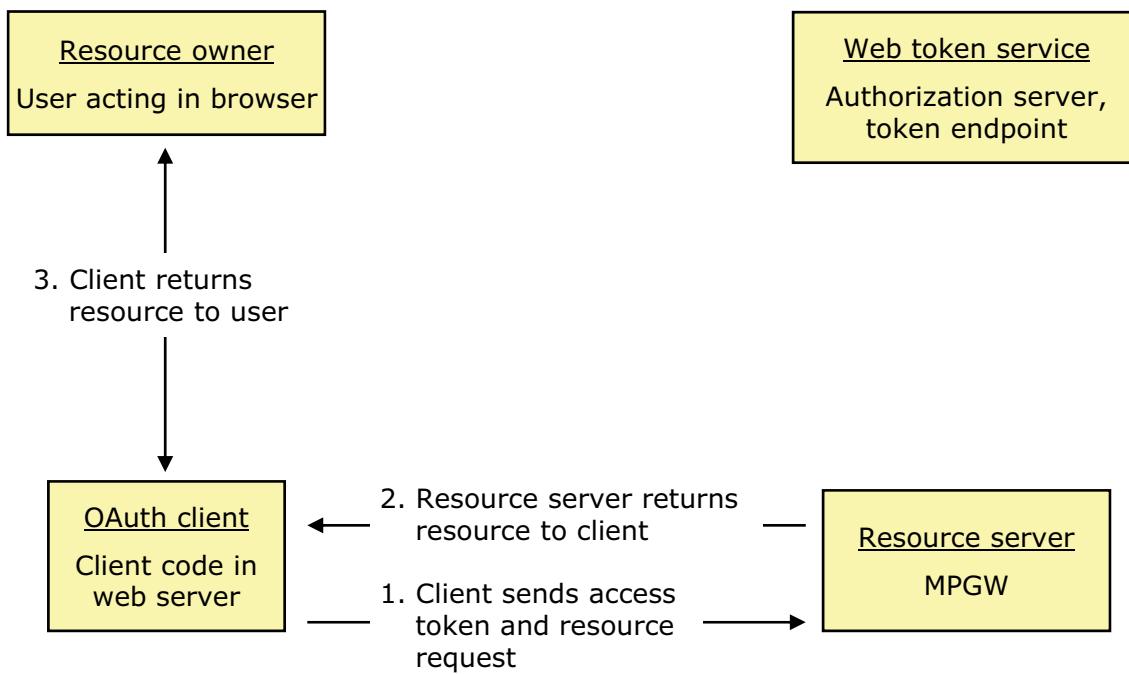
Figure 11-23. Sample three-legged scenario in DataPower (3 of 4)

WE711 / ZE7111.0

### Notes:

## Sample three-legged scenario in DataPower (4 of 4)

Access token is used to get resource



© Copyright IBM Corporation 2015

Figure 11-24. Sample three-legged scenario in DataPower (4 of 4)

WE711 / ZE7111.0

### Notes:



## 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 2015

Figure 11-25. Authorization request

WE711 / ZE7111.0

### Notes:



## 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 2015

Figure 11-26. Authorization response

WE711 / ZE7111.0

### 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 2015

Figure 11-27. Access token request

WE711 / ZE7111.0

### 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 2015

Figure 11-28. Access token response

WE711 / ZE7111.0

### Notes:

## 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 is 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 2015

Figure 11-29. Resource request

WE711 / ZE7111.0

### 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 2015

Figure 11-30. OAuth client and the OAuth Client Profile object

WE711 / ZE7111.0

### 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

© Copyright IBM Corporation 2015

Figure 11-31. DataPower OAuth objects: OAuth Client (1 of 4)

WE711 / ZE7111.0

### Notes:

With the **customized OAuth** option, you can write your own OAuth client behavior in a style sheet.

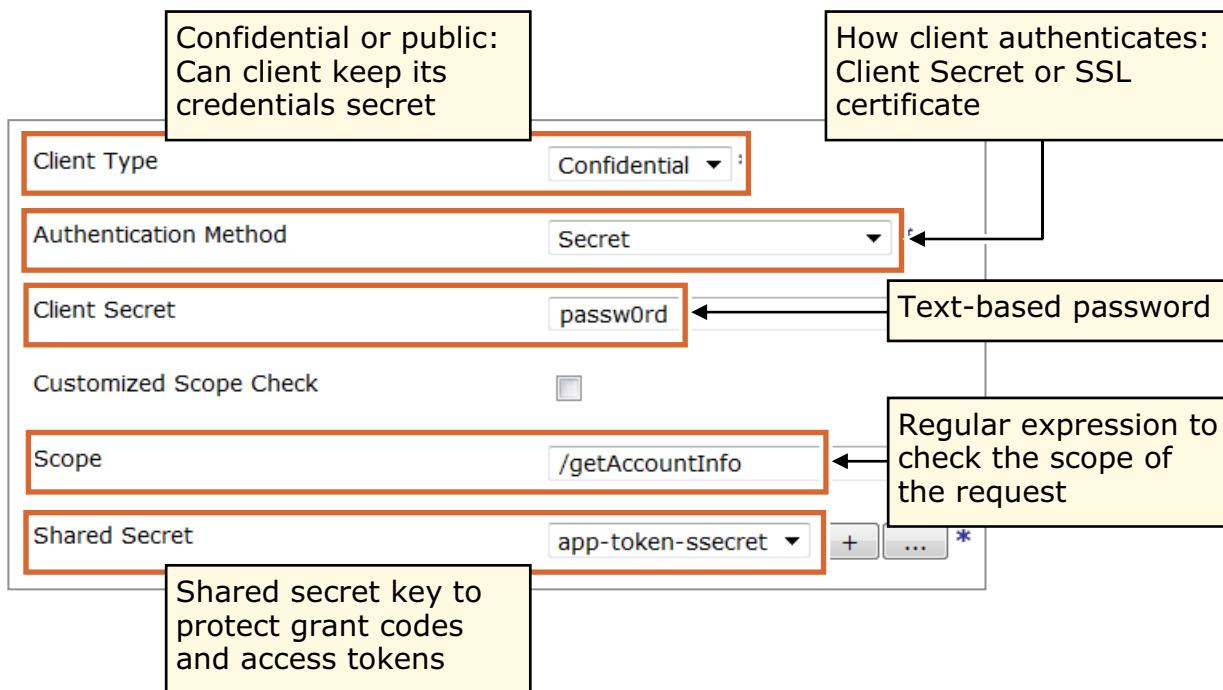
There are four options for OAuth 2.0 authorization grant types:

- With the authorization code, the authorization server sends back a custom redirect URI and an authorization code after it authenticates the resource owner. The authorization code prevents replay attacks. The client application opens the redirect URI with the authorization code to retrieve an access token for a resource.
- With the implicit grant type, the authorization server does not send back an authorization code. It sends back an access token after the resource owner authorizes the client application. This grant type is available for public clients only.
- With the resource owner password credentials grant type, the client application sends the user name and password for a user on the resource server. This grant type assumes a high level of trust between the client application and the resource server.
- With the client credentials grant type, the client application sends its own credentials when it accesses server resources under its own control, or to resources that are previously arranged with the resource server. This grant type is available to confidential client types only.

**Authorization code** and **Implicit grant** grant types are for three-legged OAuth flows. **Resource owner password credential** and **Client credential** are for two-legged OAuth flows.



## DataPower OAuth objects: OAuth Client (2 of 4)



© Copyright IBM Corporation 2015

Figure 11-32. DataPower OAuth objects: OAuth Client (2 of 4)

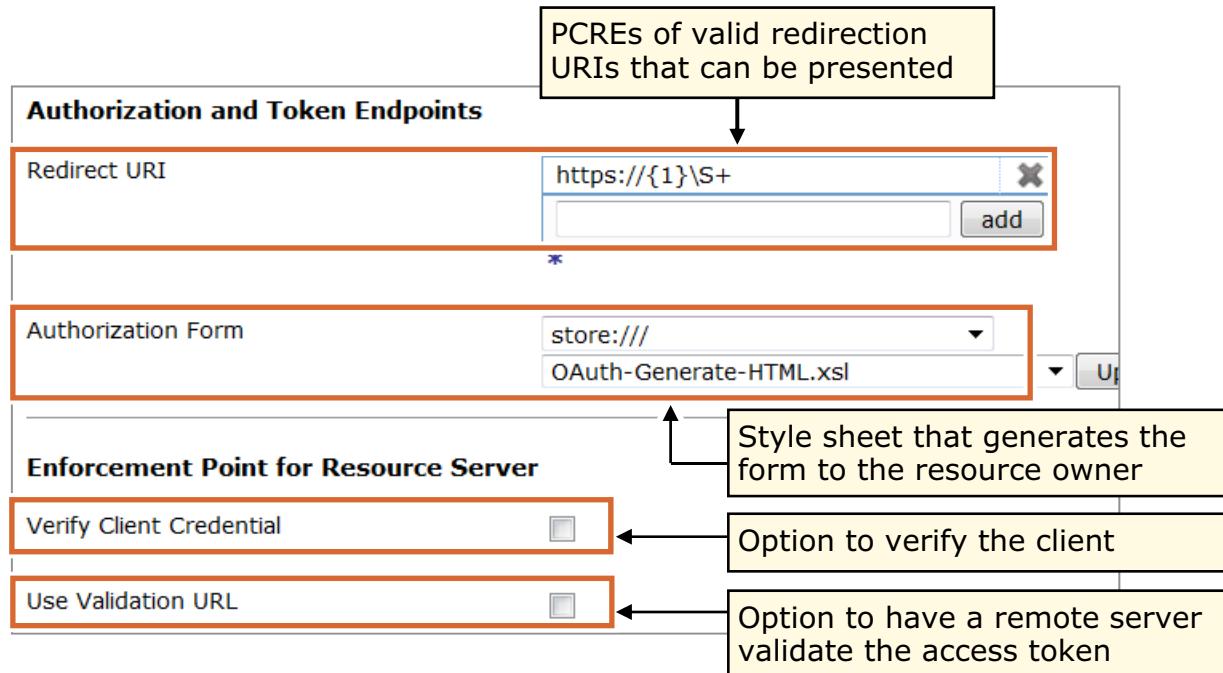
WE711 / ZE7111.0

### Notes:

With **Customized scope check**, you can use a style sheet to do the scope checking, rather than using the **Scope** field entry.



## DataPower OAuth objects: OAuth Client (3 of 4)



© Copyright IBM Corporation 2015

Figure 11-33. DataPower OAuth objects: OAuth Client (3 of 4)

WE711 / ZE7111.0

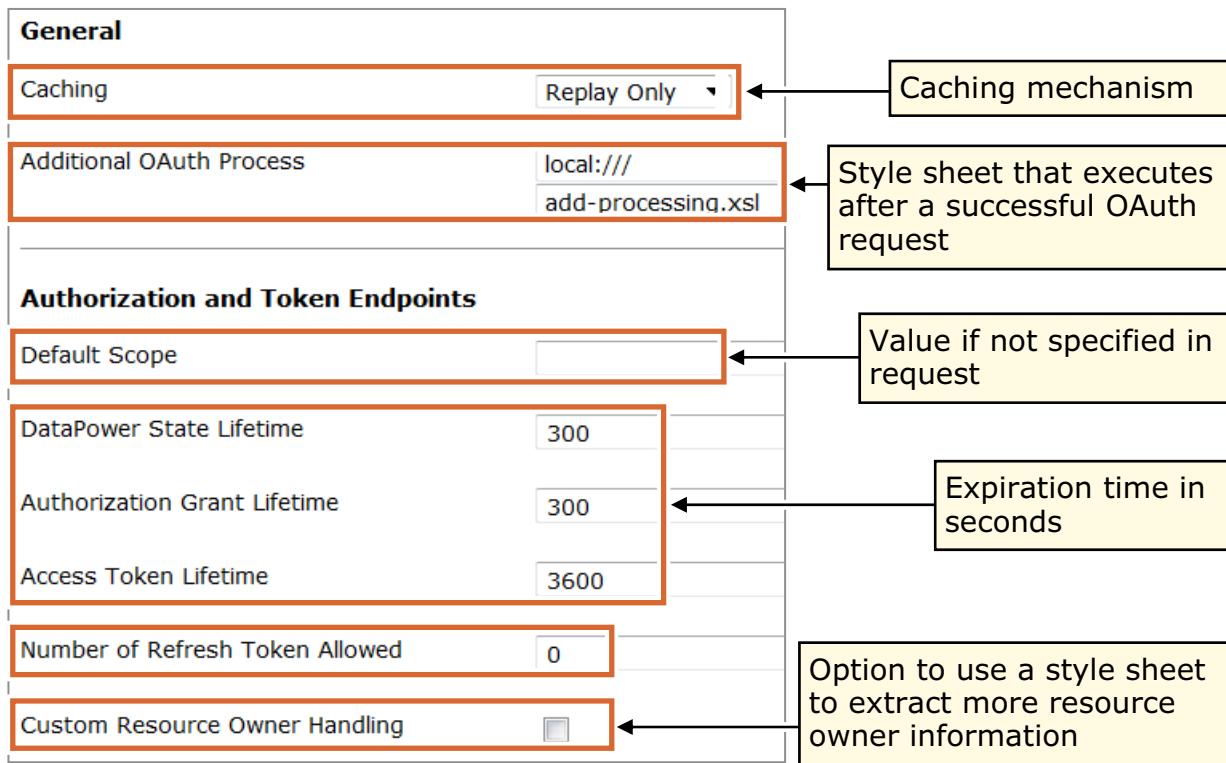
### Notes:

With **Verify client credential**, you can verify the client identity along with using the access token.

**Use validation URL** is used when you want to use a remote server to validate the access token rather than using the DataPower resource server. This situation occurs when the authorization server is not a DataPower web token service.



## DataPower OAuth objects: OAuth Client (4 of 4)



© Copyright IBM Corporation 2015

Figure 11-34. DataPower OAuth objects: OAuth Client (4 of 4)

WE711 / ZE7111.0

### Notes:

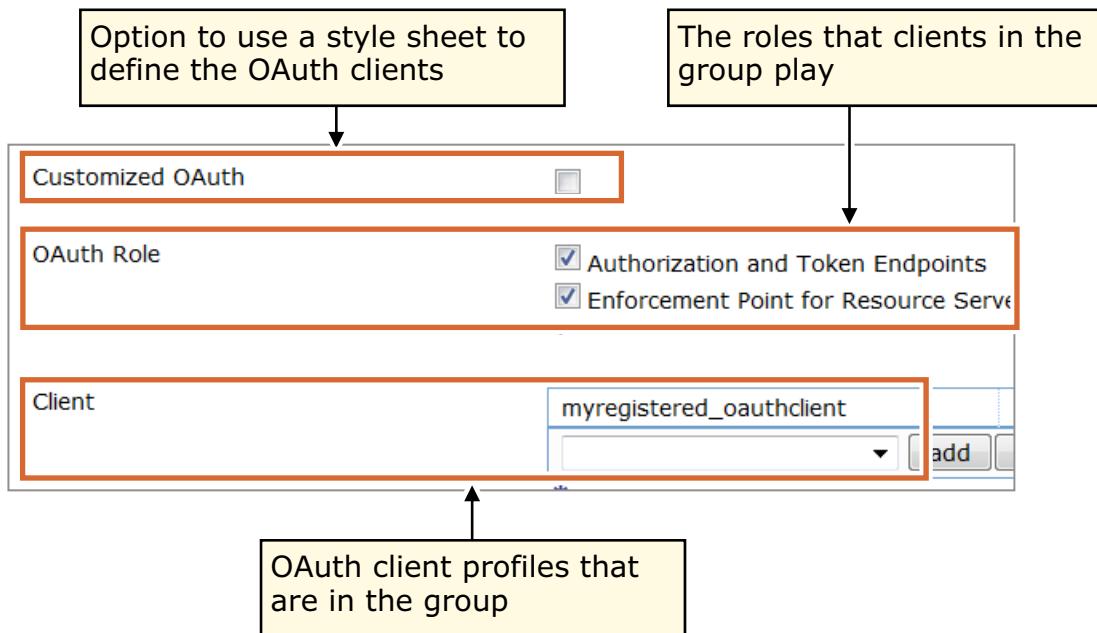
Caching specifies the caching mechanism to be used:

- **Replay Only:** Uses replay cache to prevent replay attacks
- **Token Cache:** Uses system memory to support revocation
- **Custom:** Uses a style sheet that defines how to handle revocation

Other options for adding HTTP headers for resource owner, client ID, scope, and customized information are not shown in this screen capture.

## 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 2015

Figure 11-35. DataPower OAuth objects: OAuth Client Group

WE711 / ZE7111.0

### Notes:



## 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 other parameters

The screenshot shows the XML manager interface for a DataPower Web Token Service. The configuration includes:

- Endpoints**: A table under "Source addresses" with one row:
 

| Local address | Port | SSL | Assign the SSL proxy profile | Credential Character 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 "Endpoints" table.
- Processing policy that is implemented in the service**: A callout box pointing to the "Processing Policy" dropdown.

© Copyright IBM Corporation 2015

Figure 11-36. DataPower OAuth objects: Web Token Service

WE711 / ZE7111.0

### Notes:

You can create your own DataPower service to act as the authorization server and token endpoint. It is much easier to use the web token service type that DataPower provides.

The **Client credential set** specifies the character encoding of the basic authentication values. "Protocol" indicates that the encoding is derived from what is specified in the HTTP protocol.

## 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 2015

Figure 11-37. AAA policy: Key to OAuth behavior

WE711 / ZE7111.0

### Notes:

The OAuth Client Profiles and OAuth Client Groups have no effect until they are referenced from a AAA policy.



## 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 dropdown menu set to "login". In the bottom right corner, there is another dropdown menu set to "OAuth-Code-Client". On the left side of the panel, there are two sections: "HTTP Basic Authentication Realm" and "Registered OAuth clients".

- 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 are two checkboxes: "XPath expression" (unchecked) and "Processing metadata" (checked), with an asterisk (\*) indicating it is required. Below this, there is a dropdown menu set to "oauth-scope-metadata". On the left side, there is a section labeled "Processing metadata items".

© Copyright IBM Corporation 2015

Figure 11-38. AAA policy for the web token service

WE711 / ZE7111.0

### Notes:

In the top screen capture, “HTTP authentication header” is also selected. That choice is why the authentication realm is identified.

The oauth-scope-metadata contains the scope of the request.



## AAA policy for the resource server (1 of 2)

- In Identity extraction, you must select **OAuth** and identify an **OAuth client group**

The screenshot shows a configuration interface for an AAA policy. In the top right corner, there is a checked checkbox labeled "OAuth" with an asterisk (\*) indicating it is required. Below this, a dropdown menu is set to "OAuth-Code-Client". At the bottom left, there is a link labeled "Registered OAuth clients".

- In the Resource extraction phase, you select **URL sent by client** and **Processing metadata**, and specify the **oauth-scope-metadata**

The screenshot shows a configuration interface for the Resource extraction phase. On the left, under "Resource information", there is a list of options with checkboxes: "URL sent to back end" (unchecked), "URL sent by client" (checked), "URI of top level element in..." (unchecked), "Local name of request elem..." (unchecked), "HTTP operation (GET or PO..." (unchecked), "XPath expression" (unchecked), and "Processing metadata" (checked). An asterisk (\*) is present below this list. On the right, under "Processing metadata items", a dropdown menu is set to "oauth-scope-metadata".

© Copyright IBM Corporation 2015

Figure 11-39. AAA policy for the resource server (1 of 2)

WE711 / ZE7111.0

### Notes:

For V6.0 and later, the firmware verifies that the scope agreed to in the access token matches the actual requested scope (resource extraction phase).



## 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
  - Required for pre-V6.0 firmware to compare the authorized scope to the actual requested scope

|            |                                     |   |
|------------|-------------------------------------|---|
| Method     | Custom                              | * |
| Custom URL | local:///accesstoken-check-resource |   |

© Copyright IBM Corporation 2015

Figure 11-40. AAA policy for the resource server (2 of 2)

WE711 / ZE7111.0

### Notes:

## 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 2015

Figure 11-41. Unit summary

WE711 / ZE7111.0

### 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 2015

Figure 11-42. Checkpoint questions

WE711 / ZE7111.0

### 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 2015

Figure 11-43. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 9



Defining a three-legged OAuth scenario that uses DataPower services

© Copyright IBM Corporation 2015

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

10.1

Figure 11-44. Exercise 9

WE711 / ZE7111.0

### 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 2015

Figure 11-45. Exercise objectives

WE711 / ZE7111.0

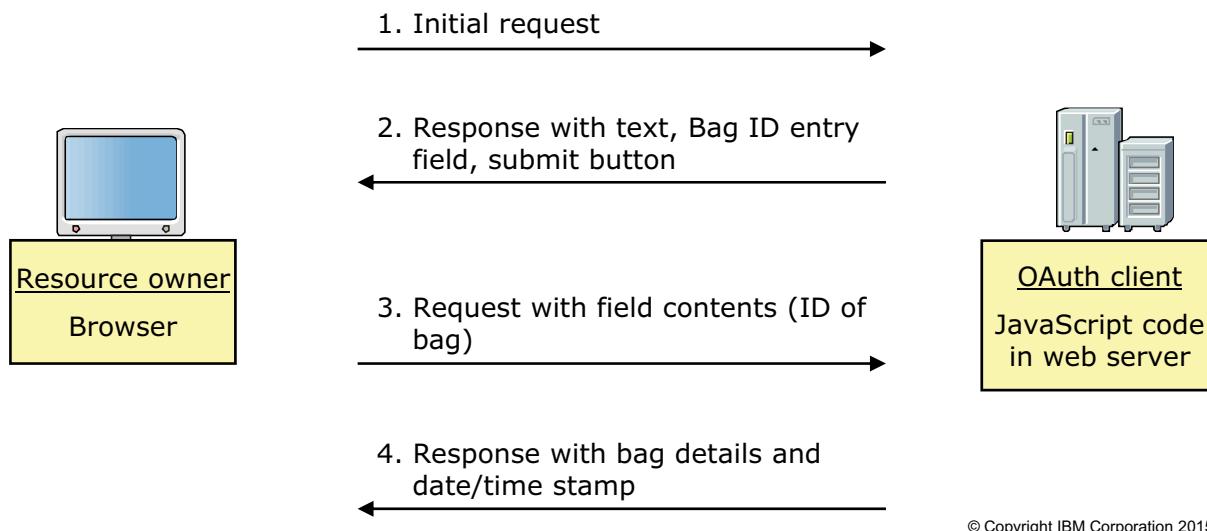
### Notes:



## Exercise overview: User interaction

Application function:

- User enters the ID of a bag to request its status
- Application responds with the current details on the bag, and the time stamp of the request



© Copyright IBM Corporation 2015

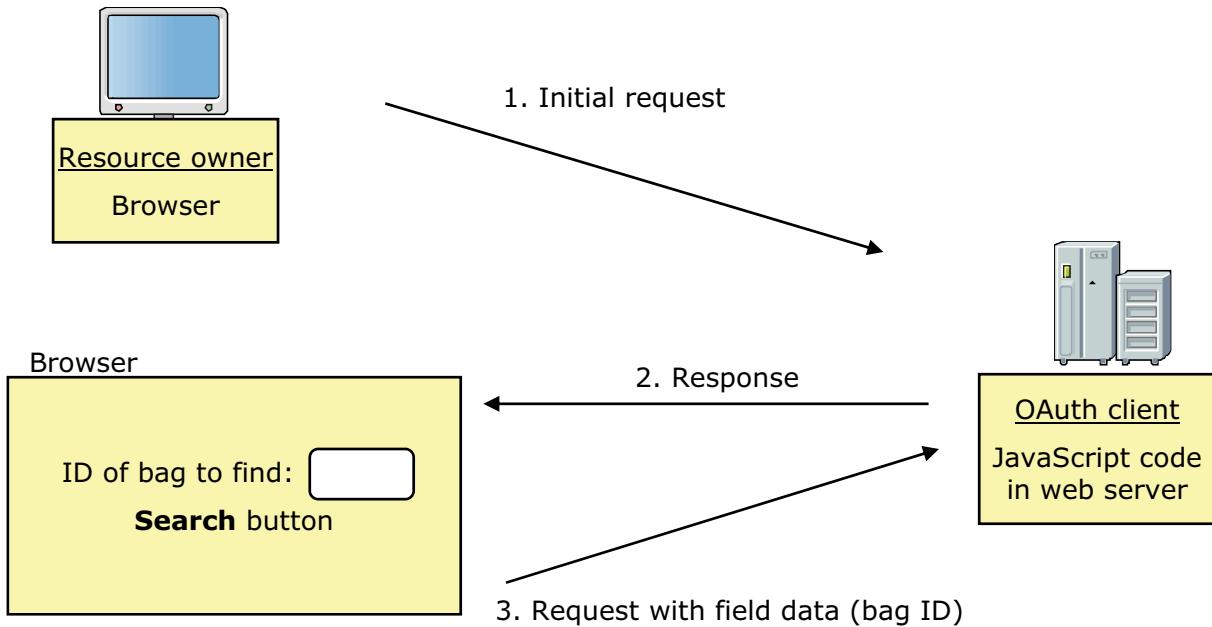
Figure 11-46. Exercise overview: User interaction

WE711 / ZE7111.0

### Notes:



## Exercise overview: OAuth interaction (1 of 4)



© Copyright IBM Corporation 2015

Figure 11-47. Exercise overview: OAuth interaction (1 of 4)

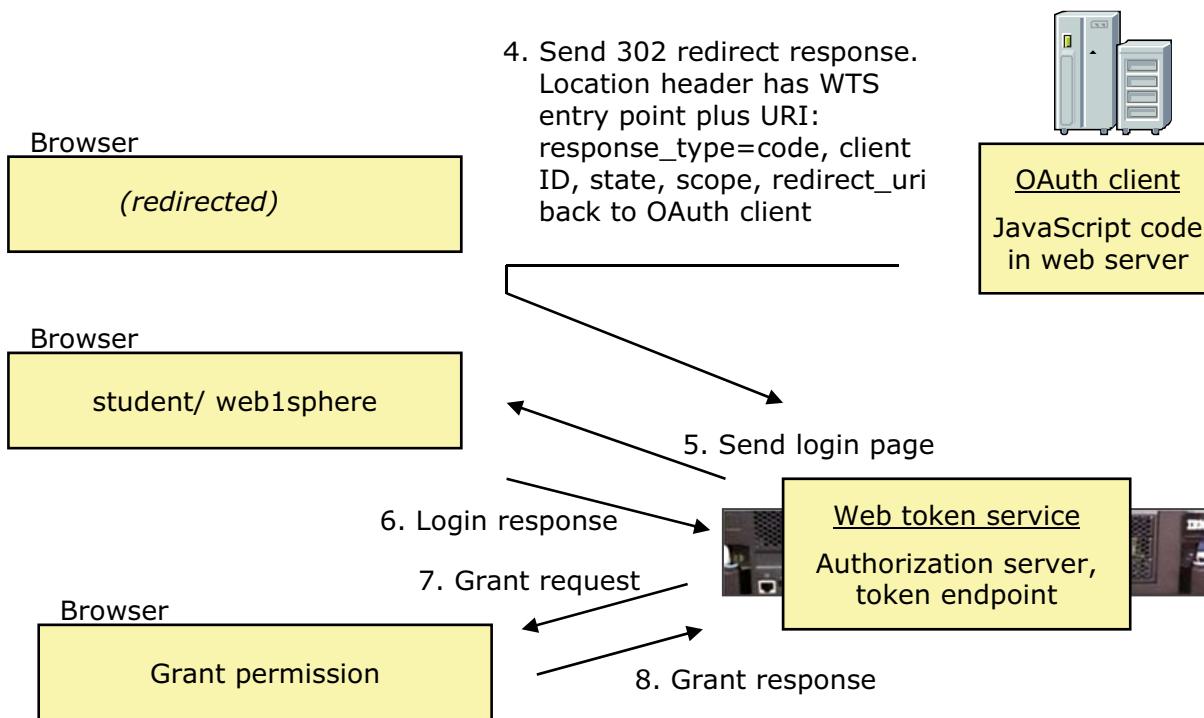
WE711 / ZE7111.0

### Notes:

The OAuth client knows:

- Client ID
- Client secret
- Scope
- State
- Web token service URL
- Resource server URL
- Its own URL

## Exercise overview: OAuth interaction (2 of 4)



© Copyright IBM Corporation 2015

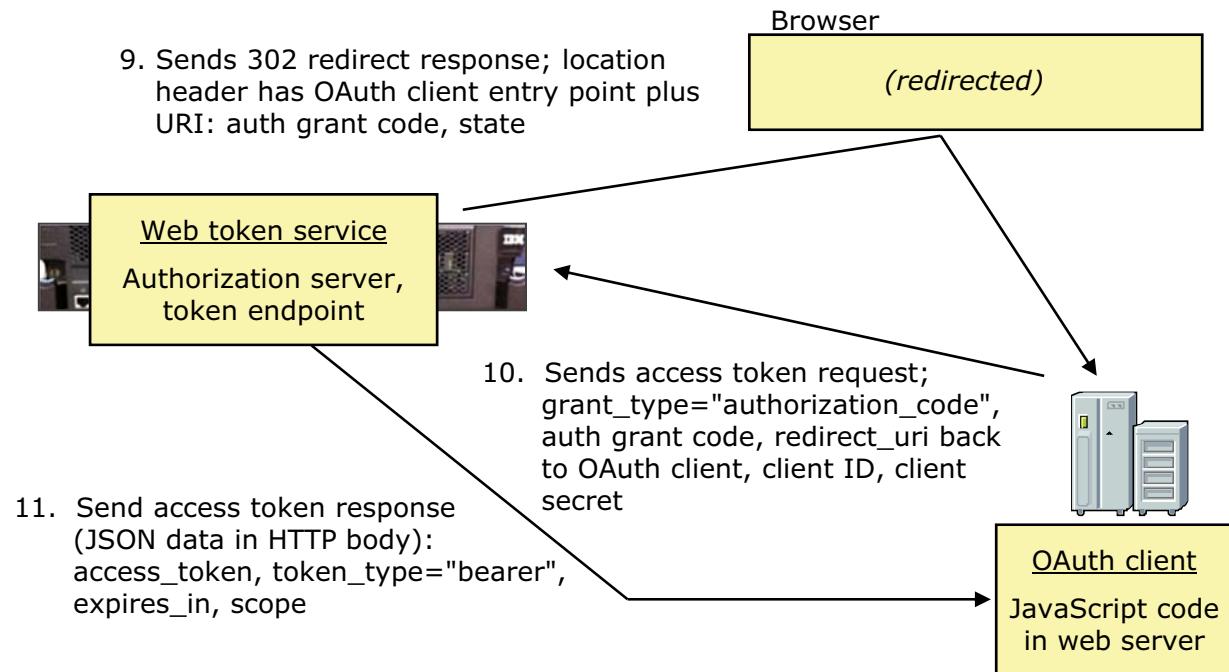
Figure 11-48. Exercise overview: OAuth interaction (2 of 4)

WE711 / ZE7111.0

### Notes:

Notice that the OAuth client is *not* involved in the login and grant interactions. The OAuth client never sees the ID and password of the user.

## Exercise overview: OAuth interaction (3 of 4)



© Copyright IBM Corporation 2015

Figure 11-49. Exercise overview: OAuth interaction (3 of 4)

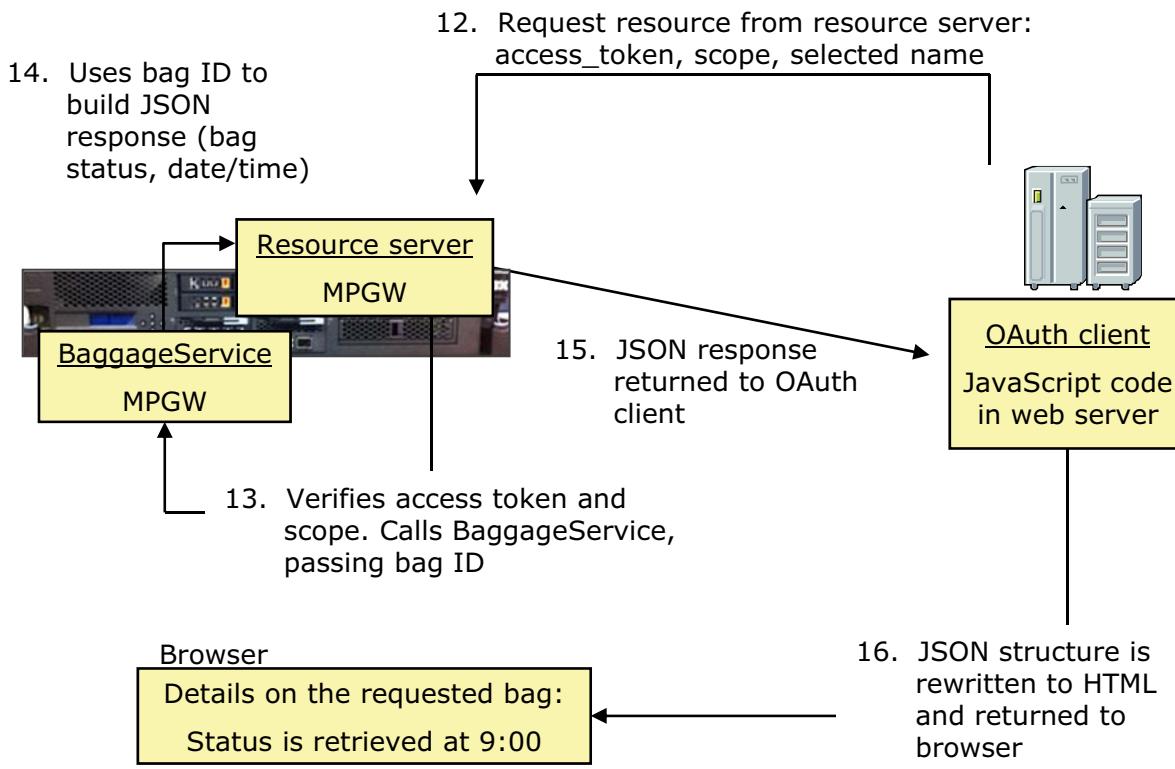
WE711 / ZE7111.0

### Notes:

The user granted access permission to the OAuth client. Now the web token service is verifying the OAuth client before giving it an access token.

Notice that the browser does not see the access token.

## Exercise overview: OAuth interaction (4 of 4)



© Copyright IBM Corporation 2015

Figure 11-50. Exercise overview: OAuth interaction (4 of 4)

WE711 / ZE7111.0

### Notes:

The user granted access permission to the OAuth client. Now the web token service is verifying the OAuth client before giving it an access token.

Notice that the browser does not see the access token.

# Unit 12. DataPower caching

## What this unit is about

DataPower can use on-appliance caching to optimize back-end resource utilization and response time. This unit covers some of the common caching options in DataPower. It covers document cache configuration, the x-dp-cache-key HTTP header, on-appliance versus side caching, and integration with DataPower XC10 and IBM WebSphere eXtreme Scale.

## What you should be able to do

After completing this unit, you should be able to:

- List some of the caching options in DataPower
- Describe the two caching approaches for documents: on-appliance and side cache
- Configure a document cache policy
- Use the DataPower x-dp-cache-key HTTP header
- Describe how to integrate a DataPower XC10 appliance and IBM WebSphere eXtreme Scale into DataPower caching

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM DataPower Gateway Knowledge Center:

[http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)

## Unit objectives

After completing this unit, you should be able to:

- List some of the caching options in DataPower
- Describe the two caching approaches for documents: on-appliance and side cache
- Configure a document cache policy
- Use the DataPower x-dp-cache-key HTTP header
- Describe how to integrate a DataPower XC10 appliance and IBM WebSphere eXtreme Scale into DataPower caching

© Copyright IBM Corporation 2015

---

Figure 12-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## Caching in DataPower

- Services in DataPower should not attempt to save any state
  - Appliance is designed to be stateless
- However, caching is supported in many places to reduce message throughput time
  - Cache sizes are limited
  - Cache entries expire or are replaced
- Cached resources
  - Services/developer-related
    - Document
    - Style sheet
    - AAA policy (authentication and authorization results)
  - Operations/administrator-related
    - DNS (responses from DNS servers)
    - LDAP (LDAP connection pool)
    - RBM (LDAP authentication)

© Copyright IBM Corporation 2015

Figure 12-2. Caching in DataPower

WE711 / ZE7111.0

### Notes:

The services and developer-related cache resources are the resources that a developer might configure in the development of DataPower services.

The XML manager object that is associated with each service specifies the style sheet and document cache options.

The “Advanced” part of the AAA policy configuration wizard shows the caching options.

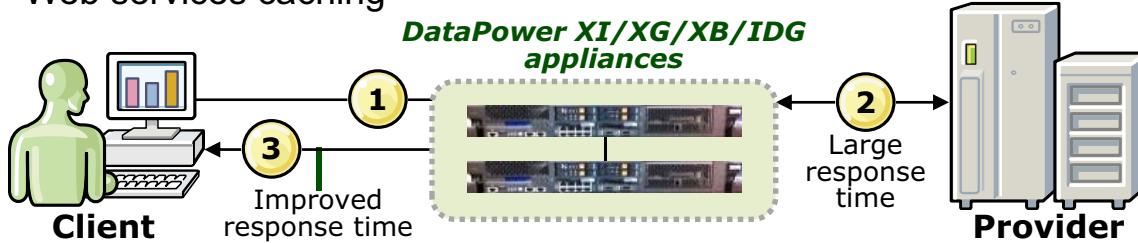
The Operations-relayed caching options are not covered in this course.

“DNS” is domain name server.

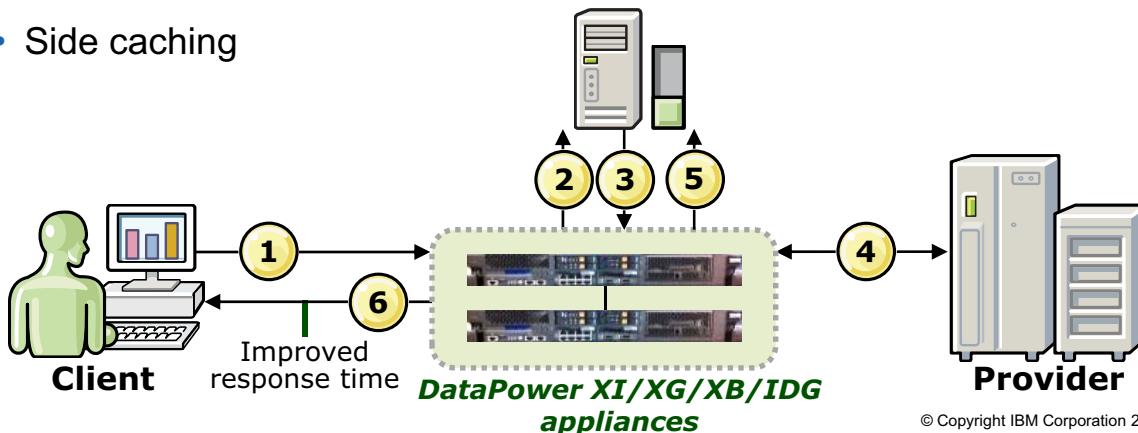
“RBM” is role-based management.

## Document caching: On and off the appliance

- Web application caching (caching reverse proxy)
- Web services caching



- Side caching



© Copyright IBM Corporation 2015

Figure 12-3. Document caching: On and off the appliance

WE711 / ZE7111.0

### Notes:

The web application and web services caching uses the document cache on the appliance. The purpose is to find responses that are cached on the appliance and return them from the cache. By avoiding a call to the application servers, you greatly reduce the response time to the client, resource usage on the appliance, and load on the application servers.

The client requests a resource (1). If the resource is not in the document cache, the request is sent to the application server (2). The response is cached, and then sent to the client. On a subsequent request for the same resource (1), the response is found already in the cache, and it is returned immediately to the client (3).

The on-appliance document cache is restricted in size. For larger caching requirements, you can implement side-caching. In this scenario, the cache resides in an off-appliance cache such as an XC10 appliance or an IBM WebSphere eXtreme Scale server. The flow for this scenario is similar to the on-appliance scenario, except you must code some of the caching behavior in the service. On first request for a resource (1), the service sends a REST call to the cache (2). If the resource is not there (3), the service sends the request to the application server (4). On response, the service writes the resource to the side cache by using a REST call (5), and then returns it to the client (6).

On the second request (1), the REST call from the service (2) retrieves the resource (3), and the response is sent to the client (6), bypassing the request to the application server.



## Document and style sheet caching in the XML manager

The screenshot shows the 'Configure XML Manager' interface. The top navigation bar includes tabs for Main, XML Parser, Document Cache, Extension Functions, Document Cache Policy (which is highlighted with a red box), and Schema Validation. Below the tabs, there's a sidebar with an 'Apply' button and a list of cache settings:

- Max cache size
- Max number of documents

The main configuration area has several sections:

- Administrative state:** A radio button group where 'enabled' is selected.
- Comments:** A text input field containing 'Default XML-Manager'.
- URL Refresh Policy:** A dropdown menu currently set to '(none)'.
- Compile Options Policy:** A dropdown menu currently set to '(none)', with a yellow callout box labeled 'Maximum number of style sheets' pointing to it.
- XSL Cache Size:** An input field containing '256' followed by a dropdown menu set to 'stylesheets'.

At the bottom right of the interface, there are links for Export, View Log, View Status, and Help, along with Flush StyleSheet Cache, Flush Document Cache, and Flush LDAP Cache buttons.

© Copyright IBM Corporation 2015

Figure 12-4. Document and style sheet caching in the XML manager

WE711 / ZE7111.0

### Notes:

## Document cache policy (1 of 2)

- URL match expression: Shell-style match pattern that identifies which documents to cache
- Policy type
  - Fixed: Cached documents use the TTL specified in this policy
  - Protocol-based: Cached documents use the TTL as determined by the HTTP 1.1 caching rules
  - No cache: Documents that match the match expression are not cached
- TTL: Specifies the validity period in seconds for a document in the cache, for **fixed** policy types only
- Priority: Value from 1 - 255 that specifies the priority of a document to add or remove from the cache
- XC10 grid: XC10 grid object to use for caching documents

| Document Cache Policy |                |     |          |           |                          |                       |                         |                       |   |   |
|-----------------------|----------------|-----|----------|-----------|--------------------------|-----------------------|-------------------------|-----------------------|---|---|
| URL Match Expression  | Policy Type    | TTL | Priority | XC10 Grid | Cache Back-end Responses | HTTP Cache Validation | Return Expired Document | RESTful Invalidations | Cache Response to POST and PUT Requests |   |
| *                     | Protocol-Based | 0   | 128      |           | on                       | on                    | on                      | on                    | off                                     |   |

© Copyright IBM Corporation 2015

Figure 12-5. Document cache policy (1 of 2)

WE711 / ZE7111.0

### Notes:

You can use wildcards to define a match pattern as follows:

- \*: The string wildcard matches 0 or more occurrences of any character.
- ?: The single character wildcard matches one occurrence of any single character.
- [ ]: The delimiters bracket a character or number range or set of specific values.

If two different policies have the same priority, the first policy in the alphabetized list is selected.

If you specify an XC10 grid object, the DataPower firmware manages the REST calls to the XC10 appliance for the document cache. This option gives you the benefits of off-appliance caching with the simplicity of a document cache policy. No extra style sheet coding is needed.



## Document cache policy (2 of 2)

- Cache back-end responses: Whether to cache the response of an HTTP GET request.
- HTTP cache validation: Whether to respect the HTTP request header cache validation requests.
- Return expired document: Whether to return expired content if the client requests it and if the appliance cannot contact the origin server.
- RESTful invalidation: Whether to invalidate the document cache entry when a PUT, POST, or DELETE request matches the entry.
- Cache response to POST and PUT requests: Whether to cache the response to POST and PUT requests. “Cache back-end responses” must also be enabled.

| Document Cache Policy |                |     |          |           |                          |                       |                         |                      |   |  |
|-----------------------|----------------|-----|----------|-----------|--------------------------|-----------------------|-------------------------|----------------------|---|--|
| URL Match Expression  | Policy Type    | TTL | Priority | XC10 Grid | Cache Back-end Responses | HTTP Cache Validation | Return Expired Document | RESTful Invalidation | Cache Response to POST and PUT Requests |  |
| *                     | Protocol-Based | 0   | 128      |           | on                       | on                    | on                      | on                   | off                                     |  |

© Copyright IBM Corporation 2015

Figure 12-6. Document cache policy (2 of 2)

WE711 / ZE7111.0

### Notes:

If HTTP cache validation is enabled, the cache accepts the HTTP request headers: If-Modified-Since, If-Unmodified-Since, If-Match, and If-None-Match.

## Document cache programming interface

### Unique cache-key "x-dp-cache-key" request header

- In a style sheet, select something in the request that makes it unique (order number, customer ID) and specify it as the “cache key”
  - GatewayScript example that uses the incoming URI as the cache key (URI has unique ID in it):

```
var sm = require('service-metadata');
var hm = require('header-metadata');
// get incoming URI using dotted notation
var URI = sm.URI;
// set x-dp-cache-key header to incoming URI
hm.current.set("x-dp-cache-key", URI);
```

- Policy is still enforced based on URL matching
  - If a header is available, and its value is non-empty, the URL and value is used for a unique identifier into the cache
  - If a header is available and its value is empty, no cache operation is performed
  - If header is not available, the URL is used for a unique identifier
- x-dp-cache-key HTTP header exists in appliance only, does not “leave the box”

© Copyright IBM Corporation 2015

Figure 12-7. Document cache programming interface

WE711 / ZE7111.0

### Notes:



## Document cache providers

- Document Cache list shows use counts and sizes, **Flush** button

| Document Cache         |           |            |                      |                       |                                      |
|------------------------|-----------|------------|----------------------|-----------------------|--------------------------------------|
| manager                | max count | used count | max size (kilobytes) | used size (kilobytes) |                                      |
| caching-key-test       | 5000      | 0          | 1024                 | 0                     | <input type="button" value="Flush"/> |
| caching-mgr            | 5000      | 0          | 1024                 | 0                     | <input type="button" value="Flush"/> |
| default                | 5000      | 0          | 0                    | 0                     | <input type="button" value="Flush"/> |
| default-attempt-stream | 5000      | 0          | 0                    | 0                     | <input type="button" value="Flush"/> |

- Document Status lists the URL and cache key for the cache entries

| Document Status  |            |   |         |                             |  |
|------------------|------------|---|---------|-----------------------------|--|
| XML Manager      | Identifier | URL                                       | Expiry  | Cache Key                   |  |
| CacheXML_Manager | 1          | http://dp_internal_ip:2068/BaggageService | expired | /BaggageService/Bag?id=1589 |  |
| CacheXML_Manager | 1          | http://dp_internal_ip:2068/BaggageService | expired |                             |  |

© Copyright IBM Corporation 2015

Figure 12-8. Document cache providers

WE711 / ZE7111.0

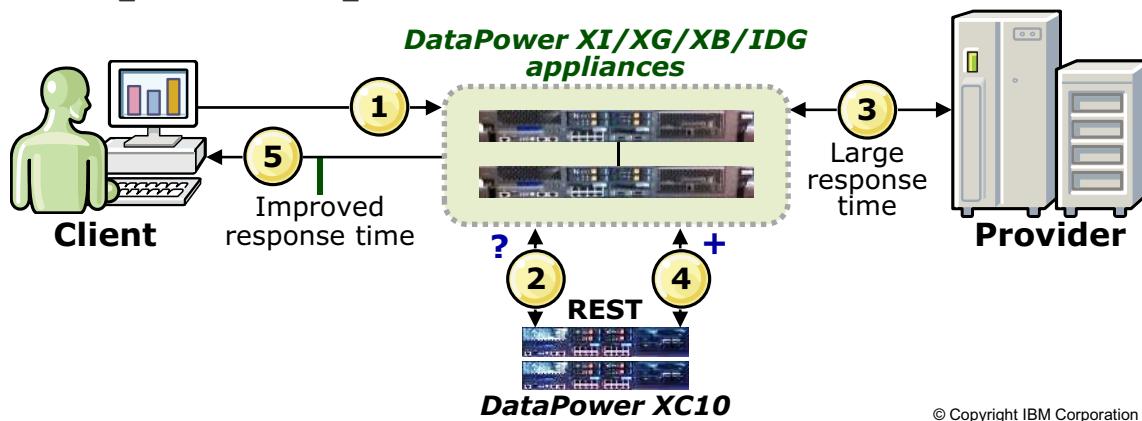
### Notes:

The reference to "kilobytes" in the **max size** and **used size** columns is an error. It should be "kilobytes". This error is in plan to be corrected.

## XC10 integration overview

WebSphere DataPower XC10 is a caching appliance

- Hardware-based, optimized, and hardened “drop-in” cache solution
  - Scalable, highly available, and simple to use
- Interaction with XC10 resources from within DataPower
  - Discover XC10 collective members, create an XC10 data grid, clear an XC10 data grid
- XC10 can be integrated with document cache automatically
  - Hand-coding of integration still possible, see developerWorks article:  
[http://www.ibm.com/developerworks/websphere/library/techarticles/1111\\_nijhawan/1111\\_nijhawan.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1111_nijhawan/1111_nijhawan.html)



© Copyright IBM Corporation 2015

Figure 12-9. XC10 integration overview

WE711 / ZE7111.0

### Notes:

## XC10 integration in document cache logic flow

Cacheable request arrives (1), DataPower reaches out to the XC10:

- If the resource is stored on the XC10, DataPower loads it into the on-appliance document cache, and then DataPower inspects it
  - For example, is it expired?
  - Does it meet the requested validation?
  - If it is expired, DataPower reaches out to the origin server (3), loads the response into the on-appliance document cache, and then writes the response to the XC10 (4)
  - If it is not expired, but it does not pass validation, DataPower reaches out to the origin server (3)
  - If it is not expired, and it does pass validation, DataPower uses the cached document (5)
- If the resource is *not* stored on the XC10, DataPower reaches out to the origin server (3), loads the response into the on-appliance document cache, and then writes the response to the XC10 (4)
- If DataPower is unable to reach the XC10, it marks the temporary document that is stored in the on-appliance document cache as “expired,” and then reaches out to the origin server (3)
  - DataPower does not attempt to update the XC10 with the response from the origin server

© Copyright IBM Corporation 2015

Figure 12-10. XC10 integration in document cache logic flow

WE711 / ZE7111.0

### Notes:



## XC10 integration URL Opener user interface

XC10 integration supplies a URL opener so that a style sheet can call XC10 REST interface

`XC10://object/operation?parameters`

- XC10: Protocol identifier
- Object: XC10 Grid configuration
- Operation: Key-value store operation
  - Read, write, and delete
- Parameters: Query parameters
  - `Key=key`: Required; `key` can be empty for a delete operation
  - `TTL=ttl`: Optional; only used on write and `EvictionStrategy` set to either “lat” and “lut,” and it defaults to a TTL associate with the grid
  - `Map=map`: Optional; defaults to the `object`’s grid name property
  - `EvictionStrategy=strategy`: Optional; none, ct (creation time), lat (the last access time), and lut (the last update time)
  - `LockingStrategy=strategy`: Optional; optimistic, or pessimistic
- URL Opener not a valid back-end destination

© Copyright IBM Corporation 2015

Figure 12-11. XC10 integration URL Opener user interface

WE711 / ZE7111.0

### Notes:



## IBM WebSphere eXtreme Scale (1 of 2)

- IBM WebSphere eXtreme Scale is an elastic, scalable, in-memory data grid (IMDG)
- The IMDG
  - Dynamically caches, partitions, replicates, and manages application data and business logic throughout multiple servers
- eXtreme Scale is not currently as integrated with DataPower as is the XC10
  - But it uses the same REST API as the XC10
- eXtreme Scale 7.1.1 Knowledge Center: Accessing data with the REST data service  
[http://www.ibm.com/support/knowledgecenter/SSTVLU\\_7.1.1/com.ibm.websphere.extremescale.doc/txsrestprogram.html](http://www.ibm.com/support/knowledgecenter/SSTVLU_7.1.1/com.ibm.websphere.extremescale.doc/txsrestprogram.html)

© Copyright IBM Corporation 2015

---

Figure 12-12. IBM WebSphere eXtreme Scale (1 of 2)

WE711 / ZE7111.0

### Notes:

## IBM WebSphere eXtreme Scale (2 of 2)

- To use eXtreme Scale as a side cache for DataPower, you must write style sheets that:
  - Determines whether the message should use the cache
  - Queries the eXtreme Scale cache by using the REST API and a url-open statement
  - Returns the result to the client, or sends the request on to the back-end server
  - Writes the response to the eXtreme Scale cache and sending it to the client
- Use the developerWorks article on DataPower and XC10 integration to guide you in the eXtreme Scale integration:  
[http://www.ibm.com/developerworks/websphere/library/techarticles/1111\\_nijhawan/1111\\_nijhawan.html](http://www.ibm.com/developerworks/websphere/library/techarticles/1111_nijhawan/1111_nijhawan.html)
  - Included materials contain sample style sheets

© Copyright IBM Corporation 2015

Figure 12-13. IBM WebSphere eXtreme Scale (2 of 2)

WE711 / ZE7111.0

### Notes:

## Unit summary

Having completed this unit, you should be able to:

- List some of the caching options in DataPower
- Describe the two caching approaches for documents: on-appliance and side cache
- Configure a document cache policy
- Use the DataPower x-dp-cache-key HTTP header
- Describe how to integrate a DataPower XC10 appliance and IBM WebSphere eXtreme Scale into DataPower caching

© Copyright IBM Corporation 2015

Figure 12-14. Unit summary

WE711 / ZE7111.0

### Notes:

## Checkpoint questions

1. True or False: To specify a cache key other than the URL, specify the x-dp-cache-key private header along with the request.
2. If a service response is cacheable and still valid, DataPower functions by (Select 4):
  - A. Performing a cache lookup
  - B. Using the cached response to fulfill the request
  - C. Checking whether the cached entry's lifetime is expired
  - D. Loading the response in a cached bitmap
  - E. Reaching out to the origin server to fulfill the request

© Copyright IBM Corporation 2015

Figure 12-15. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

1.

2.



## Checkpoint answers

1. **True.** To specify a cache key other than the URL, specify the x-dp-cache-key private header along with the request.
2. **A, B, C, and D.** If a service response is cacheable and still valid, DataPower functions by:
  - A. Performing a cache lookup
  - B. Using the cached response to fulfill the request
  - C. Checking whether the cached entry's lifetime is expired
  - D. Loading the response in a cached bitmap

© Copyright IBM Corporation 2015

Figure 12-16. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 10



Configuring a response cache

© Copyright IBM Corporation 2015

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

10.1

Figure 12-17. Exercise 10

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Create and configure an XML Manager
- Create and configure a document cache policy
- Use the x-dp-cache-key HTTP header

© Copyright IBM Corporation 2015

Figure 12-18. Exercise objectives

WE711 / ZE7111.0

### Notes:

# Unit 13. Integrating with IBM MQ

## What this unit is about

This unit describes how to configure the DataPower appliance to communicate with IBM MQ. You learn how to receive and put messages on IBM MQ queues. You also learn how DataPower manages transactions between IBM 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 an IBM MQ front-side handler
- Configure an IBM MQ back-end URL
- Manage transactionality between IBM MQ queue managers

## How you will check your progress

- Checkpoint
- Hands-on exercise

## References

IBM DataPower Gateway Appliances Version 7.1 Knowledge Center:

[www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## Unit objectives

After completing this unit, you should be able to:

- Create a multi-protocol gateway with an IBM MQ front-side handler
- Configure an IBM MQ back-end URL
- Manage transactionality between IBM MQ queue managers

© Copyright IBM Corporation 2015

---

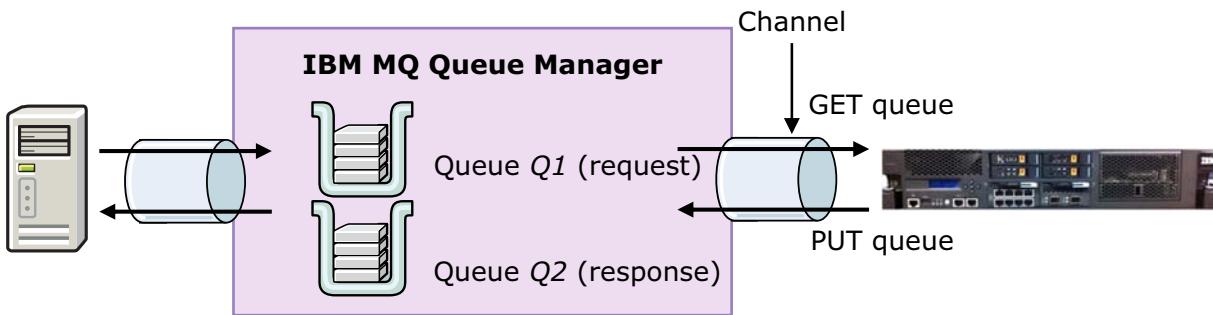
Figure 13-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## IBM MQ fundamentals

- A *queue manager* manages a container for messages that are sent over an IBM 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*
  - An IBM MQ client (such as the 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 2015

Figure 13-2. IBM MQ fundamentals

WE711 / ZE7111.0

### Notes:

IBM MQ allows asynchronous message communication across a network. If HTTP communication is analogous to telephone calls, then message delivery over IBM 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.

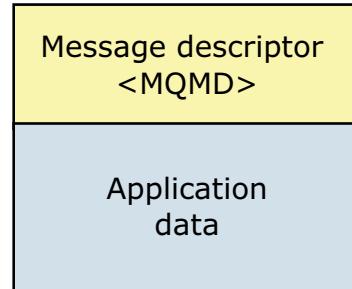
FIFO stands for first-in first-out. Queues mainly work in a FIFO fashion unless a special weighting for messages is implemented.

IBM MQ does all network communications over a **channel**. More specifically, the software program that allows network communication between an IBM MQ client and the IBM MQ queue manager is known as a **client channel**. The channel is a program that runs on the same host as the IBM 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.



## IBM MQ message

- IBM 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 2015

Figure 13-3. IBM MQ message

WE711 / ZE7111.0

### Notes:

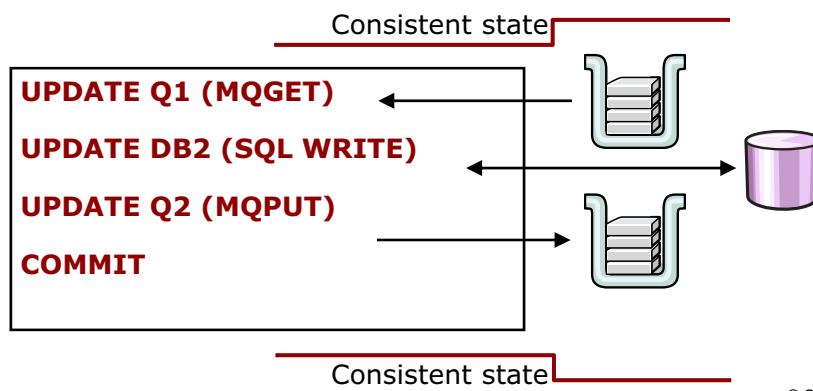
Controlling information within an IBM MQ message can include the message priority, reply queue name, correlation ID, and more.

Every message has a message identifier, which is determined from 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. Therefore, in a reply message 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 2015

Figure 13-4. Transactions

WE711 / ZE7111.0

### 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 IBM 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 IBM MQ

- The DataPower XI52 device can exchange messages with IBM MQ systems
  - DataPower XI52 provides an enhanced implementation of the IBM IBM MQ client
  - DataPower IBM 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 IBM MQ
  - Messages originating within or outside of IBM MQ can flow easily to and from another IBM 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 2015

Figure 13-5. DataPower support for IBM MQ

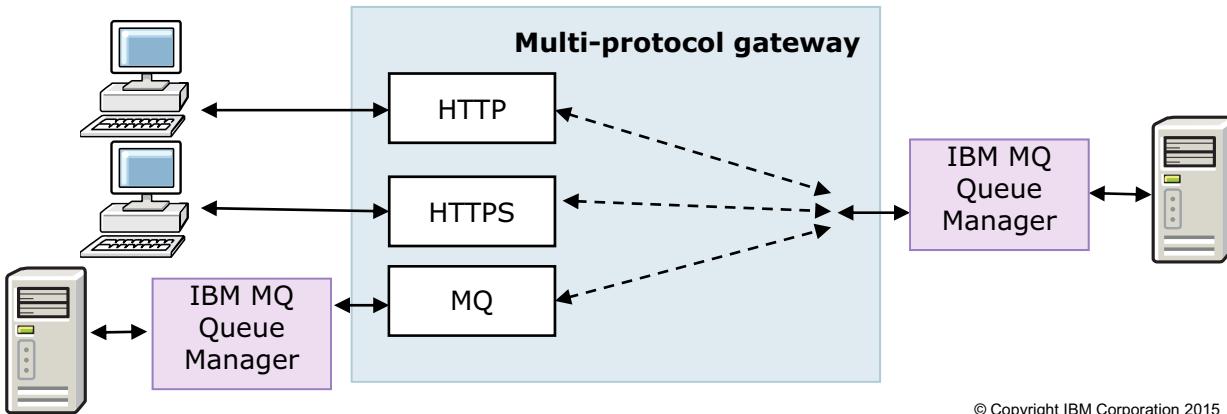
WE711 / ZE7111.0

### Notes:

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 IBM MQ access

- The multi-protocol gateway can be configured to accept requests from an IBM MQ system
  - Request and response messages are placed in queues on an IBM MQ Queue Manager
  - All requests are sent to the back-end web service over another set of IBM MQ queues
  - Web service request messages that pass through the gateway execute a service policy



© Copyright IBM Corporation 2015

Figure 13-6. Provide IBM MQ access

WE711 / ZE7111.0

### Notes:

Contact your IBM MQ administrator for the host name, port, and queue names in your application. Setting up an IBM 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 IBM 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.

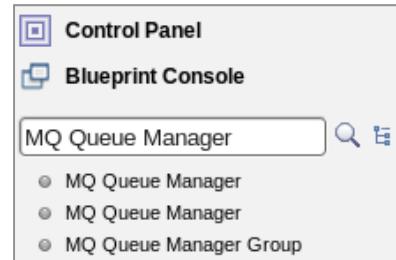
If such a message is found, the multi-protocol gateway can again apply any configured processing policy actions to the response and returns the reply to the requesting HTTP client. This message includes error responses from the back-end application server. If no response is found, the MPGW generates an error to the front side client.

The web service proxy can also use the MQ front side handler.

## WebSphere MQ queue manager overview

### 1. To create a WebSphere MQ Queue Manager object:

- The WebSphere MQ Queue Manager object can be selected from the Control Panel
- The WebSphere MQ Queue Manager object can be created when creating a WebSphere MQ front side handler by clicking the **Add (+)** button under Queue Manager



### 2. The WebSphere MQ Queue Manager object contains four tabs:



- Main:** The main configuration items
- Connections:** Connections, retry, and security
- CCSI:** Coded character set ID
- MQCSP:** WebSphere MQ connection security parameter

© Copyright IBM Corporation 2015

Figure 13-7. WebSphere MQ queue manager overview

WE711 / ZE7111.0

### Notes:

In IBM MQ, a component that is called a queue manager manages distributed send and receive queues. The queue manager provides messaging services for communicating applications by periodically monitoring and polling queues, by ensuring that sent messages are directed to the correct receive queue, or that messages are routed to another queue manager. This queue manager object on the device corresponds to a queue manager that is running on another host on the network. The properties set here enable communication between the device and the queue manager.



## WebSphere MQ queue manager overview: Main tab

- General configuration
  - Administration state
  - Host address and port (1414)
  - Channel name
  - IBM MQ user name
  - XML manager name
  
- Units of work and backout
  - Units of work: 0 or 1
    - When set to 0 (zero) messages are put with no provision for rollback
    - When set to 1, the appliance uses sync points
  - Automatic backout: Automatic backout of poison\* messages
  - Backout threshold: Maximum number of attempts to try reprocessing
  - Backout queue name: The name of the backout queue



\* A poison message is any message that the receiving application does not know how to process

© Copyright IBM Corporation 2015

Figure 13-8. WebSphere MQ queue manager overview: Main tab

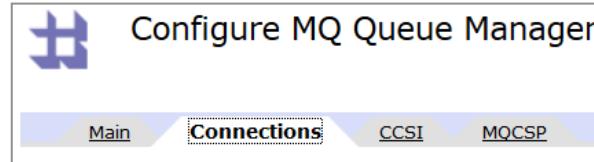
WE711 / ZE7111.0

### Notes:



## WebSphere MQ queue manager overview: Connections tab

- Open connections: The total number of connections allowed
- Retry behavior: The details of how IBM MQ should try again to process messages
- Conversation sharing: The maximum number of conversations to share a single TCP/IP connection
- Security: Secure communication with the remote queue manager in one of two ways, SSL proxy profile and the GSKit\* artifacts



With an SSL proxy profile: Specify the SSL proxy profile

- Must use this method for IBM MQ for z/OS

With artifacts from GSKit: Specify the SSL key repository and cipher specification

- SSL proxy profile
- SSL key repository
- SSL cipher specification

\*GSKit: IBM Global Security Kit

© Copyright IBM Corporation 2015

Figure 13-9. WebSphere MQ queue manager overview: Connections tab

WE711 / ZE7111.0

### Notes:



## WebSphere MQ queue manager overview: CCSI tab

- CCSI: Coded character set ID:  
The ID number to present to the IBM MQ Manager.
- Convert input: Click **off** to disable automatic conversion of incoming messages to the CCSI established for this MQ Queue Manager object.

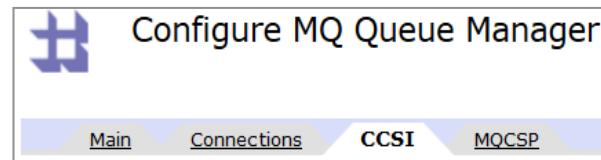


Figure 13-10. WebSphere MQ queue manager overview: CCSI tab

© Copyright IBM Corporation 2015

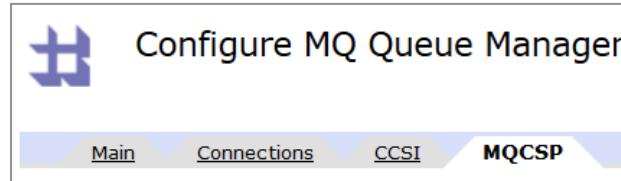
WE711 / ZE7111.0

### Notes:



## WebSphere MQ queue manager overview: MQCSP tab

- MQCSP: WebSphere MQ connection security parameter
  - MQCSP User
  - MQCSP Password



The MQCSP support enables the authorization service to authenticate a user ID and password

© Copyright IBM Corporation 2015

Figure 13-11. WebSphere MQ queue manager overview: MQCSP tab

WE711 / ZE7111.0

### Notes:

## Step 1: Create a WebSphere MQ queue manager (1 of 3)

1. Create a WebSphere MQ Queue Manager object from the Control Panel
2. From the **Main** tab, 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 alternative **Channel Name** if necessary
5. Enter a user name that identifies the client (the IBM MQ Queue Manager object) to the queue manager

The screenshot shows the 'MQ Queue Manager' configuration page in the Control Panel. The 'Name' field is set to 'EastAddressQM'. Under 'General Configuration', the 'Host Name' is '172.16.80.77(1414)' (step 2), 'Queue Manager Name' is 'QM\_1' (step 3), 'Channel Name' is 'EASTADDRESS.CHANNEL' (step 4), and 'User Name' is 'mqm' (step 5). Other settings like 'Administrative state' (enabled), 'Channel Heartbeat' (300 seconds), and 'Maximum Message Size' (1048576 bytes) are also visible.

Figure 13-12. Step 1: Create a WebSphere MQ queue manager (1 of 3)

WE711 / ZE7111.0

### Notes:

A WebSphere MQ queue manager object allows a back side handler on the DataPower SOA appliance to access an IBM MQ queue manager. The same object also allows IBM MQ queue managers to connect to a DataPower service through a front side handler.

The **Queue Manager Name** field is necessary only if a non-default 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 IBM MQ queue manager. You provide a user name with administrative permissions on the local operating system.

During the installation of IBM MQ, it creates a user in the local Windows registry that is called MUSR\_MQADMIN in the IBM 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 3)

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

**Units of Work and Backout**

|                    |   |   |
|--------------------|---|---|
| Units of Work      | 6   | 1 |
| Automatic Backout  | <input checked="" type="radio"/> on <input type="radio"/> off |   |
| Backout Threshold  |   |   |
| Backout Queue Name |   |   |

**Retry Behavior**

|                     |      |   |
|---------------------|------|---|
| Automatic Retry     | 7    | <input checked="" type="radio"/> on <input type="radio"/> off |
| Retry Interval      | 1    |   |
| Retry Attempts      | 0    |   |
| Long Retry Interval | 1800 |   |
| Reporting Interval  | 1    |   |

© Copyright IBM Corporation 2015

Figure 13-13. Step 1: Create a WebSphere MQ queue manager (2 of 3)

WE711 / ZE7111.0

### Notes:

#### Units of Work

When set to 0 (zero), the default, it causes the appliance to get and put messages with no provision for rollback. Either the operation succeeds or not. Undeliverable messages are silently discarded, which leaves higher-level protocols with the responsibility to detect and retransmit lost packets.

When set to 1, the DataPower appliance uses sync points. A sync point commits and rolls back each IBM MQ message, not the entire transaction. When specified, the DataPower appliance does not remove the message that it gets from a queue until it uses that message (such as placing the message on a server queue for processing) to complete its transaction. If the transaction fails and the message is left available on the queue, the DataPower appliance can attempt to get the message and process it again.

Only the values 0 (zero) and 1 are valid.

#### Automatic Retry

Define whether to attempt to reconnect to the remote server after a connection failure. When set to **on**, the DataPower appliance automatically attempts to reconnect to the remote host, which is on by default.

### **Retry Interval**

Specify the time interval (in seconds) between attempts to retry the failed connections to a remote host.

### **Retry Attempts**

Specify the number of attempts to retry the failed connections. After the number of attempts is reached, the **Long Retry Interval** will be used instead.

### **Long Retry Interval**

Specify the retry interval to use after the number of short retry attempts is reached. Enter an integer of 0 – 65535 seconds. The default value is 1800.



## Step 1: Create a WebSphere MQ queue manager (3 of 3)

8. Set the total number of open connections

| Open Connections       |                                    |
|------------------------|------------------------------------|
| Total Connection Limit | <input type="text" value="250"/> 8 |
| Initial Connections    | <input type="text" value="1"/>     |
| Local Address          | <input type="text"/>               |

9. Configure an automatic retry interval to automatically reconnect to the queue manager (SSL is detailed on the next slide)

| SSL Proxy Profile        |                                      |
|--------------------------|--------------------------------------|
| (none) ▾                 | + ... 9                              |
| SSL Key Repository       | cert:/// (none) ▾ Upload... Fetch... |
| SSL Cipher Specification | None                                 |

© Copyright IBM Corporation 2015

Figure 13-14. Step 1: Create a WebSphere MQ queue manager (3 of 3)

WE711 / ZE7111.0

### Notes:

**Total Connection Limit:** Specify the total number of open connections to allow. Use an integer in the range 1 – 5000.

**Initial Connections:** Specify the number of connections to open immediately when the MQ Queue Manager object starts. Use an integer in the range 0 – 5000.

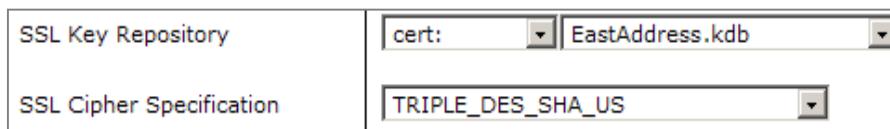
**SSL Proxy Profile:** Select an SSL proxy profile.

**SSL Key Repository:** Specify the location of the key database file in which keys and certificates are stored. Use this property with the SSL Cipher property to enable SSL communication when the SSL artifacts were created with IBM Global Security Kit (GSKit).

**SSL Cipher Specification:** Select the cipher suite for SSL communication when using an SSL key repository. The cipher suite must match the ciphers in use by the IBM MQ Queue Manager.

## 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:
  1. Configure the remote IBM MQ Queue Manager to use SSL
  2. In DataPower, generate a self-signed certificate
  3. DataPower generates the certificate key pair in the PEM format
  4. Use an external application to convert the PEM format to pkcs12
    - This step is required because IBM MQ does not understand the PEM format
  5. Import the converted certificate key into IBM MQ
  6. Obtain the IBM MQ key database file, import it into DataPower, and select it in the **SSL Key Repository** field



© Copyright IBM Corporation 2015

Figure 13-15. Step 1: Use SSL in mutual authentication mode

WE711 / ZE7111.0

### Notes:

To set up SSL between the DataPower WebSphere MQ client and an IBM 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 IBM MQ. DataPower generates certificates in the PEM format, which IBM 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 IBM MQ key database. Finally, you export the IBM 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 an MQ front side handler

1. Open the multi-protocol gateway from the previous scenario
2. Create an **MQ Front Side Handler** to accept requests from an IBM 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:**

- 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
- MQTE Front Side Handler**
- MQ Front Side Handler**
- SFTP Server Front Side Handler
- Stateless Raw XML Handler

**General**

Name: **2**

Administrative State

Comments

Queue Manager: **3** AddressQM

Get Queue: **4** ADDRESS\_REQ \*

Put Queue: ADDRESS\_RESP

The number of concurrent MQ connections: 1

Get Message Options: 4097

Polling Interval: 30

Retrieve Backout Settings:  on  off

Use Queue Manager in URL:  on  off

CCSI: **5** 0

Example for point-to-point mode

© Copyright IBM Corporation 2015

Figure 13-16. Step 2: Add an MQ front side handler

WE711 / ZE7111.0

### Notes:

The administrator on the IBM 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 an IBM 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 **User Identifier** header field, if a processing action is added as 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 the following configuration steps:

1. Backend URL: http://9.65.242.185:9080/EastAddress/\*
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)

Example for point-to-point mode

© Copyright IBM Corporation 2015

Figure 13-17. Step 3: Configure an IBM MQ back-end transport

WE711 / ZE7111.0

### 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 IBM MQ back-end transport to add a value to the user identifier header field. This setting adds `?PMO=2052` to the URL. Header injection or some processing action must set the actual header value.

The back-end URL for an IBM 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: 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 form titled "Publish and Subscribe". It contains three input fields: "Subscribe Topic String" (empty), "Subscription Name" (empty), and "Publish Topic String" (empty). The form is enclosed in a light gray border.

© Copyright IBM Corporation 2015

Figure 13-18. Publish/subscribe: MQ front side handler support

WE711 / ZE7111.0

### Notes:

Publish/subscribe applies to IBM MQ V7.



## Publish/subscribe: IBM 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 2015

Figure 13-19. Publish/subscribe: IBM MQ back-end transport support

WE711 / ZE7111.0

### Notes:

The helper constructs a DataPower IBM MQ URL like:

dpmq://QM/?SubscribeTopicString=yyyy;SubscriptionName=zzz

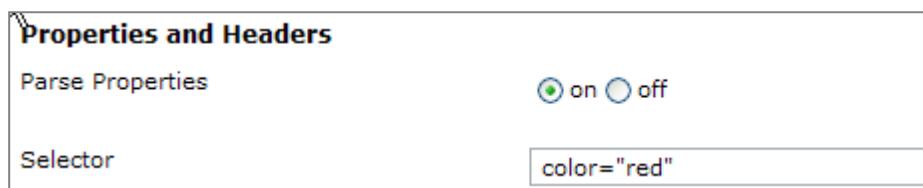
In the MQHelper dialog box:

- If both the RequestQueue and PublishTopicString are entered, the RequestQueue overrides.
- If both the ReplyQueue and SubscribeTopicString are entered, the ReplyQueue overrides.
- If the IBM 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 (IBM MQ V7)
- Allow DataPower awareness of properties by enabling parsing in 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 2015

Figure 13-20. Message properties

WE711 / ZE7111.0

### Notes:

Messages are retrieved from the queue only if the message property satisfies the selector statement.

A selector can also be specified in the IBM MQ URL.



## Ordered processing of IBM MQ messages

- Enforces serial processing of IBM 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

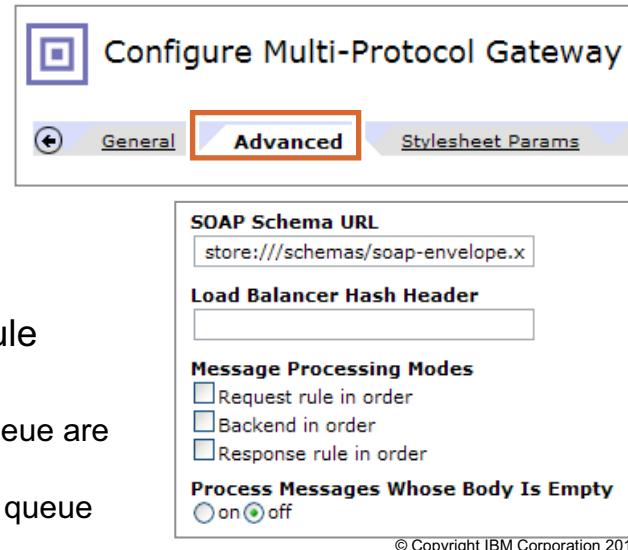


Figure 13-21. Ordered processing of IBM MQ messages

WE711 / ZE7111.0

### 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"/>          |

- MQ front side handler
  - Setting **Retrieve Backout Settings** to **on** causes the appliance to retrieve the backout settings from the actual IBM MQ server

|  |   |
|--|---|
| <input type="button" value="Retrieve Backout Settings"/> | <input checked="" type="radio"/> on <input type="radio"/> off |
|--|---|

© Copyright IBM Corporation 2015

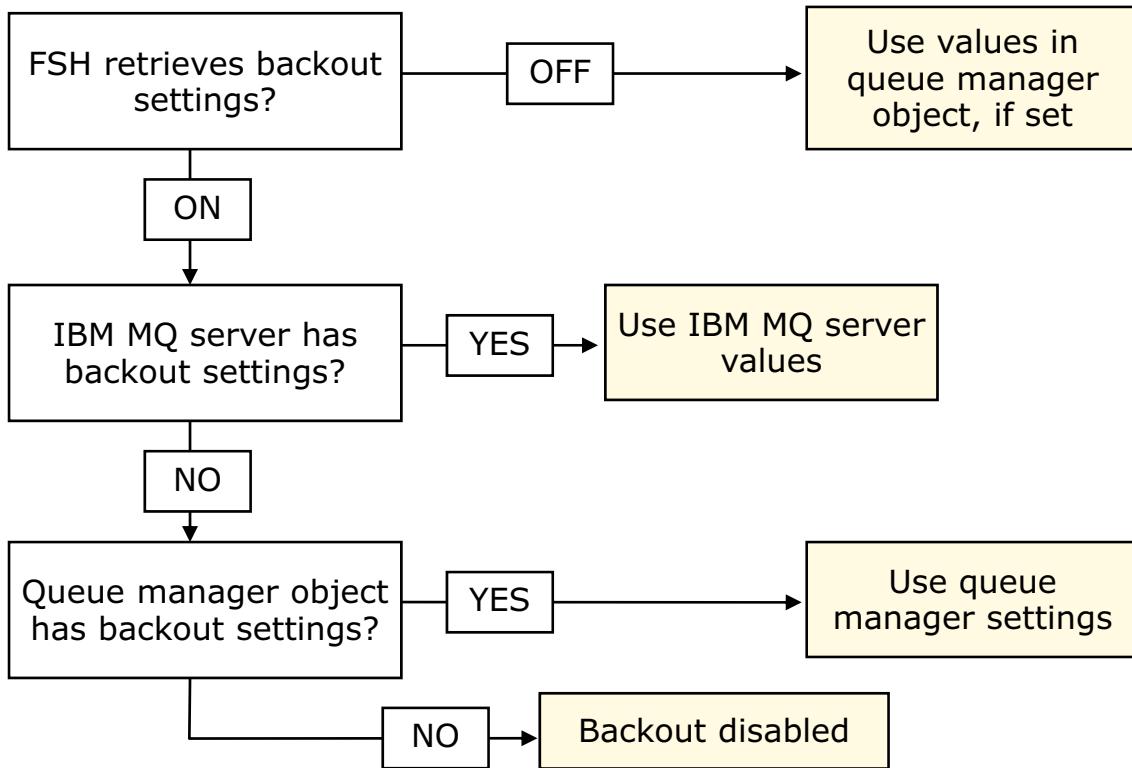
Figure 13-22. Controlling backout of WebSphere MQ messages

WE711 / ZE7111.0

### 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 2015

Figure 13-23. Decision tree for the backout settings

WE711 / ZE7111.0

### Notes:

## MQ Header action in service policy

- In policy editor:  
**Advanced > MQ Header**
- Enables manipulation of IBM 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**

|                                     |   |
|-------------------------------------|---|
| <b>Asynchronous</b>                 | <input checked="" type="radio"/> on <input type="radio"/> off   |
| <b>MQ Processing Type</b>           | <input checked="" type="radio"/> request<br><input type="radio"/> response<br><input type="checkbox"/> Save |
| <b>MQ Request Header Processing</b> | MQMD for PUT <input checked="" type="checkbox"/> Save   |
| <b>Override Existing MQMD</b>       | <input checked="" type="radio"/> on <input type="radio"/> off <input type="checkbox"/> Save                 |
| <b>Message Id</b>                   | [Text Box]  |
| <b>Correlation Id</b>               | [Text Box]  |
| <b>Character Set Id</b>             | [Text Box]  |
| <b>Format Name</b>                  | [Text Box]  |
| <b>ReplyToQ</b>                     | [Text Box]  |
| <b>ReplyToQMgr</b>                  | [Text Box]  |

© Copyright IBM Corporation 2015

Figure 13-24. MQ Header action in service policy

WE711 / ZE7111.0

### 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 an MQ Header action

- Retrieve a response message by IBM MQ message ID or IBM 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 2015

Figure 13-25. Typical uses of an MQ Header action

WE711 / ZE7111.0

### Notes:



## Transactions and IBM MQ

DataPower allows control of transactions at both ends:

- Front side
  - **Units of Work** parameter in WebSphere MQ Queue Manager object

- Back side
  - **Backend URL:**  
Sync query parameter,  
transactional query parameter

© Copyright IBM Corporation 2015

Figure 13-26. Transactions and IBM MQ

WE711 / ZE7111.0

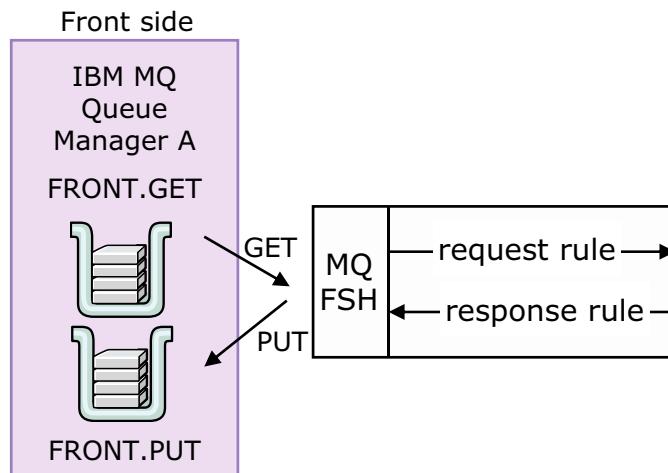
### Notes:

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 IBM MQ queue managers.

## IBM MQ front side transactions



- The 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

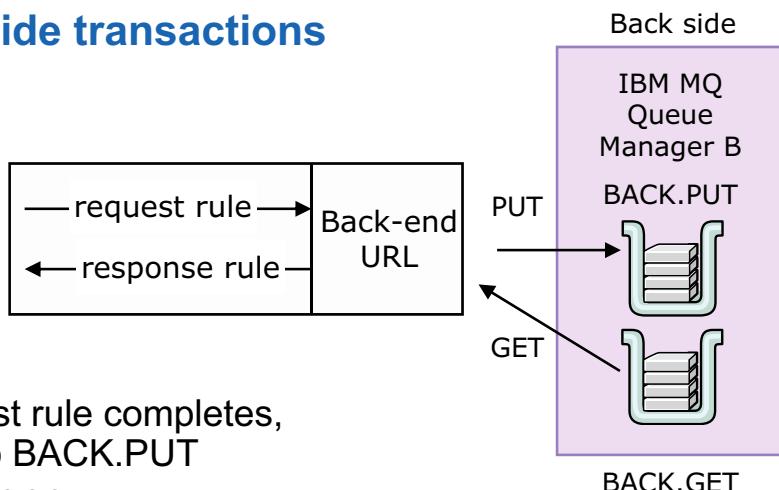
© Copyright IBM Corporation 2015

Figure 13-27. IBM MQ front side transactions

WE711 / ZE7111.0

### Notes:

## IBM 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 IBM 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 2015

Figure 13-28. IBM MQ back side transactions

WE711 / ZE7111.0

### Notes:

**Sync=true** is necessary if Queue Manager B is using transactions.

**Sync = true|false** cannot be set by the IBM MQ URL Builder. You must edit the back-end URL field to add the `Sync` parameter.



## IBM MQ DataPower URL

- The IBM 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
  - Sync (true or false): Is used to send MQCOMMIT to back side queue manager after MQPUT
  - Transactional (true or false): Is used to enforce transactional units of communication with back side queue managers
  - PMO: Set options on the MQPUT call
- The url-open extension element supports the IBM MQ DataPower URL
  - A style sheet can GET and PUT to queues

© Copyright IBM Corporation 2015

Figure 13-29. IBM MQ DataPower URL

WE711 / ZE7111.0

### Notes:

Existing MQMD headers can be accessed from the variable:

`var://context/contextname/_extension/header/MQMD`



## MQ Queue Manager Group object

- Provides failover of WebSphere MQ Queue Manager objects
  - Consists of a single primary IBM MQ Queue Manager and many backup IBM MQ Queue Managers
- If the primary queue manager becomes unavailable, the DataPower appliance can attempt to place the message on one of the backup IBM MQ Queue Managers

MQ Queue Manager Group

|                          |  |           |   |           |  |
|--------------------------|--|-----------|---|-----------|--|
| Name                     | EastAddressQMGroup *   |           |   |           |  |
| Administrative State     | <input checked="" type="radio"/> enabled <input type="radio"/> disabled  |           |   |           |  |
| Comments                 | <input type="text"/>   |           |   |           |  |
| Primary MQ Queue Manager | EastAddressQM <input type="button" value="+"/> <input type="button" value="..."/> *  |           |   |           |  |
| Backup MQ Queue Managers | <table border="1"> <tr> <td>AddressQM</td> <td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td> </tr> <tr> <td>AddressQM</td> <td><input type="button" value="Add"/> <input type="button" value="+"/> <input type="button" value="..."/></td> </tr> </table> | AddressQM | <input type="button" value="Edit"/> <input type="button" value="Delete"/> | AddressQM | <input type="button" value="Add"/> <input type="button" value="+"/> <input type="button" value="..."/> |
| AddressQM                | <input type="button" value="Edit"/> <input type="button" value="Delete"/>  |           |   |           |  |
| AddressQM                | <input type="button" value="Add"/> <input type="button" value="+"/> <input type="button" value="..."/>   |           |   |           |  |

© Copyright IBM Corporation 2015

Figure 13-30. MQ Queue Manager Group object

WE711 / ZE7111.0

### 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 an IBM MQ front-side handler
- Configure an IBM MQ back-end URL
- Manage transactionality between IBM MQ queue managers

© Copyright IBM Corporation 2015

Figure 13-31. Unit summary

WE711 / ZE7111.0

### Notes:

## Checkpoint questions

1. True or False: IBM MQ support is available only 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<br>C. DataPower supports |

© Copyright IBM Corporation 2015

Figure 13-32. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

- 1.
- 2.
3. (1)  
(2)



## 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 2015

Figure 13-33. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 11



Configuring a multi-protocol gateway service with IBM MQ

© Copyright IBM Corporation 2015

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

10.1

Figure 13-34. Exercise 11

WE711 / ZE7111.0

### Notes:

## Exercise objectives

After completing this exercise, you should be able to:

- Create an IBM MQ front-side handler (FSH) that gets messages from a queue and puts responses on a queue
- Describe and configure an IBM MQ queue manager object
- Create an IBM MQ multi-protocol gateway proxy that receives HTTP messages on the front end while communicating by using an IBM MQ queue out the back end
- Use various techniques to route the IBM MQ message to the correct queue: static, dynamic using a style sheet (XSL), and dynamic using a GatewayScript
- Send messages from a multi-protocol gateway service to a queue in IBM MQ in a request/response messaging pattern

© Copyright IBM Corporation 2015

Figure 13-35. Exercise objectives

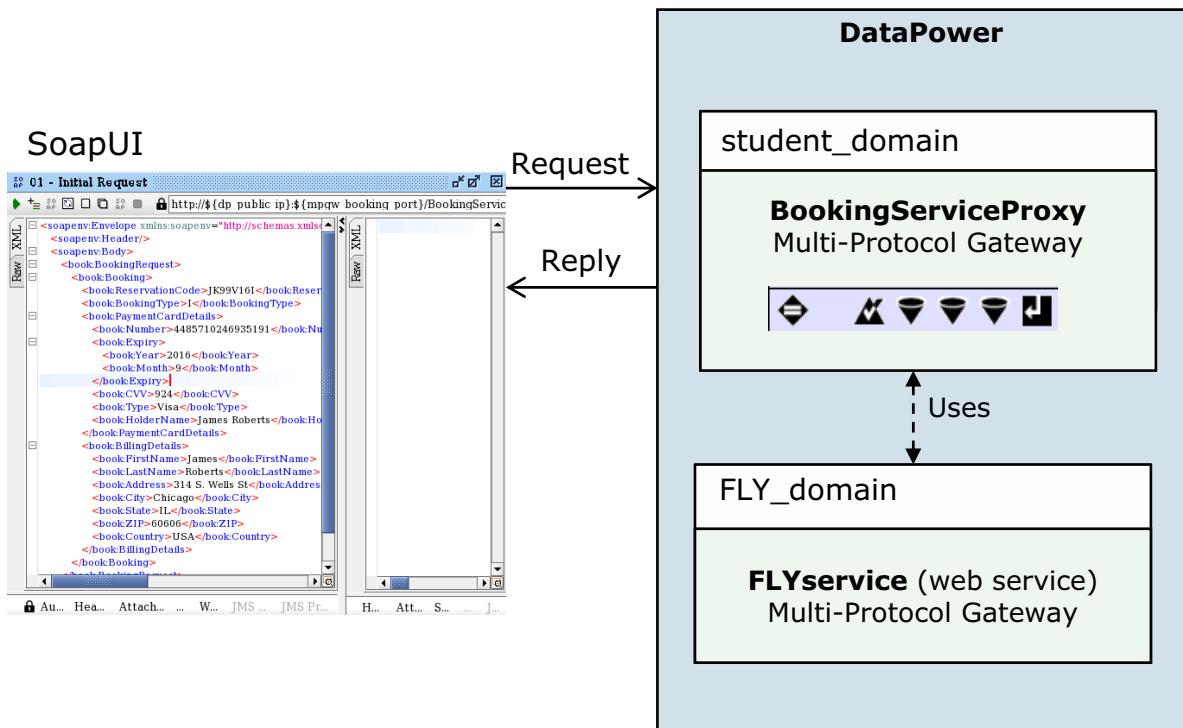
WE711 / ZE7111.0

### Notes:

This exercise shows you how to add support for IBM MQ to a multi-protocol gateway service. You add an MQ front side handler to the BookingServiceProxy service that you created in an earlier exercise. You create another MPGW service to demonstrate calling an IBM MQ queue from the back side of a service. This MPGW service is used as an IBM MQ client, similar to the IBM MQ client RFHUtil, to get and put messages from queues.



## Exercise overview



© Copyright IBM Corporation 2015

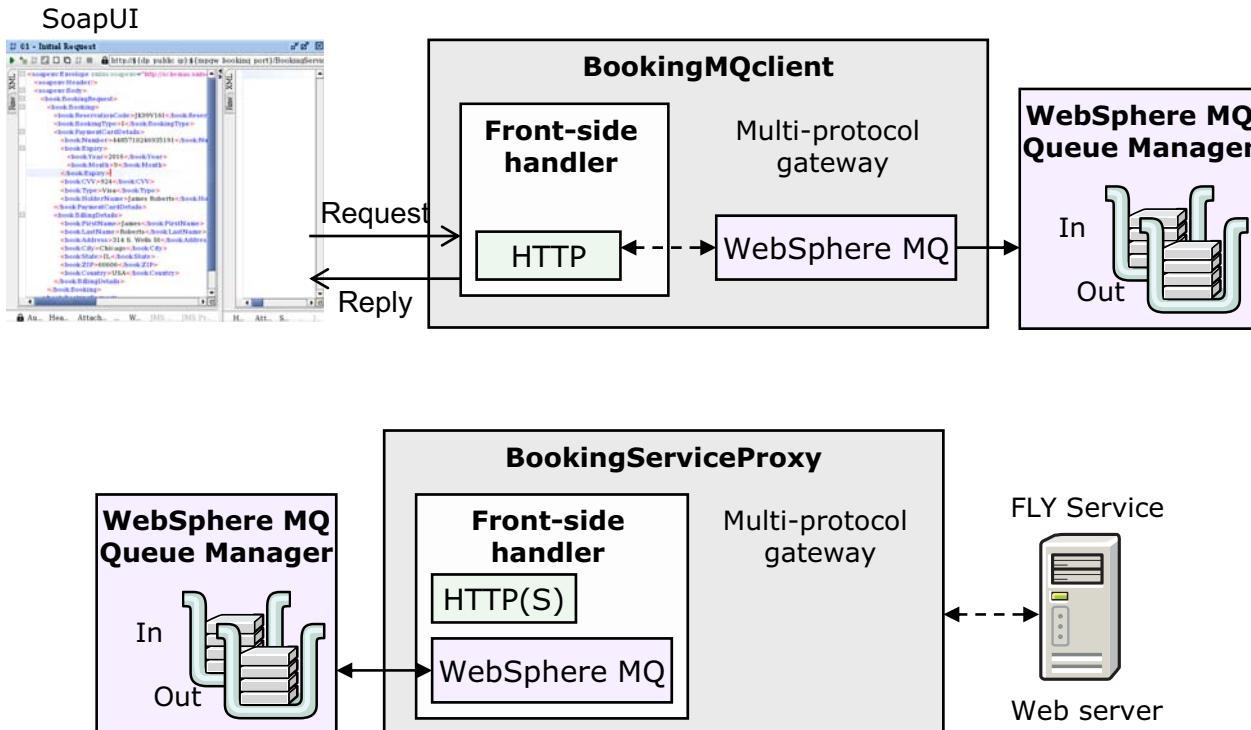
Figure 13-36. Exercise overview

WE711 / ZE7111.0

### Notes:



## Exercise overview 1 of 2



© Copyright IBM Corporation 2015

Figure 13-37. Exercise overview 1 of 2

WE711 / ZE7111.0

### Notes:

An MPGW BookingMQclient is created. Then, an MQ FSH is added to the existing BookingServiceProxy MPGW.

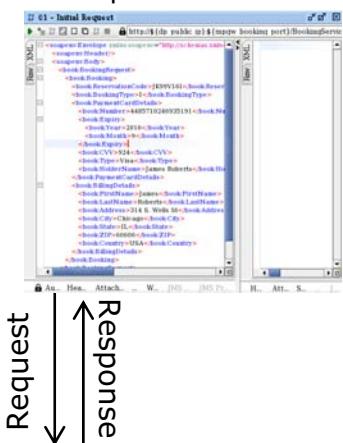
The BookingMQclient receives an HTTP SOAP request message, and places the message on an IBM MQ put queue. The BookingServiceProxy polls the IBM MQ Get queue, reading the message and sending the message to the FLY Service backend web service. The FLY Service web service processes the message and sends the message back to the BookingServiceProxy MPGW that places the message on the get queue. The BookingMQclient reads the Get queue and send the response message over HTTP back to the requesting SoapUI client.

The MQ portion of processing is known as a request/reply scenario.



## Exercise overview 2 of 2

SoapUI

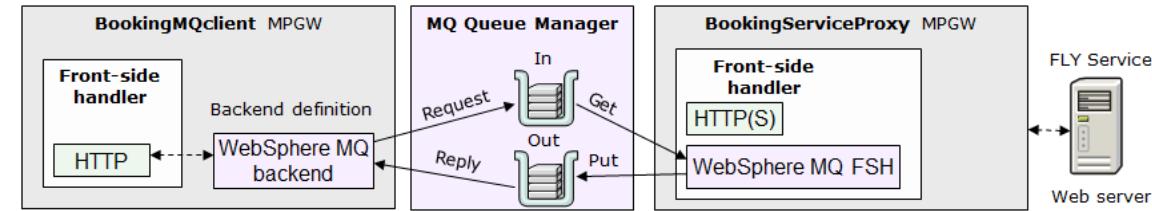


Request

Response

### MQ configuration:

- SoapUI – MQ Request sending an SOAP message over HTTP
- BookingMQclient MPGW with MQ back-end destination
- BookingServiceProxy MPGW with MQ Polling FSH
- MQ Request/Reply paradigm



© Copyright IBM Corporation 2015

Figure 13-38. Exercise overview 2 of 2

WE711 / ZE7111.0

### Notes:



# Unit 14. Web service proxy service

## What this unit is about

This unit describes the web service proxy service and its role in a web-services-based network. It explains the configuration steps that are required to create and manage a web services proxy. It also explains 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
- Hands-on exercise

## References

IBM DataPower Gateway Appliances Version 7.1 Knowledge Center:

[www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)



## 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 2015

---

Figure 14-1. Unit objectives

WE711 / ZE7111.0

### Notes:



## Web service proxy overview

- A web service proxy is a middleware component that exists between the client and the web service
  - Decouples the web service client from the actual web service
  - Hides the web service endpoint address from the client
  - Flexibility to change the 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
- You can use DataPower appliances to create a web service proxy to accelerate and mediate communication between a client and a web service
  - Rules 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 2015

Figure 14-2. Web service proxy overview

WE711 / ZE7111.0

### 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.

## Conceptual architecture of a web service proxy

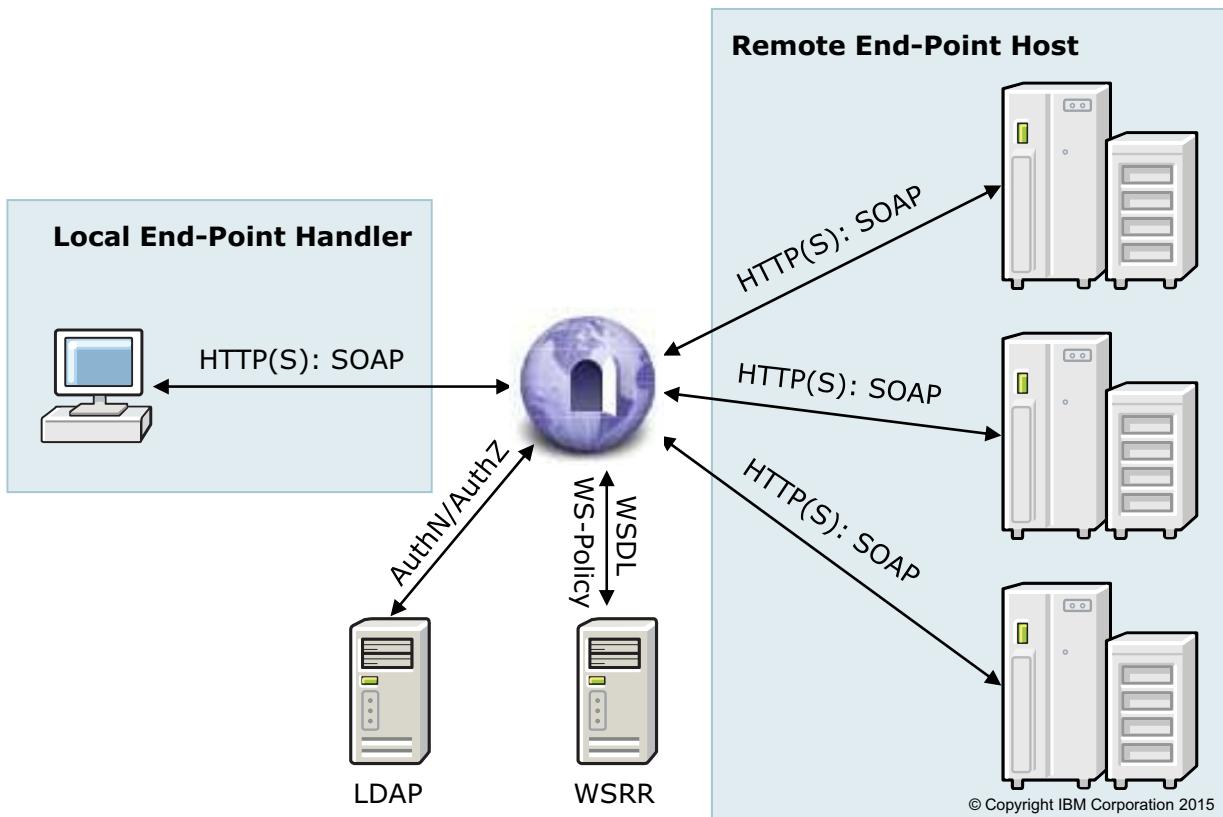


Figure 14-3. Conceptual architecture of a web service proxy

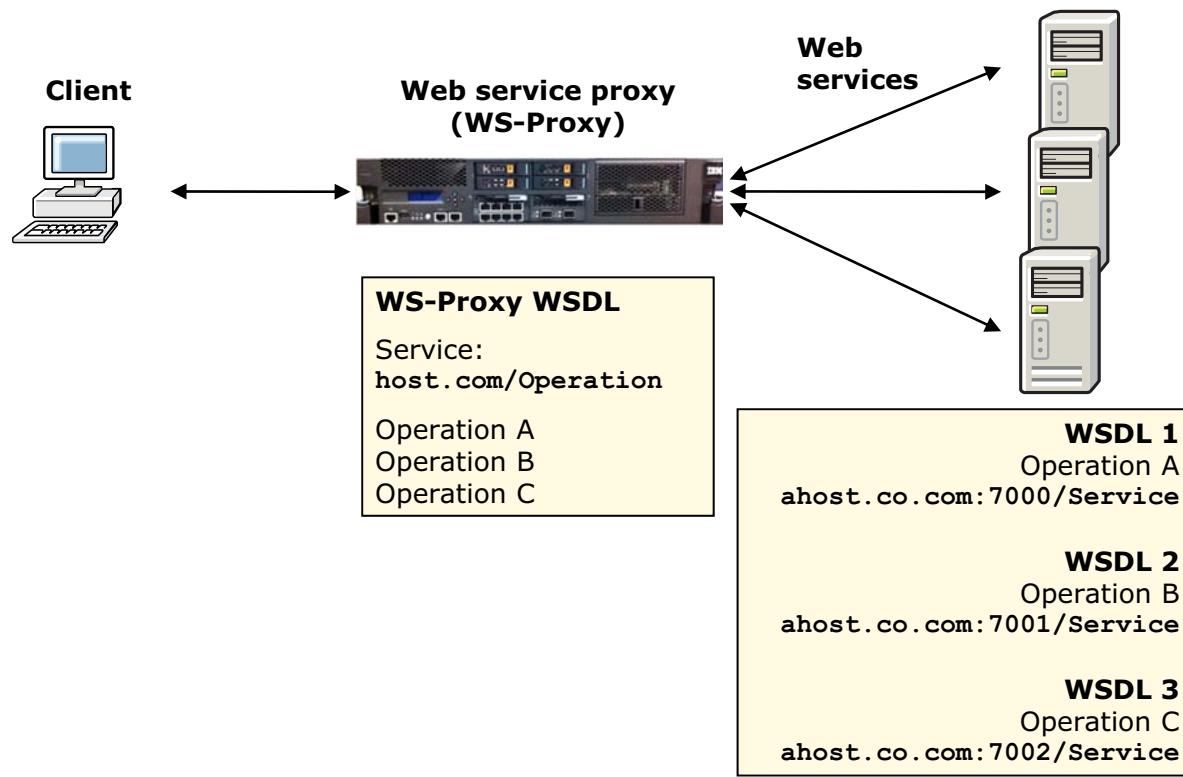
WE711 / ZE7111.0

### Notes:

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

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

## Web service virtualization



© Copyright IBM Corporation 2015

Figure 14-4. Web service virtualization

WE711 / ZE7111.0

### 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 by loading a WSDL document
  - Clients now connect directly to the web service proxy and not the back-end service
- Creates processing policy with rules and actions at a 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 the 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 2015

Figure 14-5. Web service proxy benefits

WE711 / ZE7111.0

### Notes:

You can schema validate request, response, and fault messages by using a **user policy**. 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 functions 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.

**Web service proxy configuration tabs (1 of 2)**

Configure Web Service Proxy

- WSDL files
- SLM Policy
- Services
- Policy
- SLA Policy Details
- Proxy Settings
- Advanced Proxy Settings
- Headers/Params

- **WSDL files**
  - Uploads or associates a WSDL document with a web service proxy
  - Configures the 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**
  - Lists services that are defined in each WSDL document
  - Can publish services to a UDDI registry
- **Policy**
  - Configures a web service proxy policy
- **SLA Policy Details**
  - View WSDL attachments that relate to a service level agreement
- **Proxy Settings**
  - Specifies a method of forwarding to a service, security, XML manager, and HTTP settings

© Copyright IBM Corporation 2015

Figure 14-6. Web service proxy configuration tabs (1 of 2)

WE711 / ZE7111.0

### 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 and 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 screenshot shows the 'Web service proxy configuration tabs (2 of 2)' section. At the top left is the 'WebSphere Education' logo. At the top right is the 'IBM' logo. Below the title, there is a navigation bar with several tabs: 'Proxy Settings' (selected), 'Advanced Proxy Settings', 'Headers/Params', 'WS-Addressing', 'WS-ReliableMessaging', 'Monitors', 'XML Threat Protection', and a refresh icon. The main content area contains a bulleted list of configuration tabs:

- **Advanced Proxy Settings**
  - Configures advanced connection settings
- **Headers/Params**
  - Add or remove HTTP headers and passes stylesheet 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 with this service
- **XML Threat Protection**
  - Provides protection against XML threats

© Copyright IBM Corporation 2015

Figure 14-7. Web service proxy configuration tabs (2 of 2)

WE711 / ZE7111.0

### 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 the DataPower WebGUI Control Panel  
or
  - Using the 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 the endpoint of services that are defined in a WSDL document
  - Define both the proxy URI and the endpoint URI for each service in the WSDL document
5. Specify a service policy that consists of rules for the web service (optional)

© Copyright IBM Corporation 2015

Figure 14-8. Web service proxy basic configuration steps

WE711 / ZE7111.0

### 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 2015

Figure 14-9. Step 1: Obtain WSDL document

WE711 / ZE7111.0

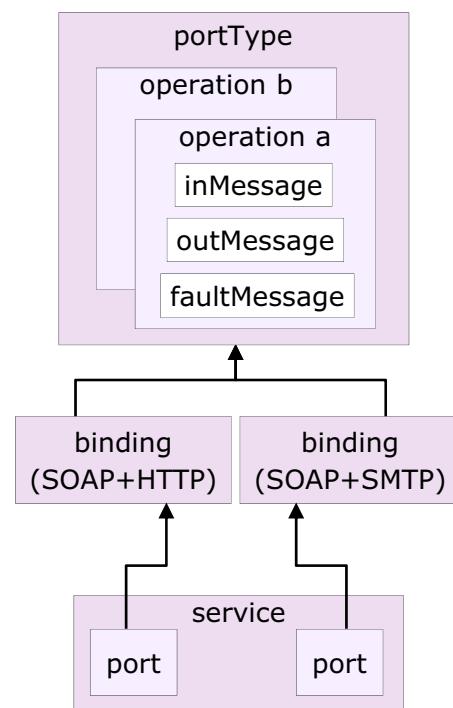
### Notes:

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 2015

Figure 14-10. WSDL structure

WE711 / ZE7111.0

### 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**

| Web Service Proxy Name | Op-State | Logs | Type             | Req-Type | Back Side URL | Resp-Type |
|------------------------|----------|------|------------------|----------|---------------|-----------|
| BookingServiceWSProxy  | up       |      | Static from WSDL | SOAP     | NA            | SOAP      |

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

BookingServiceWSProxy

Create Web Service Proxy

© Copyright IBM Corporation 2015

Figure 14-11. Step 2: Creating a web service proxy

WE711 / ZE7111.0

### 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

Blueprint Console

Search

Main    Proxy Settings    HTTP Options    Parser Limits    Monitors    WS-Addressing

Web Service Proxy: BookingServiceWSProxy [up]

Apply    Cancel    Delete    Undo    Export | View Log | View Status

Administrative state:  enabled  disabled

Comments:

Service Priority: Normal

XML Manager: default

© Copyright IBM Corporation 2015

Figure 14-12. An alternative: Web service proxy object editor

WE711 / ZE7111.0

### Notes:

The WSDL cache policy and some attachment processing specifications are example configurations that are possible only by using this editor.



## Step 3: Add WSDL document to web service proxy

**Configure Web Service Proxy**

**Web Service Proxy Name** [up] \*  
CustomerServiceProxy

Apply Cancel Delete Refresh

**WSDLS**

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

**WSDL File URL**  
local: /  
(none) Upload... Fetch... Edit... View... E

**Use WS-Policy References**  
on off

**WS-Policy Parameter Set**  
(none) + ...

**WS-Policy Enforcement Mode**  
Enforce

**SLA Enforcement Mode**  
Allow

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 2015

Figure 14-13. Step 3: Add WSDL document to web service proxy

WE711 / ZE7111.0

### 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)

### WSDLs

| BookingService - BookingServiceSOAP |   |   |                   |
|-------------------------------------|---|---|-------------------|
| <b>Local</b>                        |   |   |                   |
| <b>Local Endpoint Handler</b>       | <b>URI</b>                                    | <b>Binding (Suffix)</b>   | Edit/Remove       |
| BookingServiceWSProxyFSH            | /BookingService/                              | SOAP 1.1()  | Edit Remove       |
| (none)                              | /BookingService/                              | <input checked="" type="checkbox"/> SOAP 1.1 <input type="checkbox"/> SOAP 1.2<br><input type="checkbox"/> HTTP GET | Add               |
| <b>Remote</b>                       |   |   |                   |
| <b>Protocol</b>                     | <b>Remote Endpoint Host</b>                   | <b>Port</b>   | <b>Remote URI</b> |
| HTTP                                | 172.16.78.23                                  | 9080  | /BookingService/  |
| <b>Published</b>                    | <input checked="" type="checkbox"/> Use Local |   |                   |

© Copyright IBM Corporation 2015

Figure 14-14. Step 4: Configure WSDL endpoint

WE711 / ZE7111.0

### 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 Knowledge 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

The screenshot shows a configuration interface for a local endpoint handler. In the 'Local' section, there is a table with one row. The 'Protocol' column is set to 'HTTP' and the 'Remote Endpoint' column is set to '172.16.78.23'. Below the table, there is a checkbox labeled 'Published' and a checked checkbox labeled 'Use Local'. To the right of the table, there is a 'Create a New:' dropdown menu with a list of various handlers. The 'HTTP Front Side Handler' option is highlighted with a red box.

The screenshot shows a configuration form for a local endpoint handler. It includes fields for 'Administrative state' (radio buttons for 'enabled' and 'disabled'), 'Comments' (a text input field), 'Local IP address' (set to 'dp\_public\_ip'), 'Port' (set to '12315'), and 'HTTP version to client' (set to 'HTTP 1.1'). The 'Local IP address' and 'Port' fields are highlighted with a red box.

© Copyright IBM Corporation 2015

Figure 14-15. Step 5: Configure local endpoint handler

WE711 / ZE7111.0

### 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.



## Step 6: Add the WSDL to the service

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

↓

| BookingService - BookingServiceSOAP |                  |   |             |
|-------------------------------------|------------------|---|-------------|
| Local                               |                  |   |             |
| Local Endpoint Handler              | URI              | Binding (Suffix)  | Edit/Remove |
| BookingServiceWSProxyFSH            | /BookingService/ | SOAP 1.1()  | Edit Remove |
| (none)                              |                  | <input checked="" type="checkbox"/> SOAP 1.1 <input type="checkbox"/> SOAP 1.2<br><input type="checkbox"/> HTTP GET | Add         |

**Remote**

| Protocol | Remote Endpoint Host | Port | Remote URI       |
|----------|----------------------|------|------------------|
| HTTP     | booking.FLY.com      | 9070 | /BookingService/ |

Click **Add** to add the customized WSDL information to the service; then, click **Next**

© Copyright IBM Corporation 2015

Figure 14-16. Step 6: Add the WSDL to the service

WE711 / ZE7111.0

### 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
- Other options, such as adding another WSDL, are now available

The screenshot shows the 'Configure Web Service Proxy' interface. At the top, there's a navigation bar with tabs: WSDL files, SLM Policy, Services, Policy, SLA Policy Details, Proxy Settings, Advanced Proxy Settings, and Headers/Params. The 'Policy' tab is selected. Below the navigation bar, there's a section for 'Web Service Proxy Name [up]' with a text input field containing 'BookingServiceWSPProxy'. To the right of the input field is a mandatory indicator (\*). Below this are buttons for Apply, Cancel, and Delete. To the right of the buttons are links for Export, View Log, View Status, View Operations, Show Probe, and Conformance. A large arrow points from the 'Edit WSDL or Subscription' button in the 'WSDLs' section down towards the table below. The 'WSDLs' section also contains buttons for Add WSDL, Add UDDI Subscription, Add WSRR Subscription, and Add WSRR Saved Search Subscription. Below this is a table with four columns: WSDL Source Location, Endpoint Handler Summary, WSDL Status, and WS-I BP Status. The first row in the table has a plus sign icon and the URL 'local:///BookingService.wsdl'. The 'Endpoint Handler Summary' column shows '1 up / 1 configured'. The 'WSDL Status' column shows 'Okay' and the 'WS-I BP Status' column shows 'Okay'.

| WSDL Source Location         | Endpoint Handler Summary | WSDL Status | WS-I BP Status |
|------------------------------|--------------------------|-------------|----------------|
| local:///BookingService.wsdl | 1 up / 1 configured      | Okay        | Okay           |

© Copyright IBM Corporation 2015

Figure 14-17. Initial WSDL completed

WE711 / ZE7111.0

### 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 Service Registry and Repository
- Subscribe to a WSRR Saved Search
  - Can automatically “push” changes to the web service proxy

Configure Web Service Proxy

WSDL files SLM Policy Services Policy SLA Policy Details Proxy Settings Advanced Proxy Settings Headers/Params

Web Service Proxy Name [up] \*  
BookingServiceWSProxy

Apply Cancel Delete Export | View Log | View Status | View Operations | Show Probe | Conformance

**WSDLs**

| WSDL Source Location         | Endpoint Handler Summary | WSDL Status | WS-I BP Status |
|------------------------------|--------------------------|-------------|----------------|
| local:///BookingService.wsdl | 1 up / 1 configured      | Okay        | Okay           |

© Copyright IBM Corporation 2015

Figure 14-18. Other ways to get a WSDL

WE711 / ZE7111.0

### 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

The screenshot shows a web-based configuration interface for a Web Service Proxy. At the top, there is a navigation bar with tabs: WSDL files, SLM Policy, Services (which is highlighted with a red box), Policy, SLA Policy Details, Proxy Settings, and Advanced Proxy Settings. Below the navigation bar, there is a section for 'Web Service Proxy Name [up]' with a text input field containing 'BookingServiceWSPProxy'. Underneath this are buttons for Apply, Cancel, and Delete. To the right of these buttons are links for Export, View Log, View Status, and View Operations. The main content area is titled 'Services' and contains a table. The table has a single row with a purple header labeled 'WSDL Name: BookingService.wsdl.dpdup.0'. The row contains three columns: 'Service' (containing 'BookingService'), 'Operations' (empty), and 'Publish to UDDI' (a button). At the bottom right of the interface, there is a copyright notice: '© Copyright IBM Corporation 2015'.

Figure 14-19. View WSDL services

WE711 / ZE7111.0

### 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://booking.FLYservice.com:6999/BookingService?wsdl`
    - 6999: The port number of the web service proxy
    - /BookingService: 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="BookingService">
<address location="http://training.ibm.com:9080/
  BookingService/services/BookingService" /> </wsdl:port>
```
  - Retrieved by ?wsdl
 

```
<wsdl:port binding="..." name="BookingService">
<address
  location="http://192.168.10.41:6999/BookingService" />
</wsdl:port>
```

|                              |   |
|------------------------------|---|
| Allowed Methods and Versions | <input checked="" type="checkbox"/> HTTP/1.0<br><input checked="" type="checkbox"/> HTTP/1.1<br><input checked="" type="checkbox"/> POST<br><input checked="" type="checkbox"/> GET<br><input checked="" type="checkbox"/> PUT<br><input type="checkbox"/> HEAD<br><input type="checkbox"/> OPTIONS |
|------------------------------|---|

© Copyright IBM Corporation 2015

Figure 14-20. Retrieve the client WSDL from the service

WE711 / ZE7111.0

### Notes:

The original WSDL used in defining the WS Proxy contains a location that is no longer correct for the web service that this service proxies.

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="BookingService">
  <address
    location="http://192.168.10.41:6999/BookingService" />
</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="BookingService">
  <address
    location="http://booking.FLY.com:6999/BookingService" />
</wsdl:port>
```

© Copyright IBM Corporation 2015

Figure 14-21. Modifying the location in the client WSDL

WE711 / ZE7111.0

### 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.

- 
- 
-



© Copyright IBM Corporation 2015

Figure 14-22. Step 7: Configuring web service proxy policy

WE711 / ZE7111.0

## 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.

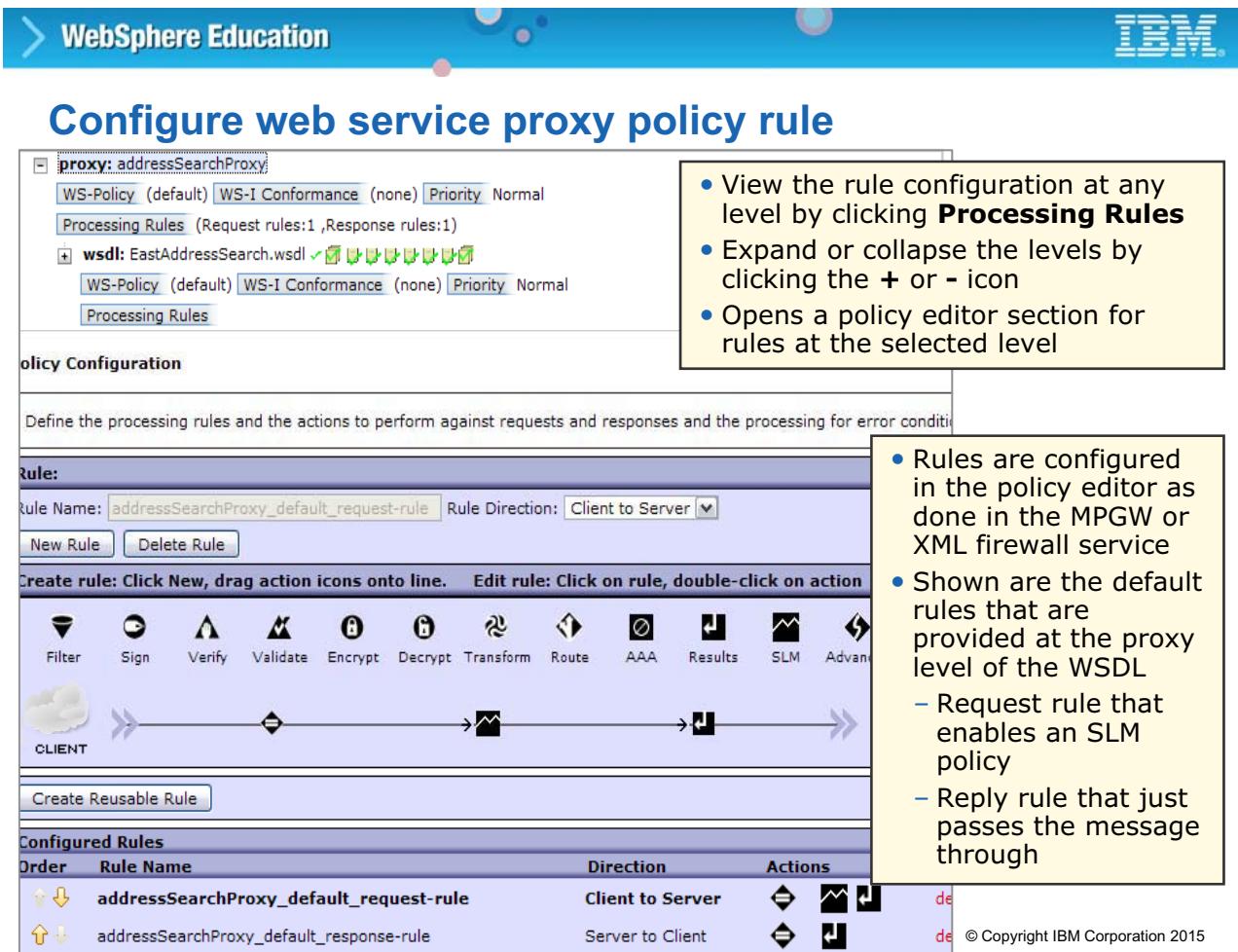


Figure 14-23. Configure web service proxy policy rule

---

WE711 / ZE711.0

## **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 the **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

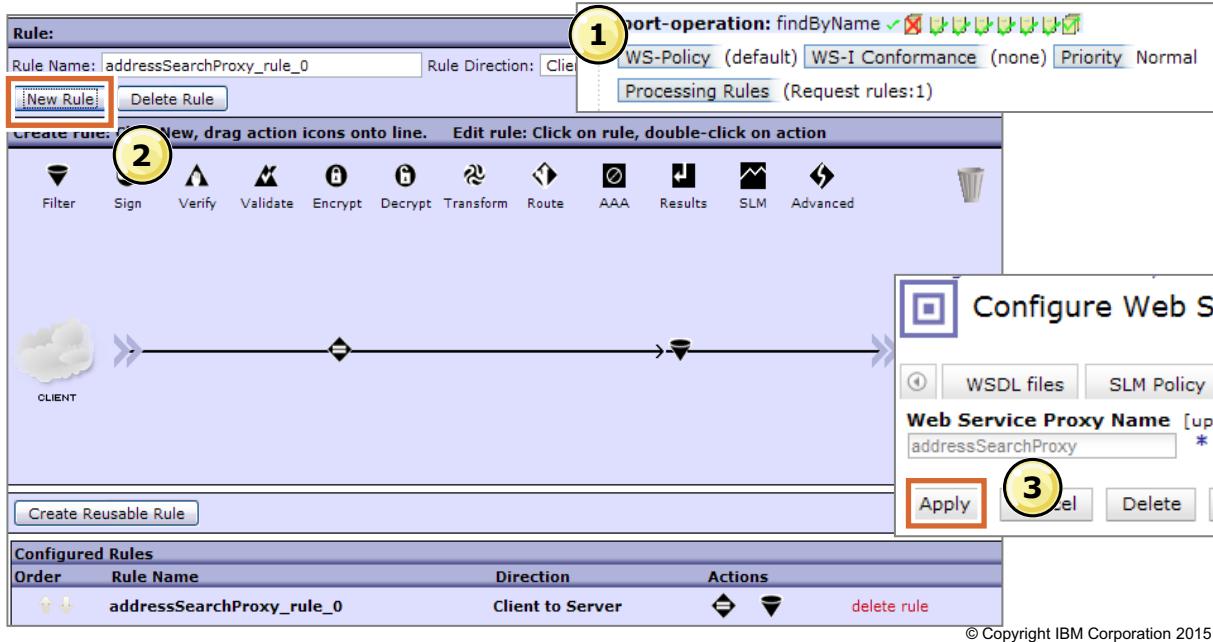


Figure 14-24. Adding a rule

WE711 / ZE7111.0

### 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.

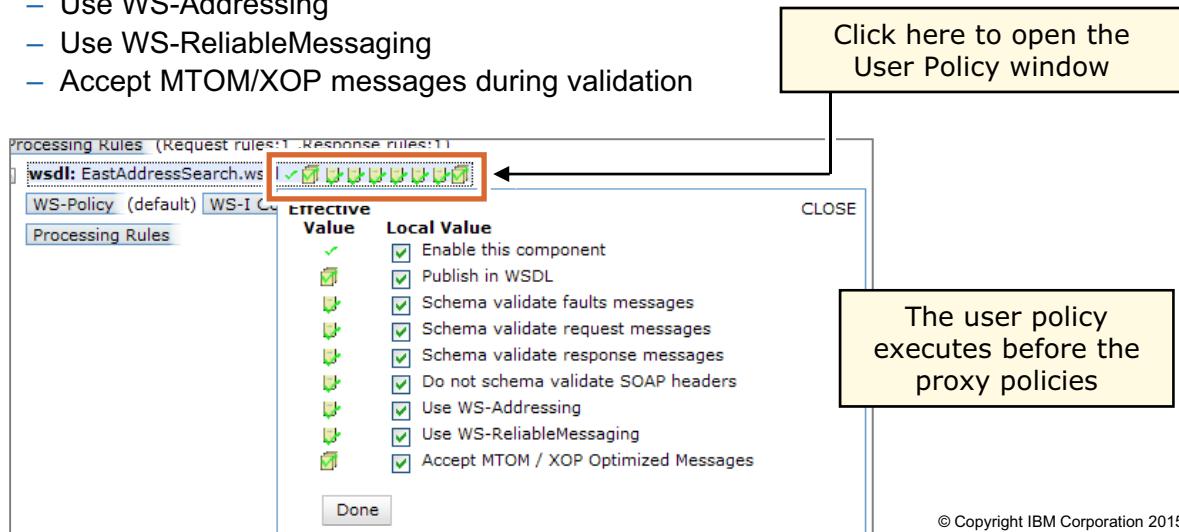
All rules that are configured at this level are in the **Configured Rules** section of the policy editor.

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 validate messages (against schema in WSDL): request, response, and fault
  - Schema validate 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



© Copyright IBM Corporation 2015

Figure 14-25. Default validation (user policies)

WE711 / ZE7111.0

### Notes:

Click any of the icons at each level to view the user policy dialog box.

The first check mark enables the component. Each option that is shown in the dialog box maps to an icon with a green check mark or red X.

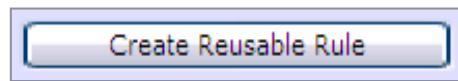
Each policy level contains a user policy that can be enabled or disabled.

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

- Use the rule configuration area to select a set of actions to be invoked as a reusable rule
  1. Click **Create Reusable Rule**.
  2. Draw a box around the actions to include in the reusable rule.
  3. 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.
  4. Use the **Advanced – Call Processing Rule** action to reuse the rule.



© Copyright IBM Corporation 2015

Figure 14-26. Create reusable rule

WE711 / ZE7111.0

### Notes:

Reusable rules are useful for applying a common set of actions at many levels of the web service proxy. More actions can be added before or after the reusable rules. With reusable rules, you can 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 are 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 stylesheet 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 2015

Figure 14-27. Advanced web service proxy configuration

WE711 / ZE7111.0

### 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, Parameters**, **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 2015

Figure 14-28. WS-Policy

---

WE711 / ZE711.0

## **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



Figure 14-29. Conformance policy

WE711 / ZE7111.0

### 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 stylesheet; 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 side-by-side configuration panels for 'Operation Conformance Policy'.

- Left Panel (Step 1): Profiles**
  - Conformance Policy Name: AddressSearchProxy
  - Basic tab selected.
  - Profiles section: WS-I BP 1.0 (unchecked), WS-I BP 1.1 (checked), WS-I AP 1.0 (checked), WS-I BSP 1.0 (checked).
  - Reject non-conforming messages section: Never (radio button selected), Failure, Warning, Always.
  - Buttons: Done, Cancel.
- Right Panel (Step 2): Ignored Requirements**
  - Conformance Policy Name: AddressSearchProxy
  - Advanced tab selected.
  - Ignored Requirements section: (empty) text input field, Add button.
  - Buttons: Done, Cancel.
- Central Area (Step 3): Record Report**
  - Never (radio button selected).
  - Buttons: Done, Cancel.
- Central Area (Step 4): Corrective Stylesheets**
  - (empty) text input field, Add button.
  - Buttons: Done, Cancel.
- Bottom Area (Step 5): Other Options**
  - Use analysis as result: On (radio button selected).
  - Off (radio button selected).
  - Buttons: Done, Cancel.

© Copyright IBM Corporation 2015

Figure 14-30. Conformance policy object

WE711 / ZE7111.0

### 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

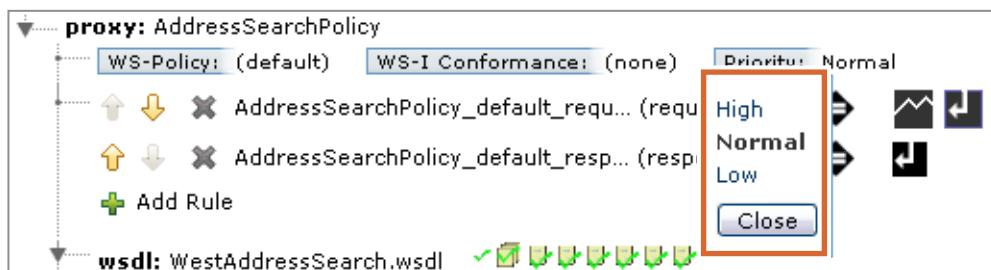
Reject nonconforming messages include:

- **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
  - Normal:** (default) Receives normal priority
  - Low:** Receives lower than normal priority

© Copyright IBM Corporation 2015

Figure 14-31. Service priority

WE711 / ZE7111.0

### 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**
  - Dynamic Backend:** Web service proxy determines the back-end server during service processing
  - Static Backend:** Web service proxy forwards to a single back-end server
  - 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, there is a 'Comments' field and an 'XML Manager' dropdown set to 'default'. The 'Authorization AAA Policy' dropdown is set to '(none)'. The 'Type' section contains three radio buttons: 'Dynamic Backend', 'Static Backend', and 'Static from WSDL', with 'Static from WSDL' being the selected option. Buttons for 'Apply', 'Cancel', 'Delete', and 'Refresh' are at the bottom left, and 'Export', 'View', and 'Show' are at the bottom right.

© Copyright IBM Corporation 2015

Figure 14-32. Proxy settings (1 of 4)

WE711 / ZE7111.0

### 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 2015

Figure 14-33. Proxy settings (2 of 4)

WE711 / ZE7111.0

### 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 because 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` or `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



Figure 14-34. Proxy settings (3 of 4)

WE711 / ZE7111.0

© Copyright IBM Corporation 2015

### 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' page in the WebSphere Education interface. At the top, there are tabs: WSDL files, SLM Policy, Services, Policy, SLA Policy Details, Proxy Settings (which is selected), and Advanced Proxy S. Below the tabs, the 'Web Service Proxy Name' is set to 'AddressSearchProxy'. There are buttons for Apply, Cancel, Delete, and Refresh. To the right, there are links for Export, View, and Show. The main area is divided into sections: 'General Configuration' (Comments field) and 'Advanced Configuration' (XML Manager and Authorization AAA Policy). The 'XML Manager' dropdown is set to 'default' and has a '+' button. The 'Authorization AAA Policy' dropdown is set to '(none)' and has a '+' button. Both sections are enclosed in a red box.

© Copyright IBM Corporation 2015

Figure 14-35. Proxy settings (4 of 4)

WE711 / ZE7111.0

### 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 that enters the web service proxy by using the **Throttle** and **Shape** action
  - Can view graph to see results of the traffic

**SLM**

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

**Auto Generated SLM Statements**

Define the request or failure SLM policy.

```

 proxy: addressSearchProxy
  Request: (none) Failure: (none)
 wsdl: EastAddressSearch.wsdl
  Request: none Failure: Interval=0,Limit=0,Action=notify Graph
 service: {http://east.address.training.ibm.com}AddressSearchService
  Request: (none) Failure: (none)
 port: {http://east.address.training.ibm.com}AddressSearch
  Request: (none) Failure: (none)
    port-operation: findByLocation
    Request: (none) Failure: (none)
    port-operation: findByName
    Request: Interval=60,Limit=500,Action=shape Failure: none Graph

```

© Copyright IBM Corporation 2015

Figure 14-36. Web service proxy: SLM Policy tab

WE711 / ZE7111.0

### 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:** Any 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.

 IBM

## 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**
  - Click an existing web service proxy
  - Use the arrows at the top to select the **WSDL Cache Policy** tab

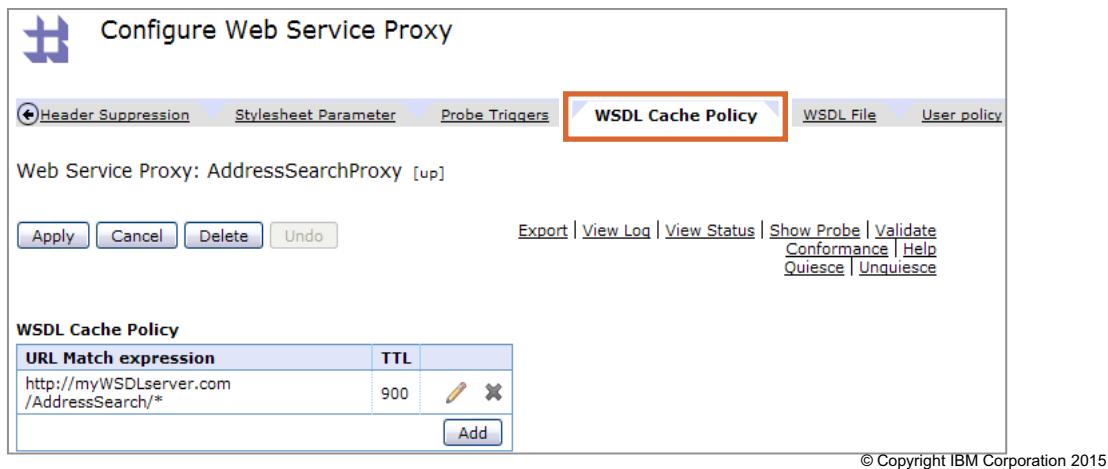


Figure 14-37. WSDL cache policy WE711 / ZE7111.0

### Notes:



## 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 is transmitted to correct back-end URL (system log)
- Response
  - Response is received from back-end service (system log)
  - Response passed automatic schema validation (user policy)
  - Response is transmitted completely to client (system log, probe)

© Copyright IBM Corporation 2015

Figure 14-38. Troubleshooting a web service proxy

WE711 / ZE7111.0

### Notes:



## 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 2015

Figure 14-39. Unit summary

WE711 / ZE7111.0

### Notes:

## Checkpoint questions

1. True or False: A web service proxy and an 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 2015

Figure 14-40. Checkpoint questions

WE711 / ZE7111.0

### 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 2015

Figure 14-41. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 12



Configuring a web service proxy

© Copyright IBM Corporation 2015

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

10.1

Figure 14-42. Exercise 12

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2015

---

Figure 14-43. Exercise objectives

WE711 / ZE7111.0

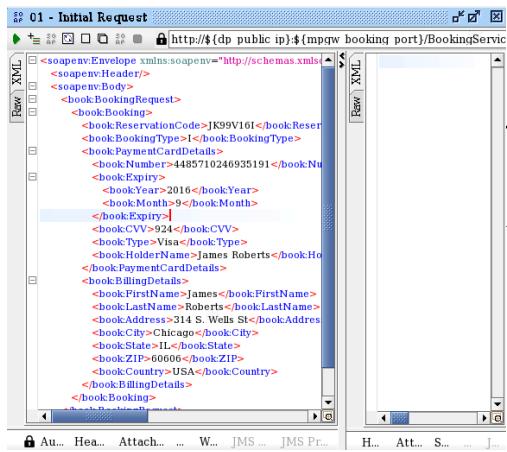
### Notes:

Future Update Configure Kerberos - Integrate Bus Messenger

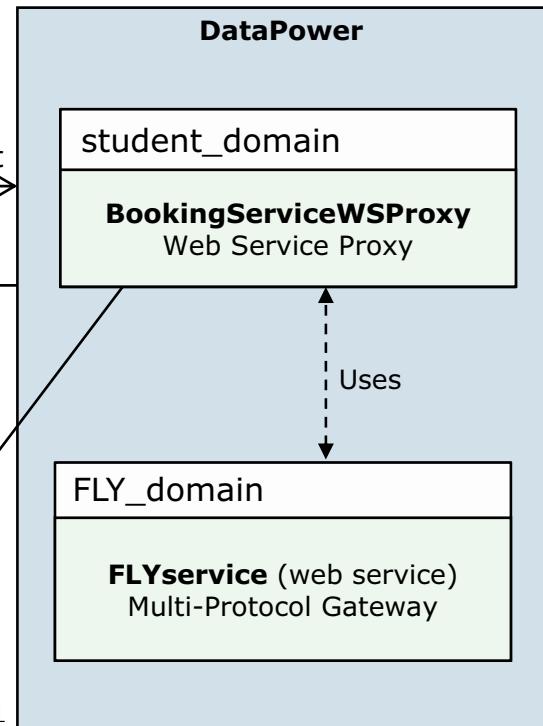


## Exercise overview

### SoapUI



BookingService.wsdl



© Copyright IBM Corporation 2015

Figure 14-44. Exercise overview

WE711 / ZE7111.0

### Notes:



# Unit 15. 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 SLM.

## What you should be able to do

After completing this unit, you should be able to:

- Identify the SLM functions that the 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
- Hands-on exercise

## References

IBM DataPower Gateway Appliances Version 7.1 Knowledge Center:

[www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0/com.ibm.dp.doc/welcome.html](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0/com.ibm.dp.doc/welcome.html)



## Unit objectives

After completing this unit, you should be able to:

- Identify the SLM functions that the 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 2015

---

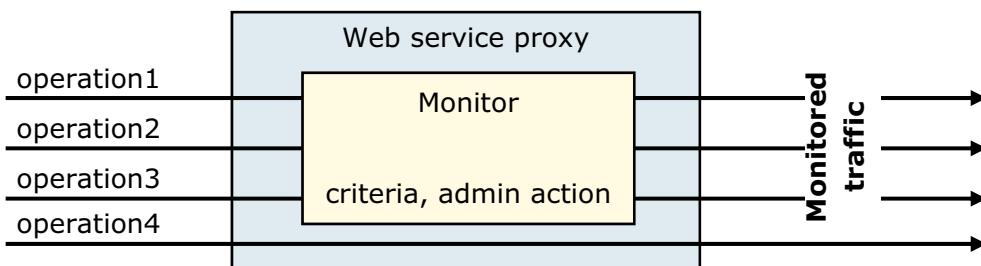
Figure 15-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## Service level monitoring (SLM) in DataPower

- A concept of monitoring message traffic within 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 2015

Figure 15-2. Service level monitoring (SLM) in DataPower

WE711 / ZE7111.0

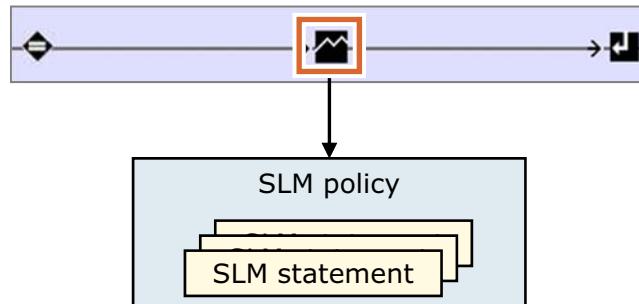
### Notes:

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

- 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 2015

Figure 15-3. The pieces of SLM

WE711 / ZE7111.0

### 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 2015

Figure 15-4. Approaches to define SLM policies

WE711 / ZE7111.0

### Notes:

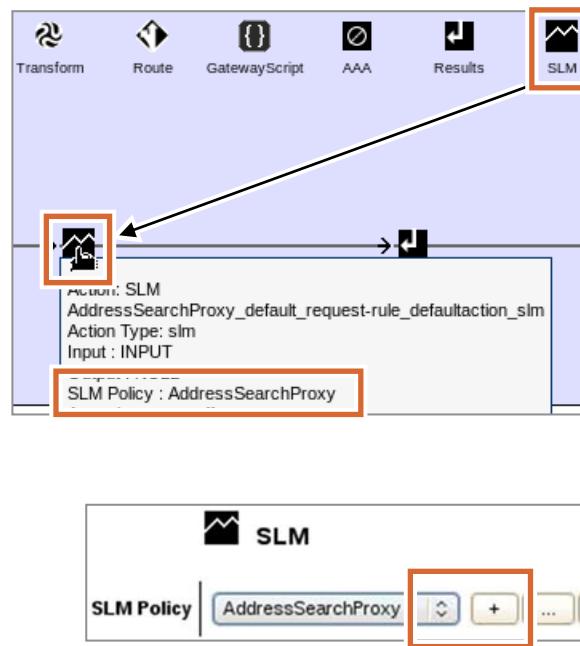
The first two approaches have been supported for many years.

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 they can 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 2015

Figure 15-5. Approach 1: Add an SLM action to a request rule

WE711 / ZE7111.0

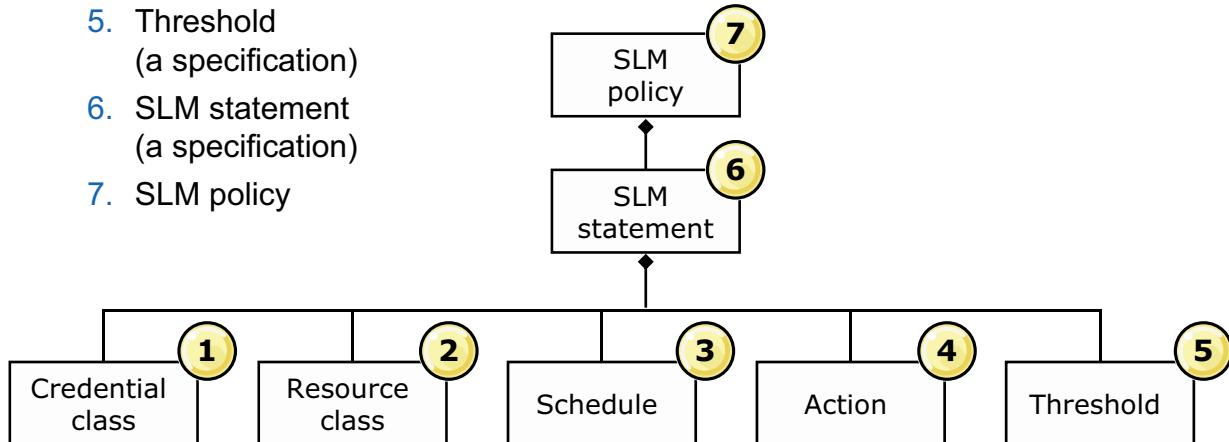
### Notes:

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

- An SLM policy requires the following objects, *if they affect the policy*:
  - SLM credential class
  - SLM resource class
  - SLM schedule
  - SLM action
  - Threshold  
(a specification)
  - SLM statement  
(a specification)
  - SLM policy



© Copyright IBM Corporation 2015

Figure 15-6. The pieces of SLM

WE711 / ZE7111.0

### 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                 | <input type="text"/>  | * |
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |   |
| Comments             | <input type="text"/>  |   |
| Credential Type      | <input type="text" value="Mapped Credential"/> *                        |   |
| Match Type           | <input type="text" value="Per Extracted Value"/> *                      |   |

© Copyright IBM Corporation 2015

Figure 15-7. The SLM credential class

WE711 / ZE7111.0

### 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 the service is using the MQ transport protocol, you can also use the name of the **MQ application** that is contained in the message. 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 only to values that match. Another field appears that lists the accepted values. For the Type of **Regular Expression**, an SLM policy applies only to values that match. Instead of a list of specific values to match, a field appears that lists PCRE-style expressions to determine whether 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 that is 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 2015

Figure 15-8. The SLM resource class

WE711 / ZE7111.0

### 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 <a href="#">*</a>   |
| Match Type     | Exact <a href="#">*</a>  |
| Resource Value | {http://east.address.training.ibm.com}AddressSearch/findByName <a href="#">↑</a> <a href="#">↓</a> <a href="#">×</a> |

© Copyright IBM Corporation 2015

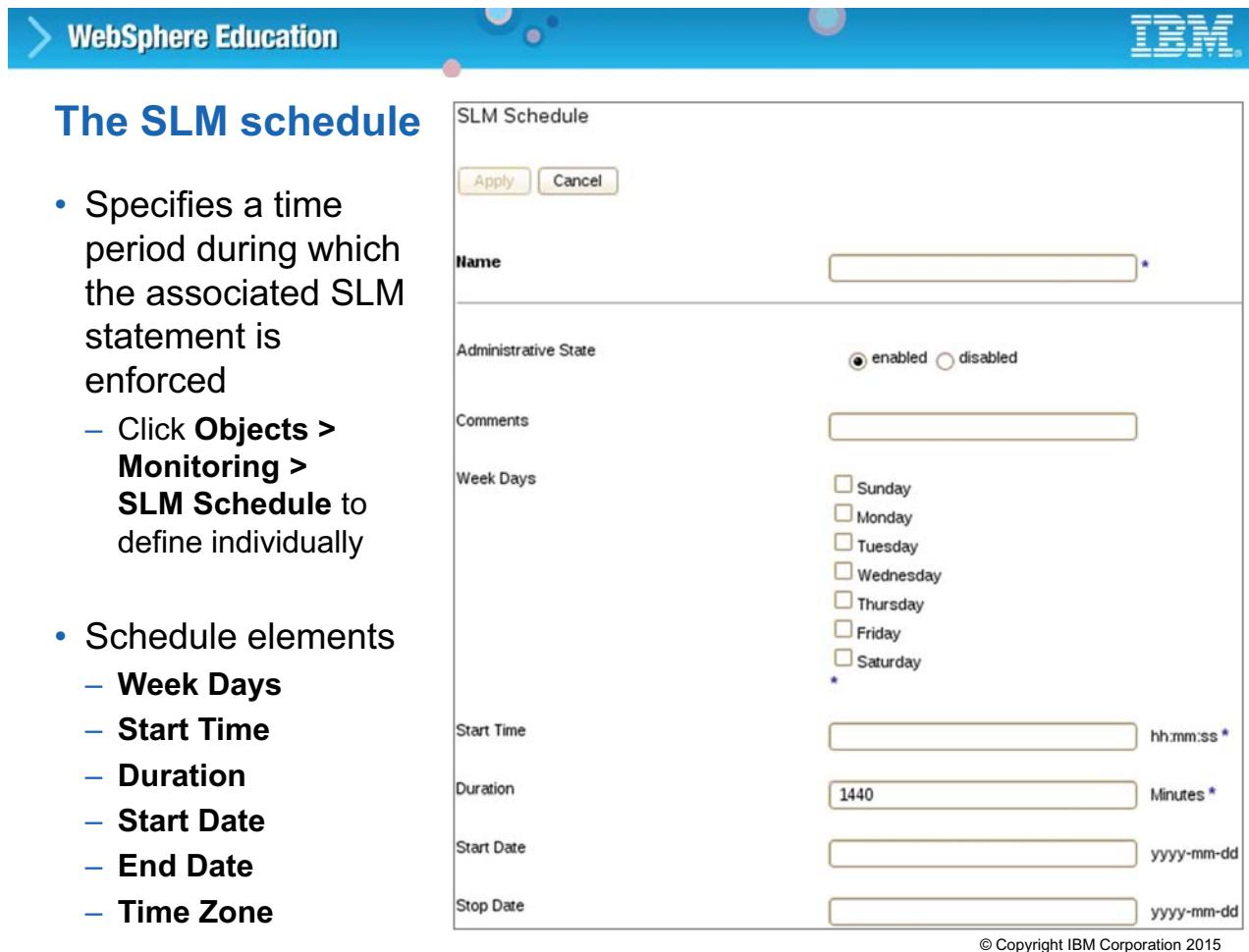
Figure 15-9. SLM resource class example

WE711 / ZE7111.0

### 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 screenshot shows the 'SLM Schedule' configuration page. At the top, there are 'Apply' and 'Cancel' buttons. The 'Name' field is empty and marked with a red asterisk. The 'Administrative State' is set to 'enabled'. The 'Comments' field is empty. Under 'Week Days', there is a list of days of the week with checkboxes: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and a red asterisk. Below this are fields for 'Start Time' (hh:mm:ss), 'Duration' (1440 Minutes), 'Start Date' (yyyy-mm-dd), and 'Stop Date' (yyyy-mm-dd).

© Copyright IBM Corporation 2015

Figure 15-10. The SLM schedule

WE711 / ZE7111.0

## 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 selected 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 SLM action

- 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

SLM Action

Apply Cancel

|              |   |
|--------------|---|
| Name         |   |
| Admin State  | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments     |   |
| Type         | Log Only *  |
| Log Priority | debug   |

Configure SLM Action

Refresh List

| Name     | Status | Op-State | Logs | Admin State | Comments |
|----------|--------|----------|------|-------------|----------|
| notify   | saved  | up       | 🔍    | enabled     |          |
| shape    | saved  | up       | 🔍    | enabled     |          |
| throttle | saved  | up       | 🔍    | enabled     |          |

Add

© Copyright IBM Corporation 2015

Figure 15-11. The SLM action

WE711 / ZE7111.0

### 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, it writes a log entry and continues to process subsequent transactions.
- Reject**: After the action is triggered, it writes a log entry and rejects traffic until the monitored entity is within conformance levels.
- Shape**: After the action is triggered, it 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
  - SLM 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 2015

Figure 15-12. SLM statement (1 of 2)

WE711 / ZE7111.0

### 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 fields

- 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
  - Concurrent:** Use concurrent number of transactions

|  |   |                                     |
|--|---|-------------------------------------|
| Threshold Interval Length  | 0   | Seconds                             |
| Threshold Interval Type  | Fixed   | <input checked="" type="checkbox"/> |
| Threshold Algorithm  | Greater Than  | <input checked="" type="checkbox"/> |
| Threshold Type   | Count All   | <input checked="" type="checkbox"/> |
| 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 Algorithm:** Greater than, less than, token bucket, high-low threshold
- Threshold Type:** Count all, count errors, back-end/internal/total latency, request/response/total message payload
- Threshold Level:** Value that triggers the threshold

© Copyright IBM Corporation 2015

Figure 15-13. SLM statement (2 of 2)

WE711 / ZE7111.0

### 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. With this property, you can 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 from 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** (the default)

SLM Policy: AddressSearchProxy [up]

Administrative State:  enabled  disabled

Comments:

Evaluation Method:

Peer Group:

© Copyright IBM Corporation 2015

Figure 15-14. SLM policy: Main tab

WE711 / ZE7111.0

### 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

**Main      Statement**

SLM Policy: AddressSearchProxy [up]

Apply    Cancel    Delete    Undo

| <b>Statement</b>  |                        |                         |   |                 |                   |                        |  |
|-------------------|------------------------|-------------------------|---|-----------------|-------------------|------------------------|--|
| <b>Identifier</b> | <b>User Annotation</b> | <b>Credential Class</b> | <b>Resource Class</b>                                   | <b>Schedule</b> | <b>SLM Action</b> | <b>Thre Inter Leng</b> |  |
| 1                 | Auto Generated         |                         | AddressSearchProxy_a552283b-ce25-442f-b609-14de727fbe6e |                 | notify            | 60                     |  |
| 2                 | Auto Generated         |                         | AddressSearchProxy_91bec166-0909-4e55-96d2-7f1ae2d29ae6 |                 | throttle          | 60                     |  |
| 3                 | limit on web service   |                         | EastAddressSeach_URI                                    |                 | throttle          | 60                     |  |

© Copyright IBM Corporation 2015

Figure 15-15. SLM policy: Statements tab

WE711 / ZE7111.0

### Notes:



## Getting SLM statements into the Statement list

- Because 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 |            |
|---------|---------------------|----------------|-----------------|------------------------|-------------|--------------------------------|----------------------------------|-----------------------|---|------------|
| 1       | Greater Than        | Count All      | 2               | 0                      | 0           | 0                              | 5000                             | on                    | 5000                                      |            |
| 2       | Greater Than        | Count All      | 5               | 0                      | 0           | 0                              | 5000                             | on                    | 5000                                      |            |
| 3       | Greater Than        | Count All      | 200             | 0                      | 0           | 0                              | 5000                             | off                   | 5000                                      |            |
|         |                     |                |                 |                        |             |                                |                                  |                       |   | <b>Add</b> |

© Copyright IBM Corporation 2015

Figure 15-16. Getting SLM statements into the Statement list

WE711 / ZE7111.0

### 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)

**Auto Generated SLM Statements**

Define the request or failure SLM policy.

**proxy:** AddressSearchProxy  
 Request (none)  Failure (none)  
 **wsdl:** EastAddressSearch.wsdl  
 Request (none)  Failure (none)  
 **wsdl:** WestAddressSearch.wsdl  
 Request (none)  Failure (none)  
 **service:** {http://west.address.training.ibm.com}AddressSearchService  
 Request (none)  Failure (none)  
 **port:** {http://west.address.training.ibm.com}AddressSearch  
 Request (none)  Failure (none)  
**port-operation:** findByName  
 Request Interval=60,Limit=200,Action=notify  Failure none  Graph  
**port-operation:** retrieveAll  
 Request (none)  Failure (none)  
**port-operation:** findByLocation  
 Request Interval=60,Limit=150,Action=throttle  Failure none  Graph

© Copyright IBM Corporation 2015

Figure 15-17. Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2)

WE711 / ZE7111.0

### 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.

**WebSphere Education**

**Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2)**

- **SLM Peers** are a cluster of appliances that support the same service and SLM policy
- **SLM Statements** list only custom SLM statements
- **Create New Statement** allows creation of a custom SLM statement

The screenshot shows the 'SLM Peers' and 'SLM Statements' sections of the SLM Policy tab. The 'SLM Peers' section includes fields for Type (SLM Unicast) and URL. The 'SLM Statements' section includes a table with columns: ID, Credential Class, Resource Class, Schedule, Threshold Level, Threshold Type, and Action. The 'Create New Statement' button in the 'Action' column is highlighted.

© Copyright IBM Corporation 2015

Figure 15-18. Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2)

WE711 / ZE7111.0

### 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.

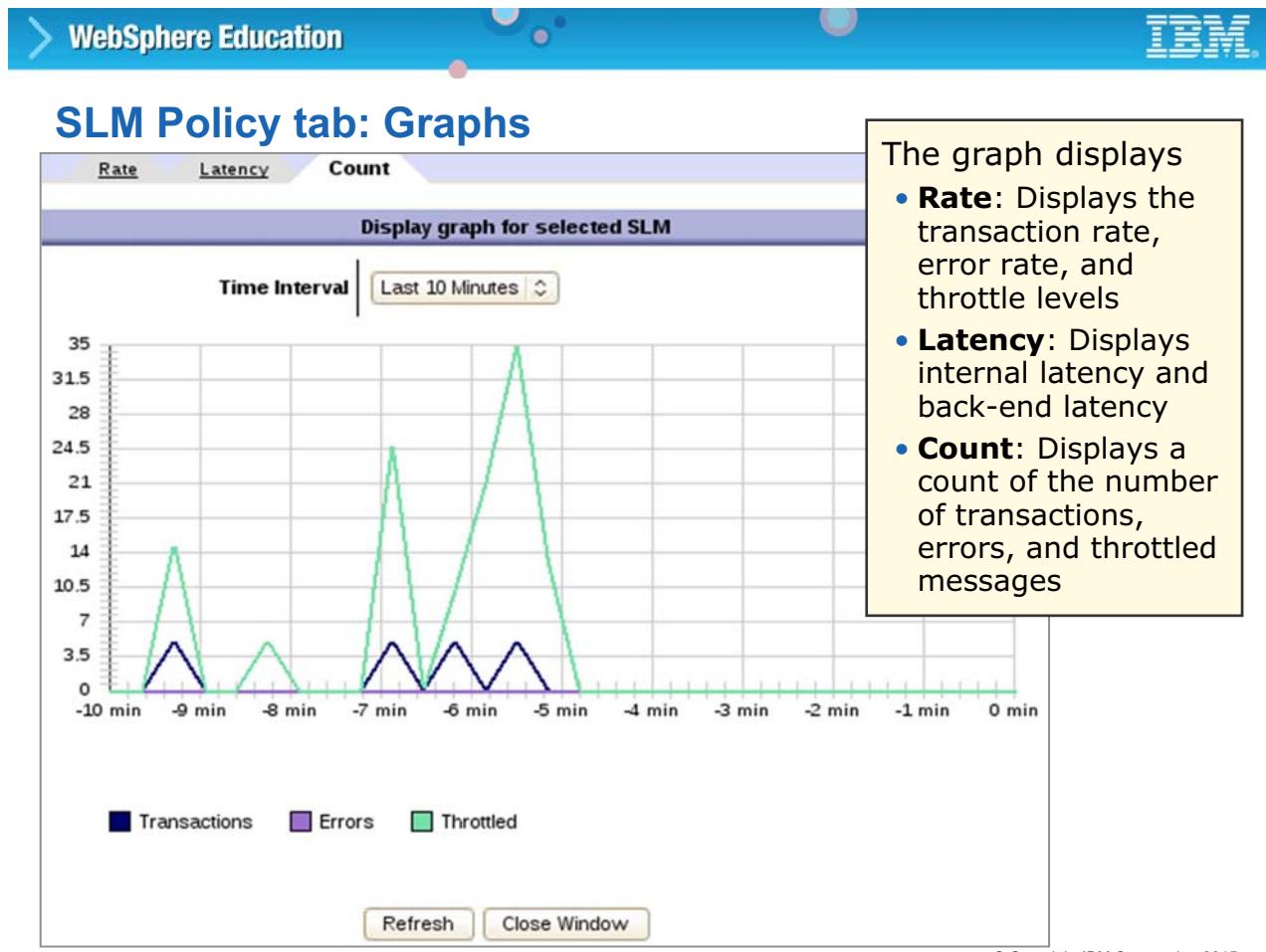


Figure 15-19. SLM Policy tab: Graphs

WE711 / ZE7111.0

## Notes:

The SLM graph is displayed by selecting the appropriate **Graph** radio button in the web service proxy **SLM Policy** tab.

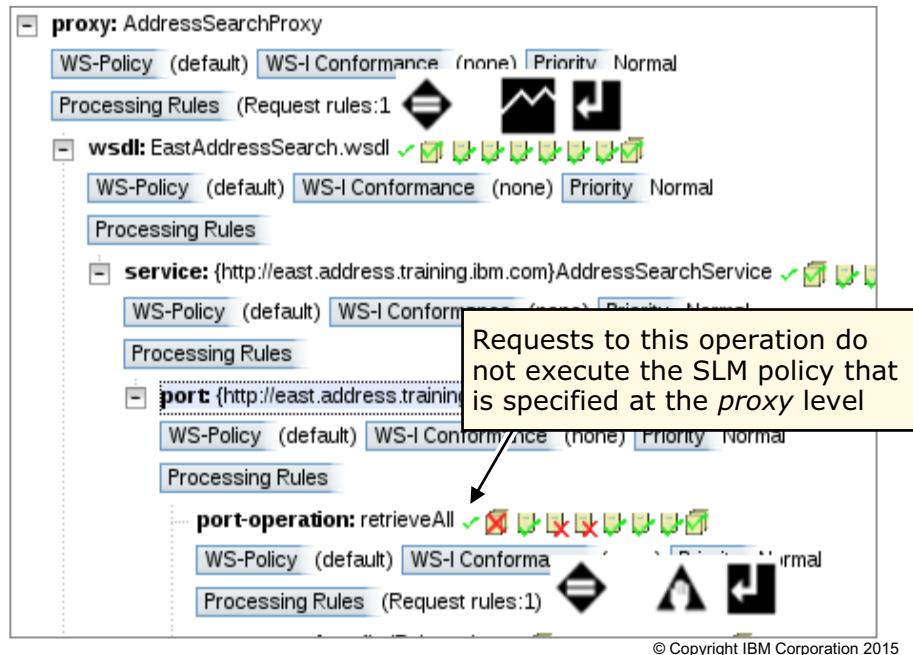
The possible time intervals are last 10 minutes, last 30 minutes, last hour, and last 3 hours.

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** 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 2015

Figure 15-20. SLM action granularity

WE711 / ZE7111.0

### 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 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 2015

---

Figure 15-21. Unit summary

WE711 / ZE7111.0

### 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 function 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 2015

Figure 15-22. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

- 1.
  
2.
  - (1)
  - (2)
  
- 3.



## Checkpoint answers

1. A
2. 1 – A, 2 – B
3. True

© Copyright IBM Corporation 2015

Figure 15-23. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 13



Implementing a service level monitor  
in a web service proxy

© Copyright IBM Corporation 2015

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

10.1

Figure 15-24. Exercise 13

WE711 / ZE7111.0

### 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 2015

Figure 15-25. Exercise objectives

WE711 / ZE7111.0

### Notes:



## Exercise overview

- Test the existing BookingServiceWSProxy by using the load test facility in SoapUI
- Create a log target for SLM log messages
- Add SLM criteria to the web service proxy
- Test the SLM action by using the SoapUI load test
- Add an SLM action to a port-operation request rule
- Use the SoapUI load test to test the operation-level SLM action

© Copyright IBM Corporation 2015

Figure 15-26. Exercise overview

WE711 / ZE7111.0

### Notes:

The SoapUI load test sends a message a specific number of times within a specific interval.



# Unit 16. 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, and how a new service can be created 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 describe 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 DataPower Gateway Appliances Version 7.1 Knowledge Center:

[www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0/  
com.ibm.dp.doc/welcome.html](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0/com.ibm.dp.doc/welcome.html)



## Unit objectives

After completing this unit, you should be able to:

- Explain what a DataPower pattern is, and describe its purpose
- Describe how a pattern is created
- Generate a new service from a pattern

© Copyright IBM Corporation 2015

---

Figure 16-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## What is a pattern?

- A pattern is an importable and 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 or specifications
- The pattern presents only the limited set of variables and specifications
- 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 “Blueprint Console” GUI is used to work with patterns
- Several sample patterns are supplied with the firmware

© Copyright IBM Corporation 2015

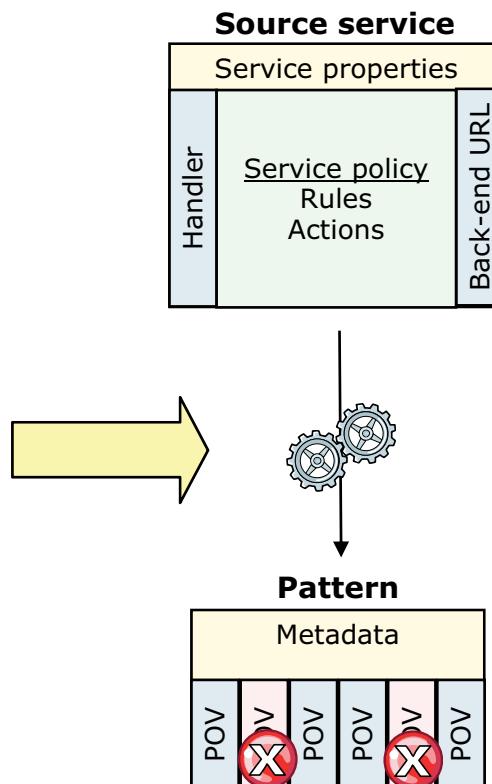
Figure 16-2. What is a pattern?

WE711 / ZE7111.0

### 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
- Up to three services can be selected to be “chained” together in the same pattern
- 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 2015

Figure 16-3. Creating a pattern

WE711 / ZE7111.0

### Notes:

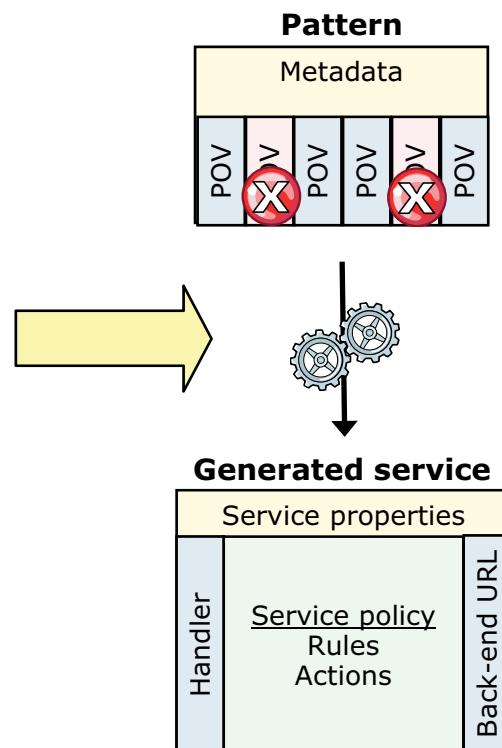
There is a list of the configuration variables in a service and its related objects that a pattern can expose. The pattern creation dialog box displays them.

The pattern creator decides to expose the variable in the pattern, or locks the value so the deployer cannot change it.

Service chaining within a pattern is available in firmware V7.1 and later.

## 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 2015

Figure 16-4. Deploying a pattern

WE711 / ZE7111.0

### Notes:

The deployer can use only the POVs that the pattern creator exposed in the pattern. The non-exposed POVs are hidden from the deployer.



## The Blueprint Console

- Any work that involves patterns must be done in the **Blueprint Console**
- The Blueprint Console UI is started in a single, separate browser tab or browser window
  - Existing WebGUI remains intact
- Blueprint Console includes the following capabilities
  - Browse and deploy patterns
  - Customize and create patterns
  - Clone patterns
  - View service status
  - Manage the appliance configuration, the network environment, network and user access, application domains, and data storage (**default** domain only)
- Several predefined “best practice” patterns are included in the firmware
  - Read-only, but they can be cloned

© Copyright IBM Corporation 2015

---

Figure 16-5. The Blueprint Console

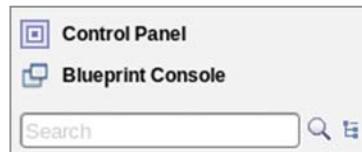
WE711 / ZE7111.0

### Notes:



## Getting to the Blueprint Console

- Two ways to get to the Blueprint Console:
  - Its own URL  
`https://<appliance_address>:<WebGUI_port>/dp`
  - Click **Blueprint Console** from the WebGUI:



- It opens a second browser tab or browser window
  - Shared session between Blueprint Console and WebGUI, so WebGUI login and timeout apply to both
  - Blueprint Console has its own Login page that you can use if a WebGUI timeout occurs

© Copyright IBM Corporation 2015

Figure 16-6. Getting to the Blueprint Console

WE711 / ZE7111.0

### Notes:



## The Blueprint Console

- Interface is different from the WebGUI

**Get started**

**Menu bar**

**Tabs**

**Welcome to DataPower.** The console provides appliance management and service pattern functionalities. The tasks described in this page get you started with the console. More information is available in the IBM Knowledge Center ([here](#)) or by clicking the help icon in the banner.

**Shortcuts**

**Manage appliance**

Manage the appliance configuration, the network environment, network and user access, application domains, and data storage. [Learn more](#)

**Manage patterns**

Create and deploy services from patterns. [Learn more](#)

**View services**

View services in the domain. [Learn more](#)

© Copyright IBM Corporation 2015

Figure 16-7. The Blueprint Console

WE711 / ZE7111.0

### Notes:



## The Blueprint Console: Menu bar

- Different from WebGUI

Drop-down list has:

- “About” choice to show the firmware version
- “Information Center” to open a tab or window in the DataPower product documentation

Drop-down list of available domains

Drop-down has the **Logout** option



© Copyright IBM Corporation 2015

Figure 16-8. The Blueprint Console: Menu bar

WE711 / ZE7111.0

### Notes:

“Save configuration” shows only in domains where the user has write permissions.



## The Blueprint Console: Getting Started tab

- Shortcuts to tasks available under the other tabs
- Clicking **Learn more** opens a text pane that provides some help on the task

**Get started**

Welcome to DataPower. The console provides appliance management and service pattern functionalities. The tasks described in this page get you started with the console. More information is available in the IBM Knowledge Center that is accessed through the help icon in the banner.

**Manage appliance**

Manage the appliance configuration, the network environment, network and user access, application domains, and data storage.  
[Learn more](#)

**Manage patterns**

Create and deploy services from patterns.  
[Learn more](#)

**View services**

View services in the domain.  
[Learn more](#)

© Copyright IBM Corporation 2015

Figure 16-9. The Blueprint Console: Getting Started tab

WE711 / ZE7111.0

### Notes:

The “manage appliance” functions are available in the **default** domain only. The option is not shown in other domains.



## The Blueprint Console: Services tab

- Read-only list of *all* the services that are defined in a specific domain
- List can be sorted by clicking each column header

The screenshot shows the DataPower Blueprint Console interface. The top navigation bar includes 'DataPower | XI52 console at 172.16.76.23', 'student32\_domain', 'admin', and a help icon. Below the navigation is a toolbar with 'Get started', 'Patterns', 'Services' (which is selected), 'Save configuration', 'Export configuration', and 'Import'. A search bar labeled 'Filter' is also present.

| Service                        | Status | Service Type           | Front side URL  | Actions |
|--------------------------------|--------|------------------------|---|---------|
| BookingServiceProxy            | Up     | Multi-Protocol Gateway | mq://FLYIN32<br>http://dp_public_ip:12321<br>https://dp_public_ip:12322 |         |
| FindBagEnforcementServer       | Up     | Multi-Protocol Gateway | https://dp_public_ip:7323   |         |
| BookingServiceProxyFromPattern | Up     | Multi-Protocol Gateway | http://dp_public_ip:7326  |         |
| BaggageServiceProxy            | Up     | Multi-Protocol Gateway | http://dp_public_ip:12329   |         |
| BookingMQclient                | Up     | Multi-Protocol Gateway | http://dp_public_ip:12324   |         |
| HelloWorld                     | Up     | Multi-Protocol Gateway | http://0.0.0.0:7328   |         |
| JimTestWSP                     | Up     | Web Service Proxy      | http://0.0.0.0:9871   |         |

© Copyright IBM Corporation 2015

Figure 16-10. The Blueprint Console: Services tab

WE711 / ZE7111.0

### Notes:

The list displays all the services in the domain, regardless of whether they were created from a pattern or not.

Click the service to open the WebGUI configuration page for that service.

The BookingServiceProxy has three front side handlers: MQ, HTTP, and HTTPS.

“dp\_public\_ip” is a host alias that is specified in the service or handler definition. If you hover over the “Front side URL” entries, the IP address of that host alias as defined on this appliance appears in the hover help.

The icon entries for the Actions column open the system log for that service.



## The Blueprint Console: Patterns tab

- Where all the pattern creation, editing, and deployment happens

**Toolbar**

**List of patterns in this domain**

**Documentation on selected pattern as written by the pattern creator**

**© Copyright IBM Corporation 2015**

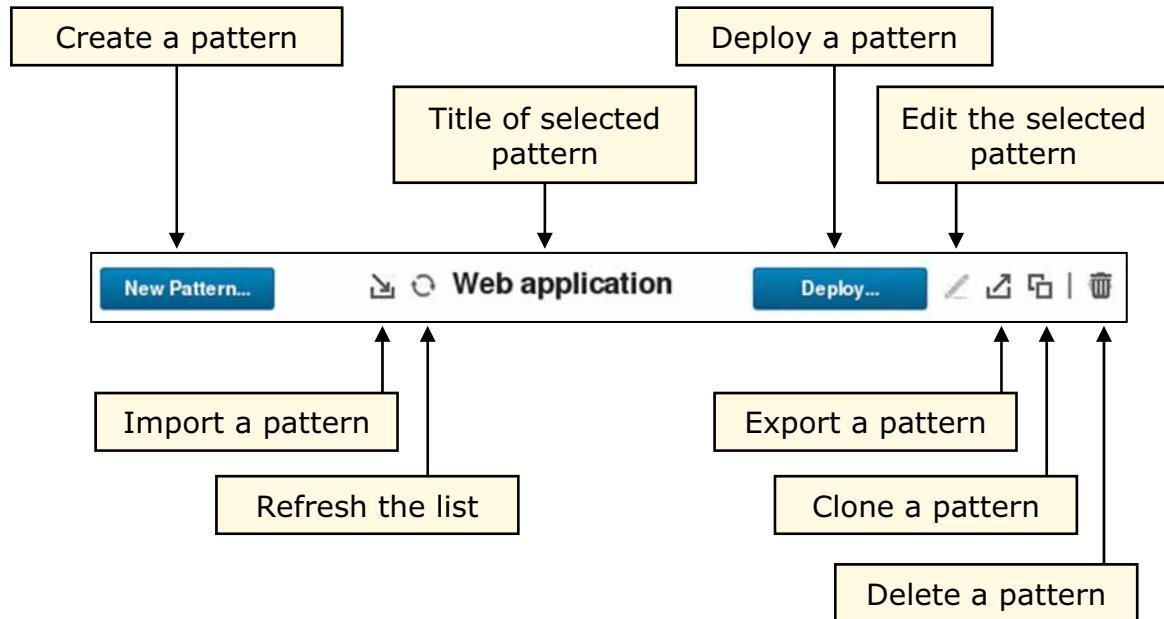
Figure 16-11. The Blueprint Console: Patterns tab

WE711 / ZE7111.0

### Notes:

Because “REST Proxy” is one of the supplied sample patterns, you must be in the **default** domain to be able to see it in the patterns list. Therefore, because it is the **default** domain, the **Manage** tab is also visible.

## The pattern toolbar



© Copyright IBM Corporation 2015

Figure 16-12. The pattern toolbar

WE711 / ZE7111.0

### Notes:

The supplied sample patterns in the default domain cannot be edited. They can be copied.

“Web application” is another of the supplied patterns in the default domain.



## The Blueprint Console: Manage tab

- Manage some appliance aspects, RAID storage, domains, users
- Available only in **default** domain

The screenshot shows the Blueprint Console interface. The top navigation bar includes 'Get started', 'Manage' (which is highlighted with a red box), 'Patterns', and 'Services'. On the right, there are 'Save configuration' and 'Export configuration' buttons. The left sidebar has sections for 'User access' (highlighted with a red box) and 'User Account' (also highlighted with a red box). Under 'User access', there are links for 'Appliance', 'Data storage', 'Lifecycle management', 'Network', 'Network access', and 'User access'. The main content area is titled 'User Account' and displays a table of user accounts:

|                          | Name            | Status | Comment                                    |
|--------------------------|-----------------|--------|--|
| <input type="checkbox"/> | admin           | up     | Administrator                              |
| <input type="checkbox"/> | student20       | up     | OAuth test account for Warren              |
| <input type="checkbox"/> | student21       | up     | Developer account on the student 21 domain |
| <input type="checkbox"/> | student21_admin | up     |  |
| <input type="checkbox"/> | student32       | up     | JB   |
| <input type="checkbox"/> | student40       | up     | Kevin                                      |

© Copyright IBM Corporation 2015

Figure 16-13. The Blueprint Console: Manage tab

WE711 / ZE7111.0

### Notes:

The available functions under this tab are primarily administrative functions that appliance administrators perform.

- **Appliance:** System settings, time settings, language
- **Data storage:** RAID array
- **Lifecycle management:** Domains
- **Network:** Ethernet interfaces
- **Network access:** Telnet Service, SSH Service, Web Management Service, XML Management Interface
- **User access:** User Account, User Group, RBM Settings

The screen capture is the User Account page for the User access selection.



## 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 that are 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 2015

Figure 16-14. Steps to generate a service from a pattern

WE711 / ZE7111.0

### 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 the 'New Pattern...' screen in the WebSphere Education interface. On the left, there is a dropdown menu labeled 'All Categories' which is currently set to 'All Categories'. Below this, there are two more categories: 'Web Application' and 'Web Service'. A red box highlights the 'All Categories' dropdown. On the right, there is a search bar with the text 'wsrr' entered. Below the search bar, a list of results is displayed, including:
 

- Web application with WSRR Saved Search subscription
- Web application with WSRR Service Version subscription
- Web service with WSRR Saved Search subscription

- Select the pattern in the list
- Click **Deploy** on the page to start the deployment wizard

© Copyright IBM Corporation 2015

Figure 16-15. Deployment: Selecting a pattern

WE711 / ZE7111.0

### Notes:

**Deployment: Filling out the wizard**

**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, options 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**

**Deploy Pattern** **Cancel**

- Specify the name of the service to be generated
- Complete 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 as any other service can
  - The service name is used as the prefix for the name of the generated dependent objects

© Copyright IBM Corporation 2015

Figure 16-16. Deployment: Filling out the wizard

WE711 / ZE7111.0

## 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
  - WebSphere Service Registry and Repository subscription, WebSphere Service Registry and Repository saved search subscription
  - Multi-protocol gateway back-end URL
  - Authentication with LTPA token, SSL certificate
  - Authorization and authentication with LDAP
  - Authorization and authentication with IBM Security Asset Manager
  - Identity extraction from OAuth
- The pattern creator decides which POVs in the source service are exposed to a pattern deployer

© Copyright IBM Corporation 2015

Figure 16-17. Points of variability

WE711 / ZE7111.0

### Notes:

IBM Security Asset Manager was previously called Tivoli Access Manager.



## Unit summary

Having completed this unit, you should be able to:

- Explain what a DataPower pattern is, and describe its purpose
- Describe how a pattern is created
- Generate a new service from a pattern

© Copyright IBM Corporation 2015

Figure 16-18. Unit summary

WE711 / ZE7111.0

### Notes:



## Checkpoint questions

1. True or False: If a pattern is updated, all services that are generated from that pattern are also updated.
  
2. True or False: Patterns are available in the default domain only.

© Copyright IBM Corporation 2015

Figure 16-19. Checkpoint questions

WE711 / ZE7111.0

### Notes:

Write your answers here:

1.

2.



## Checkpoint answers

1. **False.** As soon as 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 2015

Figure 16-20. Checkpoint answers

WE711 / ZE7111.0

### Notes:

## Exercise 14



Using a DataPower pattern with the Blueprint console

© Copyright IBM Corporation 2015

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

10.1

Figure 16-21. Exercise 14

WE711 / ZE7111.0

### Notes:



## Exercise objectives

After completing this exercise, you should be able to:

- Use the DataPower Blueprint 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 2015

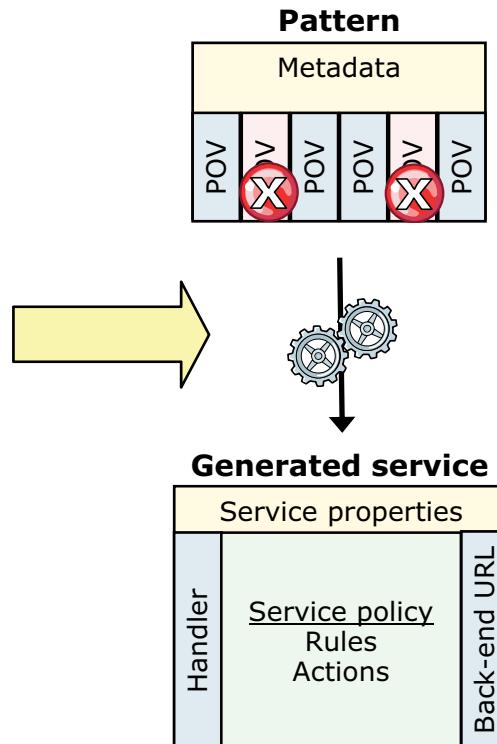
Figure 16-22. Exercise objectives

WE711 / ZE7111.0

### 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 2015

Figure 16-23. Exercise overview

WE711 / ZE7111.0

### 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 2015

---

Figure 17-1. Unit objectives

WE711 / ZE7111.0

### Notes:

## Course learning objectives

After completing this course, you should be able to:

- Describe how DataPower 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 DataPower appliances
- Configure a multi-protocol gateway (MPGW) to handle multiple protocols from a single service
- Configure a service level monitoring (SLM) policy to control message traffic

© Copyright IBM Corporation 2015

Figure 17-2. Course learning objectives

WE711 / ZE7111.0

### Notes:



## Course learning objectives

After completing this course, you should be able to:

- Configure support for IBM MQ
- 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
- Configure message transformation and routing by using style sheets (XSL) and GatewayScripts
- Handle errors in service policies

© Copyright IBM Corporation 2015

Figure 17-3. Course learning objectives

WE711 / ZE7111.0

### Notes:

## Course review (1 of 3)

- The XG45, XI52, and IDG appliances secure XML and web service applications through *two main services*:
  - Multi-protocol gateway (MPGW)
  - Web service proxy (WS-Proxy)
- 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 MQ systems
- A *service policy* within the MPGW or WS-Proxy controls much of the behavior of a service
- 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

© Copyright IBM Corporation 2015

Figure 17-4. Course review (1 of 3)

WE711 / ZE7111.0

### Notes:

## Course review (2 of 3)

- The **Authentication, Authorization, and Auditing** processing action restricts access to resources
  - A wide range of message-level security specifications are supported, including WS-Security, SAML, OAuth, XACML, SPNEGO, LTPA, Kerberos, and others
- The MPGW supports JSON-structured data and can operate as part of a RESTful interface
  - An MPGW can be used to bridge between a REST client and a web services back end
- 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
- An appliance can use a *side cache* or *on-appliance cache* to store response documents to HTTP or HTTPS requests

© Copyright IBM Corporation 2015

Figure 17-5. Course review (2 of 3)

WE711 / ZE7111.0

### Notes:

## Course review (3 of 3)

- *IBM MQ* is a protocol that is commonly used for DataPower front-end or back-end transport
  - IBM MQ header manipulation and transactionality are supported
- 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 levels
  - Web service virtualization acts as a single endpoint for a group of operations that different back-end services implement
- *Message monitors* and *service-level monitors* can filter, shape, and throttle traffic through a DataPower appliance
- DataPower *Patterns* streamline the creation of services

© Copyright IBM Corporation 2015

Figure 17-6. Course review (3 of 3)

WE711 / ZE7111.0

### Notes:



## Lab exercise solutions

- Solutions are available in the `Solution` subdirectory:

`<lab_files>/Solutions`

- Remember to change
  - Port numbers
  - Back-end server (**Network > Interface > DNS Settings > Static Hosts**)
  - Front IP addresses (**Network > Interface > Host Alias**)

© Copyright IBM Corporation 2015

Figure 17-7. Lab exercise solutions

WE711 / ZE7111.0

### Notes:



## To learn more on the subject

- IBM Training website:
  - <http://www.ibm.com/training>
  - Search on “DataPower training path”
- Webcast: How to define developer resources in DataPower Virtual Edition for Developers v7
  - <http://youtu.be/EaQyQWwQVIY>
- DataPower knowledge center
  - [http://www.ibm.com/support/knowledgecenter/SS9H2Y\\_7.1.0](http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.1.0)
- IBM Redbooks:
  - <http://www.redbooks.ibm.com>
  - Search on “DataPower”
- developerWorks articles:
  - <http://www.ibm.com/developerworks/>
  - Search on “DataPower”
- DataPower Gateway Appliance Customer Forum:
  - <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1198>

© Copyright IBM Corporation 2015

Figure 17-8. To learn more on the subject

WE711 / ZE7111.0

### Notes:

The 12-minute webcast was developed to teach DataPower developers how to define a user account, user group, and application domain. The individual developer who is working on DataPower Virtual Edition for Developers V7 might need to perform these typically administrative functions.



## 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

© Copyright IBM Corporation 2015

Figure 17-9. More DataPower education opportunities

WE711 / ZE7111.0

### 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 2015

Figure 17-10. Unit summary

WE711 / ZE7111.0

### Notes:



# Appendix A. List of abbreviations

## A

|               |  |
|---------------|--|
| <b>AAA</b>    | authentication, authorization, and auditing        |
| <b>ACL</b>    | access control list                                |
| <b>ADT</b>    | Android Development Tools                          |
| <b>AES</b>    | Advanced Encryption Standard                       |
| <b>AMP</b>    | Appliance Management Protocol                      |
| <b>APAR</b>   | authorized program analysis report                 |
| <b>API</b>    | application programming interface                  |
| <b>AP-REQ</b> | Authentication Protocol - Request                  |
| <b>AS</b>     | Applicability Statement                            |
| <b>ASCII</b>  | American Standard Code for Information Interchange |

## B

|            |                             |
|------------|-----------------------------|
| <b>B2B</b> | business-to-business        |
| <b>BPM</b> | business process management |

## C

|              |                                   |
|--------------|-----------------------------------|
| <b>CA</b>    | certificate authority             |
| <b>CBA</b>   | context-based access              |
| <b>CBR</b>   | content-based routing             |
| <b>CCS</b>   | coded character set               |
| <b>CCSID</b> | coded character set ID            |
| <b>CGI</b>   | Common Gateway Interface          |
| <b>cHTML</b> | Compact HTML                      |
| <b>CLI</b>   | command-line interface            |
| <b>CN</b>    | common name                       |
| <b>COBOL</b> | Common Business Oriented Language |
| <b>CR</b>    | carriage return                   |
| <b>CRL</b>   | certificate revocation list       |
| <b>CSR</b>   | certificate signing request       |
| <b>CSS</b>   | cascading style sheet             |

## **D**

|             |   |
|-------------|---|
| <b>DAP</b>  | Directory Access Protocol                                 |
| <b>DB</b>   | database  |
| <b>DER</b>  | Distinguished Encoding Rules                              |
| <b>DES</b>  | Data Encryption Standard                                  |
| <b>DH</b>   | Diffie-Hellman  |
| <b>DHCP</b> | Dynamic Host Configuration Protocol                       |
| <b>DIME</b> | Direct Internet Message Encapsulation                     |
| <b>DIT</b>  | directory information tree                                |
| <b>DL/I</b> | Data Language/I   |
| <b>DMZ</b>  | A firewall configuration for securing local area networks |
| <b>DN</b>   | distinguished name  |
| <b>DNS</b>  | Dynamic Name Server                                       |
| <b>DOM</b>  | Document Object Model                                     |
| <b>DOP</b>  | data-oriented programming                                 |
| <b>DoS</b>  | denial-of-service   |
| <b>DP</b>   | DataPower   |
| <b>DPL</b>  | distributed program link                                  |
| <b>DSS</b>  | Digital Signature Standard                                |
| <b>DTD</b>  | document type definition                                  |
| <b>DVD</b>  | digital versatile disc                                    |

## **E**

|                |   |
|----------------|---|
| <b>EAR</b>     | enterprise archive  |
| <b>ebMS</b>    | ebXML Message Service   |
| <b>ECMA</b>    | European Computer Manufacturers Association                             |
| <b>EDI</b>     | Electronic Data Interchange   |
| <b>EDIFACT</b> | Electronic Data Interchange for Administration, Commerce, and Transport |
| <b>EDIINT</b>  | Electronic Data Interchange-Internet Integration                        |
| <b>EJB</b>     | Enterprise JavaBeans  |
| <b>EMS</b>     | Enterprise Messaging System   |
| <b>EON</b>     | Edge of Network   |
| <b>EP</b>      | enforcement point   |
| <b>ESB</b>     | enterprise service bus  |

---

|              |   |
|--------------|---|
| <b>ESR</b>   | extended support release                                    |
| <b>EXCI</b>  | external CICS interface                                     |
| <b>EXSLT</b> | Extensions to Extensible Stylesheet Language Transformation |

## F

|              |   |
|--------------|---|
| <b>FEPI</b>  | Front End Programming Interface         |
| <b>FIFO</b>  | first-in first-out                      |
| <b>FIPS</b>  | Federal Information Processing Standard |
| <b>FIX</b>   | Financial Information Exchange          |
| <b>FLWOR</b> | for, let, where, order by, return       |
| <b>FO</b>    | formatting object                       |
| <b>FSH</b>   | front side handler                      |
| <b>FTP</b>   | File Transfer Protocol                  |
| <b>FTPS</b>  | FTP over SSL                            |

## G

|              |                           |
|--------------|---------------------------|
| <b>GB</b>    | gigabyte                  |
| <b>GDB</b>   | GNU Project Debugger      |
| <b>GNU</b>   | GNU's Not UNIX            |
| <b>GSKit</b> | Global Security Kit       |
| <b>GSS</b>   | Generic Security Services |
| <b>GUI</b>   | graphical user interface  |

## H

|              |                                  |
|--------------|----------------------------------|
| <b>HMAC</b>  | hash message authentication code |
| <b>HR</b>    | human resources                  |
| <b>HREF</b>  | hypertext reference              |
| <b>HSM</b>   | Hardware Security Module         |
| <b>HSRP</b>  | Hot Standby Router Protocol      |
| <b>HTML</b>  | Hypertext Markup Language        |
| <b>HTTP</b>  | Hypertext Transfer Protocol      |
| <b>HTTPS</b> | HTTP over SSL                    |

## I

|             |          |
|-------------|----------|
| <b>ICAL</b> | IMS Call |
|-------------|----------|

|              |   |
|--------------|---|
| <b>ICAP</b>  | Internet Content Adaptation Protocol              |
| <b>ICMP</b>  | Internet Control Message Protocol                 |
| <b>ICRX</b>  | Extended Identity Context Reference               |
| <b>IDE</b>   | integrated development environment                |
| <b>IDEA</b>  | International Data Encryption Algorithm           |
| <b>IDG</b>   | IBM DataPower Gateway appliance                   |
| <b>IEEE</b>  | Institute of Electrical and Electronics Engineers |
| <b>IETF</b>  | Internet Engineering Task Force                   |
| <b>ILD</b>   | Intelligent load distribution                     |
| <b>IMDG</b>  | in-memory data grid                               |
| <b>IMS</b>   | Information Management System                     |
| <b>IP</b>    | Internet Protocol                                 |
| <b>IPSec</b> | IP Security                                       |
| <b>ISAM</b>  | IBM Security Access Manager                       |
| <b>iSCSI</b> | Internet Small Computer Systems Interface         |

## **J**

|             |                                     |
|-------------|-------------------------------------|
| <b>J2SE</b> | Java Platform, Standard Edition     |
| <b>JAXP</b> | Java API for XML Processing         |
| <b>JDBC</b> | Java Database Connectivity          |
| <b>JFAP</b> | JetStream Formats and Protocols     |
| <b>JKS</b>  | Java Key Store                      |
| <b>JMS</b>  | Java Message Service                |
| <b>JNDI</b> | Java Naming and Directory Interface |
| <b>JRE</b>  | Java runtime environment            |
| <b>JSON</b> | JavaScript Object Notation          |
| <b>JVM</b>  | Java virtual machine                |

## **K**

|           |          |
|-----------|----------|
| <b>KB</b> | kilobyte |
|-----------|----------|

## **L**

|             |                                       |
|-------------|---------------------------------------|
| <b>LAN</b>  | local area network                    |
| <b>LDAP</b> | Lightweight Directory Access Protocol |
| <b>LDIF</b> | LDAP Data Interchange Format          |

---

|             |  |
|-------------|--|
| <b>LED</b>  | light-emitting diode                   |
| <b>LLM</b>  | Low Latency Messaging                  |
| <b>LTPA</b> | Lightweight Third Party Authentication |

## M

|               |   |
|---------------|---|
| <b>MAC</b>    | message authentication code                 |
| <b>Mb</b>     | megabit                                     |
| <b>MB</b>     | megabyte                                    |
| <b>MDB</b>    | message-driven bean                         |
| <b>MFA</b>    | message filter action                       |
| <b>MIB</b>    | Management Information Base                 |
| <b>MIME</b>   | Multipurpose Internet Mail Extensions       |
| <b>MM</b>     | message monitor                             |
| <b>MMXDoS</b> | multiple message XML denial-of-service      |
| <b>MP3</b>    | MPEG-1 or MPEG-2 Audio Layer III            |
| <b>MPGW</b>   | multi-protocol gateway                      |
| <b>MQ</b>     | Message Queue                               |
| <b>MQCSP</b>  | MQ connection security parameter            |
| <b>MQFSH</b>  | MQ front side handler                       |
| <b>MQFTE</b>  | MQ File Transfer Edition                    |
| <b>MQMD</b>   | message queuing message descriptor          |
| <b>MQOD</b>   | message queuing object descriptor           |
| <b>MT</b>     | message type                                |
| <b>MTOM</b>   | Message Transmission Optimization Mechanism |

## N

|                |   |
|----------------|---|
| <b>NAT</b>     | network address translation   |
| <b>NFS</b>     | Network File System   |
| <b>NG</b>      | New Generation  |
| <b>NIC</b>     | network interface card  |
| <b>npm</b>     | node package manager  |
| <b>NSS</b>     | Network Security Services   |
| <b>NSTISSC</b> | National Security Telecommunications and Information Systems Security Committee |
| <b>NTP</b>     | Network Time Protocol   |

## O

|              |  |
|--------------|--|
| <b>OASIS</b> | Organization for the Advancement of Structured Information Standards |
| <b>OAuth</b> | Open standard for Authorization                                      |
| <b>OID</b>   | Object ID  |
| <b>OSI</b>   | Open Systems Interconnection   |
| <b>OTMA</b>  | Open Transaction Management Access                                   |
| <b>OTP</b>   | One-Time Password  |

## P

|             |   |
|-------------|---|
| <b>PAM</b>  | Pluggable Authentication Module                         |
| <b>PC</b>   | personal computer                                       |
| <b>PCF</b>  | Processing Control File                                 |
| <b>PCRE</b> | Perl-compatible regular expressions                     |
| <b>PDF</b>  | Portable Document Format                                |
| <b>PDP</b>  | policy decision point                                   |
| <b>PED</b>  | PIN Entry Device  |
| <b>PEM</b>  | Privacy-Enhanced Mail                                   |
| <b>PEP</b>  | policy enforcement point                                |
| <b>PI</b>   | processing instruction                                  |
| <b>PIN</b>  | personal identification number                          |
| <b>PKCS</b> | Public Key Cryptography Standard                        |
| <b>PKI</b>  | public key infrastructure                               |
| <b>PKIX</b> | Public Key Infrastructure for X.509 Certificates (IETF) |
| <b>PMR</b>  | program maintenance request                             |
| <b>POP</b>  | Post Office Protocol                                    |
| <b>POV</b>  | point of variability                                    |
| <b>POX</b>  | plain old XML   |

## Q

|            |                    |
|------------|--------------------|
| <b>QoS</b> | quality of service |
|------------|--------------------|

## R

|               |  |
|---------------|--|
| <b>RADIUS</b> | Remote Authentication Dial-In User Service |
| <b>RAID</b>   | Redundant Array of Independent Disks       |
| <b>RAM</b>    | random access memory                       |

---

|              |                                       |
|--------------|---------------------------------------|
| <b>RBM</b>   | role-based management                 |
| <b>RDBMS</b> | relational database management system |
| <b>RDN</b>   | relative distinguished name           |
| <b>RDO</b>   | resource definition online            |
| <b>REL</b>   | Rights Expression Language            |
| <b>REQ</b>   | Request                               |
| <b>REST</b>  | Representational State Transfer       |
| <b>RFC</b>   | Request for Comments                  |
| <b>RPC</b>   | Remote Procedure Call                 |
| <b>RPM</b>   | RPM Package Manager, utility in Linux |
| <b>RSA</b>   | Rational Software Architect           |
| <b>RSS</b>   | Really Simple Syndication             |

## S

|              |  |
|--------------|--|
| <b>SAF</b>   | System Authorization Facility  |
| <b>SAML</b>  | Security Assertion Markup Language   |
| <b>SAS</b>   | Serial Attached SCSI   |
| <b>SAX</b>   | Simple API for XML   |
| <b>SCP</b>   | Secure Copy Protocol   |
| <b>SCSI</b>  | Small Computer System Interface  |
| <b>SDK</b>   | software development kit   |
| <b>SFTP</b>  | Secured File Transfer Protocol   |
| <b>SHA1</b>  | Secure Hash Algorithm, Version 1   |
| <b>SIBus</b> | service integration bus  |
| <b>SLA</b>   | service level agreement  |
| <b>SLES</b>  | SUSE Linux Enterprise Server   |
| <b>SLM</b>   | service level management   |
| <b>SLM</b>   | service level monitoring   |
| <b>SMS</b>   | session management server  |
| <b>SMTP</b>  | Simple Mail Transfer Protocol  |
| <b>SNMP</b>  | Simple Network Management Protocol   |
| <b>SOA</b>   | service-oriented architecture  |
| <b>SOAP</b>  | Usage note: SOAP is not an acronym; it is a word in itself (formerly an acronym for Simple Object Access Protocol) |

|               |  |
|---------------|--|
| <b>SOMA</b>   | SOAP management                                    |
| <b>SPNEGO</b> | Simple and Protected GSS-API Negotiation Mechanism |
| <b>SPVC</b>   | self-paced virtual classroom                       |
| <b>SQL</b>    | Structured Query Language                          |
| <b>SSH</b>    | Secure Shell                                       |
| <b>SSL</b>    | Secure Sockets Layer                               |
| <b>SSO</b>    | single sign-on                                     |
| <b>STS</b>    | Security Token Service                             |
| <b>SUSE</b>   | A Linux based operating system                     |
| <b>SwA</b>    | SOAP with Attachments                              |

## **T**

|               |  |
|---------------|--|
| <b>Tcl</b>    | Tool Control Language (often pronounced as “tickle”) |
| <b>TCO</b>    | total cost of ownership                              |
| <b>TCP</b>    | Transmission Control Protocol                        |
| <b>TCP/IP</b> | Transmission Control Protocol/Internet Protocol      |
| <b>TDES</b>   | Triple Data Encryption Standard                      |
| <b>TFIM</b>   | Tivoli Federated Identity Manager                    |
| <b>TIA</b>    | Telecommunications Industry Association              |
| <b>TIBCO</b>  | The Information Bus Company                          |
| <b>TIM</b>    | Tivoli Identity Manager                              |
| <b>TLS</b>    | Transport Layer Security                             |
| <b>TTL</b>    | Time to Live   |

## **U**

|             |   |
|-------------|---|
| <b>UDDI</b> | Universal Description, Discovery, and Integration |
| <b>UDP</b>  | User Datagram Protocol                            |
| <b>UNIX</b> | Uniplexed Information and Computing System        |
| <b>URI</b>  | Uniform Resource Identifier                       |
| <b>URL</b>  | Uniform Resource Locator                          |
| <b>USB</b>  | Universal Serial Bus                              |
| <b>UTC</b>  | Coordinated Universal Time                        |

## **V**

|            |                    |
|------------|--------------------|
| <b>VIP</b> | virtual IP address |
|------------|--------------------|

---

|             |                                    |
|-------------|------------------------------------|
| <b>VM</b>   | virtual machine                    |
| <b>VLAN</b> | virtual local area network         |
| <b>VRRP</b> | Virtual Router Redundancy Protocol |

**W**

|                 |   |
|-----------------|---|
| <b>W3C</b>      | World Wide Web Consortium                 |
| <b>WAFW</b>     | web application firewall                  |
| <b>WAMC</b>     | WebSphere Appliance Management Center     |
| <b>WML</b>      | Wireless Markup Language                  |
| <b>WS</b>       | web services                              |
| <b>WSDL</b>     | Web Services Description Language         |
| <b>WSDM</b>     | Web Services Distributed Management       |
| <b>WSP</b>      | web service proxy                         |
| <b>WS-Proxy</b> | web service proxy                         |
| <b>WSRR</b>     | WebSphere Service Registry and Repository |
| <b>WTX</b>      | IBM WebSphere Transformation Extender     |
| <b>WWW</b>      | World Wide Web                            |

**X**

|               |   |
|---------------|---|
| <b>XA</b>     | Extended Architecture                         |
| <b>XACML</b>  | Extensible Access Control Markup Language     |
| <b>XCF</b>    | cross-system coupling facility                |
| <b>XDoS</b>   | XML denial of service                         |
| <b>XHTML</b>  | Extensible Hypertext Markup Language          |
| <b>XMI</b>    | XML Management Interface                      |
| <b>XML</b>    | Extensible Markup Language                    |
| <b>XMLDS</b>  | XML digital signature                         |
| <b>XMLFW</b>  | XML firewall                                  |
| <b>XML-PI</b> | XML processing instructions                   |
| <b>XPath</b>  | XML Path Language                             |
| <b>XSD</b>    | XML Schema Definition                         |
| <b>XSL</b>    | Extensible Stylesheet Language                |
| <b>XSLT</b>   | 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/IBMTrainEN>
- **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/ites.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 going to 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](https://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](http://facebook.com/ibmtraining)

- **YouTube:**

- Go to the IBM Training YouTube channel to learn about IBM training programs and courses.
- Find IBM Training at: [youtube.com/IBMTutorial](http://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**  
®