



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

(Course code WE401)

Student Notebook

ERC 3.0

Authorized

IBM | Training

WebSphere Education

Trademarks

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

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

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

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

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

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

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

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

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

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

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

March 2014 edition

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

Contents

| | |
|--|--------------|
| Trademarks | xvii |
| Course description | xxi |
| Agenda | xxiii |
| Unit 1. DataPower administration overview. | 1-1 |
| Unit objectives | 1-2 |
| Developing on the appliance | 1-3 |
| DataPower SOA appliance administration | 1-4 |
| WebGUI web administration application | 1-6 |
| Administration by using the web browser | 1-7 |
| Navigation bar categories | 1-8 |
| System control features (1 of 3) | 1-9 |
| System control features (2 of 3) | 1-11 |
| System control features (3 of 3) | 1-13 |
| File management | 1-14 |
| File directories for configuration | 1-15 |
| File directories for security | 1-16 |
| File directories for logging | 1-17 |
| Administrative access control | 1-18 |
| Create an application domain | 1-19 |
| Application domain: Configuration tab | 1-21 |
| Configuration Checkpoints | 1-22 |
| View application domain status | 1-23 |
| Create a user account and a user group | 1-24 |
| Manage user group details | 1-25 |
| Manage user account details | 1-27 |
| Export the system configuration | 1-28 |
| Import a system configuration | 1-29 |
| Saving configuration changes | 1-30 |
| Administration by using the command-line interface | 1-31 |
| First boot after unpacking | 1-32 |
| Initial CLI login screen | 1-33 |
| Quick initial configuration procedure | 1-34 |
| Retrieve system information by using the CLI | 1-35 |
| Administration by using SOAP: SOMA and AMP | 1-36 |
| SOMA: Create a user account | 1-37 |
| SOMA: Account creation response | 1-38 |
| AMP: Appliance information request | 1-39 |
| AMP: Appliance information response | 1-40 |
| Management interface summary | 1-41 |
| Unit summary | 1-43 |
| Checkpoint questions | 1-44 |

| | |
|---|------------|
| Checkpoint answers | 1-45 |
| Exercise | 1-46 |
| Exercise objectives | 1-47 |
| Unit 2. Introduction to XSL transformations..... | 2-1 |
| Unit objectives | 2-2 |
| Three parts of Extensible Stylesheet Language (XSL) | 2-3 |
| XSL Transformations (XSLT) overview | 2-4 |
| The XSLT process | 2-5 |
| What is XPath? | 2-6 |
| Example XPath expressions | 2-7 |
| XPath current context | 2-8 |
| XPath step syntax | 2-9 |
| XPath address notation | 2-10 |
| Example: XPath absolute addressing | 2-11 |
| Example: XPath relative addressing | 2-12 |
| Anatomy of an XSL style sheet | 2-13 |
| The <xsl:template> element | 2-14 |
| The <xsl:apply-templates> element | 2-15 |
| The <xsl:value-of> element | 2-16 |
| XSLT style sheet elements to generate output | 2-17 |
| XML input as a tree | 2-18 |
| Wanted HTML output | 2-19 |
| XML to HTML (1 of 4) | 2-20 |
| XML to HTML (2 of 4) | 2-21 |
| XML to HTML (3 of 4) | 2-22 |
| XML to HTML (4 of 4) | 2-23 |
| XSL style sheet control elements | 2-24 |
| The <xsl:for-each> element | 2-25 |
| The <xsl:if> element | 2-26 |
| The <xsl:choose> element (1 of 2) | 2-27 |
| The <xsl:choose> element (2 of 2) | 2-28 |
| Elements to generate output (XML to XML) | 2-29 |
| The <xsl:element> element | 2-30 |
| The <xsl:attribute> element | 2-31 |
| Using custom style sheets | 2-32 |
| How to develop style sheets with DataPower extensions | 2-33 |
| XSLT variables | 2-34 |
| DataPower variables | 2-35 |
| DataPower variable scopes | 2-36 |
| Style sheet using DataPower extension functions | 2-38 |
| Testing style sheets with DataPower content | 2-39 |
| Interoperability Test Service example | 2-40 |
| Interoperability Test Service example | 2-41 |
| Unit summary | 2-42 |
| Checkpoint questions | 2-43 |
| Checkpoint answers | 2-44 |
| Exercise | 2-45 |

| | |
|--|------------|
| Exercise objectives | 2-46 |
| Exercise overview | 2-47 |
| Unit 3. DataPower services overview | 3-1 |
| Unit objectives | 3-2 |
| Services in a DataPower appliance | 3-3 |
| Front sides and back sides, and sideways | 3-4 |
| Services available on the DataPower appliance | 3-5 |
| XSL proxy service | 3-6 |
| XML firewall service | 3-7 |
| Multi-protocol gateway service | 3-8 |
| Web service proxy service | 3-9 |
| Web application firewall service | 3-11 |
| Other services | 3-12 |
| DataPower services feature hierarchy | 3-13 |
| Unit summary | 3-15 |
| Checkpoint questions | 3-16 |
| Checkpoint answers | 3-17 |
| Unit 4. Structure of a service | 4-1 |
| Unit objectives | 4-2 |
| Object-based configuration | 4-3 |
| Approach to configuring objects | 4-4 |
| Basic architectural model | 4-5 |
| Front side access | 4-6 |
| Connection to the back side | 4-7 |
| Message processing phases | 4-8 |
| Configuring a service policy (processing policy) | 4-10 |
| Policy editor | 4-11 |
| Configuring rules within a service policy | 4-12 |
| Processing rules | 4-13 |
| Match action | 4-14 |
| Processing actions | 4-15 |
| Processing actions | 4-16 |
| More processing actions | 4-18 |
| Multi-step processing rules | 4-19 |
| Pre-defined context variables | 4-20 |
| Validate action | 4-21 |
| Transform action | 4-22 |
| Filter action | 4-23 |
| Filter action: Replay attack | 4-24 |
| Content-based routing | 4-25 |
| Route action configuration | 4-26 |
| Style sheet programming with dynamic routing | 4-27 |
| Results action | 4-29 |
| Results asynchronous and multi-way results mode | 4-30 |
| Service settings | 4-32 |
| Service types | 4-33 |

| | |
|---|----------------|
| URL rewriting | 4-34 |
| XML Manager | 4-36 |
| Default XML Manager configuration | 4-37 |
| XML parser limits | 4-38 |
| Exporting a service configuration | 4-40 |
| Cloning an XML firewall configuration | 4-41 |
| Troubleshooting a service configuration | 4-42 |
| Unit summary | 4-43 |
| Checkpoint questions | 4-44 |
| Checkpoint answers | 4-45 |
| Exercise | 4-46 |
| Exercise objectives | 4-47 |
| Exercise overview | 4-48 |
| Unit 5. Multi-protocol gateway service | 5-1 |
| Unit objectives | 5-2 |
| What is a multi-protocol gateway? | 5-3 |
| Protocol handlers at a glance (1 of 2) | 5-4 |
| Protocol handlers at a glance (2 of 2) | 5-5 |
| Front side protocol handlers | 5-6 |
| Static back-end gateway | 5-7 |
| Dynamic back-end gateway | 5-8 |
| Multi-protocol gateway and XML firewall compared | 5-9 |
| Multi-protocol gateway editor | 5-10 |
| Scenario 1: Provide HTTP and HTTPS access | 5-12 |
| Step 1: Configure the back side transport | 5-13 |
| Step 2: Create a document processing rule | 5-14 |
| Step 3: Create the front side handlers | 5-15 |
| Step 4: Configure the front side handler | 5-16 |
| Step 5: Configure the SSL proxy profile | 5-17 |
| Scenario 2: Dynamic back-end service | 5-18 |
| Step 1: Configure the back-end transport | 5-19 |
| Sample service that targets a style sheet | 5-20 |
| Scenario 3: Provide WebSphere MQ access | 5-21 |
| Scenario 4: Provide WebSphere JMS access | 5-22 |
| Scenario 5: Provide IMS Connect access | 5-23 |
| Scenario 6: Provide a RESTful interface | 5-24 |
| REST support in DataPower | 5-25 |
| Comparing services | 5-26 |
| Unit summary | 5-27 |
| Checkpoint questions | 5-28 |
| Checkpoint answers | 5-29 |
| Exercise | 5-30 |
| Exercise objectives | 5-31 |
| Exercise overview | 5-32 |
| Unit 6. Problem determination tools | 6-1 |
| Unit objectives | 6-2 |

| | |
|--|------------|
| Problem determination tools | 6-3 |
| Common problem determination tools | 6-4 |
| Appliance status information | 6-5 |
| Troubleshooting panel | 6-6 |
| Troubleshooting panel | 6-7 |
| Troubleshooting: Networking | 6-8 |
| Troubleshooting: Packet capture | 6-9 |
| Troubleshooting: Logging | 6-10 |
| Troubleshooting: System log | 6-11 |
| Filtering system log | 6-13 |
| Troubleshooting: Generate Log Event | 6-14 |
| Troubleshooting: Reporting | 6-15 |
| Troubleshooting: Advanced | 6-16 |
| Troubleshooting: XML File Capture | 6-17 |
| Troubleshooting: Send a test message | 6-18 |
| Troubleshooting: Multi-step probe | 6-19 |
| Troubleshooting: Enabling the multi-step probe | 6-20 |
| Multi-step probe window | 6-21 |
| Multi-step probe content | 6-22 |
| Problem determination with cURL | 6-23 |
| Communicating with DataPower support | 6-24 |
| Topic summary | 6-25 |
| Log targets | 6-26 |
| Logging basics | 6-27 |
| Available log levels | 6-28 |
| Log targets | 6-29 |
| Log target configuration | 6-30 |
| Ten-log target types | 6-31 |
| Event filters | 6-32 |
| Object filters | 6-33 |
| Event subscriptions | 6-34 |
| Log action | 6-35 |
| Topic summary | 6-36 |
| Unit summary | 6-37 |
| Checkpoint questions | 6-38 |
| Checkpoint answers | 6-39 |
| Exercise | 6-40 |
| Exercise objectives | 6-41 |
| Exercise overview | 6-42 |
| | |
| Unit 7. Handling errors in a service policy | 7-1 |
| Unit objectives | 7-2 |
| Error handling constructs | 7-3 |
| Configure an On Error action | 7-4 |
| Creating an error rule | 7-5 |
| Configure Transform action in error rule | 7-6 |
| Style sheet programming that use error variables | 7-7 |
| Example custom error style sheet | 7-8 |

| | |
|--|----------------|
| Error rule versus On Error action | 7-9 |
| Unit summary | 7-10 |
| Checkpoint questions | 7-11 |
| Checkpoint answers | 7-12 |
| Exercise | 7-13 |
| Exercise objectives | 7-14 |
| Exercise overview | 7-15 |
| Unit 8. DataPower cryptographic tools | 8-1 |
| Unit objectives | 8-2 |
| Security problems | 8-3 |
| Security problem 1: Message confidentiality | 8-4 |
| Symmetric key encryption | 8-5 |
| Asymmetric key encryption | 8-6 |
| Security problem 2: Message integrity | 8-7 |
| Security problem 3: Nonrepudiation | 8-8 |
| Digital signature | 8-9 |
| Security problems: Solved | 8-10 |
| Digital certificates | 8-11 |
| Distribution problem | 8-12 |
| DataPower cryptographic tools | 8-13 |
| Generating cryptographic (asymmetric) keys on the DataPower appliance (1 of 2) | 8-14 |
| Generating crypto (asymmetric) keys on board (2 of 2) | 8-15 |
| Download keys from temporary storage | 8-16 |
| Keys and certificates are objects | 8-17 |
| Crypto shared secret (symmetric) key | 8-18 |
| Crypto certificate | 8-19 |
| Certificates exist in a trust chain | 8-20 |
| Crypto identification credential | 8-21 |
| Crypto validation credential | 8-22 |
| Crypto profile | 8-24 |
| Import and export crypto objects | 8-25 |
| Uploading keys | 8-26 |
| Java keytool command | 8-27 |
| Certificates can expire or get revoked | 8-28 |
| Certificate revocation list (CRL) retrieval | 8-29 |
| Crypto certification monitor | 8-30 |
| Hardware security module (HSM) | 8-31 |
| Unit summary | 8-32 |
| Checkpoint questions | 8-33 |
| Checkpoint answers | 8-34 |
| Exercise | 8-35 |
| Exercise objectives | 8-36 |
| Exercise overview | 8-37 |
| Unit 9. Securing connections by using SSL | 9-1 |
| Unit objectives | 9-2 |

| | |
|---|-------------|
| What is SSL | 9-3 |
| SSL features | 9-4 |
| SSL terminology | 9-5 |
| SSL handshake | 9-6 |
| SSL handshake: Client hello | 9-7 |
| SSL handshake: Server hello | 9-8 |
| SSL handshake: Verify server certificate | 9-9 |
| SSL handshake: Client key exchange | 9-10 |
| SSL handshake: Reply with secret key | 9-11 |
| Securing the connection by using the SSL handshake | 9-12 |
| DataPower support for SSL | 9-13 |
| SSL proxy profile | 9-14 |
| SSL proxy profile: Crypto objects relationship | 9-15 |
| Securing connections from client to appliance | 9-16 |
| Step 1: Appliance supplies cryptographic certificate | 9-17 |
| Step 2: SSL server crypto profile: non-FSH | 9-18 |
| Step 2: SSL server crypto profile: FSH | 9-19 |
| If you do not have an SSL server crypto profile... | 9-20 |
| Securing the connection from appliance to external application server | 9-21 |
| Step 1: Appliance validates presented certificate | 9-22 |
| Step 2: Configuring an SSL client crypto profile | 9-23 |
| If you do not have an SSL client crypto profile... | 9-24 |
| SSL Proxy Profile list | 9-25 |
| User agent | 9-26 |
| Configuring a user agent | 9-27 |
| Creating a user agent configuration | 9-28 |
| Unit summary | 9-29 |
| Checkpoint questions | 9-30 |
| Checkpoint answers | 9-31 |
| Exercise | 9-32 |
| Exercise objectives | 9-33 |
| Exercise overview | 9-34 |
| Unit 10. XML threat protection | 10-1 |
| Unit objectives | 10-2 |
| What are the security concerns? | 10-3 |
| Traditional systems and exposure | 10-4 |
| Addressing the security concerns | 10-5 |
| Three high-level deployment patterns | 10-6 |
| Four types of XML attacks | 10-7 |
| XML denial of service (XDoS): Single-message attacks | 10-8 |
| XML denial of service (XDoS): Multiple-message attacks | 10-9 |
| Unauthorized access attacks | 10-10 |
| Data integrity and confidentiality attacks | 10-11 |
| System compromise attacks | 10-12 |
| XML parser limits | 10-13 |
| XML threat protection | 10-15 |
| XML threat protection: Single message XDoS | 10-16 |

| | |
|---|-----------------|
| XML threat protection: Multiple message XDoS | 10-18 |
| XML threat protection: Protocol threats | 10-20 |
| XML threat protection: XML virus | 10-21 |
| XML threat protection: Dictionary attack | 10-23 |
| Message tampering | 10-25 |
| SQL injection attack | 10-26 |
| SQL injection attack protection | 10-28 |
| Unit summary | 10-29 |
| Checkpoint questions | 10-30 |
| Checkpoint answers | 10-31 |
| Exercise | 10-32 |
| Exercise objectives | 10-33 |
| Exercise overview | 10-34 |
| Unit 11. Web service proxy service | 11-1 |
| Unit objectives | 11-2 |
| Web service proxy overview | 11-3 |
| Web service virtualization | 11-4 |
| Web service proxy benefits | 11-5 |
| Web service proxy configuration tabs (1 of 2) | 11-6 |
| Web service proxy configuration tabs (2 of 2) | 11-7 |
| Web service proxy basic configuration steps | 11-8 |
| Step 1: Obtain WSDL document | 11-9 |
| WSDL structure | 11-10 |
| Step 2: Creating a web service proxy | 11-11 |
| An alternative: Web service proxy object editor | 11-12 |
| Step 3: Add WSDL document to web service proxy | 11-13 |
| Step 4: Configure WSDL endpoint | 11-14 |
| Step 5: Configure local endpoint handler | 11-16 |
| Step 6: Add the WSDL to the service | 11-17 |
| Initial WSDL completed | 11-18 |
| Other ways to get a WSDL | 11-19 |
| View WSDL services | 11-20 |
| Retrieve the client WSDL from the service | 11-21 |
| Modifying the location in the client WSDL | 11-22 |
| Step 7: Configuring web service proxy policy (optional) | 11-23 |
| Configure web service proxy policy rule | 11-24 |
| Adding a rule | 11-25 |
| Default validation (user policies) | 11-26 |
| Create reusable rule | 11-27 |
| Advanced web service proxy configuration | 11-28 |
| WS-Policy | 11-29 |
| Conformance policy | 11-30 |
| Conformance policy object | 11-31 |
| Service priority | 11-32 |
| Proxy settings (1 of 4) | 11-33 |
| Proxy settings (2 of 4) | 11-34 |
| Proxy settings (3 of 4) | 11-36 |

| | |
|--|-------------|
| Proxy settings (4 of 4) | 11-37 |
| Web service proxy: SLM Policy tab | 11-38 |
| WSDL cache policy | 11-39 |
| Troubleshooting a web service proxy | 11-40 |
| Unit summary | 11-41 |
| Checkpoint questions | 11-42 |
| Checkpoint answers | 11-43 |
| Exercise 9 | 11-44 |
| Exercise objectives | 11-45 |
| Exercise overview | 11-46 |
| Unit 12. XML and web services security overview | 12-1 |
| Unit objectives | 12-2 |
| Review of basic security terminology | 12-3 |
| Web services security | 12-5 |
| Components of WS-Security | 12-6 |
| Specifying security in SOAP messages | 12-7 |
| Scenario 1: Ensure confidentiality with XML encryption | 12-8 |
| XML encryption and WS-Security | 12-9 |
| DataPower support for XML encryption | 12-10 |
| Encrypt action | 12-11 |
| Decrypt action | 12-13 |
| Field-level encryption and decryption | 12-14 |
| XPath tool | 12-15 |
| Sample encrypted SOAP message | 12-16 |
| Scenario 2: Ensure integrity with XML signatures | 12-17 |
| DataPower support for XML signature | 12-19 |
| Sign action | 12-20 |
| Verify action | 12-22 |
| Verify action: Advanced tab | 12-23 |
| Field-level message signature and verification | 12-24 |
| Sample signed SOAP message | 12-25 |
| Summary of security and keys | 12-26 |
| Unit summary | 12-27 |
| Checkpoint questions | 12-28 |
| Checkpoint answers | 12-29 |
| Exercise 10 | 12-30 |
| Exercise objectives | 12-31 |
| Exercise overview | 12-32 |
| Unit 13. Authentication, authorization, and auditing (AAA)..... | 13-1 |
| Unit objectives | 13-2 |
| Authentication, authorization, and auditing | 13-3 |
| Authentication and authorization framework | 13-4 |
| AAA action and access control policy | 13-6 |
| How to define an access control policy (1 of 2) | 13-7 |
| How to define an access control policy (2 of 2) | 13-8 |
| Access control policy processing | 13-9 |

| | |
|--|-------------|
| Scenario 1: Authorize authenticated clients | 13-10 |
| Scenario 1: Sample SOAP request message | 13-11 |
| Scenario 1: Identify the client | 13-12 |
| Scenario 1: Authorize access to resources | 13-13 |
| Scenario 2: Security token conversion | 13-14 |
| Scenario 2: Sample HTTP request message | 13-15 |
| Scenario 2: Identify the client | 13-16 |
| Scenario 2: Authorize access to resources | 13-17 |
| Scenario 3: Multiple identity extraction methods | 13-18 |
| Scenario 3: Identify the client | 13-19 |
| Scenario 3: Authorize access to resources | 13-20 |
| Internal access control resources | 13-21 |
| AAA XML file | 13-22 |
| Example AAA XML file | 13-23 |
| Lightweight Third Party Authentication | 13-24 |
| External access control resource | 13-25 |
| Lightweight Directory Access Protocol | 13-26 |
| Security Assertion Markup Language | 13-27 |
| Types of SAML assertions | 13-28 |
| Scenario 4: Authorize valid SAML assertions | 13-29 |
| Scenario 4: SAML authentication statement (1 of 2) | 13-30 |
| Scenario 4: SAML authentication statement (2 of 2) | 13-31 |
| Scenario 4: SAML attribute statement | 13-32 |
| Scenario 4: Identify the client | 13-33 |
| Scenario 4: Authorize access to resources | 13-34 |
| Scenario 4: Match SAML attributes | 13-35 |
| Access control policy by SAML information | 13-36 |
| Unit summary | 13-37 |
| Checkpoint questions | 13-38 |
| Checkpoint answers | 13-39 |
| Exercise 11 | 13-40 |
| Exercise objectives | 13-41 |
| Exercise overview | 13-42 |
| Unit 14. Monitoring objects | 14-1 |
| Unit objectives | 14-2 |
| Message monitors | 14-3 |
| Monitor objects | 14-4 |
| Defining monitor objects | 14-5 |
| Step 1: Specifying particular traffic to monitor | 14-6 |
| Step 1: Matching on HTTP headers | 14-7 |
| Step 2: Message type configuration | 14-8 |
| Step 3: Message Filter Action configuration | 14-9 |
| Step 4C: Message Count Monitor configuration | 14-10 |
| Step 4C: Thresholds/Filters for count monitor | 14-11 |
| Step 4D: Message Duration Monitor configuration | 14-12 |
| Step 4D: The transaction life cycle | 14-13 |
| Step 4D: Thresholds/Filters for duration monitor | 14-14 |

| | |
|---|-------------|
| Step 5: Service-monitor association example | 14-15 |
| Other types of monitors | 14-16 |
| Other types of monitors: Service level monitors | 14-17 |
| Other types of monitors: Web services monitors | 14-18 |
| Monitor types that are supported by a service | 14-19 |
| Unit summary | 14-21 |
| Checkpoint questions | 14-22 |
| Checkpoint answers | 14-23 |
| Exercise 12 | 14-24 |
| Exercise objectives | 14-25 |
| Unit 15. Configuring LDAP by using AAA | 15-1 |
| Unit objectives | 15-2 |
| External access control resource | 15-3 |
| Lightweight Directory Access Protocol | 15-4 |
| Directory services | 15-5 |
| Directories | 15-6 |
| Common LDAP attributes | 15-7 |
| Directory services structure | 15-8 |
| LDAP operations | 15-9 |
| LDAP URL | 15-10 |
| Directory services implementations | 15-11 |
| Example scenario | 15-12 |
| Use LDAP to authenticate the client | 15-13 |
| Use LDAP to authorize the client | 15-15 |
| Unit summary | 15-16 |
| Checkpoint questions | 15-17 |
| Checkpoint answers | 15-18 |
| Exercise 13 | 15-19 |
| Exercise objectives | 15-20 |
| Exercise overview | 15-21 |
| Unit 16. OAuth overview and DataPower implementation | 16-1 |
| Unit objectives | 16-2 |
| What is OAuth? | 16-3 |
| Why OAuth? | 16-4 |
| Before OAuth | 16-5 |
| 3-legged OAuth is the pattern OAuth was designed for | 16-6 |
| 2-legged OAuth is a traditional client/server pattern | 16-8 |
| OAuth support in DataPower (version 5.0+) (1 of 2) | 16-9 |
| OAuth support in DataPower (version 5.0+) (2 of 2) | 16-10 |
| Web Token Service | 16-11 |
| AAA: Extract identity and extract resource | 16-12 |
| Configure OAuth client group | 16-13 |
| Configure OAuth Client (1 of 2) | 16-15 |
| Configure OAuth Client (2 of 2) | 16-17 |
| Unit summary | 16-20 |
| Checkpoint questions | 16-21 |

| | |
|---|-------------|
| Checkpoint answers | 16-22 |
| Unit 17. Service level monitoring | 17-1 |
| Unit objectives | 17-2 |
| Service level monitoring (SLM) in DataPower | 17-3 |
| The pieces of SLM (1 of 2) | 17-4 |
| Approaches to define SLM policies | 17-5 |
| Approach 1: Add an SLM action to a request rule | 17-6 |
| The pieces of SLM (2 of 2) | 17-7 |
| The SLM credential class | 17-8 |
| The SLM resource class | 17-9 |
| SLM resource class example | 17-10 |
| The SLM schedule | 17-11 |
| The SLM action | 17-12 |
| SLM statement (1 of 2) | 17-14 |
| SLM statement (2 of 2) | 17-15 |
| SLM policy: Main tab | 17-17 |
| SLM policy: Statements tab | 17-18 |
| Getting SLM statements into the Statement list | 17-19 |
| Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2) | 17-20 |
| Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2) | 17-21 |
| SLM Policy tab: Graphs | 17-22 |
| SLM action granularity | 17-23 |
| Unit summary | 17-24 |
| Checkpoint questions | 17-25 |
| Checkpoint answers | 17-26 |
| Exercise 14 | 17-27 |
| Exercise objectives | 17-28 |
| Unit 18. Integration with WebSphere MQ | 18-1 |
| Unit objectives | 18-2 |
| WebSphere MQ fundamentals | 18-3 |
| WebSphere MQ message | 18-5 |
| Transactions | 18-6 |
| DataPower support for WebSphere MQ | 18-8 |
| Provide WebSphere MQ access | 18-9 |
| Step 1: Create a WebSphere MQ queue manager (1 of 2) | 18-10 |
| Step 1: Create a WebSphere MQ queue manager (2 of 2) | 18-11 |
| Step 1: Use SSL in mutual authentication mode | 18-12 |
| Step 2: Add a WebSphere MQ front side handler | 18-13 |
| Step 3: Configure a WebSphere MQ back-end transport | 18-14 |
| Publish/subscribe: WebSphere MQ front side handler support | 18-16 |
| Publish/subscribe: WebSphere MQ back-end transport support | 18-17 |
| Message properties | 18-18 |
| Ordered processing of WebSphere MQ messages | 18-19 |
| Controlling backout of WebSphere MQ messages | 18-21 |
| Decision tree for the backout settings | 18-22 |
| WebSphere MQ header action in service policy | 18-23 |

| | |
|--|-------------|
| Typical uses of a WebSphere MQ header action | 18-24 |
| Transactions and WebSphere MQ | 18-25 |
| WebSphere MQ front side transactions | 18-26 |
| WebSphere MQ back side transactions | 18-27 |
| WebSphere MQ DataPower URL | 18-28 |
| WebSphere MQ queue manager Group object | 18-29 |
| Unit summary | 18-30 |
| Checkpoint questions | 18-31 |
| Checkpoint answers | 18-32 |
| Exercise 15 | 18-33 |
| Exercise objectives | 18-34 |
| Exercise overview | 18-35 |
| Unit 19. DataPower and WebSphere JMS | 19-1 |
| Unit objectives | 19-2 |
| Messaging middleware | 19-3 |
| DataPower support of messaging middleware | 19-4 |
| Java Message Service (JMS) | 19-5 |
| JMS models | 19-6 |
| WebSphere service integration bus (SIBus) | 19-7 |
| JMS queue resources on SIBus | 19-8 |
| JMS topic resources on SIBus | 19-9 |
| WebSphere JMS support | 19-10 |
| DataPower and WebSphere JMS interaction | 19-11 |
| WebSphere JMS object: Main (1 of 2): Messaging bus | 19-12 |
| WebSphere JMS object: Main (2 of 2): Optional settings | 19-13 |
| WebSphere JMS object: WebSphere JMS Endpoint | 19-15 |
| Communicating to WebSphere JMS | 19-16 |
| WebSphere JMS Front Side Handler | 19-17 |
| WebSphere JMS back-end URL | 19-18 |
| TIBCO EMS support | 19-19 |
| Ordered processing of JMS messages | 19-20 |
| Unit summary | 19-22 |
| Checkpoint questions | 19-23 |
| Checkpoint answers | 19-24 |
| Unit 20. Course summary | 20-1 |
| Unit objectives | 20-2 |
| Course learning objectives | 20-3 |
| Course review (1 of 3) | 20-4 |
| Course review (2 of 3) | 20-5 |
| Course review (3 of 3) | 20-6 |
| DataPower services feature hierarchy | 20-7 |
| Class evaluation | 20-9 |
| Lab exercise solutions | 20-10 |
| To learn more on this subject | 20-11 |
| Additional DataPower education | 20-12 |
| References | 20-13 |

| | |
|--|------------|
| Unit summary | 20-14 |
| Appendix A. Web application firewall service. | A-1 |
| Glossary of abbreviations and acronyms | X-1 |

Trademarks

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

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

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

Approach®

DB™

Domino®

RACF®

Redbooks®

WebSphere®

BladeCenter®

DB2®

IMS™

Rational®

Tivoli®

z/OS®

DataPower®

developerWorks®

Lotus®

RDN®

U®

zSeries®

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

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

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

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

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

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

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

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

Course description

Accelerate, Secure and Integrate with WebSphere DataPower SOA Appliances V5

Duration: 5 days

Purpose

In this 5-day, instructor-led course, you learn the fundamental skills that are required to implement IBM WebSphere DataPower SOA Appliances with firmware version 5.0.

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

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

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

Audience

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

Prerequisites

Before taking this course, you should be familiar with:

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

- Web service fundamentals and the web services security specifications

You should also successfully complete course VW406, Technical Introduction to WebSphere DataPower V5, to gain basic DataPower knowledge that you need in this course.

Objectives

After completing this course, students should be able to:

- Describe how WebSphere DataPower SOA Appliances are configured, including the role of XSL Transformations (XSLT)
- Configure a DataPower service to protect against a new class of XML-based threats
- Create a web service proxy to virtualize web service applications
- Implement web services security
- Create and configure cryptographic objects
- Configure Secure Sockets Layer (SSL) to and from WebSphere DataPower SOA Appliances
- Configure a multi-protocol gateway (MPGW) to handle multiple protocols for a single service
- Configure a service level monitoring (SLM) policy to handle service processing violations
- Enforce service level policies to manage traffic to and from WebSphere DataPower SOA Appliances
- Configure support for IBM WebSphere MQ and WebSphere Java Message Service (JMS)
- Use logs and probes to troubleshoot services
- Handle errors in service policies

Agenda

Day 1

- Course introduction
- Unit 1: DataPower administration overview
- Exercise 1: Exercise setup
- Unit 2: Introduction to XSL transformations
- Exercise 2: Creating XSL transformations
- Unit 3: DataPower services overview
- Unit 4: Structure of a service
- Exercise 3: Creating a simple XML firewall
- Unit 5: Multi-protocol gateway service

Day 2

- Unit 6: Problem determination tools
- Exercise 4: Creating an advanced multi-protocol gateway
- Unit 7: Handling errors in a service policy
- Exercise 5: Adding error handling to a service policy
- Unit 8: DataPower cryptographic tools
- Exercise 6: Creating cryptographic objects
- Unit 9: Using SSL to secure connections
- Exercise 7: Using SSL to secure connections

Day 3

- Unit 10: XML threat protection
- Exercise 8: Protecting against XML threats
- Unit 11: Web service proxy service
- Exercise 9: Configuring a web service proxy
- Unit 12: XML and web services security overview
- Exercise 10: Web service encryption and digital signatures
- Unit 13: Authentication, authorization, and auditing (AAA)
- Exercise 11: Web service authentication and authorization

Day 4

- Unit 14: Monitoring objects
- Exercise 12: Creating message counter monitors for a AAA policy
- Unit 15: Configuring LDAP by using AAA
- Exercise 13: Creating a AAA policy by using LDAP
- Unit 16: OAuth overview and DataPower implementation
- Unit 17: Service level monitoring
- Exercise 14: Implementing an SLM monitor in a web service proxy

Day 5

- Unit 18: Integration with WebSphere MQ
- Exercise 15: Configuring a multi-protocol gateway service with WebSphere MQ
- Unit 19: DataPower and WebSphere JMS
- Unit 20: Course summary

Appendices

- Appendix Unit A: Web application firewall service
- Appendix Exercise A: Creating a firewall and HTTP proxy for a web application
- Appendix Exercise B: Configuring WebSphere JMS

Unit 1. DataPower administration overview

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

- Checkpoint
- Exercise 1: Exercise setup

Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 1-1. Unit objectives

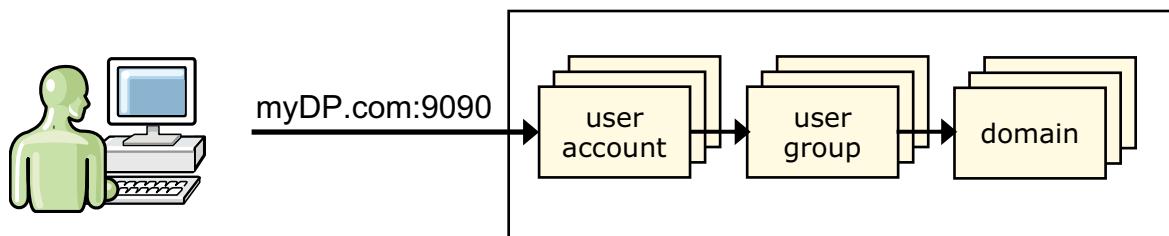
WE4013.0

Notes:

Developing on the appliance

To develop services on the appliance, you need

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



“Administration” defines all of these resources

© Copyright IBM Corporation 2013

Figure 1-2. Developing on the appliance

WE4013.0

Notes:

DataPower SOA appliance administration

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



Serial (one port)

Command-line interface



Ethernet (multiple ports)

WebGUI web application
Command-line interface
▪ (Telnet or SSH)
XML Management web service



cKVM
Command-line interface



Serial-over-LAN (one port)
Command-line interface

© Copyright IBM Corporation 2013

Figure 1-3. DataPower SOA appliance administration

WE4013.0

Notes:

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

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

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

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

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



WebGUI web administration application

- The WebGUI web application allows administrators to configure and troubleshoot the DataPower SOA appliance
 - The WebGUI application must be activated through the command-line interface before its first use
 - Role-based management restricts access to predefined administrators
 - Same web interface that is used for service development
- Navigate to the network address and port that are assigned to the WebGUI application
 - The default port for the WebGUI application is 9090
 - Requires HTTPS for both administrative tasks and development work



© Copyright IBM Corporation 2013

Figure 1-4. WebGUI web administration application

WE4013.0

Notes:

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

The officially supported browsers for firmware 5.0.0 are:

- Microsoft Internet Explorer, version 9 in compatibility mode
- Microsoft Internet Explorer, version 8
- Mozilla Firefox, versions 3.5 and 3.6

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

Figure 1-5. Administration by using the web browser

WE4013.0

Notes:

1. The navigation bar provides access to configuration or management options.
2. The Control Panel allows quick access to common administration functions. The services section allows you to create or modify the primary DataPower services.
3. The monitoring and troubleshooting section provides a view of the DataPower SOA appliance status, traffic, and load.
4. The files and administration section manages the configuration files, access levels, and cryptographic keys and certificates on the appliance.

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



WebSphere Education

Navigation bar categories

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

© Copyright IBM Corporation 2013

Figure 1-6. Navigation bar categories

WE4013.0

Notes:

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



System control features (1 of 3)

The screenshot shows the 'Control Panel' interface on the left and the 'System Control' page on the right.

- Control Panel:**
 - Search bar
 - Navigation tree: Status, Services, Network, Administration (selected), Main, File Management, System Control (highlighted with a red box and circled by a yellow circle labeled 3).
 - Firmware: X152.5.0.0.1
 - Build: 215672
 - IBM Web Sphere DataPower
 - Copyright IBM Corporation 1999-2012
 - [View License Agreement](#)
- System Control Page:**
 - Set Time and Date:** Shows date (2012-10-03) and time (16:28:37). A yellow circle labeled 3 is over the 'Set Time and Date' button.
 - Boot Image:** Shows a checkbox for accepting license agreements (unchecked) and a 'Firmware File' section with upload, fetch, edit, and view buttons. A yellow circle labeled 4 is over the checkbox.
 - Firmware Roll-Back:** Shows a 'Firmware Roll-Back' button. A yellow circle labeled 5 is over this button.
 - Select Configuration:** Shows a 'Configuration File' section with upload, fetch, edit, and view buttons. A yellow circle labeled 6 is over the 'Select Configuration' button.

© Copyright IBM Corporation 2013

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

WE4013.0

Notes:

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

1. Access the system control page through the Administration section of the navigation bar.
2. Alternatively, select the System Control icon in the Control Panel to open the same page.
3. Use the time and date features to set the current time in your locale. To modify the time zone, select **Administration > Device > Time Settings** from the navigation bar. The DataPower SOA appliance sets the clock in Coordinated Universal Time (UTC). You can also define a reference to Network Time Protocol (NTP) server instead.
4. The Boot Image feature allows you to upgrade the system to a newer firmware level. Use the Upload function to copy a new firmware image onto the DataPower appliance.

Once it is complete, click **Boot Image** to restart the DataPower appliance with the new firmware.

5. If you encounter problems when using a new firmware level, click **Firmware Roll-Back** to revert the DataPower appliance to the previous firmware level.
6. The **Select Configuration** section determines which configuration file is used on the next system restart.

System control features (2 of 3)

Secure Backup

Crypto certificate (none) *

Destination *

Include iSCSI on off

Include RAID on off

Secure Restore

Crypto Credential (none) *

Source *

Only validate the backup on off

Machine type of the backup appliance

Shutdown

Mode Reboot System *

Delay 1 Second(s)

Change User Password

Old Password *

New Password Confirm Password: *

© Copyright IBM Corporation 2013

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

WE4013.0

Notes:

This slide continues examining the system control page.

1. Use the **Secure Backup** option to create an encrypted backup of all files on the appliance, including keys and certificates.
2. Use the **Secure Restore** option to reload files onto an appliance from the encrypted backup.
3. Use the **Shutdown** option to restart the DataPower appliance in one of three modes:
 - a. **Reload firmware** restarts the device without rebooting the DataPower SOA appliance. Temporary files and applied, but unsaved, changes are kept intact.
 - b. **Reboot system** restarts the DataPower appliance. All temporary files and unsaved configuration changes are lost.
 - c. **Halt system** shuts down the DataPower appliance.

4. The **Change User Password** section allows you to assign a new password to the user who is currently logged in. If you are logged in as `admin`, this operation changes both the WebGUI and CLI passwords for the user.

System control features (3 of 3)

The screenshot shows a configuration interface for system control features. It consists of several sections, each with a button and a corresponding numbered callout circle:

- Restart Domain** (1): A button labeled "Restart Domain".
- Reset Domain** (2): A button labeled "Reset Domain".
- Generate Device Certificate** (3): A section containing "Common Name (CN)" input field, "Generate Self-Signed Certificate" radio buttons (on selected), and a "Generate Device Certificate" button.
- Control Locate LED** (4): A section containing "Blue Locator LED" radio buttons (off selected) and a "Control Locate LED" button.
- Quiesce** (5): A section containing "Timeout" input field and a "Quiesce" button.
- Unquiesce** (6): A section containing a "Unquiesce" button.

© Copyright IBM Corporation 2013

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

WE4013.0

Notes:

This slide finishes examining the system control page.

1. **Restart Domain** reloads the configuration for the current application domain. Any unsaved configuration changes are lost.
2. **Reset Domain** erases any configured objects in the domain. Be careful when using this feature. It does not delete any files in the file system.
3. **Generate Device Certificate** creates a digital certificate that represents the current DataPower appliance. The common name must be mapped to a valid IP host address.
4. **Control Locate LED** controls a blue "locate" LED on the front panel of the appliance, or on the XI50B Blade and the chassis.
5. **Quiesce** can be used to bring down a handler, service, domain, or appliance in a controlled manner.
6. **Unquiesce** returns the handler, service, domain, or appliance to an **up** state after a previous quiesce.

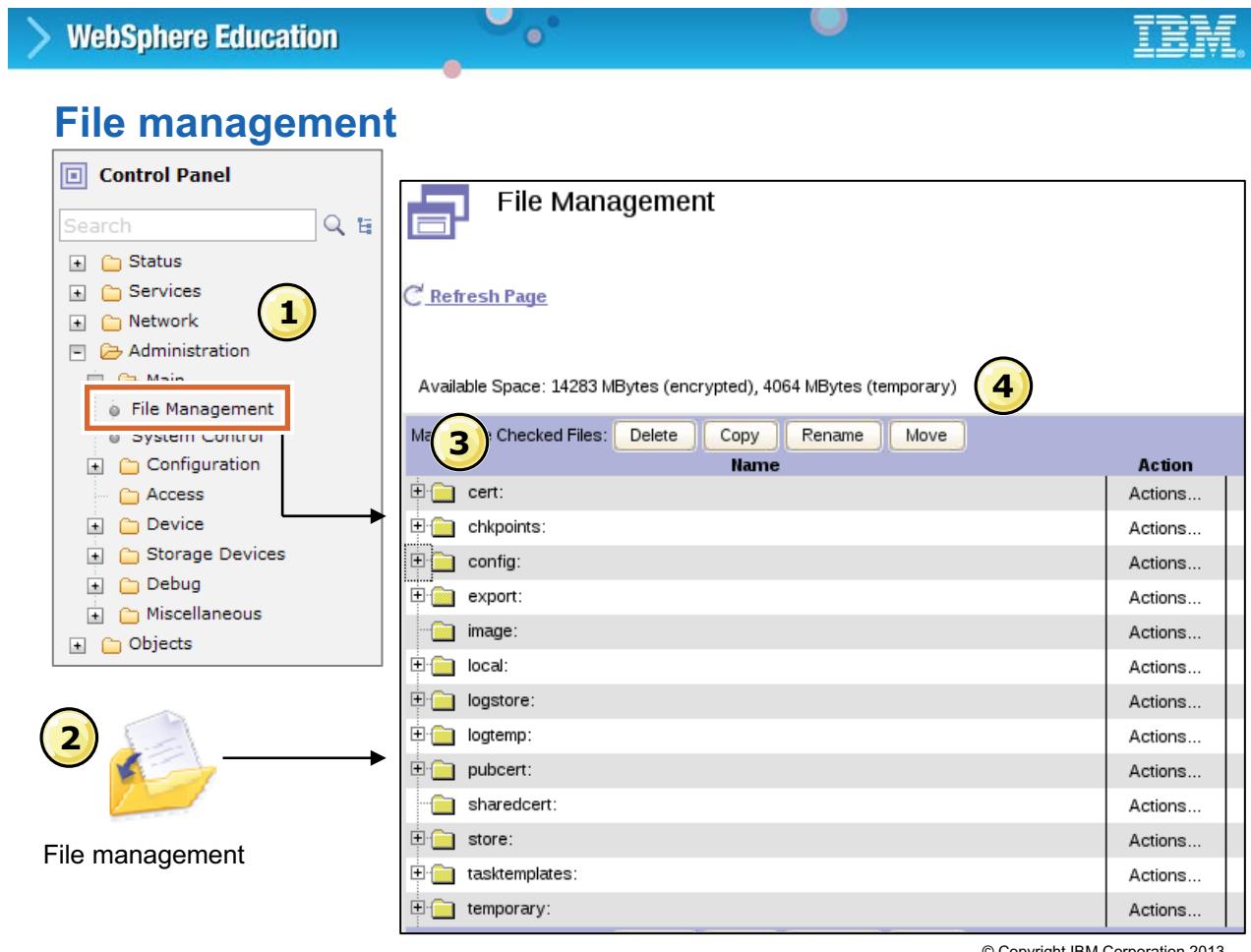


Figure 1-10. File management

WE4013.0

Notes:

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

File directories for configuration

| Store | Scope | Usage |
|------------|---|---|
| chkpoints: | Per application domain; not shared | Stores different versions of the configuration files for the current application domain |
| config: | Per application domain; not shared | Stores configuration files for the current application domain |
| export: | Per application domain; not shared | Holds any exported configuration that is created with the export configuration operation |
| local: | Per application domain; shareable | Stores files that local services use, including XML style sheets, XML schemas, and WSDL documents |
| store: | System-wide; shared | Stores sample and default style sheets that DataPower services use |
| temporary: | Per application domain; not shared | Temporary disk space that document processing rules and actions use |

© Copyright IBM Corporation 2013

Figure 1-11. File directories for configuration

WE4013.0

Notes:

The flash drive (9004 machines) or hard disk drive array (9004 and 9005 machines, and 4195 blades) are supported as a subdirectory of the `local:` and `logstore:` directories.

File directories for security

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

© Copyright IBM Corporation 2013

Figure 1-12. File directories for security

WE4013.0

Notes:

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

File directories for logging

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

© Copyright IBM Corporation 2013

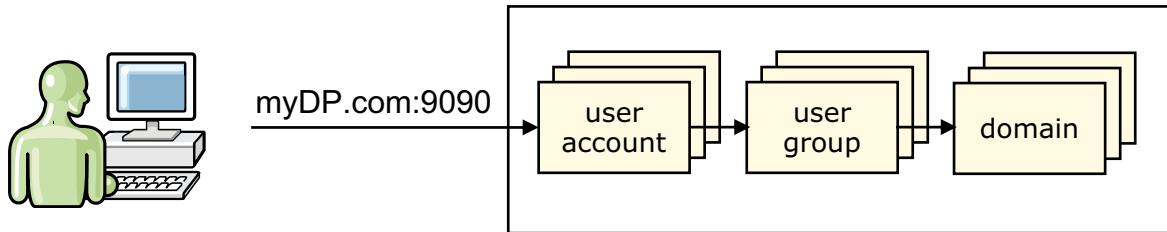
Figure 1-13. File directories for logging

WE4013.0

Notes:

Administrative access control

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



© Copyright IBM Corporation 2013

Figure 1-14. Administrative access control

WE4013.0

Notes:

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

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

User accounts that are created through the WebGUI interface also apply to the command-line interface (CLI) and XML Management Interface as well.

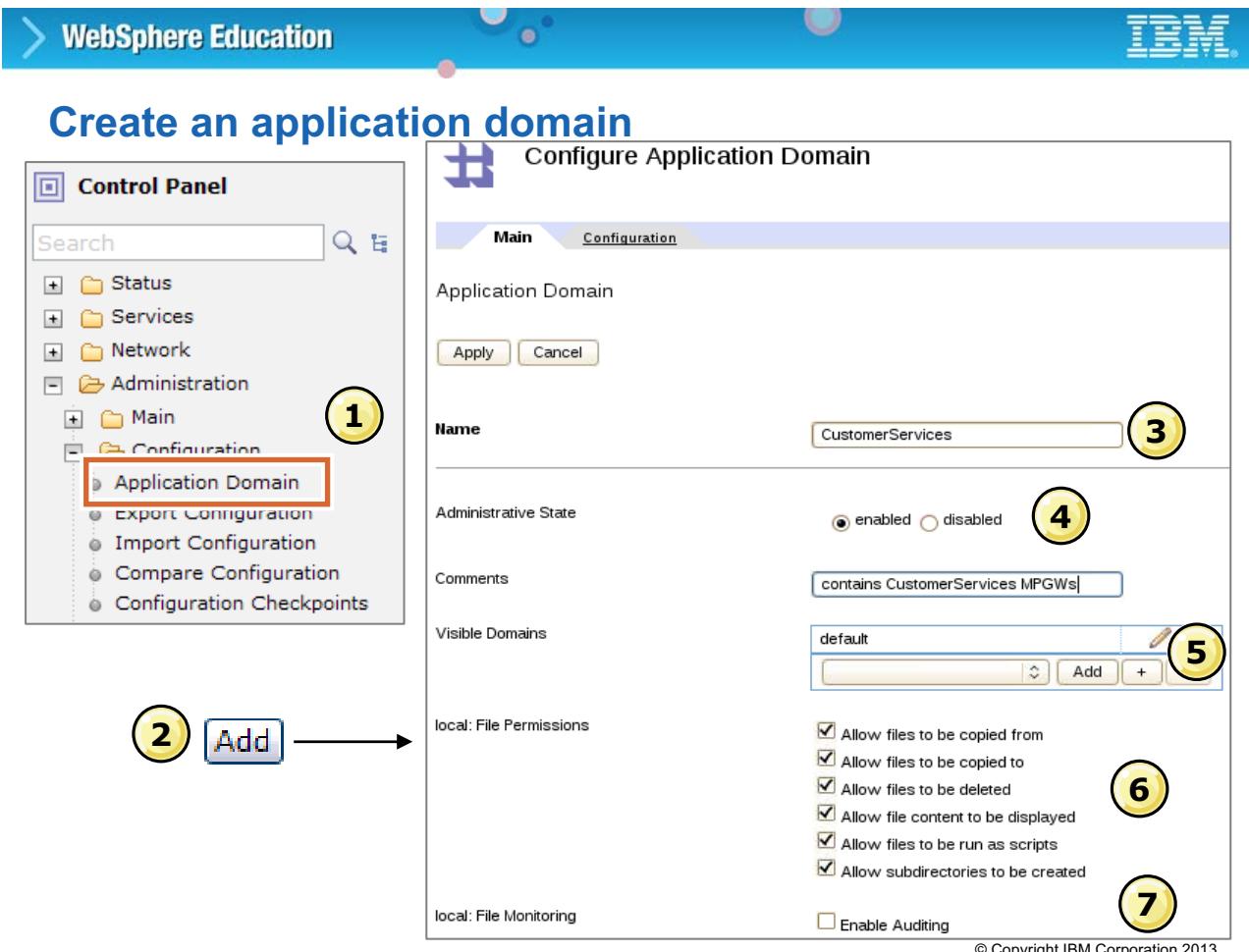


Figure 1-15. Create an application domain

WE4013.0

Notes:

1. From the WebGUI navigation bar, expand the **Administration** section and select **Configuration > Application Domain**.
2. In the listing of available application domains (not shown), click the **Add** button.
3. Provide a name for the new application domain; this field is mandatory.
4. Leave the Admin State at **enabled**. The administration state setting determines whether a particular DataPower object is available for use.
5. The visible domains setting determines whether this domain can access files in the local: file store of another application domain. In the figure, the **student01-domain** can access the files in the **local** file store of the **default** domain.
6. Local file permissions determine the access rights to files stored in the local file store of the current domain.
7. When enabled, changes to files in the local file store generate auditing or logging events.

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



Application domain: Configuration tab

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

The screenshot shows the 'Configure Application Domain' dialog box with the 'Configuration' tab selected. The 'Main' tab is also visible. The dialog has several fields:

- Name:** CustomerServices
- Configuration Checkpoint Limit:** 3
- Configuration Mode:** Import (selected)
- Import URL:** (empty field)
- Import Format:** ZIP (selected)
- Deployment Policy:** (none) (selected), with a '+' button and an ellipsis button (...).
- Local IP Rewrite:** on (radio button selected)

At the bottom are 'Apply' and 'Cancel' buttons.

© Copyright IBM Corporation 2013

Figure 1-16. Application domain: Configuration tab

WE4013.0

Notes:

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



Configuration Checkpoints

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

The screenshot shows a web-based administrative interface titled "Configuration Checkpoints". At the top, there is a "Refresh List" button. Below it is a table with three columns: "Name", "Time", and "Actions". A single row is visible, showing "Checkpoint1" and the timestamp "2012-10-03 22:12:35 GMT" under the "Time" column. 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 text input field labeled "Checkpoint Name:" and a "Save Checkpoint" button.

© Copyright IBM Corporation 2013

Figure 1-17. Configuration Checkpoints

WE4013.0

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).

View application domain status

 Configure Application Domain

[C Refresh List](#)

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

© Copyright IBM Corporation 2013

Figure 1-18. View application domain status

WE4013.0

Notes:

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

Create a user account 1

Should the user be restricted to a domain?

Selecting 'Yes' will restrict the user to a domain. Selecting 'No' will allow the user to login to all domains.

Create a user account 2

To which domain should the user be restricted?

Select the domain:

User Domain: student01-domain *

Create a user account 3

What kind of user account do you want to create?

Select one of the following:

Domain Account Type

- Developer (configuring services in a domain)
- Backup User (domain backup)
- Guest (read-only in domain)
- User-Defined Group

© Copyright IBM Corporation 2013

Figure 1-19. Create a user account and a user group

WE4013.0

Notes:

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

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

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

The screenshot shows two overlapping windows. The top window is titled 'Configure User Group' and displays a 'User Group : developer_student01-domain [up]' configuration. It includes tabs for 'Main' and 'CLI Command Groups'. Under 'Main', there are fields for 'Admin State' (enabled), 'Comments' (Developer group for the student), and an 'Access Profile' section. The 'Access Profile' section contains two entries: '/default/*?Access=r' and '/student01-domain/*?Access=r+w+a+d+x'. A yellow circle labeled '1' is around the second entry, and a yellow circle labeled '2' is around the 'Access Profile' section. The bottom window is titled 'Editing Access Profile property of User Group' and shows settings for 'Device Address', 'Application Domain' (studentXX-domain), 'Resource Type' (Web Service Proxy), 'Name Match (PCRE)' (EastAddressSearch), and 'Permissions' (Read, Write, Add, Delete, Execute). Buttons for 'Save' and 'Cancel' are at the bottom.

Access Profile property syntax

`address/domain/resource?Access=permissions & [field=value]`

© Copyright IBM Corporation 2013

Figure 1-20. Manage user group details

WE4013.0

Notes:

User groups provide a convenient way for applying an access profile to a set of user accounts. The access profile policy syntax restricts the access permission of any user to which the user group is applied. If two access profile policies affect the same resource, the most specific policy is applied.

- `address` refers to the DataPower appliance host name, IP address, or local host alias.
- `domain` specifies the name of one particular application domain.
- `resource` represents one type of DataPower object within the configuration.
- `permissions` are r (read), w (write), a (add), d (delete), or x (execute).
- `field` and `value` allow you to specify a particular object, such as the name of a web service proxy.

1. In the figure that is shown, users in the group have read access to all resources within the default domain and read, write, add, delete, and execute permissions for all resources in the `student01-domain` domain.

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

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

The screenshot shows the IBM WebSphere Education interface. At the top, there's a blue header bar with the 'WebSphere Education' logo and the 'IBM' logo. Below the header, the main title 'Manage user account details' is displayed in a large blue font. The left side features a 'Control Panel' sidebar with a search bar and a tree view of administration sections. A yellow circle labeled '1' highlights the 'Manage User Accounts' option under the 'Access' section. The right side shows the 'Configure User Account' page for a user named 'student88'. It includes tabs for 'Main' and 'SNMP V3 User Credentials'. The 'Main' tab displays fields for 'Comments' (with a yellow circle '2'), 'Password' (with a yellow circle '2'), 'Access Level' (set to 'Group-Defined' with a yellow circle '3'), 'User Group' (containing 'developer_student88_domain' with a yellow circle '2'), and 'Domain Restriction' (which is currently empty). Navigation links like 'View Log', 'View Status', 'Help', 'Reset Password', 'Force Password Change', 'Reset Failed Login', and 'Logout' are also present.

Figure 1-21. Manage user account details

WE4013.0

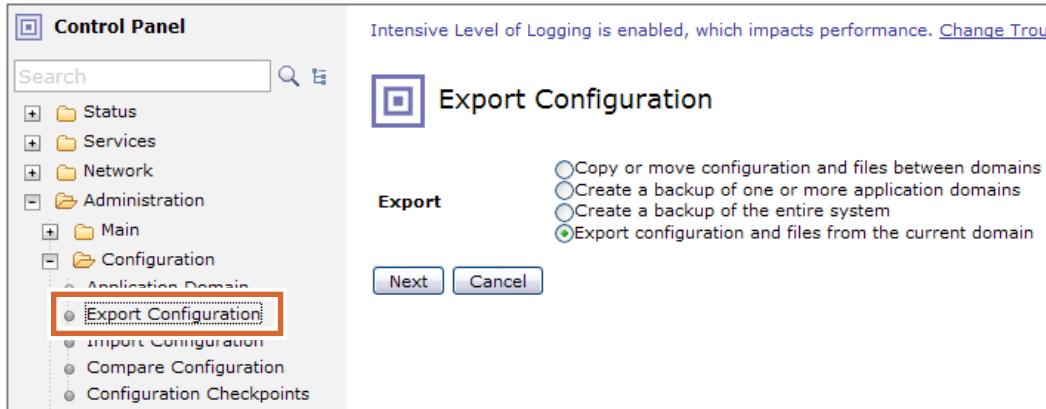
Notes:

1. From the navigation bar **Administration** section, select **Access > Manage User Accounts**.
2. The Configure User Account page allows you to modify the comments and the password for a specified user in the application domain.
3. Select one of three access level settings: privileged, user, or group-defined.

The **Domain Restriction** list limits which domains a user account can access. An existing access policy, or an RBM access policy can supercede this list.

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

Export the system configuration



- 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 2013

Figure 1-22. Export the system configuration

WE4013.0

Notes:

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

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



Import a system configuration

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



Import Configuration

Select options for Import

From XML ZIP

File [Browse...](#) *

Use Deployment Policy

Rewrite Local Service Addresses on off

[Next](#) [Cancel](#)

© Copyright IBM Corporation 2013

Figure 1-23. Import a system configuration

WE4013.0

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.

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

The screenshot shows the WebSphere Education interface. At the top, there is a navigation bar with a back arrow, the text "WebSphere Education", and the IBM logo. Below the navigation bar, the title "Saving configuration changes" is displayed. A horizontal row of buttons includes "Apply" (highlighted with a red box), "Cancel", and "Delete". Below this, a warning dialog box appears with the message: "The page at https://[REDACTED]:9090 says: Your configuration changes have not been saved! Click OK to continue without saving or Cancel to go and save your changes." It has "OK" and "Cancel" buttons. At the bottom of the page, there is a navigation bar with "Domain: student01-domain ▾", "Save Config" (highlighted with a red box), and "Logout".

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

© Copyright IBM Corporation 2013

Figure 1-24. Saving configuration changes

WE4013.0

Notes:

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

Administration by using the command-line interface

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

© Copyright IBM Corporation 2013

Figure 1-25. Administration by using the command-line interface

WE4013.0

Notes:

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

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

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

First boot after unpacking

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

© Copyright IBM Corporation 2013

Figure 1-26. First boot after unpacking

WE4013.0

Notes:

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

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

Initial CLI login screen

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

Welcome to DataPower XI52 console configuration.
Copyright IBM Corporation 1999-2012
```

```
Version: XI52.5.0.0.1 build 215672 on 2012/08/03 23:57:15 2
Serial number: 6XXXXXXX
```

```
xi52# show system 3

description: DataPower XI52
serial number: 6800358
entitlement id: 6800358
product id: 719942X [Rev 0001]
OID: 1.3.6.1.4.1.14685.1.3
uptime: 6 days 22:30:52
contact: Jim Brown/Los Angeles
```

© Copyright IBM Corporation 2013

Figure 1-27. Initial CLI login screen

WE4013.0

Notes:

1. You must provide a valid user login and password in order to access the command-line interface. At installation time, the default user name and password are `admin` and `admin`. After reviewing and accepting the license agreement (which is not shown), you must provide a new password for the `admin` account.
2. The initial welcome message displays the firmware build level and date, as well as the serial number of the appliance.
3. Use the `show` command to display system information about the interfaces, objects, and the appliance itself.

Quick initial configuration procedure

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

```

xi52# configure terminal
Global configuration mode
xi52(config)# interface mgt0
Interface configuration mode (mgt0)
xi52(config-if[mgt0])# ip address 10.0.0.0/8
xi52(config-if[mgt0])# exit
xi52(config)# web-mgmt 10.0.0.0 9090
Web management: successfully started
xi52(config)# ssh 10.0.0.0 22
%      Pending

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

```

1

2

3

4

5

© Copyright IBM Corporation 2013

Figure 1-28. Quick initial configuration procedure

WE4013.0

Notes:

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

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

These steps are performed as part of the installation wizard.

Retrieve system information by using the CLI

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

© Copyright IBM Corporation 2013

Figure 1-29. Retrieve system information by using the CLI

WE4013.0

Notes:

Administration by using SOAP: SOMA and AMP

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

© Copyright IBM Corporation 2013

Figure 1-30. Administration by using SOAP: SOMA and AMP

WE4013.0

Notes:

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

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

SOMA: Create a user account

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

© Copyright IBM Corporation 2013

Figure 1-31. SOMA: Create a user account

WE4013.0

Notes:

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



SOMA: Account creation response

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

    <dp:response
      xmlns:dp="http://www.datapower.com/schemas/management">
      <dp:timestamp>
        2012-09-22T15:22:04-05:00
      </dp:timestamp>
      <dp:result>
        OK
      </dp:result>
    </dp:response>

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

© Copyright IBM Corporation 2013

Figure 1-32. SOMA: Account creation response

WE4013.0

Notes:

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

AMP: Appliance information request

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

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

</soapenv:Envelope>
```

© Copyright IBM Corporation 2013

Figure 1-33. AMP: Appliance information request

WE4013.0

Notes:

The AMP WSDL defines multiple operations, unlike SOMA.

AMP: Appliance information response

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Body>
  <amp:DeviceInfoResponse xmlns:amp=
    "http://www.datapower.com/schemas/appliance/management/3.0">
    <amp:DeviceName>DP #15</amp:DeviceName>
    <amp:DeviceSerialNo>XXXXXXXX</amp:DeviceSerialNo>
    <amp:DeviceID>719942X</amp:DeviceID>
    <amp:DeviceType>XI52</amp:DeviceType>
    <amp:FirmwareVersion>XI52.5.0.0.1</amp:FirmwareVersion>
    <amp:FailureDetected>false</amp:FailureDetected>
    <amp:CurrentAMPVersion>3.0</amp:CurrentAMPVersion>
    <amp:ManagementInterface type="web-mgmt">
      9090</amp:ManagementInterface>
    <amp:DeviceFeature>MQ</amp:DeviceFeature>
    <amp:DeviceFeature>TAM</amp:DeviceFeature>
    . . .
  </amp:DeviceInfoResponse>
</env:Body></env:Envelope>
```

© Copyright IBM Corporation 2013

Figure 1-34. AMP: Appliance information response

WE4013.0

Notes:

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



Management interface summary

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

© Copyright IBM Corporation 2013

Figure 1-35. Management interface summary

WE4013.0

Notes:

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

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

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

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



Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 1-36. Unit summary

WE4013.0

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. Which user account and application domain do you need to use in order to perform an upgrade?
 - A. Any domain
 - B. default
 - C. Linux root
 - D. admin
3. Match the advantages in performing administration tasks through the diverse DataPower interfaces:

| Description | Definition |
|-------------------------------------|--|
| 1. The WebGUI web application | A. Creation of scripts, less bandwidth |
| 2. The command-line interface (CLI) | B. Easier to use |
| 3. The XML Management Interface | C. Programmatic |

© Copyright IBM Corporation 2013

Figure 1-37. Checkpoint questions

WE4013.0

Notes:

Checkpoint answers

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

| Description | Definition |
|-------------------------------------|--|
| 1. The WebGUI web application | A. Creation of scripts, less bandwidth |
| 2. The command-line interface (CLI) | B. Easier to use |
| 3. The XML Management Interface | C. Programmatic |

© Copyright IBM Corporation 2013

Figure 1-38. Checkpoint answers

WE4013.0

Notes:

Exercise 1



© Copyright IBM Corporation 2013

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

7.0.1

Figure 1-39. Exercise

WE4013.0

Notes:

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



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 1-40. Exercise objectives

WE4013.0

Notes:

Unit 2. Introduction to XSL transformations

What this unit is about

This unit introduces you to XSL transformations. You learn how to create XSLT style sheets to transform XML documents into other formats. You also learn how to write XPath expressions to retrieve information from an XML document.

What you should be able to do

After completing this unit, you should be able to:

- Describe the Extensible Stylesheet Language (XSL) model
- Construct XPath expressions
- Create XSL style sheets to apply XSL transformations
- Use and apply XSL templates in XSLT
- Describe the use of DataPower variables and extensions in XSL style sheets
- Test a style sheet by using the Interoperability Test Service

How you will check your progress

- Checkpoint
- Hands-on exercise

Unit objectives

After completing this unit, you should be able to:

- Describe the Extensible Stylesheet Language (XSL) model
- Construct XPath expressions
- Create XSL style sheets to apply XSL transformations
- Use and apply XSL templates in XSLT
- Describe the use of DataPower variables and extensions in XSL style sheets
- Test a style sheet by using the Interoperability Test Service

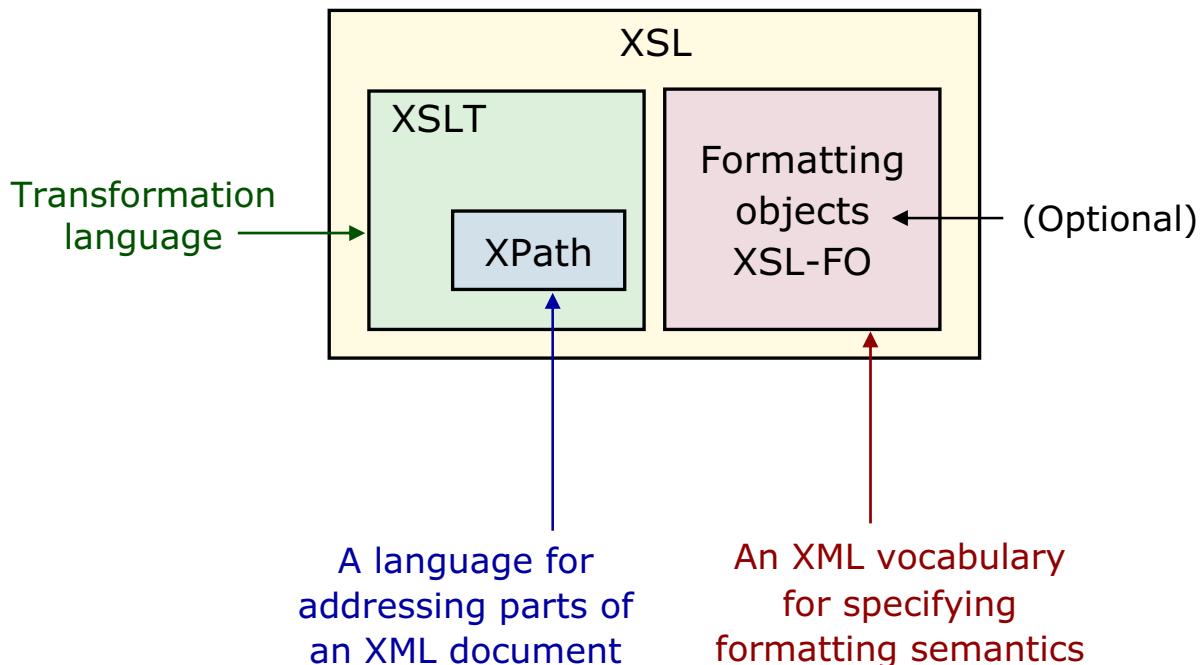
© Copyright IBM Corporation 2013

Figure 2-1. Unit objectives

WE4013.0

Notes:

Three parts of Extensible Stylesheet Language (XSL)



© Copyright IBM Corporation 2013

Figure 2-2. Three parts of Extensible Stylesheet Language (XSL)

WE4013.0

Notes:

Extensible Stylesheet Language (XSL) describes how to display XML data. It is analogous to how cascading style sheets (CSS) describe the presentation of web pages. XSL itself comprises two main specifications:

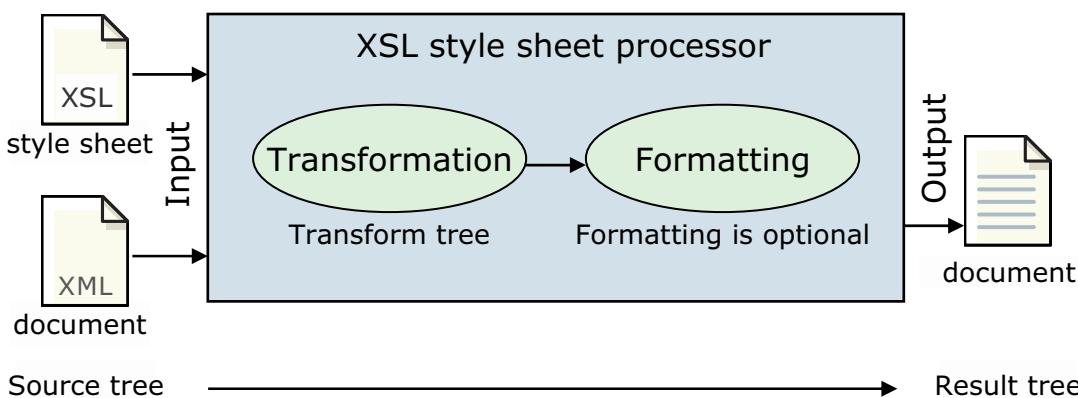
- A transformation language, XSLT, for processing XML data.
- Formatting objects - XSL-FO is a vocabulary for formatting objects that are received from XSLT into a result tree. It allows for a large array of print, display, or oral presentations.

XSLT also provides a language to describe the location of data within an XML document, which is known as XPath.

XSL Transformations (XSLT) overview

XSLT Transformation overview

1. An XML document and an XSL style sheet are passed to an XSL processor
2. The XSL processor uses the instructions in the style sheet to transform the input document (source tree) into the output document (result tree)
3. Formatting can optionally be applied to the document
4. The output exits the XSL processor in XML, HTML, or any other structured document format



© Copyright IBM Corporation 2013

Figure 2-3. XSL Transformations (XSLT) overview

WE4013.0

Notes:

An XSL style sheet processor accepts an XML document that is represented as a tree structure and processes it to produce a result tree.

The XSL style sheet defines the rules for transformation, which is based on the XML elements and attributes in the source tree. The style sheet might also contain formatting information called formatting objects (or FOs) and applies those objects against the transformation.

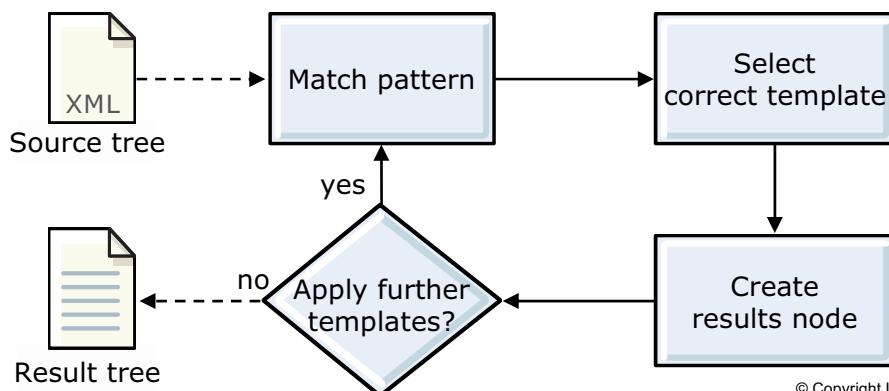
XSL does not require the use of XSL-FO for formatting.

An example use of XSL is to transform XML into well-formed HTML, that is, XML that uses the element types and attributes that are defined by HTML.

The XSLT process

XSLT processing within the style sheet:

1. The XSL processor reads an element (node) from the input document (source tree)
2. The element is compared to the template patterns within the style sheet
3. The XSL processor selects the template that matches the pattern
4. The source tree is read, and part of the result tree (output document) is constructed from the template instructions within the style sheet
5. The style sheet can specify other templates that are going to be tested and applied as the source tree is read and transformed into more of the result tree
6. As the end of the source tree is reached, no further templates are applied, and the result tree is complete



© Copyright IBM Corporation 2013

Figure 2-4. The XSLT process

WE4013.0

Notes:

XSLT uses the ideas of *pattern matching* and *templates*. A style sheet includes templates, which contain rules that associate them with one or more elements or attributes in the XML document.

The templates contain the rules for transformation and, optionally, the formatting that is applied to the matching nodes.

A template can also contain further pattern matching and instructions to apply further templates.

What is XPath?

- A specification for describing a location with an XML document
 - Shared by many XML-based standards and technologies
 - Used by XSLT, XPointer, and XQuery
- Allows you to address elements of a document that meet specified criteria
 - Example: in XML for a book on Java, find the chapters with JDBC in the title
- Allows you to retrieve a subset of an XML document in any direction
 - Forwards, backwards, or sideways

© Copyright IBM Corporation 2013

Figure 2-5. What is XPath?

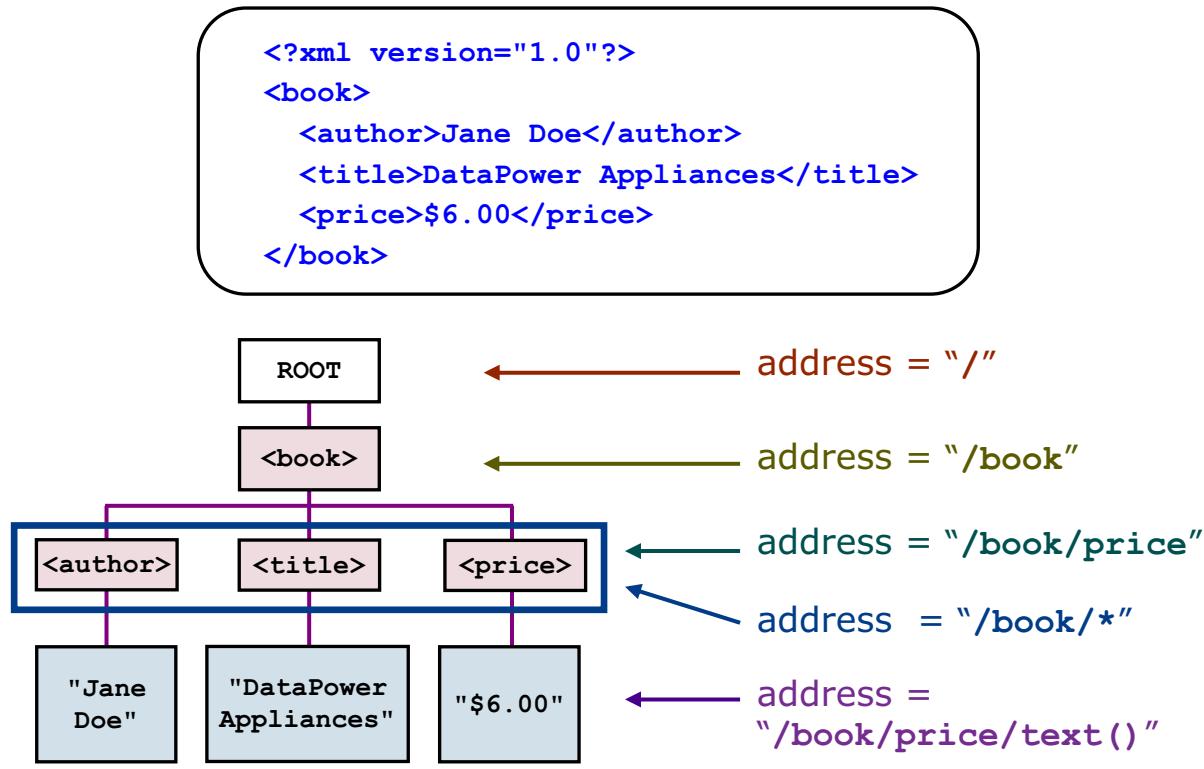
WE4013.0

Notes:

W3C recommends (Nov. 16, 1999):

- XQuery provides standardized access to RDBMS data stores by using XML.
- XPointer allows forward and backward addressing to specific XML locations internal to a document and to locations in external XML documents. Consider XPointer as an enhanced version of HTML HREF linking.
- When XPath is used in an XML document, it usually appears as an attribute value. For example, it appears as an attribute value within an `<xsl:template>` element in XSLT.

Example XPath expressions



© Copyright IBM Corporation 2013

Figure 2-6. Example XPath expressions

WE4013.0

Notes:

There is a single root node, which contains several other types of nodes.

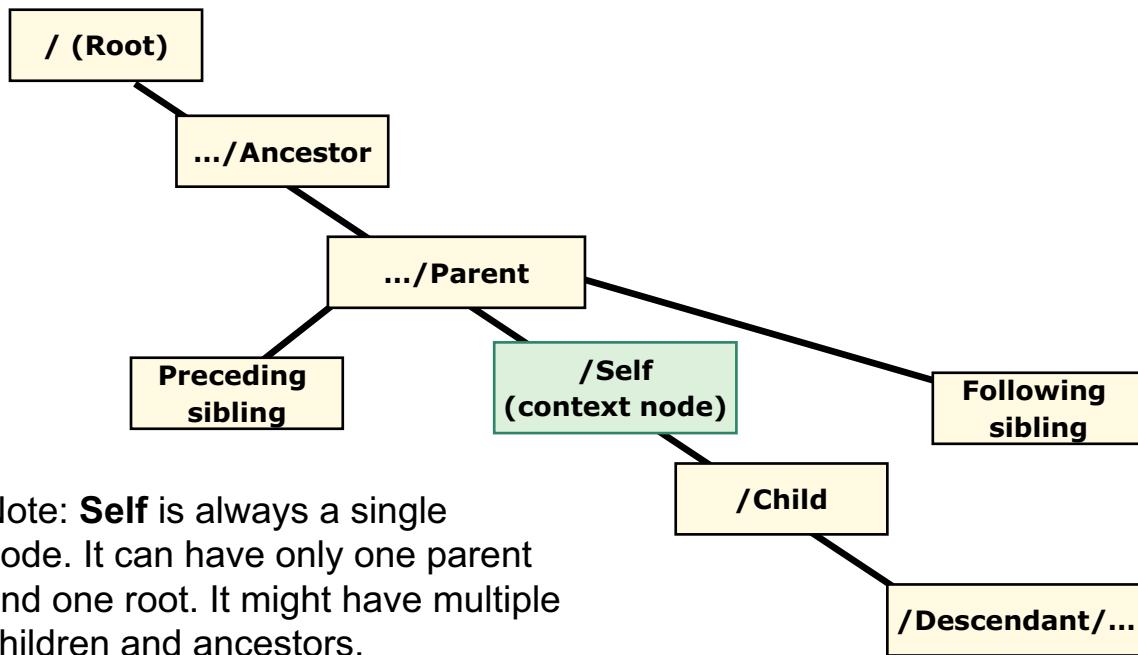
There are seven node types in XML:

- Root nodes
- Element nodes
- Text nodes
- Attribute nodes
- Namespace nodes
- Processing instruction nodes
- Comment nodes

XPath current context

- The active element within the XPath address step

- /Root/.../Ancestor/Parent/**SELF**/Child/Descendent



© Copyright IBM Corporation 2013

Figure 2-7. XPath current context

WE4013.0

Notes:

The current context is simply a "you are here" designation within a complete XPath address.

When using XPath traversals in pattern matching, the leftmost value becomes the context (current) node, even though you navigated to a child node that exists further down. For example, if book/title is the path, then book remains the context node, even though you are not matching against it.

XPath step syntax

- An XPath location path is made up of one or more steps that are separated with a forward slash (/)
- Each step within the path consists of:
 - **Axis**: branch of the node tree relative to the current context node (use keywords such as `ancestor`, `attribute`, `child`, `descendant`)
 - **NodeTest**: consists of the node name that is used to test node for inclusion
 - **Predicate**: optional filter of matched nodes
- Abbreviated syntax is allowed for several different axes
 - "`child::`" has an empty default as it is the default axis
 - "`/child::catalog/child::tools/`" is the same as "`/catalog/tools/`"
- Expression shortcuts
 - "`//[element]`" selects element node regardless of location
 - "`.`" selects the current node
 - "`..`" selects the parent of the current node
 - "`@[attribute-name]`" selects an attribute
- Example:
 - Locate all titles in the book that contain the string '`XPath`'
 - `/book/child::title[contains(text(), 'XPath')]/`

.../**axis::nodetest[predicate]**/...

© Copyright IBM Corporation 2013

Figure 2-8. XPath step syntax

WE4013.0

Notes:

XPath uses a path notation similar to URLs. Location paths are specified by using a list of steps that are separated by a forward slash (/).

XPath provides a simple method to traverse an XML tree structure and select a slice of information in any direction that the axis defines.

Paths starting with a forward slash (/) are absolute paths from the root downward through the document tree. Paths that do not begin with a slash are relative to the current (context) node of the node list.

For a complete listing of XPath axes, see <http://www.w3.org/TR/xpath#axes>.

XPath address notation

An XPath expression consists of two types of paths

- Absolute location path:
 - Starts search at the root of the tree
 - Search begins with a forward slash (/)
- Relative location path:
 - Sequence of one or more location steps, or referenced from the current context node

© Copyright IBM Corporation 2013

Figure 2-9. XPath address notation

WE4013.0

Notes:

The absolute path is addressed based from the document root.

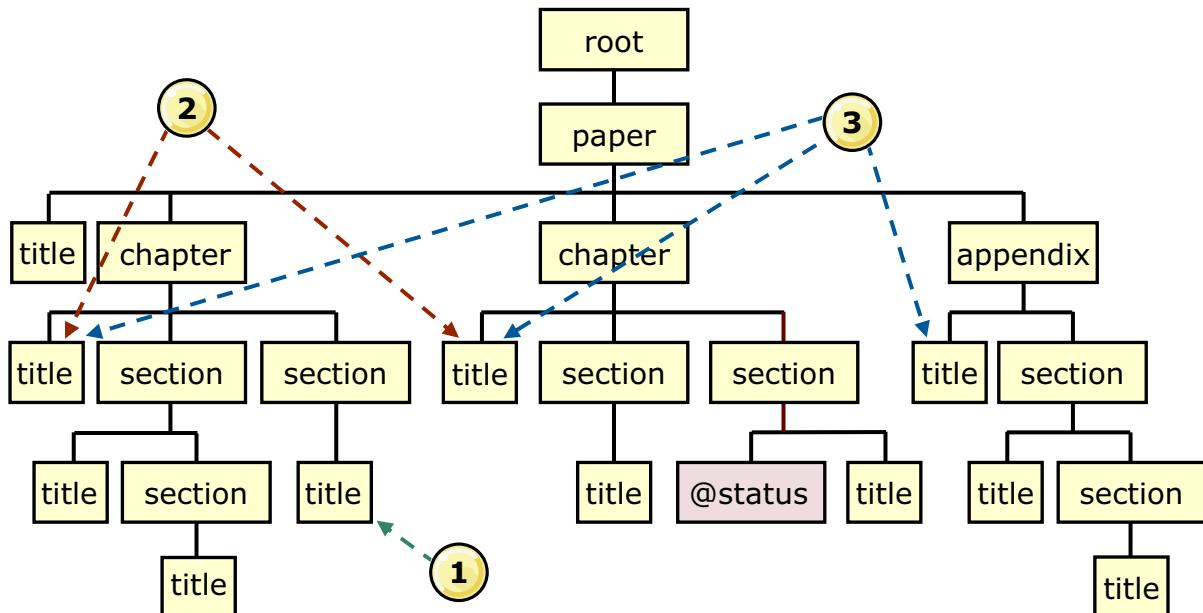
- /child::catalog/child::tools - The full syntactic expression that returns all child elements of the catalog element that appear under the document root
- /catalog/tools - the short form

The relative path is based on the current context of the addressing path.

- child::tools/child::saw - the full expression of a path relative to the context node
- tools/saw - the short form

Example: XPath absolute addressing

1. `/paper/chapter[1]/section[2]/title` → Title for first chapter, second section
2. `/paper/chapter/title` → Titles for all chapters
3. `/paper/*/title` → Any title that is a child of any element child of paper



© Copyright IBM Corporation 2013

Figure 2-10. Example: XPath absolute addressing

WE4013.0

Notes:

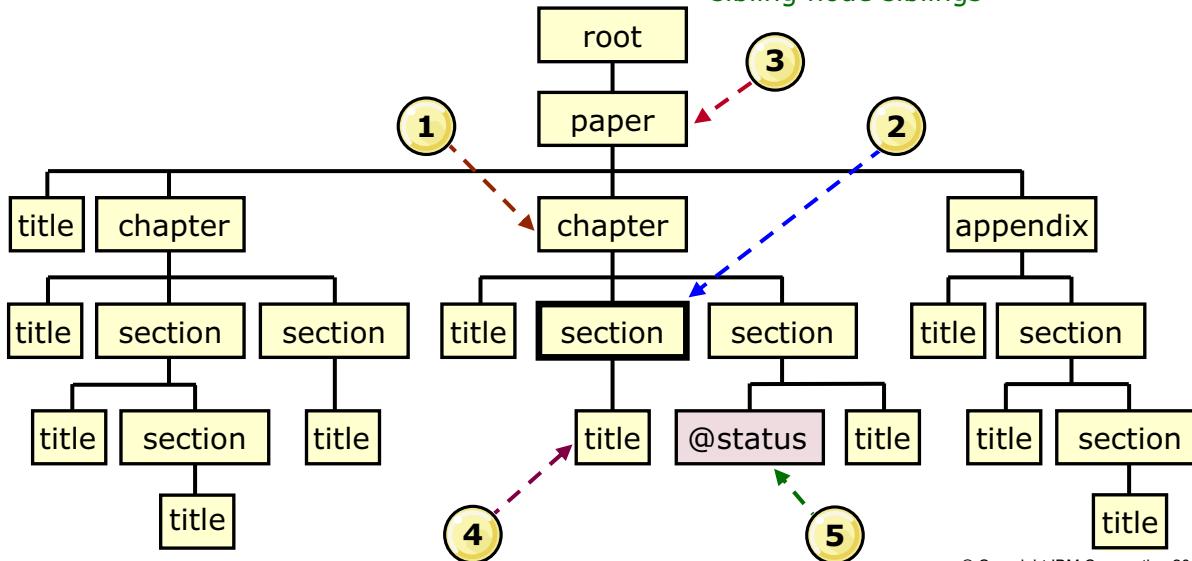
All direct addressing starts at 1.

The results of running the XPath expression against the XML source tree are shown.

1. Section 1.2 title
2. Chapter 1 title, chapter 2 title
3. Chapter 1 title, chapter 2 title, appendix A.1 title

Example: XPath relative addressing

| | |
|--|---|
| <pre>/paper/chapter[2]/section[1]</pre> <ol style="list-style-type: none"> 1. parent::node() or .. 2. self::node() or . 3. ../../.. 4. child::* (default) 5. ./following-sibling::node() /@status | → Absolute path to "current context" → Parent of current context → Context node (self) → Parent of parent of context node → Children of the current context node → Status attribute of any following sibling node siblings |
|--|---|



© Copyright IBM Corporation 2013

Figure 2-11. Example: XPath relative addressing

WE4013.0

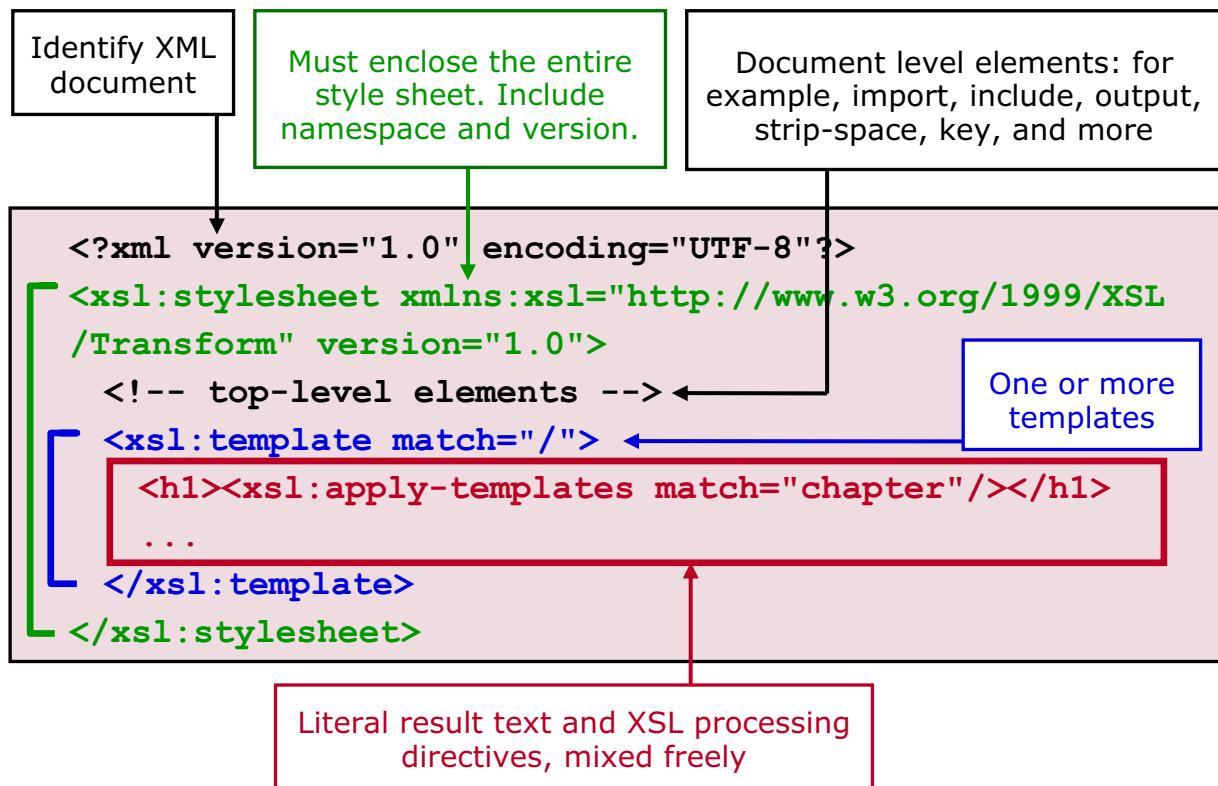
Notes:

The expressions (1–5) are run after the initial absolute path to the current context is executed. The black box indicates the current context.

The results of running the XPath expression against the XML source tree are in the elements shown.

1. Chapter 2, chapter 2 title, section 2.1, section 2.1 title, section 2.2, section 2.2 title
2. Section 2.1, section 2.1 title
3. Paper (**everything in the instance file**)
4. Section 2.1 title
5. Section 2.2 status

Anatomy of an XSL style sheet



© Copyright IBM Corporation 2013

Figure 2-12. Anatomy of an XSL style sheet

WE4013.0

Notes:

The figure shows the main elements that make up the XSL style sheet and the order in which they appear.

The forward slash (/) expression in XSLT also matches the root element node.

The <xsl:template> element

A style sheet has one or more template tags with the structure:

```
<xsl:template match="match expression">
    <!-- literal result text or XSLT elements -->
</xsl:template>
```

Specifies:

- A **match expression** defines when the template is called
 - An XPath expression
 - Test against the nodes in the XML source tree
- Literal result text is written to the output tree or XSLT elements are executed

© Copyright IBM Corporation 2013

Figure 2-13. The <xsl:template> element

WE4013.0

Notes:

The `<xsl:template>` tag is a container for a set of rules that apply actions against the source tree to yield a result tree.

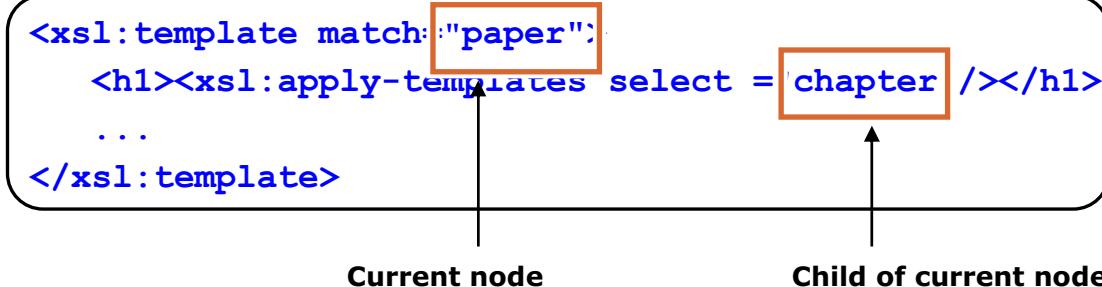
The `match="/" /` attribute matches the root node, which provides access to namespaces, processing instructions, and comments, if they are available.

The following functions are available for accessing the respective elements:

- Namespaces: `namespace()`
- Processing instructions: `processing-instruction()`
- Comments: `comment()`

The <xsl:apply-templates> element

- Looks for a matching template rule in the XSL style sheet
 - Each child of the current node in the XML source tree is evaluated for a matching template rule
- The rules that can be matched are:
 - None: it is not required to have a template rule for each child node
 - The template match rules that you define by using the **select** attribute



© Copyright IBM Corporation 2013

Figure 2-14. The <xsl:apply-templates> element

WE4013.0

Notes:

The `apply-templates` tag gives you automatic recursion because the template executes for each instance of the node.

The <xsl:value-of> element

- **<xsl:value-of select="patternToMatch"/>**
 - Is used to extract a specific value from the source tree
 - Inserts literal values into result tree as a string, element, or attribute from **patternToMatch**
- Example:

```
<list>
  <book ID = "999">
    <author>Dan Big</author>
    <title>Large Stories</title>
    <price>$7.00</price>
  </book>
</list>
```

Result

```
<td>Large Stories</td>
```

```
<xsl:template match="/list/book">
  <td><xsl:value-of select="title"/></td>
</xsl:template>
```

© Copyright IBM Corporation 2013

Figure 2-15. The <xsl:value-of> element

WE4013.0

Notes:

In this example, the contents of the `book` child tag `title` are extracted into a `td` element in the result tree. This element produces an output text node, since the `td` element markup was supplied explicitly.

XSLT style sheet elements to generate output

The following elements are used to generate output in the transformed document:

- **<xsl:apply-templates />**
 - Outputs the value of the current node when no matching template is found
- **<xsl:value-of select="validXPathExprOrFunction">**
 - Extracts a specific value from source tree
- **<xsl:text>**
 - Inserts text into result tree verbatim
 - Used when outputting special characters, particularly whitespace
- **<xsl:processing-instruction name="piName"/>**
 - Inserts a processing instruction into the result tree
- **<xsl:comment>**
 - Inserts a comment into the result tree

© Copyright IBM Corporation 2013

Figure 2-16. XSLT style sheet elements to generate output

WE4013.0

Notes:

The namespace prefix `xsl` is a namespace alias declared inside the XSL style sheet element.

Using the `<xsl:comment>` is not the same as testing for the presence of a comment within the source nodes. That test is done by using the `comment()` function in the test predicate.

XML input as a tree

```

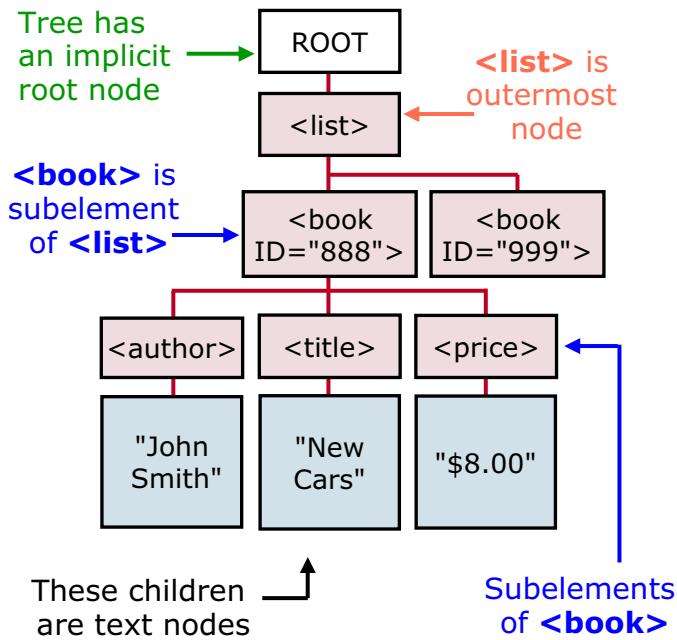
<?xml version = "1.0" encoding
= "UTF-8"?>
<!DOCTYPE list SYSTEM
"books.dtd">

<list>
  <book ID = "888">
    <author>John Smith
    </author>
    <title>New Cars</title>
    <price>$8.00</price>
  </book>
  <book ID = "999">
    <author>Dan Big</author>
    <title>Large Stories
    </title>
    <price>$7.00</price>
  </book>
</list>

```

Books.xml

The tree is created by parsing the XML document.



Note: The children of the second book are not shown

© Copyright IBM Corporation 2013

Figure 2-17. XML input as a tree

WE4013.0

Notes:

A XSL style sheet transforms the Books.xml file.

HTML produced by XSLT must be XHTML-compliant so that a valid XML tree structure is produced. If you have invalid HTML (for example, with no closing tag), the XSLT processor throws an error.

Intended HTML output

```
<html>
<head><title>Book List </title></head>
<body>
  <h1>Book List</h1>
  <table border="1" cols="3" width="100%">
    <tbody>
      <tr>
        <td>888</td>
        <td>New Cars</td>
        <td>$8.00</td>
      </tr>
      <tr>
        <td>999</td>
        <td>Large Stories</td>
        <td>$7.00</td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

The HTML that is produced must be well-formed

Data that is taken from the XML document (nodes)

Book List

| | | |
|-----|---------------|--------|
| 888 | New Cars | \$8.00 |
| 999 | Large Stories | \$7.00 |

© Copyright IBM Corporation 2013

Figure 2-18. Wanted HTML output

WE4013.0

Notes:

HTML produced by XSLT must be XHTML-compliant so that it produces a valid XML tree structure.

If you have invalid HTML (for example, no closing tag is used), the XSLT processor throws an error.

XML to HTML (1 of 4)

```
<list>
  <book ID = "888">
    <author>John Smith</author>
    <title>New Cars</title>
    <price>$8.00</price>
  </book>
...
</list>
```

Books.xml

The processor looks for a `<xsl:template match = "/">` tag, which matches the root element `<list>`.

It copies non-XSLT elements to the output tree in list template so you get the first part of your HTML

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Book List</title></head>
    <body>
      <table border="1" cols="3" width="100%" >
        <tbody>
          <xsl:apply-templates />
        </tbody>
      </table>
    </body>
  </html>
...
(remaining templates are omitted for clarity)
```

Books.xsl

```
<html>
  <head><title>
    Book List
  </title></head>

  <body>
    <table borders="1"
      cols="3"
      width="100%">
      <tbody>
```

HTML output

© Copyright IBM Corporation 2013

Figure 2-19. XML to HTML (1 of 4)

WE4013.0

Notes:

The first pattern match is the root element. In this case, it would not matter if it is `match="/" or match="list"`, since plain HTML code is transferred over to the output tree.

XML to HTML (2 of 4)

```
<list>
  <book ID = "888">
    <author>John Smith</author>
    <title>New Cars</title>
    <price>$8.00</price>
  </book>
  ...
</list>
```

Books.xml

```
<xsl:template match="/">
  ...
  <xsl:apply-templates />
  ...
</xsl:template>

<xsl:template match="book">
  <tr>
    <td><xsl:value-of select="@ID" /></td>
    <xsl:apply-templates
      select="title|price"/>
  </tr>
</xsl:template>
```

Books.xsl

Inside the `<xsl:template match="/">` tag, the `<xsl:apply-templates/>` tag looks for templates for the children of "list" (that is, `<book>`). It finds `<xsl:template match="book">`, and processes that template.

`<xsl:value-of select="@ID" />` writes the value of the attribute ID to the output tree.

```
<html>
<head><title>Book List
</title></head>
<body>
<table borders="1"
       cols="3"
       width="100%">
  <tbody>
    <tr>
      <td>888</td>
```

HTML output

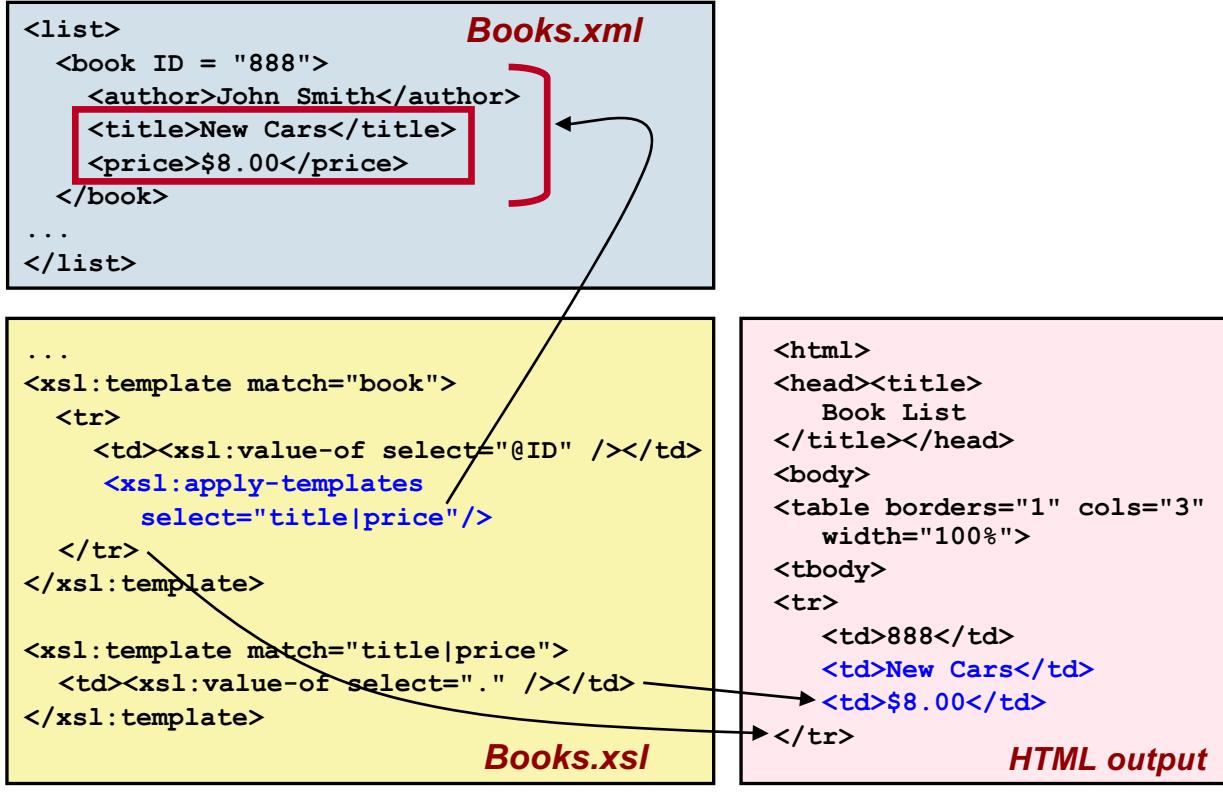
© Copyright IBM Corporation 2013

Figure 2-20. XML to HTML (2 of 4)

WE4013.0

Notes:

XML to HTML (3 of 4)



© Copyright IBM Corporation 2013

Figure 2-21. XML to HTML (3 of 4)

WE4013.0

Notes:

The `<xsl:apply-templates select="title|price"/>` tag accepts matches for both the `<title>` and `<price>` children of `<book>`.

The `<xsl:value-of select =". ." />` writes the value of the element node to the output tree.

The `|` is "or" from XPath.

The processor calls the `<xsl:apply-templates select="author|price"/>` template:

- Once for each `<author>` node, which is a child of `<book>`
- Once for each `<price>`, which is a child of `<book>`

XML to HTML (4 of 4)

```
<list>
...
<book ID = "999">
    <author>Dan Big</author>
    <title>Large Stories</title>
    <price>$7.00</price>
</book>
</list>
```

Books.xml

The processor now looks for and finds another `<book>` node to process. Output for that `<book>` node is added to the output tree.

```
<xsl:template match="book">
    <tr>
        <td><xsl:value-of select="@ID" /></td>
        <xsl:apply-templates
            select="title|price"/>
    </tr>
</xsl:template>

<xsl:template match="title|price">
    <td><xsl:value-of select="." /></td>
</xsl:template>
```

(Other templates have been omitted for clarity)

Books.xsl

```
<html>
...
<tr>
    <td>888</td>
    <td>New Cars</td>
    <td>$8.00</td>
</tr>
<tr>
    <td>999</td>
    <td>Large Stories</td>
    <td>$7.00</td>
</tr>
```

HTML output

© Copyright IBM Corporation 2013

Figure 2-22. XML to HTML (4 of 4)

WE4013.0

Notes:

After processing the first node, processing of the next `<book>` element takes place.

XSL style sheet control elements

The following elements are used to control flow in an XSL style sheet:

- **<xsl:apply-templates/>**
 - Looks for matching template that is based on current node or by using expression in `select` attribute, if specified
- **<xsl:call-template name= "templateName" />**
 - Calls a template; that is, `<xsl:template ...>` with a `name` attribute “`templateName`”
- **<xsl:for-each>**
 - Used to iterate over the result of an XPath expression
- **<xsl:if>**
 - Allows a conditional test or tests
- **<xsl:choose>**
 - Allows a choice of one or more tests and permits a default condition:
 - `<xsl:when>` – the tested condition
 - `<xsl:otherwise>` – the default condition

© Copyright IBM Corporation 2013

Figure 2-23. XSL style sheet control elements

WE4013.0

Notes:

Unlike the `<xsl:apply-template>` tag, the `<xsl:call-template>` tag does not call the template multiple times for each instance of the node.

The invoked `<xsl:template>` by the `<xsl:call-template>` must have a `name` attribute defined.

Here is how to define a named template:

```
<xsl:template name="bookTitle" > <h1> <xsl:value-of select=". /></h1>
</xsl:template>
```

Here is how to call a named template:

```
<xsl:template match="title">
  <xsl:call-template name="bookTitle"/>
  <xsl:with-param name="Author">John Smith<xsl:with-param>
    <!-- any other template actions -->
</xsl:template>
```

The `<xsl:for-each>` element

- `<xsl:for-each select="nodeSetExpression">`

- Used to iterate over the result of the select expression
- Selected node becomes the current node

```
<list>
  <book ID="666">
    <author>Jim Blue</author>
    <title>Blue Flowers</title>
  </book>
  <book ID="888">
    <author>John Smith</author>
    <title>New Cars</title>
  </book>
  <book ID="999">
    <author>Dan Big</author>
    <title>Large Stories</title>
  </book>
</list>
```

Books.xml

```
<xsl:template match="/">
  <xsl:for-each select="//book">
    <p><xsl:value-of
      select="title"/></p>
  </xsl:for-each>
</xsl:template>
```

Books.xsl

```
<p>Blue Flowers</p>
<p>New Cars</p>
<p>Large Stories</p>
```

Books.html

© Copyright IBM Corporation 2013

Figure 2-24. The `<xsl:for-each>` element

WE4013.0

Notes:

The `"//"` in `"//book"` is used as a wildcard to search for nodes anywhere in the XML document relative to the current context node.

The <xsl:if> element

- **<xsl:if test="patternToMatch">**
 - Used to conditionally process the matched expression
 - Can also be used with the logical not statement

```
<list>
  <book ID="666">
    <author>Jim Blue</author>
    <author>Mike Yellow</author>
    <author>Dan Farm</author>
    <title>Blue Flowers</title>
  </book>
</list>
```

Books.xml

Blue Flowers by Jim Blue, Mike
Yellow, and Dan Farm

```
<xsl:template match="list/book">
  <xsl:value-of select="title"/> by
  <xsl:for-each select="author">
    <xsl:value-of select=". " />
    <xsl:if test="position() != last()">, </xsl:if>
    <xsl:if test="position() = last() - 1"> and </xsl:if>
  </xsl:for-each>
</xsl:template>
```

Books.xsl

© Copyright IBM Corporation 2013

Figure 2-25. The <xsl:if> element

WE4013.0

Notes:

The functions `position()` and `last()` are XSLT functions.

The `position()` function returns an integer that represents the number of author nodes processed.

The `last()` function returns an integer for the number of author nodes.

The `xsl:if` conditional can be used to test for a certain situation within a template. It can be used with other actions.

More than one `xsl:if` action can appear within a template.

The <xsl:choose> element (1 of 2)

- Multiple tests can be implemented
- The `<xsl:otherwise>` element is optional and must be the last child element of `<xsl:choose>`
 - Used as a default if the other tests fail

```
<xsl:choose>
  <xsl:when test="testCondition">
    <!-- ... other actions ... -->
  </xsl:when>
  <xsl:otherwise>
    <!-- ... alternative actions ... -->
  </xsl:otherwise>
</xsl:choose>
```

© Copyright IBM Corporation 2013

Figure 2-26. The `<xsl:choose>` element (1 of 2)

WE4013.0

Notes:

The <xsl:choose> element (2 of 2)

```
<list>
  <book ID="666">
    <chapter>First Chapter</chapter>
    <chapter>Second Chapter</chapter>
    <appendix>XSLT reference</appendix>
  </book>
</list>
```

Books.xml

<p>Chapter: First Chapter</p>
<p>Chapter: Second Chapter</p>
<p>Appendix: XSLT reference</p>

Books.html

```
<xsl:stylesheet xmlns:xsl='...
  <xsl:template match="//book">
    <xsl:for-each select="*">
      <p>
        <xsl:choose>
          <xsl:when test='name()="chapter"'>Chapter: </xsl:when>
          <xsl:when test='name()="appendix"'>Appendix: </xsl:when>
          <xsl:otherwise>Index: </xsl:otherwise>
        </xsl:choose>
        <xsl:value-of select=". " />
      </p>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Books.xsl

© Copyright IBM Corporation 2013

Figure 2-27. The <xsl:choose> element (2 of 2)

WE4013.0

Notes:

Elements to generate output (XML to XML)

The following elements can be used to transform from XML-to-XML:

- **<xsl:element>**
 - Creates an element in transformed document
- **<xsl:attribute>**
 - Creates an attribute within an **<xsl:element>**
- **<xsl:copy>**
 - Copies current node and namespace node from source tree to result tree
- **<xsl:copy-of>**
 - Copies current node, namespace node, child tags, and attributes from source tree to result tree
- **<xsl:processing-instruction>**
 - Add a processing instruction node

© Copyright IBM Corporation 2013

Figure 2-28. Elements to generate output (XML to XML)

WE4013.0

Notes:

A common use of XSLT is to translate and transform from one XML vocabulary to another XML vocabulary.

XSLT provides some built-in elements to help with these types of transformations.

The **<xsl:copy-of>** function is similar to the **<xsl:copy>** tag except that the child and attribute nodes are also copied.

The `<xsl:element>` element

- Creates an element in the result tree
- Content (inside `<xsl:element>`) can be:
 - `<xsl:attribute>` (create attribute)
 - `<xsl:element>` (create child element)
 - Text

```
<xsl:element name="element-name">
  <!-- content: attributes, child elements, text -->
</xsl:element>
```

- Alternative: `<element-name>` literal result elements

```
<element-name>
  <!-- content: attributes, child elements, text-->
</element-name>
```

© Copyright IBM Corporation 2013

Figure 2-29. The `<xsl:element>` element

WE4013.0

Notes:

The attributes are always inside an element.

Elements might be inside of elements (child element).

The <xsl:attribute> element

- Creates an attribute in the result tree
- All attributes must precede the first child element

```
<xsl:attribute name="attribute-name">
  <!-- content: text value -->
</xsl:attribute>
```

- Example:

```
<xsl:attribute name="id">
  <xsl:value-of select="@no"/>
</xsl:attribute>
```

Create an attribute
named "id"

XPath:
attribute "no"
of current
node

The value of the **id** attribute is the value
of attribute "no" of the current node

© Copyright IBM Corporation 2013

Figure 2-30. The <xsl:attribute> element

WE4013.0

Notes:

The <xsl:attribute> tag must exist inside an <xsl:element> tag.

Using custom style sheets

- DataPower functionality is implemented by using XSL style sheets
 - XSL style sheets perform document processing actions, such as encryption, routing, and AAA
 - Other actions, such as Transform or Filter, explicitly require XSL style sheets as input parameters
- Develop custom actions by designing your own XSL style sheets with DataPower extension functions
 - Extension functions allow standard XSL style sheets to access features within the DataPower SOA appliance
- Map custom extension functions to DataPower extension functions with the XML Manager
 - For example, style sheets that use a SAXON node-set function can be mapped to a DataPower equivalent function
- Use only when the DataPower appliance does not provide the functionality
 - Do not modify existing built-in DataPower XSL style sheets
 - See the **store:** directory for built-in DataPower XSL style sheets

© Copyright IBM Corporation 2013

Figure 2-31. Using custom style sheets

WE4013.0

Notes:

Mapping custom extension functions to DataPower extension functions saves the effort of rewriting or modifying style sheets. To configure this mapping, use the XML Manager object.

DataPower supports the XSLT V1.1 specification and parts of the XSLT V2.0 specification.

DataPower also supports creating custom XSLT functions by using EXSLT.

How to develop style sheets with DataPower extensions

1. Use an XML editor to write the XSL style sheet:
 - For example:
 - Eclipse V3.2 Web Tools
 - IBM Rational Application Developer
2. Use the DataPower XSLT compiler to compile the style sheet
 - Use the supplied Eclipse plug-ins to upload and compile your custom style sheet on the DataPower SOA appliance
3. Add a Transform action to apply the XSL style sheet in the document processing policy



© Copyright IBM Corporation 2013

Figure 2-32. How to develop style sheets with DataPower extensions

WE4013.0

Notes:

Create custom XSL style sheets by using the XML editor of your choice. Eclipse V3.2 Web Tools is an open source solution that provides an XML editor for designing XSL style sheets and XML schema files. IBM Rational Application Developer is based on the same Eclipse platform, which provides the same level of XML functionality, in addition to XSL style sheet compile and debug features.

You can also use the DataPower Eclipse plug-ins on either product. There are two sets of plug-ins: the coprocessor and management. The former offloads complex XML processing tasks to the DataPower SOA appliance, while the latter allows you to control multiple DataPower appliances from the Eclipse workbench.

These plug-ins do not support auto-completion when using DataPower extension functions. See the DataPower reference guide for documentation.

The `cli copy` command is also useful in copying files.

XSLT variables

- XSLT variables are *immutable*: their value cannot change once set
- Retrieve the value of a variable by using the `$variable` notation
 - Use `{$variable}` to retrieve a value within an attribute that does not support a nodeset

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="/">
    <xsl:variable name="serviceOp">findByName</xsl:variable>
    The name of the called service is
    <xsl:value-of select="//soap-env:body/local-name(.) " />
    The value of the serviceOp variable is
    <xsl:value-of select="$serviceOp" />
  </xsl:template>
</xsl:stylesheet>
```

© Copyright IBM Corporation 2013

Figure 2-33. XSLT variables

WE4013.0

Notes:

XSLT variables can be global by defining them outside of any `<xsl:template>` tags.

The `<xsl:param>` is also used in style sheet programming and behaves in a way similar to the `<xsl:variable>` tag. However, unlike the `<xsl:variable>` tag, the `<xsl:param>` tag can provide an alternative value if it is passed as a style sheet parameter.

A node set represents a set of nodes on an XML document. Some XSL functions or tags expect a node set value type. For example, in the `<xsl:value-of>` tag, the `select` attribute requires a set of nodes.

DataPower variables

- DataPower variables are mutable; their value can be modified after declaration
- Create a DataPower variable
 - Within a document processing policy by using the **Set Variable** action
 - Use an Advanced action to configure a **Set Variable** action
 - Within a style sheet with the `dp:set-variable` function:
 

```
<dp:set-variable name="var://context/test/port"
  value="2068"/>
```
- Keep in mind that variable names are strings; there is no implied hierarchical structure
- Use the `dp:variable` extension function to retrieve the value of a DataPower variable

© Copyright IBM Corporation 2013

Figure 2-34. DataPower variables

WE4013.0

Notes:

DataPower variables allow for more flexibility over XSLT variables because they can be modified.

Also, use the `dp:set-local-variable` function to set local variables.

- Deleting variables inside scopes does not affect other variables
- Example: `var://context/example/level1` and
`var://context/example/level1/level2`
- Deleting `var://context/example/level1` does not affect
`var://context/example/level2`

DataPower variable scopes

- DataPower scopes
 - Local: single processing rule
 - Service: per transaction (request and response rule) – do not create variables in the service scope
 - Context: per transaction (request and response rule) – user-defined variables per transaction
 - System: global variable outside the scope of the transaction – accessible to any transform on the appliance
- Built in context variables in policy rule
 - INPUT: original document that is retrieved at start of the processing rule
 - OUTPUT: document that is returned to client
 - NULL: empty document
- Clean up variables by assigning them to an empty nodeset
 - Cleanup is required only in the `var://system` space
 - Variables that are defined in the `service` and `context` spaces are cleaned up automatically

© Copyright IBM Corporation 2013

Figure 2-35. DataPower variable scopes

WE4013.0

Notes:

A transaction is defined as the multi-step action in a processing rule for both the request and response.

The service scope contains built-in variables that are used in transactions, configuration, load balancer, and WebSphere MQ-specific services.

An example transaction variable is `var://context/serviceURI`.

DataPower variable storage views DataPower variables with an empty node set as a sign for deletion.

The multi-step probe can be used to view the value of a variable.

See the Information Center for list of read-only and read/write service variables.

Variables that are created in the context scope must define at least three levels, for example, `var://context/level1/level2` but not `var://context/level1/`. The latter example does not allow you to read the variable you define.

System variables are deleted when the appliance is shut down.

A list of the types of variables available for transactions follows.

Asynchronous transactions:

- Error handling
- Headers
- Information
- Persistent connections
- Routing
- Statistics
- URL

Style sheet that uses DataPower extension functions

```

<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:template match="/">
    <xsl:choose>
      <xsl:when test="/*[local-name()='Envelope']/*[local-name()='Body']
                    /*[local-name()='CheckRequestElement']">
        <dp:set-target>
          <host>10.10.36.11</host>
          <port>2068</port>
        </dp:set-target>
      </xsl:when>
      <xsl:otherwise>
        <dp:set-target>
          <host>10.10.36.11</host>
          <port>8080</port>
        </dp:set-target>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>

```

© Copyright IBM Corporation 2013

Figure 2-36. Style sheet using DataPower extension functions

WE4013.0

Notes:

DataPower extension functions use the namespace. See
<http://www.datapower.com/extensions>

The common namespace alias that is used for this namespace is `dp`.

In this example, the back-end host and port are set by using the `<dp:set-target>` extension function that is based on the name of the operation inside the SOAP message.

Testing style sheets with DataPower content

- To test a style sheet that contains any DataPower content, you must run the code on an appliance
 - DataPower variables
 - DataPower extension functions and extension elements
- Typical procedure:
 - Create style sheet in editor or development tool
 - Upload style sheet
 - Create service, policy, rule, transform action to test style sheet
 - Send message or test data to service to test style sheet behavior
 - If more testing needed, upload edited style sheet and test again
- Using the Interoperability Test Service (ITS)
 - Create style sheet in editor or development tool
 - Invoke ITS, passing style sheet and test data
 - If more testing needed, invoke ITS again, passing edited style sheet

© Copyright IBM Corporation 2013

Figure 2-37. Testing style sheets with DataPower content

WE4013.0

Notes:

Any style sheet that contains references to DataPower specific functions can be tested only on the appliance.



Interoperability Test Service example (1 of 2)

- A development-time service on the appliance for testing transformations
- Testing capabilities:
 - Send schema and XML file, validates XML file
 - Send XPath expression and XML file, returns XPath result
 - Send style sheet and XML file, returns transformation result
- Accepts HTTP, HTTPS, and basic authentication
- Documentation in information center tutorials
- Sample code in the Resource Kit
 - Java JAR and Secure Shell clients
 - Sample code for the information center examples
- Disabled by default
- For development use only, do **not** enable on production appliances

© Copyright IBM Corporation 2013

Figure 2-38. Interoperability Test Service example

WE4013.0

Notes:

Get the DataPower Resource Kit from your DataPower administrator.

Interoperability Test Service example (2 of 2)

```
./dp-interop-client.sh -x toBase64.xsl -i message.xml  
-h myDP.com -p 9990
```

- `./dp-interop-client.sh` is the sample Secure Shell client
- The style sheet to test is `toBase64.xsl`
- The file to transform is `message.xml`
- ITS is running on the appliance at `myDP.com`
- ITS is listening for HTTP requests on port **9990** (default)
- The **response** to the script is the result of the transformation

- For Java implementations, `java -jar DPInteropClient.jar` replaces `./dp-interop-client.sh`

© Copyright IBM Corporation 2013

Figure 2-39. Interoperability Test Service example

WE4013.0

Notes:

This sample code is from the Information Center tutorial.

Unit summary

Having completed this unit, you should be able to:

- Describe the Extensible Stylesheet Language (XSL) model
- Construct XPath expressions
- Create XSL style sheets to apply XSL transformations
- Use and apply XSL templates in XSLT
- Describe the use of DataPower variables and extensions in XSL style sheets
- Test a style sheet by using the Interoperability Test Service

© Copyright IBM Corporation 2013

Figure 2-40. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. What template would you use for extracting a specific value from the source tree?
 - A. `<xsl:choose ... />`
 - B. `<xsl:copy ... />`
 - C. `<xsl:value-of select="..." ... />`
 - D. `<xsl:text />`
2. List the three parts of XSL.
3. What is the difference between XSLT and DataPower variables?

© Copyright IBM Corporation 2013

Figure 2-41. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

1.

2.



Checkpoint answers

1. C. What template would you use for extracting a specific value from the source tree?
 - A. `<xsl:choose ... />`
 - B. `<xsl:copy ... />`
 - C. `<xsl:value-of select="..." ... />`
 - D. `<xsl:text />`
2. The three parts of XSL are:
 - XSL-FO
 - XSLT
 - XPath
3. XSLT variables are immutable; once they are set, their value cannot change. The value of a DataPower variable can change throughout its lifespan.

© Copyright IBM Corporation 2013

Figure 2-42. Checkpoint answers

WE4013.0

Notes:

Exercise 2

Creating XSL transformations

© Copyright IBM Corporation 2013

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

7.0.1

Figure 2-43. Exercise

WE4013.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

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

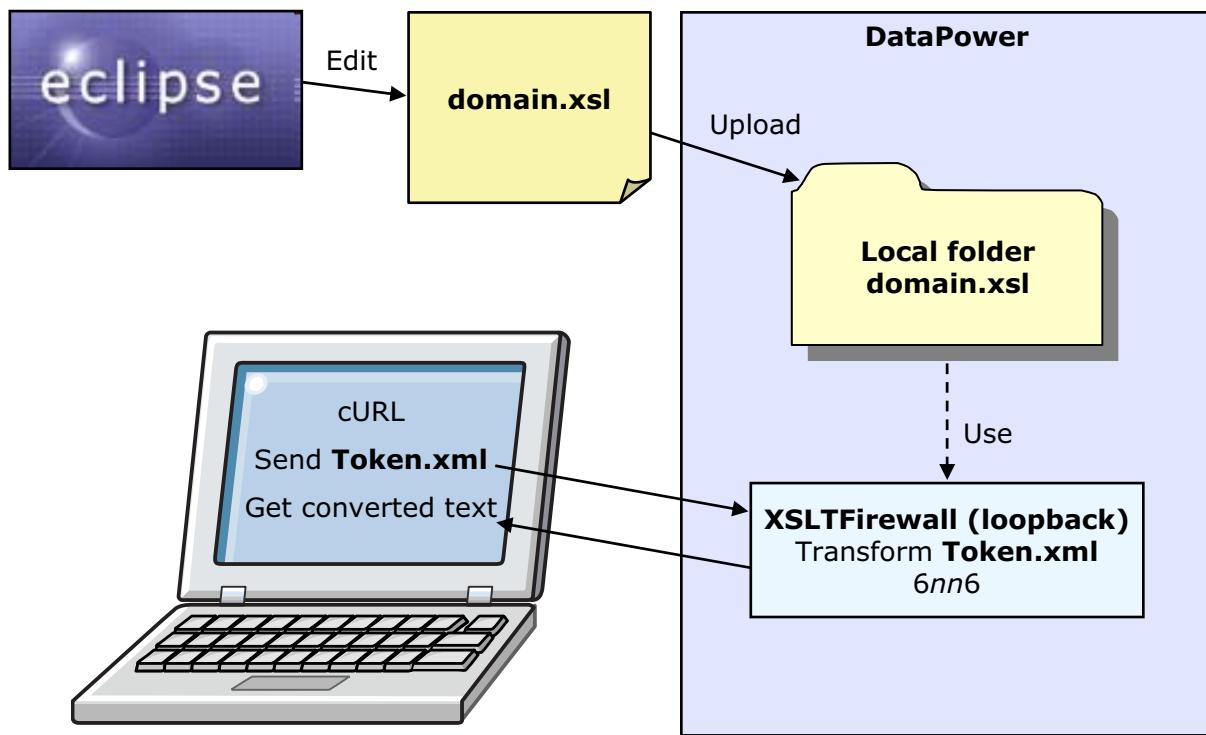
© Copyright IBM Corporation 2013

Figure 2-44. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 2-45. Exercise overview

WE4013.0

Notes:

Unit 3. DataPower services overview

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

- Define what a dataPower service is
- List the supported services on the WebSphere DataPower SOA Appliance
- Compare and contrast the features that each WebSphere DataPower service supports

How you will check your progress

- Checkpoint

Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2013

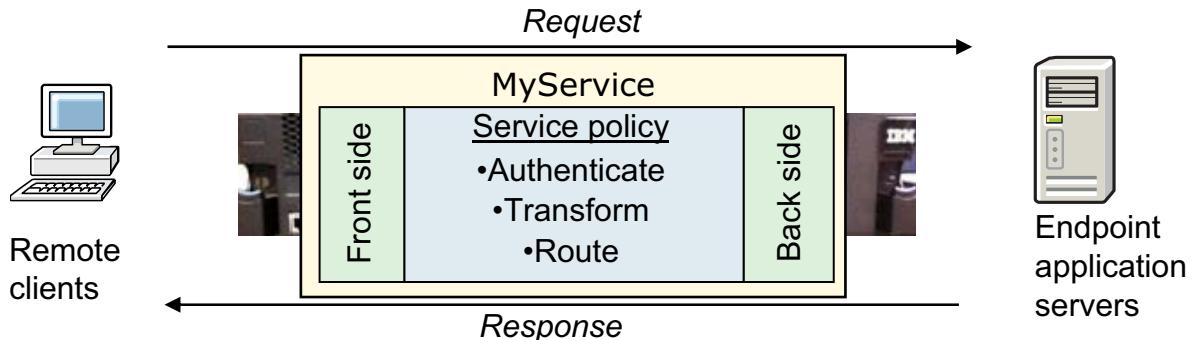
Figure 3-1. Unit objectives

WE4013.0

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
- “Front side” defines the client interface to the DataPower service
- Most services have a **service policy**, which you configure to deliver the DataPower functions that the service needs
- “Back side” defines the DataPower outbound service interface to the endpoint application servers

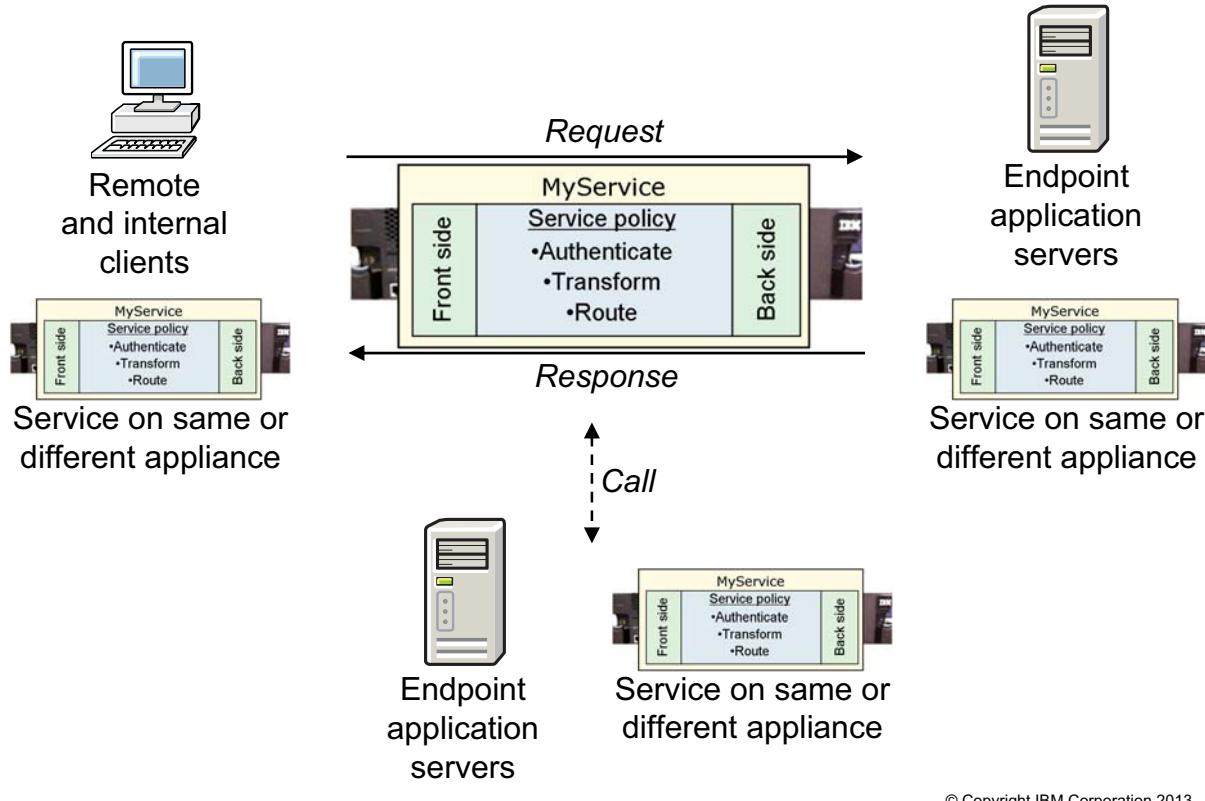
© Copyright IBM Corporation 2013

Figure 3-2. Services in a DataPower appliance

WE4013.0

Notes:

Front sides and back sides, and sideways



© Copyright IBM Corporation 2013

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

WE4013.0

Notes:

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

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

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

Services available on the DataPower appliance

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



© Copyright IBM Corporation 2013

Figure 3-4. Services available on the DataPower appliance

WE4013.0

Notes:

AAA: authentication, authorization, and auditing

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

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

All services support monitors and logging.

XSL proxy service

- Use the XSL proxy to accelerate XML processing
 - Use XML schema files to validate XML messages
 - Perform XSL transformations
 - Communicate with client and back-end servers 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 2013

Figure 3-5. XSL proxy service

WE4013.0

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



XML Firewall

© Copyright IBM Corporation 2013

Figure 3-6. XML firewall service

WE4013.0

Notes:

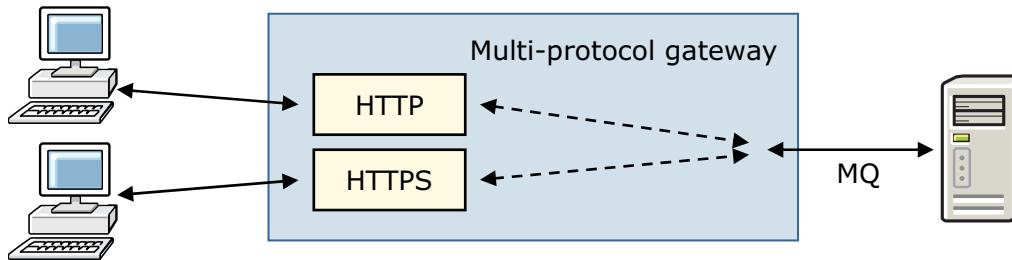
The features that are listed for the XML firewall are not exhaustive. The XML firewall also supports the same features that are mentioned previously for the XSL proxy.

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 associated actions that are used in a firewall policy.

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

Multi-protocol gateway service

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



© Copyright IBM Corporation 2013

Figure 3-7. Multi-protocol gateway service

WE4013.0

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.

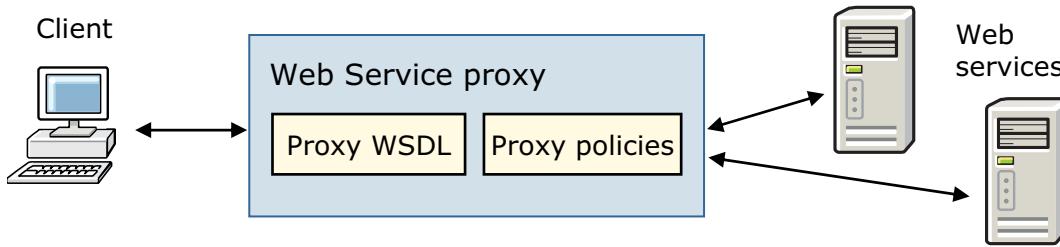
The supported protocols are HTTP(S), FTP(S), SFTP, NFS, raw XML, WebSphere MQ, WebSphere MQ File Transfer Edition (MQFTE), TIBCO EMS, WebSphere JMS, and IMS Connect.

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

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

Web service proxy service

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



© Copyright IBM Corporation 2013

Figure 3-8. Web service proxy service

WE4013.0

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 the same effect can be specified by using a DataPower variable within the service.

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

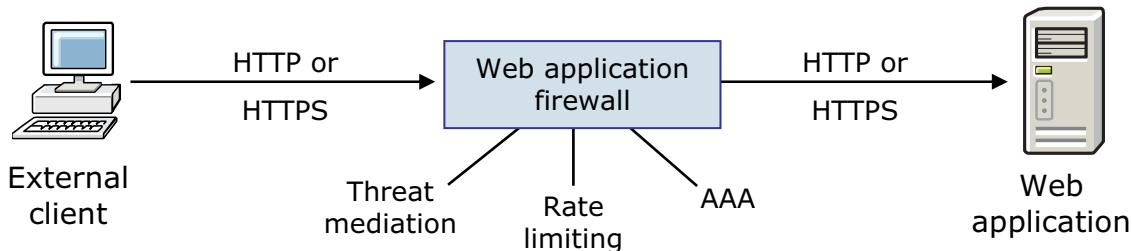
WSDLs can be retrieved from a Universal Description, Discovery and Integration (UDDI) registry and from WebSphere Service Registry and Repository (WSRR).

Governance policy, such as WS-SecurityPolicy and WS-MediationPolicy, can also be retrieved from a UDDI registry and WSRR.

WSRR can “push” changes to a web service proxy.

Web application firewall service

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



© Copyright IBM Corporation 2013

Figure 3-9. Web application firewall service

WE4013.0

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.

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

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

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

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

Other services

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

© Copyright IBM Corporation 2013

Figure 3-10. Other services

WE4013.0

Notes:

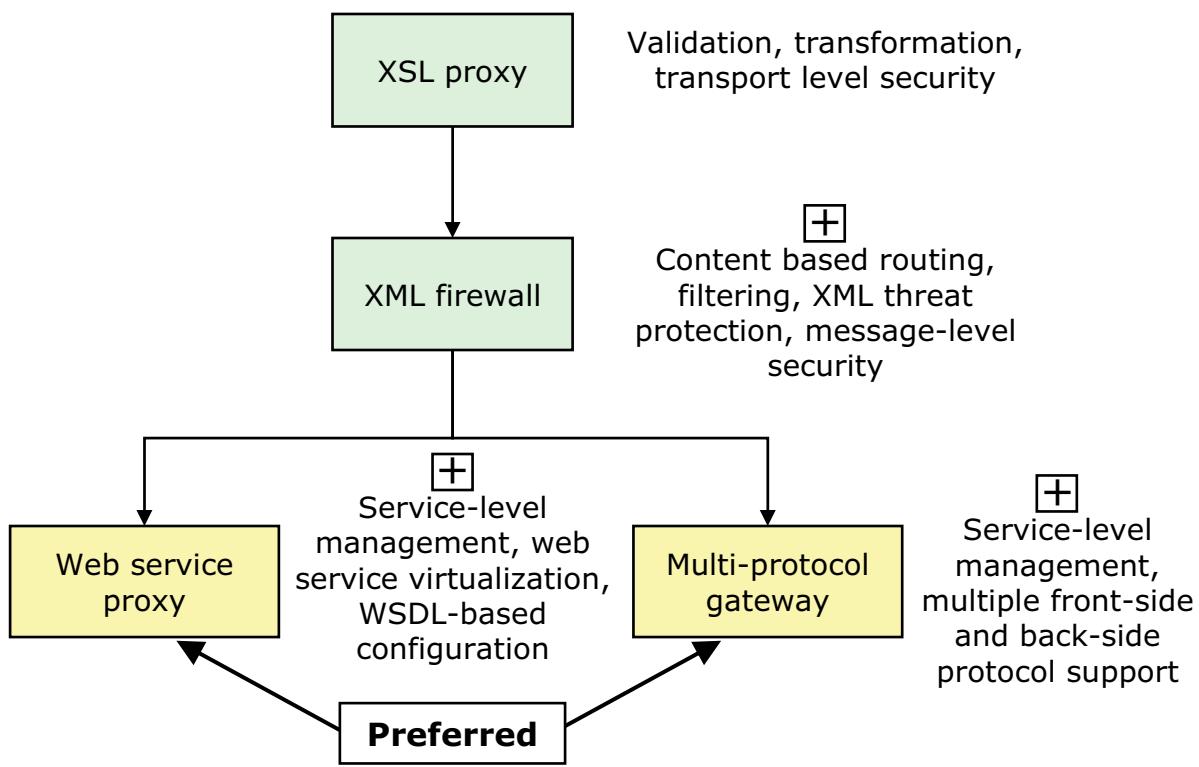
Web token service and interoperability test service are discussed in other units.

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

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

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

DataPower services feature hierarchy



© Copyright IBM Corporation 2013

Figure 3-11. DataPower services feature hierarchy

WE4013.0

Notes:

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

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

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

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

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

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

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

Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 3-12. Unit summary

WE4013.0

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 should be selected for this requirement? A service needs to schema-validate and transform a message before it is placed on a WebSphere MQ queue for mainframe processing. Input comes over HTTPS from external clients, and over HTTP from internal clients.
 - A. XML firewall
 - B. Multi-protocol gateway
 - C. Web service proxy
4. Which service type should be selected for this requirement? An enterprise has operations within several existing web services that it wants to expose to external clients as a single web service.
 - A. XML firewall
 - B. Multi-protocol gateway
 - C. Web service proxy

© Copyright IBM Corporation 2013

Figure 3-13. Checkpoint questions

WE4013.0

Notes:

Checkpoint answers

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

© Copyright IBM Corporation 2013

Figure 3-14. Checkpoint answers

WE4013.0

Notes:

Unit 4. Structure of a service

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

- Checkpoint
- Exercise 3: Create a simple service



Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 4-1. Unit objectives

WE4013.0

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

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

© Copyright IBM Corporation 2013

Figure 4-2. Object-based configuration

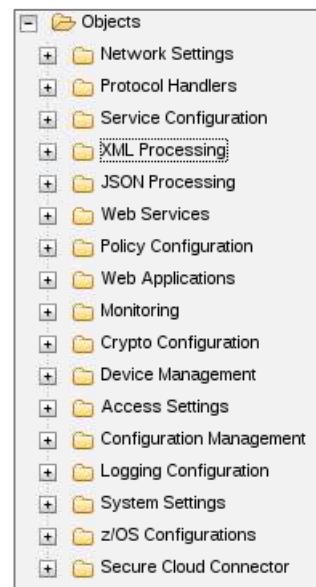
WE4013.0

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 (XML firewall and others) have wizards



The screenshot shows a configuration interface for an XML Firewall. The title is 'Configure XML Firewall'. Below it is a table with the following data:

| XML Firewall Name | Op-State | Logs | Req-Type | Local Address | Port |
|-------------------|----------|------|-------------|---------------|------|
| LoopbackXMLFW | up | 🔍 | unprocessed | dp_public_ip | 5971 |

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

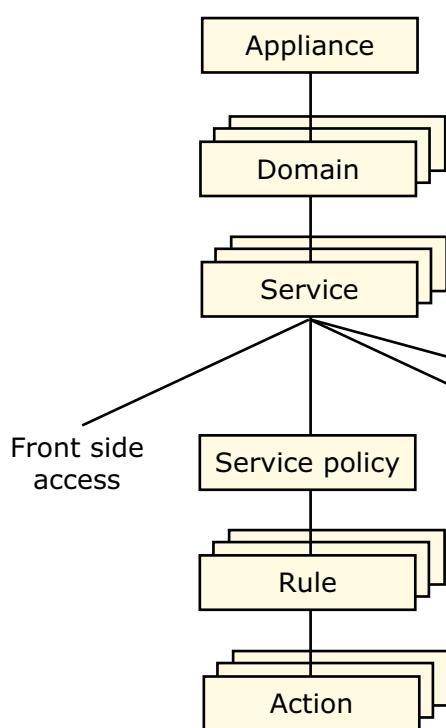
© Copyright IBM Corporation 2013

Figure 4-3. Approach to configuring objects

WE4013.0

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**, **Results**, and more
 - Custom XSLT always available using the **Transform** action

© Copyright IBM Corporation 2013

Figure 4-4. Basic architectural model

WE4013.0

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.



Front side access

- Specify how a service accepts requests
- Part of the service definition
 - XML firewall

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

- Protocol handler object
 - Multi-protocol gateway: front side protocol handler
 - Web service proxy: endpoint handler
 - Specification depends on protocol

| |
|---|
| Front side settings |
| Front Side Protocol MySSLFSH (HTTPS (SSL) Front Side Handler) |
| Local Local Endpoint Handler AddressSearchFSH |

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

© Copyright IBM Corporation 2013

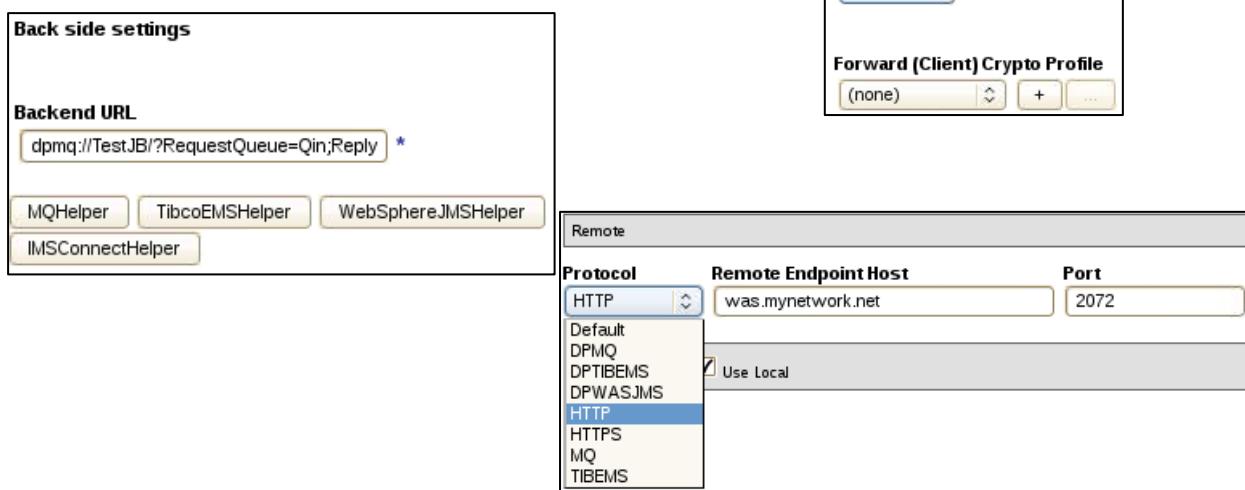
Figure 4-5. Front side access

WE4013.0

Notes:

Connection to the back side

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



© Copyright IBM Corporation 2013

Figure 4-6. Connection to the back side

WE4013.0

Notes:

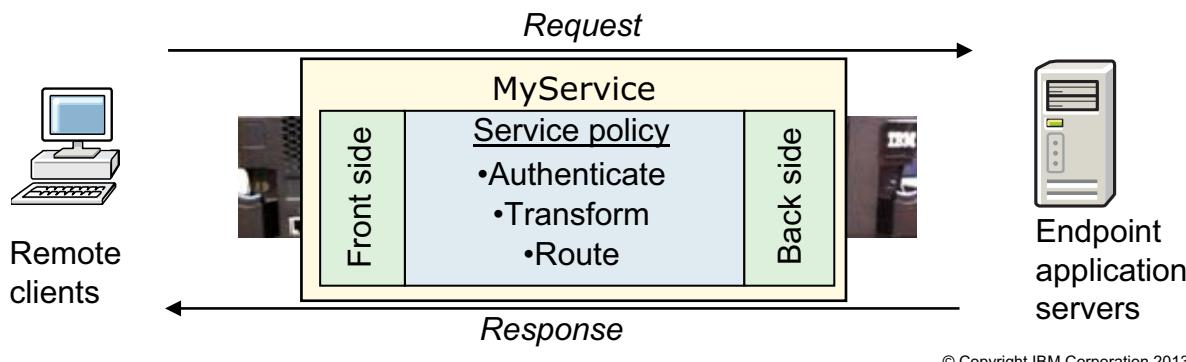
The back side connection can be dynamically determined, rather than hardcoded. In this case, the connection is made from a style sheet within the service policy.

Message processing phases

Each message passes through three phases:

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

Service settings control client-side and server-side behavior



© Copyright IBM Corporation 2013

Figure 4-7. Message processing phases

WE4013.0

Notes:

Response messages from the server then pass through these phases in reverse.

Response processing is the same as request processing except that the server must deal with errors from the back-end service.

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

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

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

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

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

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

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

Configuring a service policy (processing policy)

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

© Copyright IBM Corporation 2013

Figure 4-8. Configuring a service policy (processing policy)

WE4013.0

Notes:

Policy area

Policy Name: MyMPGWPolicy

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 → ↗ ↖ → ↘ 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 2013

Figure 4-9. Policy editor

WE4013.0

Notes:

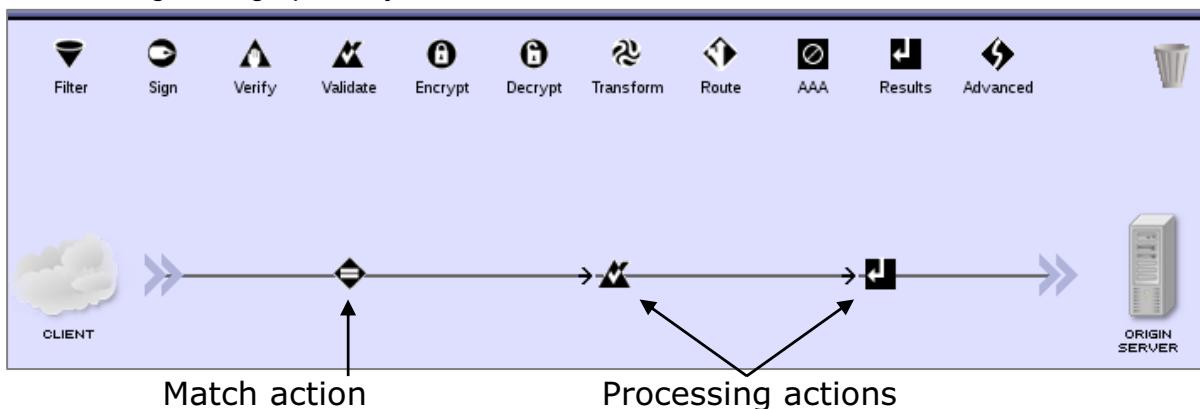
The policy area is where the service policy is named, and the configuration applied. In the web service proxy, there is no policy area in the policy editor.

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 2013

Figure 4-10. Configuring rules within a service policy

WE4013.0

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

| Configured Rules | | | | | |
|------------------|------------------|------------------|---------|--|--|
| Order | Rule Name | Direction | Actions | | |
| | LDAPTest_request | Client to Server | | | |
| | LDAPTest_Rule_1 | Server to Client | | | |
| | LDAPTest_Rule_2 | Error | | | |

© Copyright IBM Corporation 2013

Figure 4-11. Processing rules

WE4013.0

Notes:

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

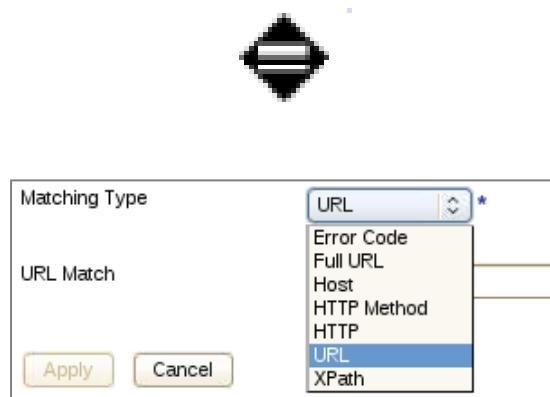
Processing in rules occurs sequentially in the order that the actions appear. Actions that allow for programmatic processing, such as looping and if-then-else statements, are available.



Match action

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

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



© Copyright IBM Corporation 2013

Figure 4-12. Match action

WE4013.0

Notes:

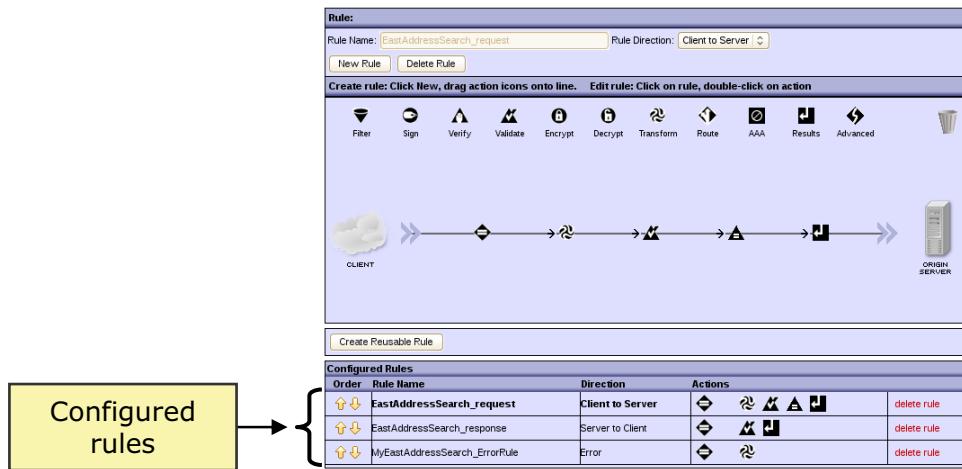
A Match action allows you to define criteria that is matched against the incoming traffic to determine whether the actions configured in the rule are applicable.

Each rule is configured with a Match action.

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

Processing actions (1 of 2)

- 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 a 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 2013

Figure 4-13. Processing actions

WE4013.0

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 (2 of 2)

| 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 or XPath-based routing |  Route |
| AAA | Invokes a AAA policy |  AAA |
| Results | Sends a message in specific context to an external destination |  Results |
| Advanced | A grouping of lesser-used actions |  Advanced |

© Copyright IBM Corporation 2013

Figure 4-14. Processing actions

WE4013.0

Notes:

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

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

The advanced actions are:

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

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

More processing actions

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

© Copyright IBM Corporation 2013

Figure 4-15. More processing actions

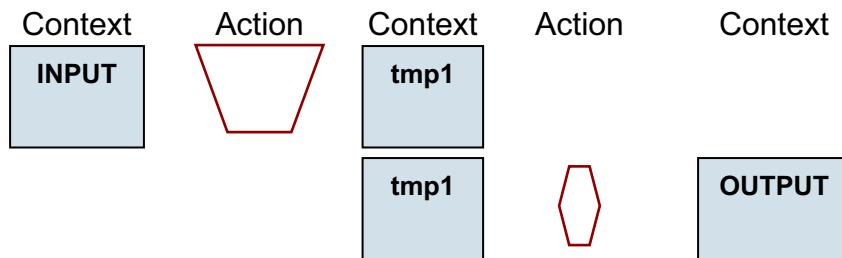
WE4013.0

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 2013

Figure 4-16. Multi-step processing rules

WE4013.0

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

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

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: 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 2013

Figure 4-17. Pre-defined context variables

WE4013.0

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

Perform schema-based validation of XML documents:

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

The screenshot shows a user interface for validating XML documents. On the left, there's a sidebar with the title 'Validate'. On the right, under 'Schema Validation Method', there's a list of five options: 'Validate Document via Attribute Rewrite Rule', 'Validate Document via Schema Attribute', 'Validate Document via Schema URL' (which is selected and highlighted in blue), 'Validate Document via WSDL URL', and 'Validate Document with Encrypted Sections'. Below this list is a note with an asterisk (*). Under 'Schema URL', there's a text input field containing 'store:///jsonx.xsd', a 'Fetch...' button, and an 'Upload...' button.

© Copyright IBM Corporation 2013

Figure 4-18. Validate action

WE4013.0

Notes:

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

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

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

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



Transform action

Use XSLT to manipulate documents

- Use XSLT specified in this action
 - Identifies the XSL style sheet that is referenced in the XSL style sheet field
- Use XSLT specified in XML document processing instructions, if available
 - Incoming XML document contains a processing instruction that identifies the XSL style sheet to use in transformation
- Use XSLT specified in this action on a non-XML message
 - XSL style sheet is used on a non-XML message (binary transform)

The screenshot shows the 'Transform' configuration dialog box. On the left, there's a sidebar with sections: 'Use Document Processing Instructions', 'XSL style sheet', 'URL Rewrite Policy', and 'Asynchronous'. The 'XSL style sheet' section contains a dropdown menu set to 'local:///jorder.xsl', a 'Upload...' button, and other buttons for 'Fetch...', 'Edit...', 'View...', and 'Validate...'. The 'Asynchronous' section has a radio button set to 'off'.

© Copyright IBM Corporation 2013

Figure 4-19. Transform action

WE4013.0

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.



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 2013

Figure 4-20. Filter action

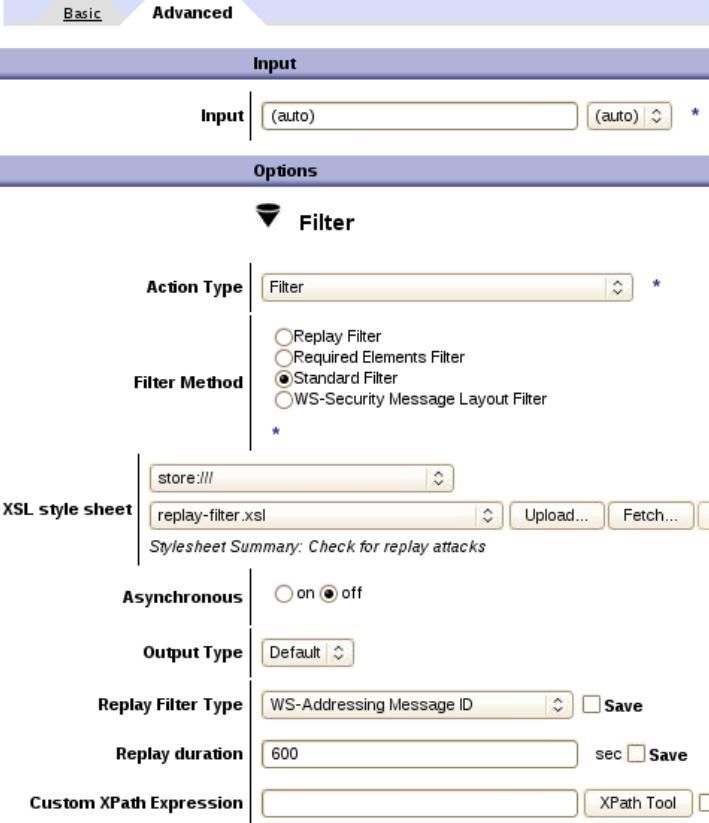
WE4013.0

Notes:

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

 WebSphere Education 

Filter action: Replay attack



The screenshot shows the 'Filter Advanced' configuration page. Key settings include:

- Input:** Set to '(auto)'.
- Options:** Under 'Filter', 'Action Type' is set to 'Filter' (selected from a dropdown). 'Filter Method' is set to 'Standard Filter' (radio button selected).
- XSL style sheet:** Set to 'store:///replay-filter.xsl'. Below it, a note says 'Stylesheet Summary: Check for replay attacks'.
- Asynchronous:** Set to 'on'.
- Output Type:** Set to 'Default'.
- Replay Filter Type:** Set to 'WS-Addressing Message ID'.
- Replay duration:** Set to '600 sec'.
- Custom XPath Expression:** An empty input field.

- 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 2013

Figure 4-21. Filter action: Replay attack

WE4013.0

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 username token with a nonce value, or a custom XPath. A nonce is a bit string that is generated to produce a unique string. It is used in authentication and security situations to create a unique ID.

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

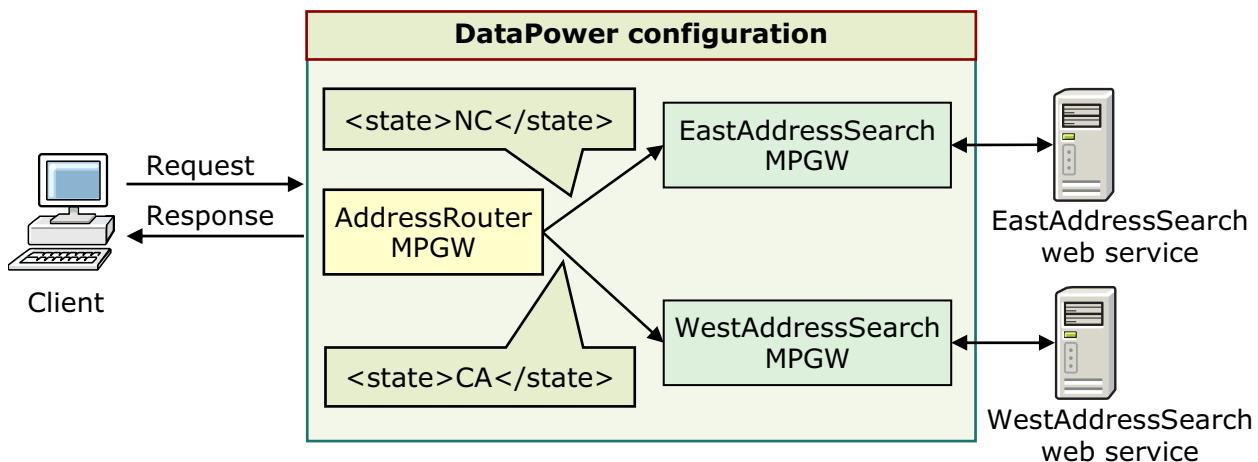
The WS-Addressing message ID is a unique message identifier.

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

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

Content-based routing

- With content-based routing, can choose 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 2013

Figure 4-22. Content-based routing

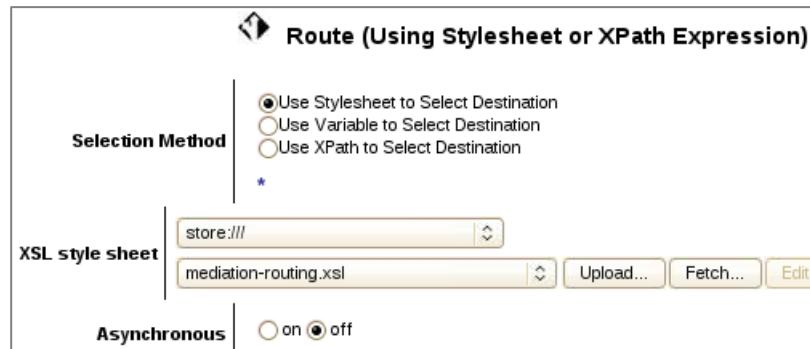
WE4013.0

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 2013

Figure 4-23. Route action configuration

WE4013.0

Notes:

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

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

Style sheet programming with dynamic routing

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

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

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

© Copyright IBM Corporation 2013

Figure 4-24. Style sheet programming with dynamic routing

WE4013.0

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

Example:

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

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

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

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

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



Results action

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

| Results | |
|--------------|--|
| Destination | <input type="text" value="cert:///"/> <input type="button" value="Up"/> <input type="button" value="(none)"/> <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <input type="button" value="Edit..."/> <input type="button" value="View..."/> <input type="button" value="Var Builder"/> |
| Asynchronous | <input checked="" type="radio"/> on <input type="radio"/> off |

© Copyright IBM Corporation 2013

Figure 4-25. Results action

WE4013.0

Notes:

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

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

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

Results asynchronous and multi-way results mode

- The **Results Asynchronous** action acts similarly to the **Results** action except that it:

- **Requires** a destination URL
- Does not wait for a response from the remote servers

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

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

© Copyright IBM Corporation 2013

Figure 4-26. Results asynchronous and multi-way results mode

WE4013.0

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 Information Center):

```
<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 of the remote servers fail.

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

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



Service settings

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

© Copyright IBM Corporation 2013

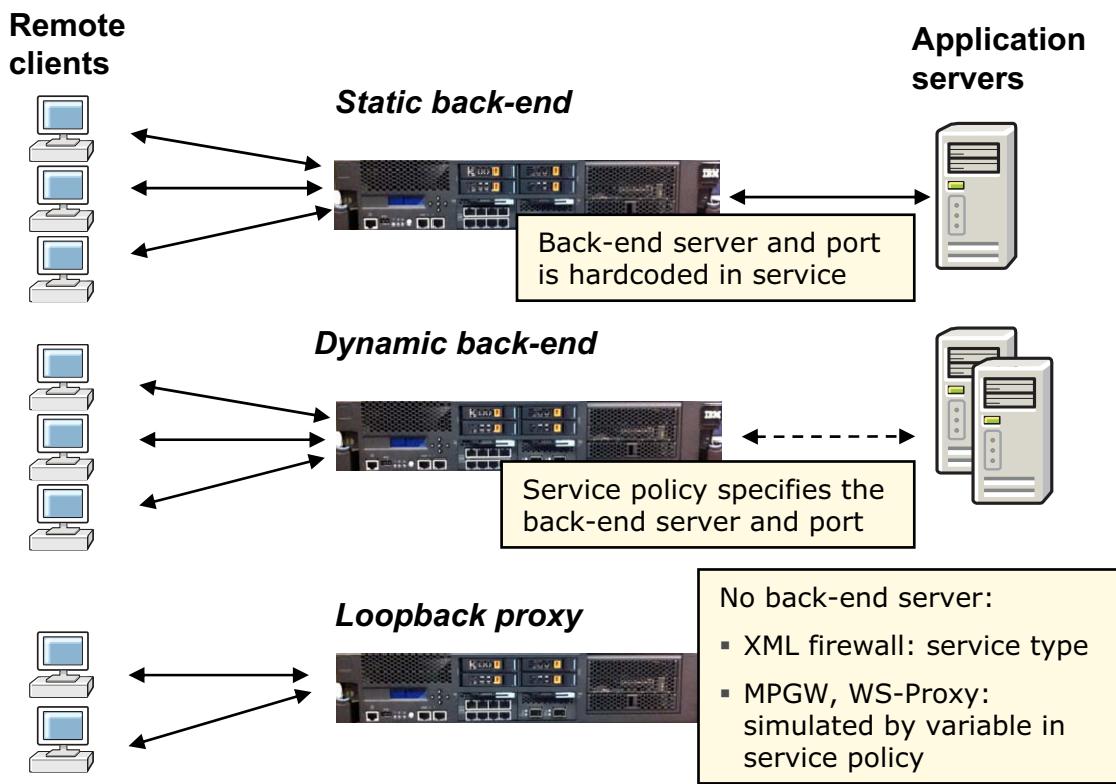
Figure 4-27. Service settings

WE4013.0

Notes:

Traffic monitors and XML threat protection are covered in other units.

Service types



© Copyright IBM Corporation 2013

Figure 4-28. Service types

WE4013.0

Notes:

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

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

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

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



URL rewriting

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

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

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

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

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

© Copyright IBM Corporation 2013

Figure 4-29. URL rewriting

WE4013.0

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 by a rewritten URL.

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

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

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

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

A URL rewrite policy can also be specified at the action level for Transform, Validate, and Header Rewrite actions.



XML Manager

- The XML Manager obtains and manages XML documents, style sheets, and other resources on behalf of one or more services
 - All services use the **default** XML Manager object
 - Accessed from the navigation bar by selecting **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 2013

Figure 4-30. XML Manager

WE4013.0

Notes:

Select **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 type="radio"/> on <input checked="" type="radio"/> off |
| Load Balance Groups | (empty) <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 2013

Figure 4-31. Default XML Manager configuration

WE4013.0

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 more.

WebSphere Education

XML parser limits

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

Configure XML Manager

Main XML Parser Document Cache Extension Functions Document

XML Manager

Apply Cancel

| | | |
|----------------------------------|---------------------------------------|-------|
| Name | <input type="text"/> | * |
| XML Bytes Scanned | <input type="text" value="4194304"/> | bytes |
| XML Element Depth | <input type="text" value="512"/> | |
| XML Attribute Count | <input type="text" value="128"/> | |
| XML Maximum Node Size | <input type="text" value="33554432"/> | bytes |
| XML Maximum Distinct Prefixes | <input type="text" value="0"/> | |
| XML Maximum Distinct Namespaces | <input type="text" value="0"/> | |
| XML Maximum Distinct Local Names | <input type="text" value="0"/> | |
| XML External Reference Handling | Forbid | ▼ |

© Copyright IBM Corporation 2013

Figure 4-32. XML parser limits

WE4013.0

Notes:

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

“DoS” is “denial of service”.

Parser limits:

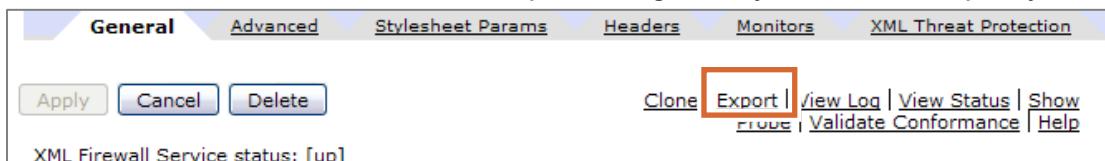
- **XML Bytes Scanned:** The maximum number of bytes scanned in one message by the XML parser. "0" indicates no restriction.
- **XML Element Depth:** The maximum depth of element nesting.
- **XML Attribute Count:** The maximum number of attributes that are allowed within an XML element.
- **XML Maximum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

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



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



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



© Copyright IBM Corporation 2013

Figure 4-33. Exporting a service configuration

WE4013.0

Notes:

Click the **Export** button 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.

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

Notice that there is an Import Configuration as well.



Cloning an XML firewall configuration

- Cloning

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

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

© Copyright IBM Corporation 2013

Figure 4-34. Cloning an XML firewall configuration

WE4013.0

Notes:

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

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

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

 WebSphere Education 

Troubleshooting a service configuration

- The System Log is the first place to start your problem determination exercise
 - Select 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

[C 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 2013

Figure 4-35. Troubleshooting a service configuration

WE4013.0

Notes:

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

Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 4-36. Unit summary

WE4013.0

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 2013

Figure 4-37. Checkpoint questions

WE4013.0

Notes:

Checkpoint answers

1. **True.** A service has a single policy with many rules, and each rule has many actions.
2. **True.** 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. **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.** 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 2013

Figure 4-38. Checkpoint answers

WE4013.0

Notes:

Exercise 3



Creating a simple XML firewall

© Copyright IBM Corporation 2013

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

7.0.1

Figure 4-39. Exercise

WE4013.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

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

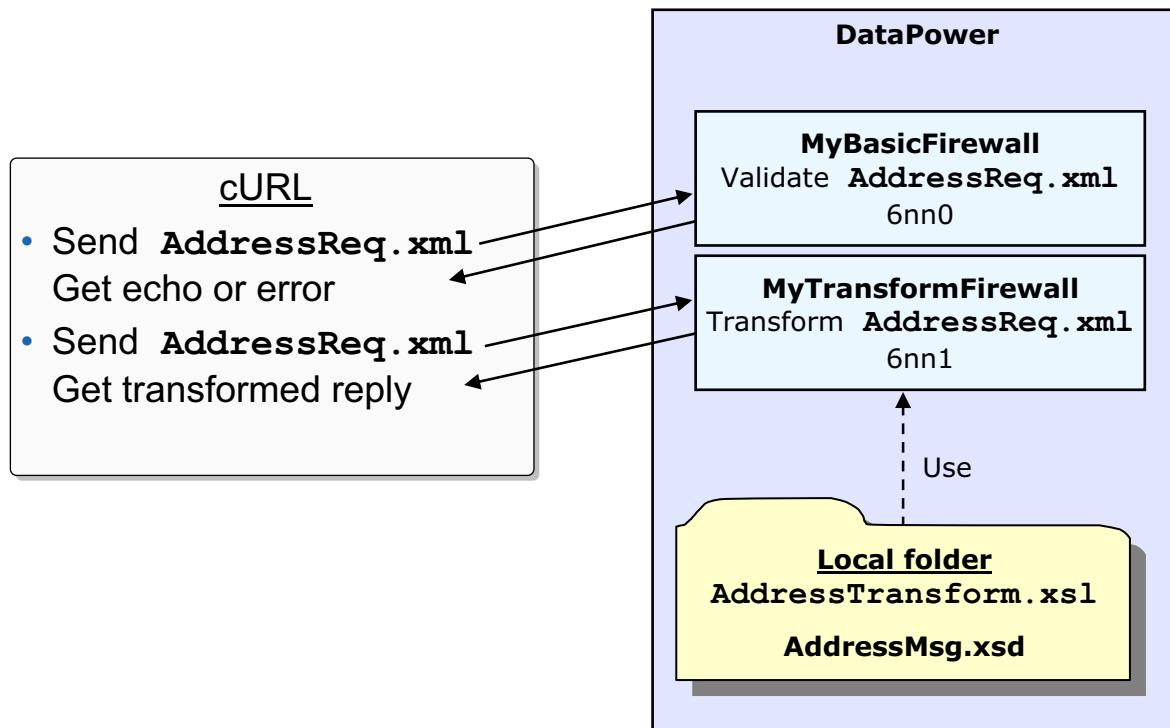
© Copyright IBM Corporation 2013

Figure 4-40. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 4-41. Exercise overview

WE4013.0

Notes:

Unit 5. Multi-protocol gateway service

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

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

Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 5-1. Unit objectives

WE4013.0

Notes:

What is a multi-protocol gateway?



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

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

© Copyright IBM Corporation 2013

Figure 5-2. What is a multi-protocol gateway?

WE4013.0

Notes:

Protocol handlers at a glance (1 of 2)

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

© Copyright IBM Corporation 2013

Figure 5-3. Protocol handlers at a glance (1 of 2)

WE4013.0

Notes:

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

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

For an overview on IBM WebSphere MQ messaging, review the WebSphere MQ white paper, "The continuing benefits of commercial messaging." This paper might be found by entering the title for a search at

<http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>.

The TIBCO Enterprise Message Service product website provides a summary of its features at <http://www.tibco.com/products/soa/messaging/enterprise-message-service/default.jsp>.

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 |
| NFS poller | Polls an NFS server for input |
| WebSphere JMS | Processes JMS messages received from WebSphere Application Server |
| IMS Connect | Accepts incoming IMS protocol requests and can initiate IMS connections on the back side |

© Copyright IBM Corporation 2013

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

WE4013.0

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. Since there is a limited amount of storage on the DataPower appliance, 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 back-end
- The following transport protocols are supported:
 - HTTP, HTTPS
 - FTP, NFS
 - IBM WebSphere MQ
 - IBM WebSphere JMS, TIBCO Enterprise Messaging System (EMS)
 - Stateless and stateful raw XML
 - IMS Connect
- Each instance of an HTTP, HTTPS, FTP, and raw XML protocol handler listens to a specific pair of IP address and port number
- Each WebSphere MQ protocol handler connects to a WebSphere MQ queue manager and the associated PUT and GET queues
- Each WebSphere JMS and TIBCO EMS handler connects to a JMS server and the associated GET and PUT queues

© Copyright IBM Corporation 2013

Figure 5-5. Front side protocol handlers

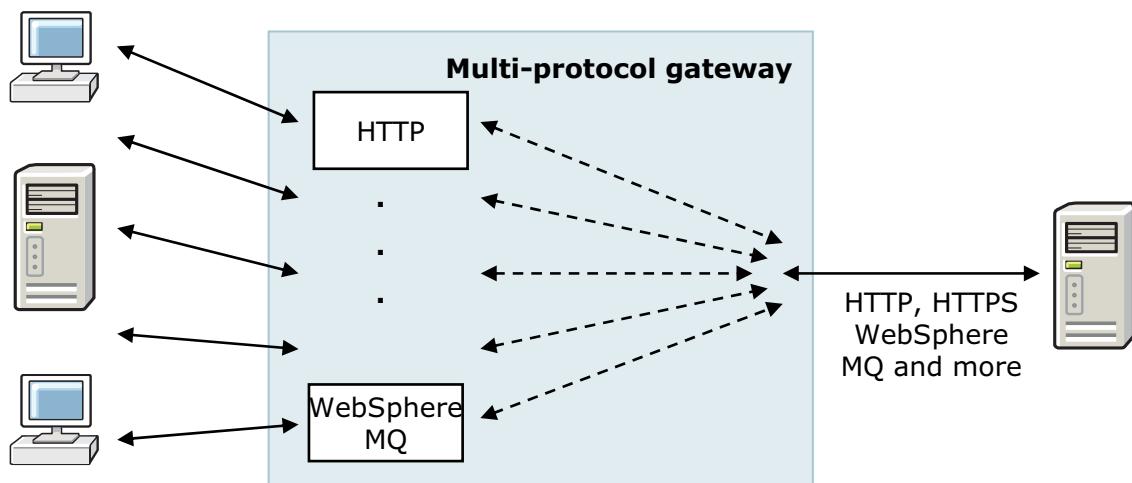
WE4013.0

Notes:

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

Static back-end gateway

- With a static back-end, 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 can employ any of the protocols shown (HTTP, HTTPS, WebSphere MQ, or TIBCO EMS)



© Copyright IBM Corporation 2013

Figure 5-6. Static back-end gateway

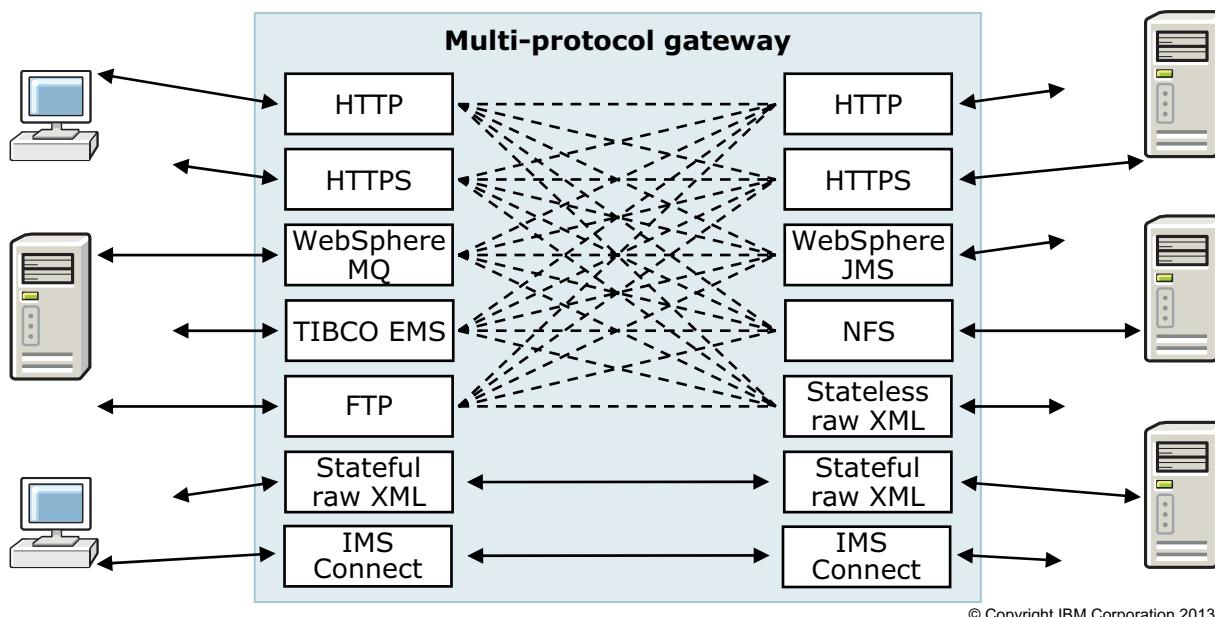
WE4013.0

Notes:

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

Dynamic back-end gateway

- A dynamic back-end 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



© Copyright IBM Corporation 2013

Figure 5-7. Dynamic back-end gateway

WE4013.0

Notes:

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

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

Multi-protocol gateway and XML firewall compared

- The multi-protocol gateway offers all of the same message processing capabilities as an XML firewall, regardless of the transport protocol chosen:
 - Encrypts and decrypts the entire message or individual token
 - Signs and verifies the entire message or individual token
 - Validates XML messages
 - Applies a custom XML transform style sheet
 - Authenticates clients and authorizes access to resources
- In addition, 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 2013

Figure 5-8. Multi-protocol gateway and XML firewall compared

WE4013.0

Notes:

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

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

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

The screenshot shows the 'Configure Multi-Protocol Gateway' page in the WebSphere Education interface. The top navigation bar includes tabs for General, Advanced, Stylesheet Params, Headers, Monitors, WS-Addressing, WS-ReliableMessaging, and XML Threat. Below the tabs are 'Apply' and 'Cancel' buttons. The main area is titled 'General Configuration' and contains the following fields:

- Multi-Protocol Gateway Name**: A text input field with a yellow circle labeled '1'.
- Summary**: A text input field with a yellow circle labeled '2'.
- Type**: A radio button group with 'dynamic-backend' and 'static-backend' options, where 'static-backend' is selected. A yellow circle labeled '3' points to it.
- XML Manager**: A dropdown menu set to 'default' with a yellow circle labeled '5'.
- Multi-Protocol Gateway Policy**: A dropdown menu set to '(none)' with a yellow circle labeled '6'.
- URL Rewrite Policy**: A dropdown menu set to '(none)'.

The 'Back side settings' section contains:

- Backend URL**: A text input field with a yellow circle labeled '4'.
- A list of helper buttons: MQHelper, TibcoEMSHelper, WebSphereJMSHelper, and IMSConnectHelper.

The 'Front side settings' section contains:

- Front Side Protocol**: A dropdown menu set to '(empty)' with a yellow circle labeled '7'.

At the bottom right of the page is the copyright notice: © Copyright IBM Corporation 2013.

Figure 5-9. Multi-protocol gateway editor

WE4013.0

Notes:

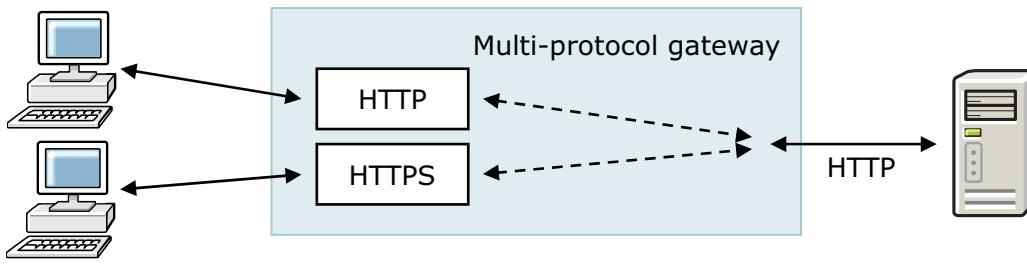
The multi-protocol gateway inherits most of the XML firewall features. The following list explains some new or modified settings that are specific to the multi-protocol gateway. Refer to the XML firewall presentation for an explanation on the remaining settings in the editor.

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.
8. 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 2013

Figure 5-10. Scenario 1: Provide HTTP and HTTPS access

WE4013.0

Notes:

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



Step 1: Configure the back side transport

General Configuration

Multi-Protocol Gateway Name: EastAddressSearch * (1)

Summary: East Address Search MPG

Type: static-backend (2)

Back side settings

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

MQHelper, TibcoEMSHelper, WebSphereJMSHelper, IIMConnectHelper

User Agent settings (4)

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

SSL Client Crypto Profile: (none) (5)

1. Provide a name and a summary for the newly created multi-protocol gateway
2. Select **static-backend** for a back-end service that is set at design time
3. Provide the HTTP address for the back-end service
4. Review the **User Agent settings**, if defined in the XML manager
 - **User Agent settings** match the identity 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 2013

Figure 5-11. Step 1: Configure the back side transport

WE4013.0

Notes:

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

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



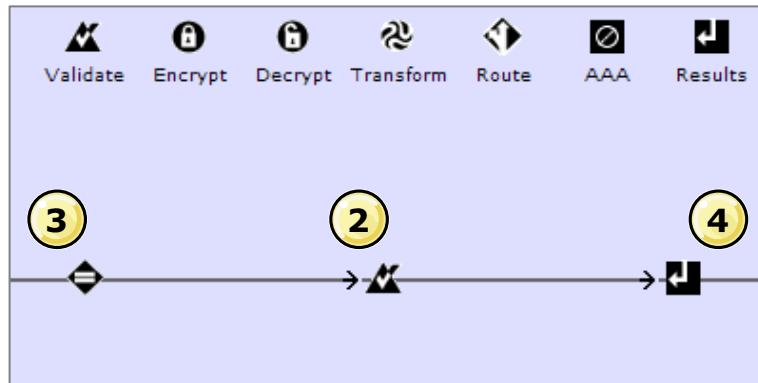
Step 2: Create a document processing rule

1. Create a multi-protocol gateway policy

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

| | |
|---|---|
| Policy: | |
| Policy Name: | <input type="text" value="AddressSearchMPGPolicy"/> 1 |
| <input type="button" value="Apply Policy"/> <input type="button" value="Cancel"/> | |

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



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

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

| | |
|--|--|
| Rule Direction: | |
| <input type="button" value="Both Directions"/> 5 | |

5

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

© Copyright IBM Corporation 2013

Figure 5-12. Step 2: Create a document processing rule

WE4013.0

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

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

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

Step 3: Create the front side handlers

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

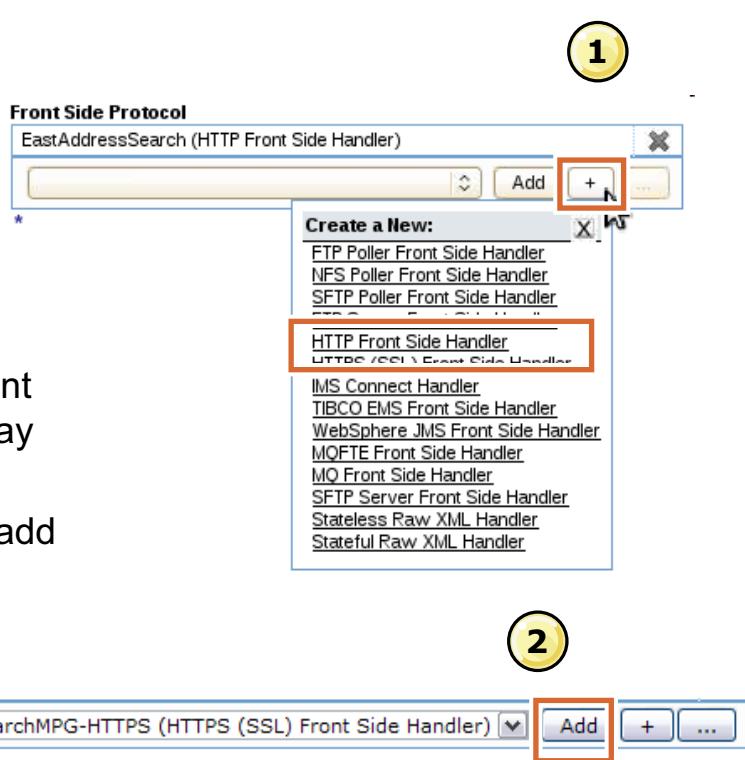


Figure 5-13. Step 3: Create the front side handlers

WE4013.0

Notes:

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

Usually, a newly created handler is automatically added to the protocol list upon completion of the handler configuration.



Step 4: Configure the front side handler

1. Activate the handler by setting the **Admin State** to **enabled**
2. Set the **Local IP Address** to **0.0.0.0** to listen to request on all external ports
3. Specify a unique port number that the handler monitors
4. Select the HTTP version reported to the client
5. Choose which HTTP version and method to permit
 - Web service clients use the HTTP POST method to send SOAP request messages
6. For **HTTPS Handlers** only, specify the **SSL Proxy** profile

HTTP Front Side Handler: EastAddressSearch [up]

Apply Cancel Undo

Administrative State **1** enabled disabled

Comments

Local IP Address **2** dp_public_ip

Port Number **3** 6957

HTTP Version to Client **4** HTTP 1.1

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

SSL Proxy **6** AddressSSL + ... *

Access Control List (none) + ...

© Copyright IBM Corporation 2013

Figure 5-14. Step 4: Configure the front side handler

WE4013.0

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

Step 5: Configure the SSL proxy profile

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

| | | |
|-----------------------------------|---|------------------|
| Name | AliceSSLProxy1 | 1 |
| Admin State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled | 2 |
| SSL Direction | Reverse * | 3 |
| Reverse (Server) Crypto Profile | AliceCryptoProfile | 4 |
| 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 2013

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

WE4013.0

Notes:

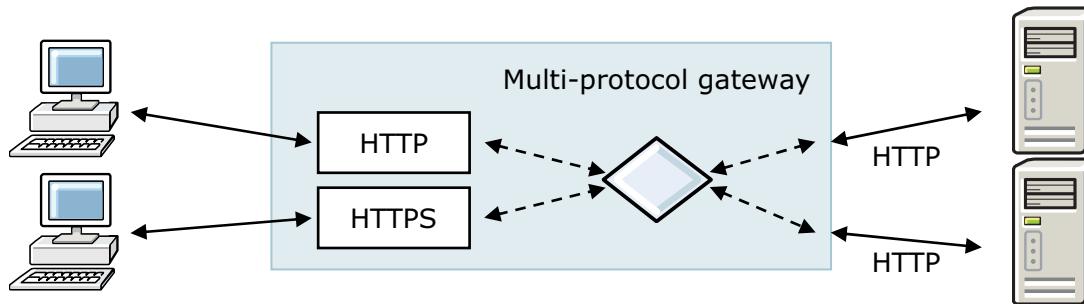
The SSL Proxy profile defines a set of keys and certificates that the gateway uses to build an SSL connection. The **Forward (Client) Crypto Profile** defines the keys and certificates that are used in an SSL connection between the gateway and the back-end service. The **Reverse (Server) Crypto Profile**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

Shown is a dynamic routing capability



© Copyright IBM Corporation 2013

Figure 5-16. Scenario 2: Dynamic back-end service

WE4013.0

Notes:

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

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

Step 1: Configure the back-end transport

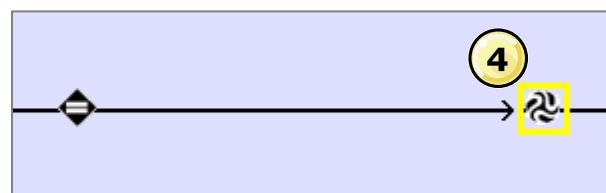
1. Open the multi-protocol gateway for the previous scenario
2. Set the back-end transport type to **dynamic-backend**
 - A processing rule must set the back-end address
3. Add a processing rule to the multi-protocol gateway policy
4. Add a **Transform** action in a request rule
5. Specify a custom style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

2

Type
 dynamic-backend
 static-backend
*

Back side settings

With a dynamic proxy back end Multi-Protocol Gateway type, the back end server address and port are determined by a stylesheet in a policy action.



5

Using Control File local:/// (none)

URL Rewrite Policy (none) **6**

© Copyright IBM Corporation 2013

Figure 5-17. Step 1: Configure the back-end transport

WE4013.0

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 2013

Figure 5-18. Sample service that targets a style sheet

WE4013.0

Notes:

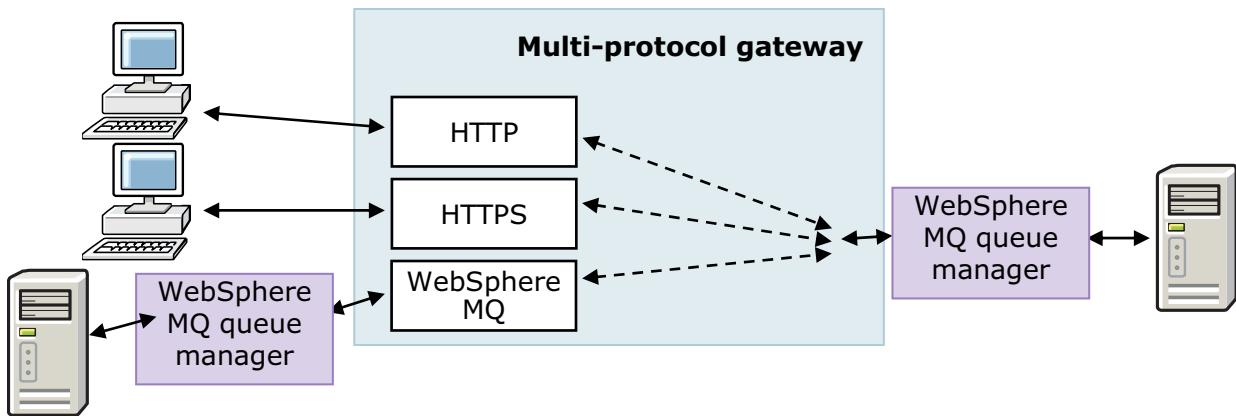
The `<dp:set-target>` built-in element defines the IP address (or host name) and the port for a particular back-end server. Additional attributes are available to set up an SSL connection to the back-end service.

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

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

Scenario 3: Provide WebSphere MQ access

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



© Copyright IBM Corporation 2013

Figure 5-19. Scenario 3: Provide WebSphere MQ access

WE4013.0

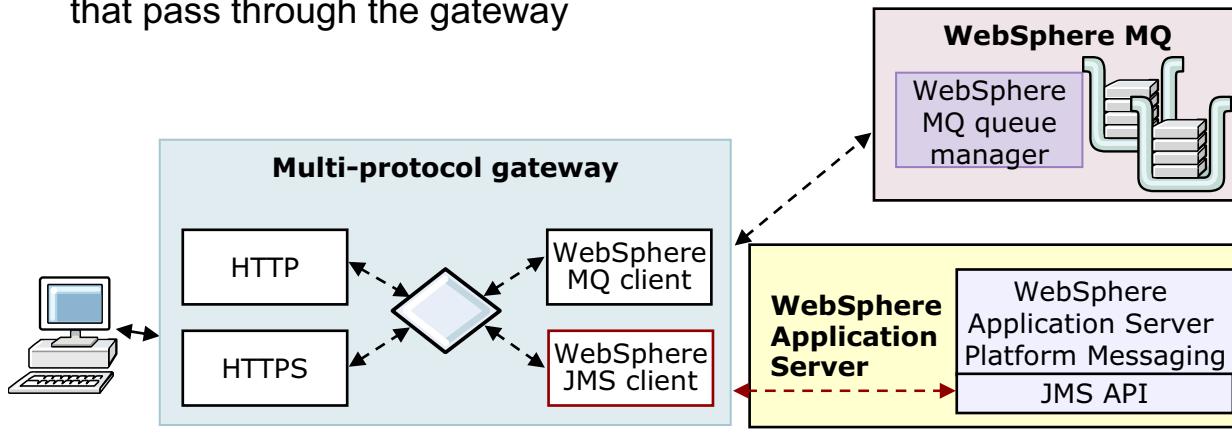
Notes:

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

Scenario 4: Provide WebSphere JMS access

Modify the multi-protocol gateway so that it:

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



© Copyright IBM Corporation 2013

Figure 5-20. Scenario 4: Provide WebSphere JMS access

WE4013.0

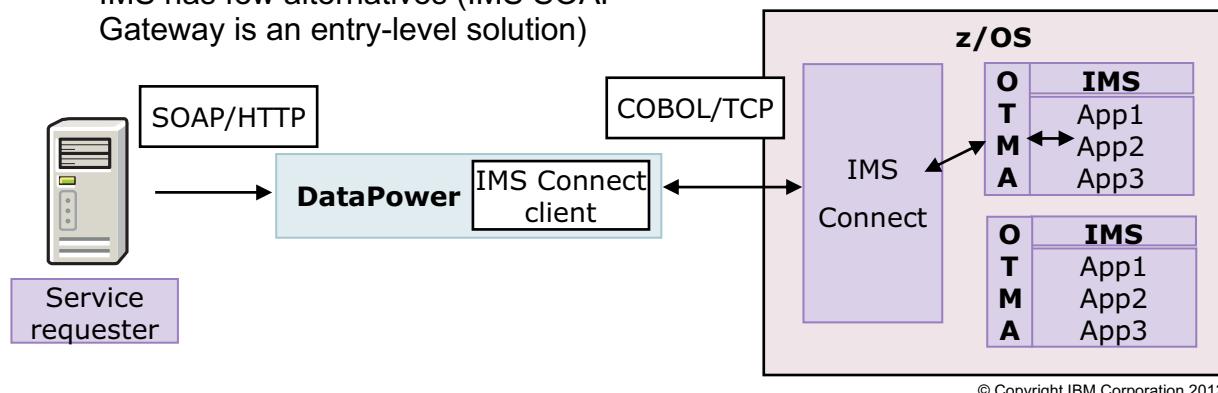
Notes:

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

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

Scenario 5: Provide IMS Connect access

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



© Copyright IBM Corporation 2013

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

WE4013.0

Notes:

OTMA: Open Transaction Management Access.

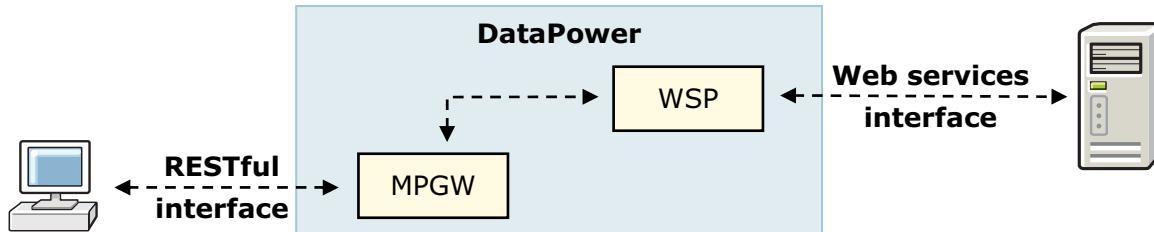
The IMS OTMA facility is a transaction-based connectionless client/server protocol that runs on IMS Version 5.1 or later. It functions as an interface for host-based communications servers that access IMS TM applications through the z/OS Cross Systems Coupling Facility (XCF).

IMS Connect communicates to OTMA by XCF.

Scenario 6: Provide a RESTful interface

Use a multi-protocol gateway to convert RESTful requests to a SOAP-based web service request

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



© Copyright IBM Corporation 2013

Figure 5-22. Scenario 6: Provide a RESTful interface

WE4013.0

Notes:

For a discussion of REST and DataPower, see: "Implementing REST services with WebSphere DataPower SOA Appliances" at http://www.ibm.com/developerworks/websphere/techjournal/0903_peterson/0903_peterson.html

Check developerWorks for more DataPower articles, and search the DataPower information center for REST support.

REST support in DataPower

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

© Copyright IBM Corporation 2013

Figure 5-23. REST support in DataPower

WE4013.0

Notes:

- **Process Messages whose body is empty** option: useful for RESTful message patterns in which some message flows might not incorporate a body but multi-step rules still need to run. It bypasses the built-in "One Way Exchange Pattern" in multi-step.
- Matching Rule type **HTTP Method**: supports processing rule that matches on the HTTP method: HEAD, DELETE, PUT, POST, and GET.
- **HTTP Method** on dp:url-open: allows control of the HTTP method on a dp:url-open.
- **Method Rewrite** Advanced Processing action: rewrites HTTP method requests to the back-end.
- **Method Rewrite** by Set Variable action: another way to rewrite the HTTP method.
- **JSON encoding** in Convert HTTP action: automates the conversion of JSON passed in a RESTful request to an XML representation called JSONX. There is also a style sheet to convert JSONX into JSON.
- **JSON Choice** for request/response types: An additional request and response type of JSON is added.



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 most other needs
 - 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 2013

Figure 5-24. Comparing services

WE4013.0

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 processing rule to select a back-end service at run time

© Copyright IBM Corporation 2013

Figure 5-25. Unit summary

WE4013.0

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. Which scenarios are better suited for a multi-protocol gateway as opposed to a web service proxy?

| Description | Definition |
|--|---|
| <ol style="list-style-type: none">1. Multi-protocol gateway2. Web service proxy | <ul style="list-style-type: none">A. WSDLB. WSRR conceptsC. Multiple front side handlersD. Easy service level rule configurationE. WebSphere MQ integration |

© Copyright IBM Corporation 2013

Figure 5-26. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

1.

2.

Checkpoint answers

1. **True.** 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. Which scenarios are better suited for a multi-protocol gateway as opposed to a web service proxy?

| Description | Definition |
|---|---|
| <p>1. C and E. Multi-protocol gateway</p> <p>2. A, B, and D. Web service proxy</p> | <p>A. WSDL (web service proxy)</p> <p>B. WSRR concepts (web service proxy)</p> <p>C. Multiple front side handlers (multi-protocol gateway)</p> <p>D. Easy service level rule configuration (web service proxy)</p> <p>E. WebSphere MQ integration (multi-protocol gateway)</p> |

© Copyright IBM Corporation 2013

Figure 5-27. Checkpoint answers

WE4013.0

Notes:

Multi-protocol gateway service



© Copyright IBM Corporation 2013

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

4.0 7/01

Figure 5-28. Exercise

WE4013.0

Notes:

Multi-protocol gateway service



© Copyright IBM Corporation 2013
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

4.0 7/01

Figure 5-29. Exercise objectives

WE4013.0

Notes:

Multi-protocol gateway service



© Copyright IBM Corporation 2013

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

4.0 701

Figure 5-30. Exercise overview

WE4013.0

Notes:

Unit 6. Problem determination tools

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

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

Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 6-1. Unit objectives

WE4013.0

Notes:

Problem determination tools

After completing this topic, you should be able to:

- List the tools available for troubleshooting traffic and service policies on the DataPower appliance
- Configure the multi-step probe to analyze message flow within a service policy

© Copyright IBM Corporation 2013

Figure 6-2. Problem determination tools

WE4013.0

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
 - Select **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
 - Select **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
 - Select **Administration > Debug > Send a Test Message**

© Copyright IBM Corporation 2013

Figure 6-3. Common problem determination tools

WE4013.0

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

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

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

© Copyright IBM Corporation 2013

Figure 6-4. Appliance status information

WE4013.0

Notes:

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

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

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

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

Troubleshooting panel

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 2013

Figure 6-5. Troubleshooting panel

WE4013.0

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; then object status for objects that have changed to the down state, and finally the default system log.

Include a generated error report to DataPower support.

Troubleshooting: Networking

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

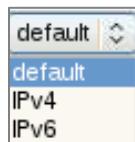


Figure 6-6. Troubleshooting panel

WE4013.0

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; then object status for objects that have changed to the down state, and finally the default system log.

Include a generated error report to DataPower support.



Troubleshooting: 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

Packet Capture

Start Packet Capture

No Packet Capture Available for Downloading

Interface Type: Ethernet *

Ethernet Interface: (none) *

Mode: Timed *

Maximum Duration: 30 seconds *

Maximum Size: 10000 KB *

Maximum Packet Size: 9000 bytes *

Filter Expression:

Stop Packet Capture

© Copyright IBM Corporation 2013

Figure 6-7. Troubleshooting: Networking

WE4013.0

Notes:

The first test that you perform when you cannot access the back side application server is to ping the remote server from the DataPower appliance to make sure that it is running and accessible.

Troubleshooting: Logging

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



© Copyright IBM Corporation 2013

Figure 6-8. Troubleshooting: Packet capture

WE4013.0

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.

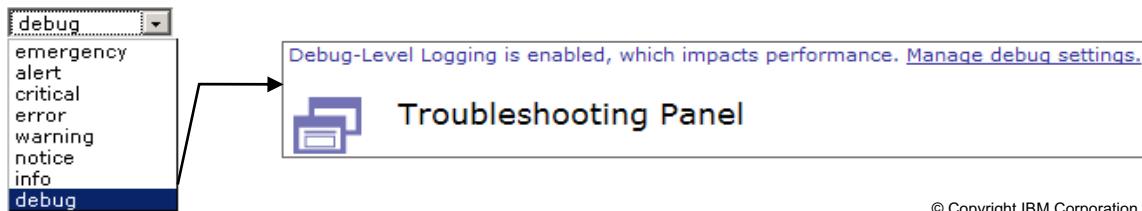
Before installing a packet capture tool, such as Wireshark (formerly called Ethereal), make sure that you have the necessary permission from your network staff.

Restarting the device automatically turns off packet capture.



Troubleshooting: System log

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



© Copyright IBM Corporation 2013

Figure 6-9. Troubleshooting: Logging

WE4013.0

Notes:

Log Level

Enable Internal Logging:

Enable internal logging to troubleshoot failing requests through the XML management interface. Internal logging provides detailed error messages for requests that are submitted to the XML management interface. When enabled, the system log in the default domain contains debug-level messages in the WebGUI category. The system log is available only in the default domain.

Enable RBM Debug logging:

Enable role-based management debug that logs to troubleshoot RBM issues such as user authentication failures or credential mapping errors. The log is available only in the default domain.

Global IP Address Log Filter:

Specify the IP address against which to filter. Global IP address that filters affects all log targets in all domains. Use this function for advanced troubleshooting only.

Filtering system log

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

The screenshot shows the 'System Log' interface. At the top, there are two yellow boxes labeled 'Category' and 'Log level'. Below them is a search bar with fields for 'Target' (set to 'default-log'), 'Filter' (set to '(none)'), and another 'Filter' dropdown also set to '(none)'. The main area displays a table of log entries. The columns are: time, category, level, tid, direction, client, msgid, and message. The table shows several entries from Tuesday, Aug 28, 2012, at 13:32:27. The last entry is: 'mpgw (EastAddressSearch): Dynamic Execution Error'. At the bottom right of the interface, it says '© Copyright IBM Corporation 2013'.

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

Figure 6-10. Troubleshooting: System log

WE4013.0

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.

Log level of the default system log.

- * **emergency**: An emergency-level message. The system is unusable.
- * **alert**: An alert-level message. Immediate action must be taken.
- * **critical**: A critical message. Immediate action should 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 that is required.
- * debug: A debug message for processing information to help during troubleshooting.

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 2013

Figure 6-11. Filtering system log

WE4013.0

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. Additional log targets can be defined and customized with the log entries from objects to post.

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

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

Troubleshooting: 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 shows a 'Reporting' section with two main options:

- Generate Error Report**: A button labeled "Generate Error Report". Below it, a message says "No Error Report Available for Viewing".
- Send Error Report**: A form with four fields:
 - SMTP Server**: An input field with an asterisk (*) indicating it is required.
 - Location**: An input field with an asterisk (*) indicating it is required.
 - E-mail Address**: An input field with an asterisk (*) indicating it is required.
 - Email Sender Address**: An input field.
 A "Send Error Report" button is located below the form.

© Copyright IBM Corporation 2013

Figure 6-12. Troubleshooting: Generate Log Event

WE4013.0

Notes:

The system log captures log messages from all objects. Log targets can be configured to capture messages from specific objects. To test these types of log targets, the generate log event tool is an excellent test tool.

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 2013

Figure 6-13. Troubleshooting: Reporting

WE4013.0

Notes:

Click the **Generate Error Report** button. A dialog window 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: 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 is meant to be enabled only in test environments
 - Significant performance penalties are incurred when mode is set to always or errors
- Default domain only

The screenshot shows a configuration interface titled "XML File Capture". It includes a "View File Capture" link with a magnifying glass icon, a "Mode" dropdown menu set to "None" with an asterisk (*) indicating it is required, and a large "XML File Capture" button.

© Copyright IBM Corporation 2013

Figure 6-14. Troubleshooting: Advanced

WE4013.0

Notes:



Troubleshooting: Send a test message

WebSphere. DataPower XI52

Send a Test Message

Request

URL: **1**

Request Headers:

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

Request Body: **3**

Response

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

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

© Copyright IBM Corporation 2013

Figure 6-15. Troubleshooting: XML File Capture

WE4013.0

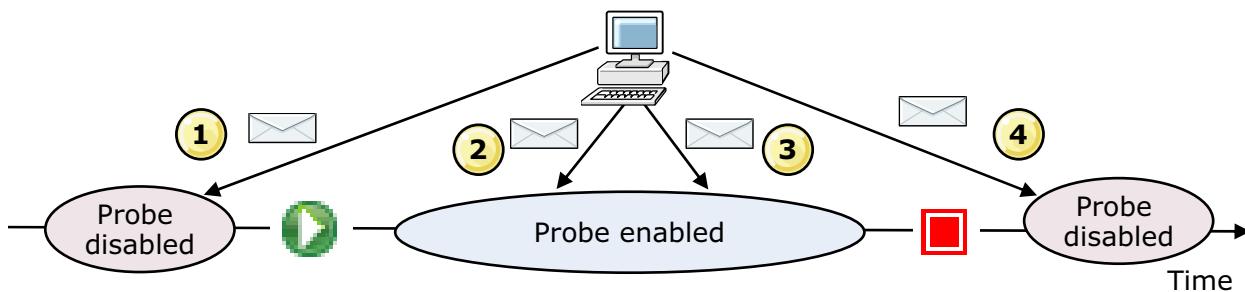
Notes:

To support file capture, the DataPower system creates a RAM-disk to store a WebGUI accessible virtual file system. A RAM-disk is a segment of RAM used for auxiliary storage.

The XML file capture tool is only available in the default domain.

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 since it impacts appliance performance



© Copyright IBM Corporation 2013

Figure 6-16. Troubleshooting: Send a test message

WE4013.0

Notes:

On 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 simply 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 has been created. The advantage of using tools like cURL is that it can send files directly from the file system.

Troubleshooting: Enabling the multi-step probe

Two ways to enable a probe for a service:

- Select the **Debug Probe** tab in the Troubleshooting panel
 - 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

Figure 6-17. Troubleshooting: Multi-step probe

WE4013.0

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; the probe captures no more messages.

The multi-step probe can be used to view:

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



Multi-step probe window

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

Enable Probe

View probe data

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

© Copyright IBM Corporation 2013

Figure 6-18. Troubleshooting: Enabling the multi-step probe

WE4013.0

Notes:

Probes are enabled for the following services:

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



Multi-step probe content

4 Input Content 'INPUT' of Step 1

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

Content Headers Attachments Local Variables Global Variables Service Variables

1 Content of context 'INPUT':

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

Select to show unformatted content

2 3

View the message content as it traverses each action

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

© Copyright IBM Corporation 2013

Figure 6-19. Multi-step probe window

WE4013.0

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.

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

Restarting the appliance disables all probes.

Problem determination with cURL

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

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

© Copyright IBM Corporation 2013

Figure 6-20. Multi-step probe content

WE4013.0

Notes:

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

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

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



Communicating with DataPower support

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

© Copyright IBM Corporation 2013

Figure 6-21. Problem determination with cURL

WE4013.0

Notes:

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

Topic summary

Having completed this topic, you should be able to:

- Identify a troubleshooting strategy to use when debugging problems on the DataPower appliance
- Use the multi-step probe to debug service policies

© Copyright IBM Corporation 2013

Figure 6-22. Communicating with DataPower support

WE4013.0

Notes:

For a comprehensive list of all of the information that is required to communicate with support, see <http://www.ibm.com/support/docview.wss?rs=2362&uid=swg21236322>

For detailed information about how to perform the steps to generate the files that are required by support, see <http://www.ibm.com/support/docview.wss?uid=swg21235587>

Log targets

After completing this topic, you should be able to:

- Create log targets to capture messages that are generated by objects on the appliance
- Use the **Log** action in a service policy to log the entire message

© Copyright IBM Corporation 2013

Figure 6-23. Topic summary

WE4013.0

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, SHTTP)
 - Off-device support for syslog, syslog-NG, SNMP
- Log targets do not capture the actual message
 - Add a **Log** action in a processing rule to capture the entire message

© Copyright IBM Corporation 2013

Figure 6-24. Log targets

WE4013.0

Notes:

Available log levels

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
- Info: an informational message only
- Debug: debug-level messages
 - This level generates numerous messages

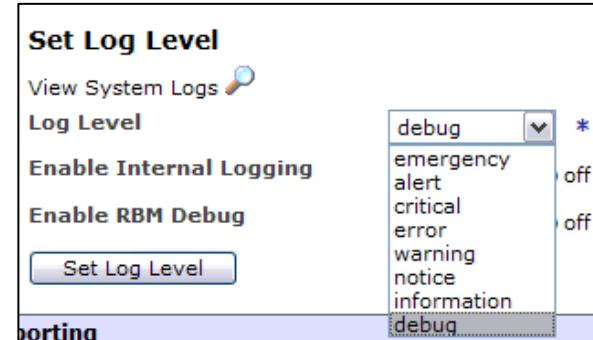


Figure 6-25. Logging basics

© Copyright IBM Corporation 2013

WE4013.0

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 message.

Log targets

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

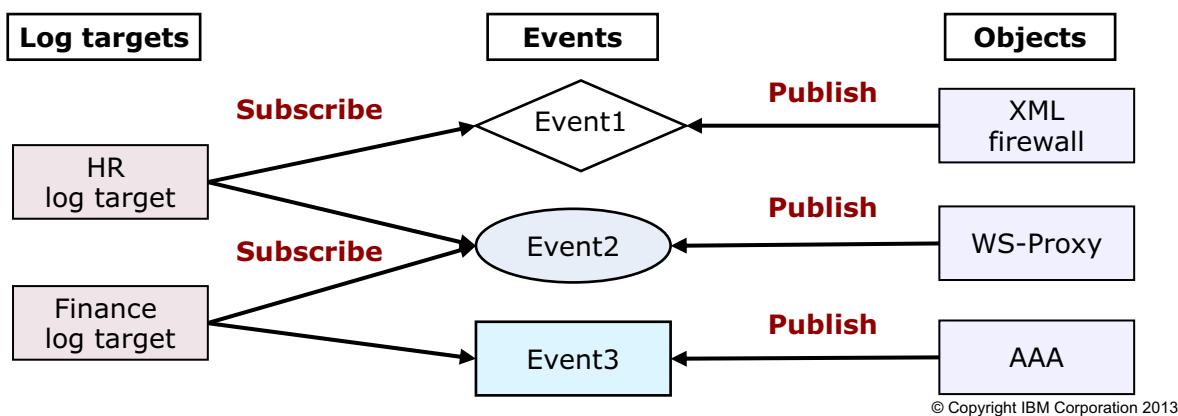


Figure 6-26. Available log levels

WE4013.0

Notes:

The default system log is set up as a log target that subscribes to all events generated by the appliance.

Log targets capture messages only at or above the configured level.

This input sets the level at which the default system log captures messages.

Enable Internal Logging and **Enable RBM Debug** are available in the default domain only.



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 (highlighted)

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 2013

Figure 6-27. Log targets

WE4013.0

Notes:

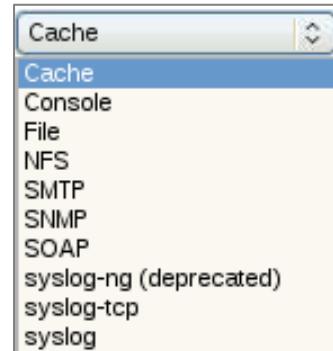
The diagram in the slide shows two-log targets: HR and Finance log targets. These log targets subscribe to certain types of events that are generated or published by objects on the DataPower appliance.

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

Ten-log target types

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

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



© Copyright IBM Corporation 2013

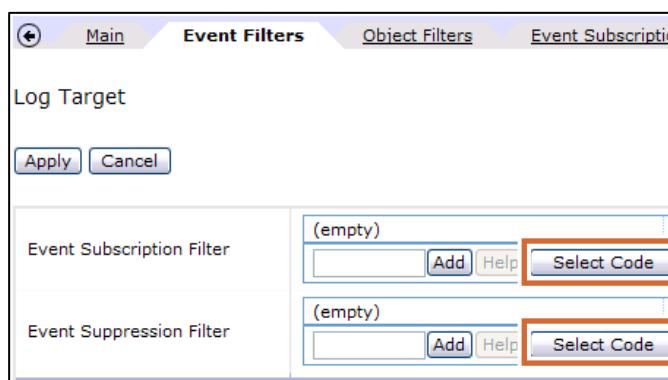
Figure 6-28. Log target configuration

WE4013.0

Notes:

Event filters

- In the “Configure Log Target” web page, select the **Event Filters** tab
- Event filters create filters for a log target that is based on **event codes**
 - Use the **Event Subscription Filter** to subscribe to specific event codes



| 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 |

- 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

© Copyright IBM Corporation 2013

Figure 6-29. Ten-log target types

WE4013.0

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 depreciated. Use syslog-tcp in place of syslog-ng.



Object filters

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

Editing Object Filters property
of **Log Target**

[Help](#)

| | |
|------------------------|---|
| Object Type | <input type="text" value="AAA Policy"/> <small>*</small> |
| Object Name | <input type="text" value="MyTest"/> <small>*</small> |
| Add Referenced Objects | <input type="radio"/> on <input checked="" type="radio"/> off |

Save **Cancel**

© Copyright IBM Corporation 2013

Figure 6-30. Event filters

WE4013.0

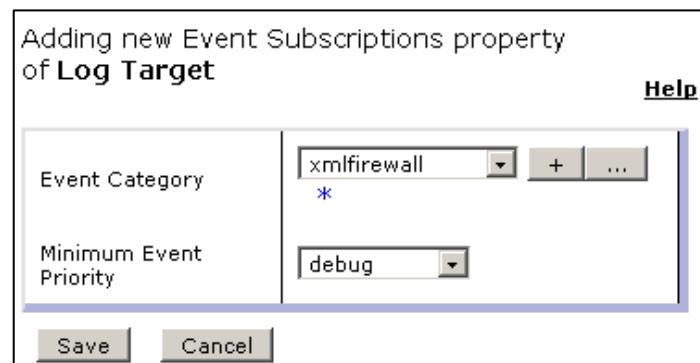
Notes:

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 more.

These event codes are DataPower specific event conditions.

Event subscriptions

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



© Copyright IBM Corporation 2013

Figure 6-31. Object filters

WE4013.0

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 named **MyFirewall** and not all XML firewall instances.

WebSphere Education

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 Type:** log priority
 - Log Level:** event category

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 2013

Figure 6-32. Event subscriptions

WE4013.0

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.

Topic summary

Having completed this topic, you should be able to:

- Create a log target to capture a customized set of log messages
- Configure a **Log** action in a service policy

© Copyright IBM Corporation 2013

Figure 6-33. Log action

WE4013.0

Notes:

If there is a response to the action, it is stored in the output context, if one is specified.

If no output context is specified, the Log action sends the contents and does not wait for a response.

An output context is specified on the Log action if the policy administrator wants the failure of the Log action in a policy rule to cause an error condition in the processing of the rule.

Physical log files can be stored on the appliance by `logtemp:///<filename>`. You can use the file management utilities to copy or view this file.

Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 6-34. Topic summary

WE4013.0

Notes:

Checkpoint questions

1. True or False: To test a Log Event, one would use the Generate Log Event option in the troubleshooting panel 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 backside 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 2013

Figure 6-35. Unit summary

WE4013.0

Notes:

Checkpoint answers

1. **True.** To test a Log Event, one would use the Generate Log Event option in the troubleshooting panel to generate a log message, and verify that it is included or excluded in a log target.
2. **A.** 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 backside connection.
3. **A, B, C, D, and F.** 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 2013

Figure 6-36. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.

Exercise 4



Creating an advanced multi-protocol gateway

© Copyright IBM Corporation 2013

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

7.0.1

Figure 6-37. Checkpoint answers

WE4013.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

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

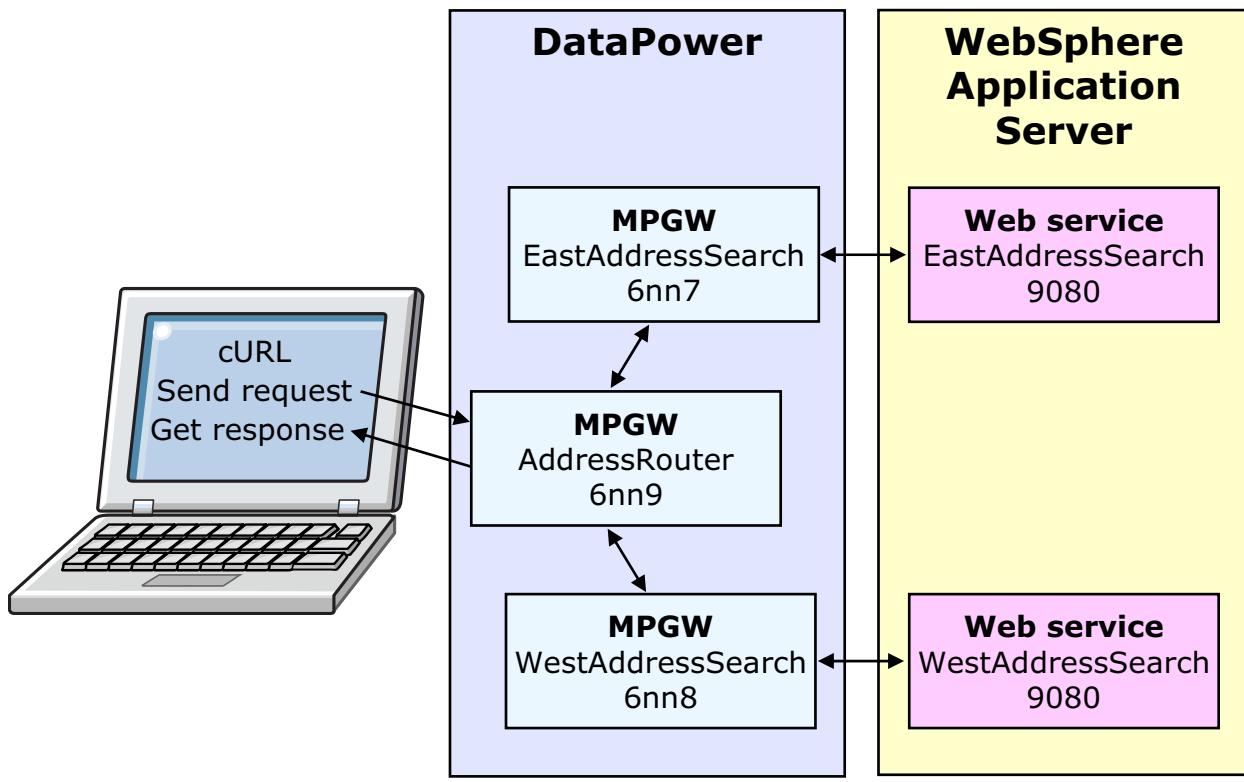
© Copyright IBM Corporation 2013

Figure 6-38. Exercise

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 6-39. Exercise objectives

WE4013.0

Notes:

Problem determination tools



© Copyright IBM Corporation 2013

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

4.0 701

Figure 6-40. Exercise overview

WE4013.0

Notes:

Unit 7. Handling errors in a service policy

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

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

How you will check your progress

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

Unit objectives

After completing this unit, you should be able to:

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

© Copyright IBM Corporation 2013

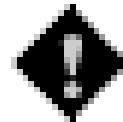
Figure 7-1. Unit objectives

WE4013.0

Notes:

Error handling constructs

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



Two methods for handling errors:

- **On Error** action
 - Lets you either cancel or continue processing
 - If *continue*, then the next action in the rule is executed; otherwise, the rule is canceled
- **Error rule**
 - Automatically executes if it is configured within the current document processing policy
 - Presence of an **On Error** action precludes the automatic selection of an **error rule** for execution

© Copyright IBM Corporation 2013

Figure 7-2. Error handling constructs

WE4013.0

Notes:

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

Configure an On Error action

The **On Error** action is used to control what happens when an error is encountered within the rule

- Optional: execute a named rule to handle the error condition

Configure the following within an **On Error** action:

- Error mode:
 - **Cancel**: stop executing the current rule
 - **Alternative**: invoke an alternative processing rule
 - **Continue**: continue with the next sequential action
- The **Processing Rule** fields specify either:
 - An error rule to execute
 - A custom variable for the processing rule
 - Use the Var Builder to create a custom variable

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

- Error Mode:** A dropdown menu showing 'Cancel' (selected), 'Alternative', and 'Continue'.
- Processing Rule:** A dropdown menu with options 'Cancel', 'Alternative', and 'Continue', followed by a 'Var Builder' button.
- Error Input:** A text input field with '(none)' selected.
- Error Output:** A text input field with '(none)' selected.
- Asynchronous:** A radio button group with 'on' (unchecked) and 'off' (checked).

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

© Copyright IBM Corporation 2013

Figure 7-3. Configure an On Error action

WE4013.0

Notes:

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

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

The **Error Input** and **Error Output** context in an **On Error** action provide the context for the actions within the error rule (if selected).

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



Creating an error rule

Rule:

Rule Name: Test_rule_1 Rule Direction: Error

New Rule Delete Rule

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

Filter Sign Verify Validate Encrypt Decrypt Transform Route AAA Results Advanced

ORIGIN SERVER → DATAPOWER ←

Create Reusable Rule

| Order | Rule Name | Direction | Actions |
|-------|---------------|------------------|---------|
| ↑ ↓ | Test_request | Client to Server | ◀ ↵ |
| ↑ ↓ | Test_response | Server to Client | ◀ ↵ |
| ↑ ↓ | Test_rule_1 | Error | ◀ ↵ |

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

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

© Copyright IBM Corporation 2013

Figure 7-4. Creating an error rule

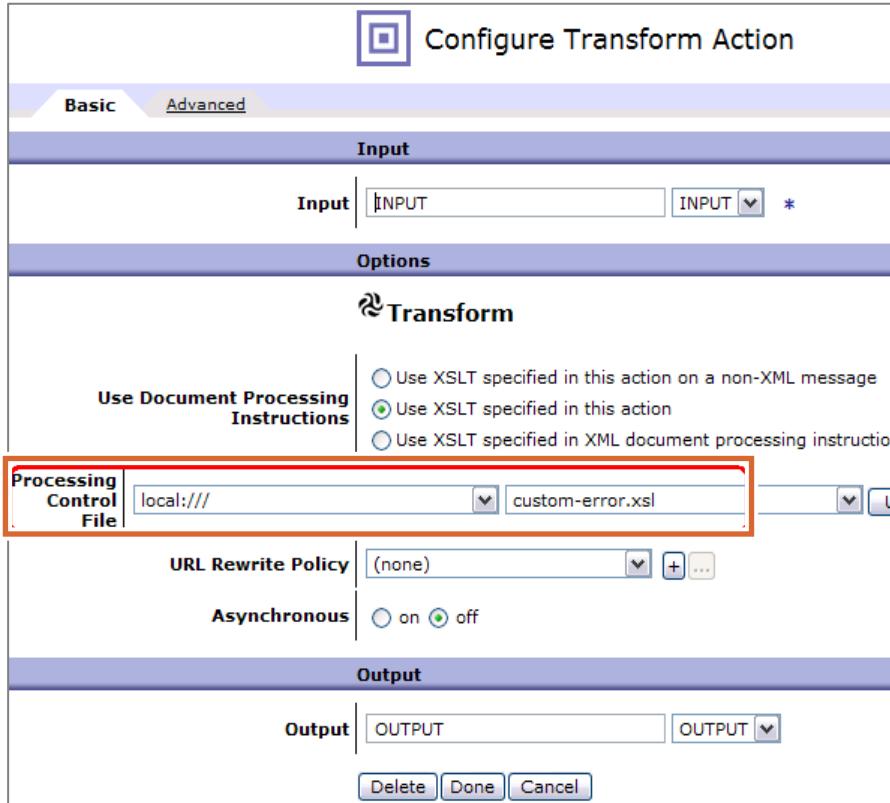
WE4013.0

Notes:

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

Configure Transform action in error rule



Configure Transform Action

Basic [Advanced](#)

Input

Input INPUT *

Options

Transform

Use Document Processing Instructions

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

Processing Control File local:///

URL Rewrite Policy (none)

Asynchronous on off

Output

Output OUTPUT *

© Copyright IBM Corporation 2013

Figure 7-5. Configure Transform action in error rule

WE4013.0

Notes:

Style sheet programming that uses 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 2013

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

WE4013.0

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 executing the multi-step probe by selecting the **Service Variables** tab.

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

Example custom error style sheet

```

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

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

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

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

```

© Copyright IBM Corporation 2013

Figure 7-7. Example custom error style sheet

WE4013.0

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
 - **On Error** action itself 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

© Copyright IBM Corporation 2013

Figure 7-8. Error rule versus On Error action

WE4013.0

Notes:

Unit summary

Having completed this unit, you should be able to:

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

© Copyright IBM Corporation 2013

Figure 7-9. Unit summary

WE4013.0

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 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 2013

Figure 7-10. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

1.

2.

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.** 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

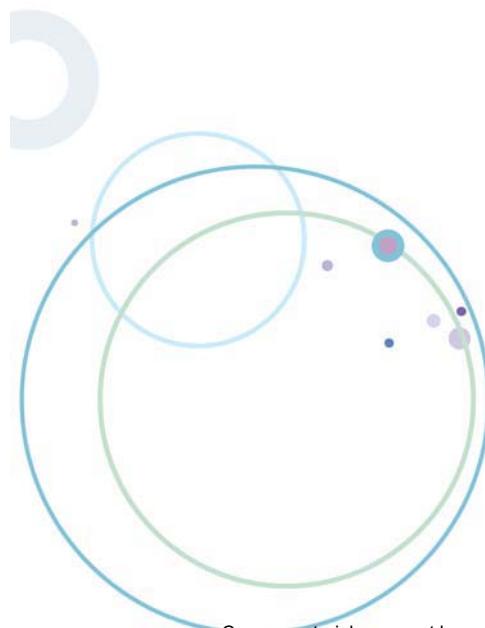
© Copyright IBM Corporation 2013

Figure 7-11. Checkpoint answers

WE4013.0

Notes:

Exercise 5



Adding error handling to a service policy

© Copyright IBM Corporation 2013

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

7.0.1

Figure 7-12. Exercise

WE4013.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

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

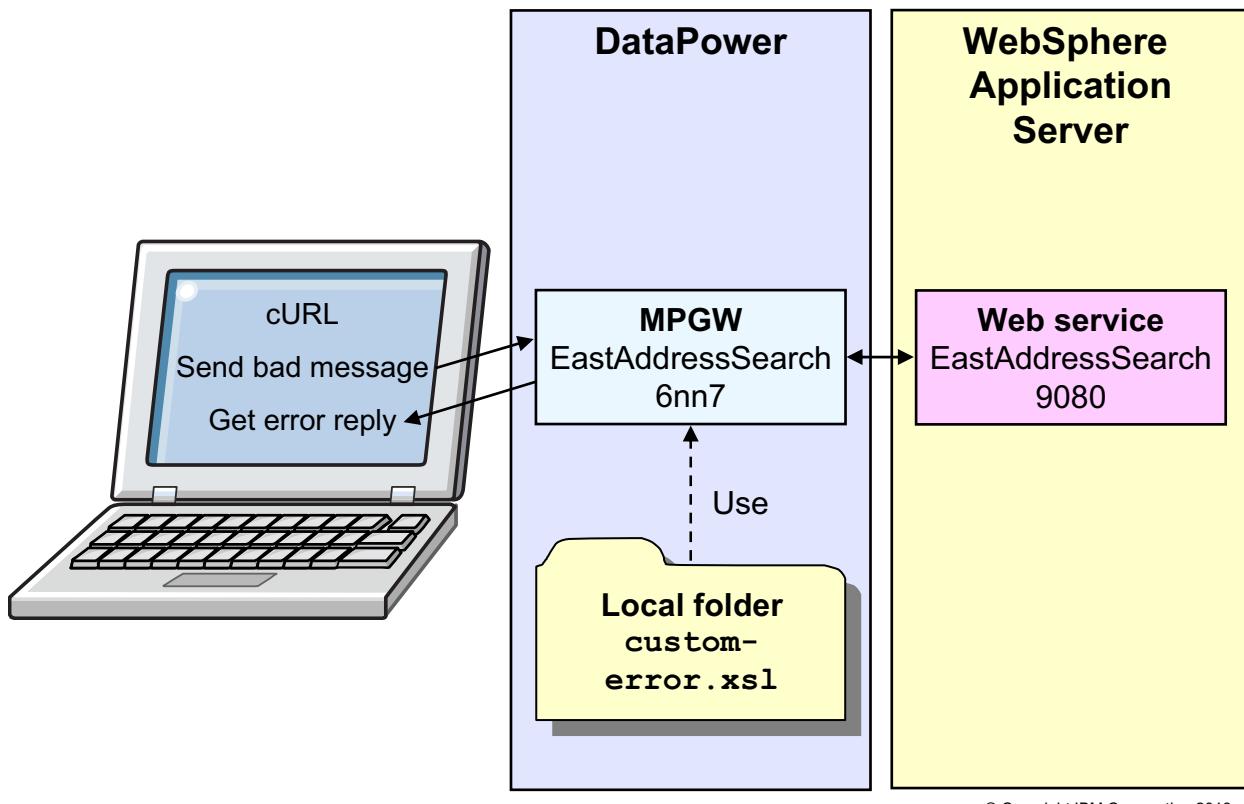
© Copyright IBM Corporation 2013

Figure 7-13. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 7-14. Exercise overview

WE4013.0

Notes:

Unit 8. DataPower cryptographic tools

What this unit is about

This unit describes how to use the cryptographic tools to create keys and certificates. You also set the DataPower objects that are used to validate certificates and configure certificate monitoring to ensure that only valid certificates exist on board.

What you should be able to do

After completing this unit, you should be able to:

- Generate cryptographic keys by using the WebSphere DataPower tools
- Create a cryptographic identification credential object that contains a matching public and private key
- Create a cryptographic validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up to date

How you will check your progress

- Checkpoint
- Exercise 6: Creating cryptographic objects

Unit objectives

After completing this unit, you should be able to:

- Generate cryptographic keys by using the WebSphere DataPower tools
- Create a cryptographic identification credential object that contains a matching public and private key
- Create a cryptographic validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up-to-date

© Copyright IBM Corporation 2013

Figure 8-1. Unit objectives

WE4013.0

Notes:

Security problems

1. Message confidentiality: how do you prevent anyone from looking at your message?
2. Message integrity: how do you know that anyone looked at the message and changed it?
3. Nonrepudiation: how do you know who the party on the other end is?

© Copyright IBM Corporation 2013

Figure 8-2. Security problems

WE4013.0

Notes:

Security problem 1: Message confidentiality

How do you prevent anyone from looking at your message?

- Use cryptography
 - Study of techniques that are used to transform information into an unreadable format, which is called a cipher
 - Only the party for whom the information is intended can decipher the message

Modern cryptography uses *algorithms* and *keys* to manipulate data

- Keys are pieces of data that are used to alter how the algorithm behaves
- Knowing the algorithm does not help any attackers; they need the key
 - Most algorithms are published publicly
 - Keys are usually protected and hidden

© Copyright IBM Corporation 2013

Figure 8-3. Security problem 1: Message confidentiality

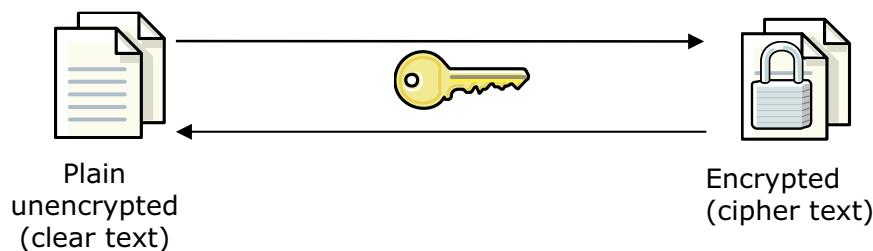
WE4013.0

Notes:

PKI (public key infrastructure) uses the processes of encryption to hide a text message and decryption to re-create the message.

Symmetric key encryption

- Symmetric key: a secret key that is used to both encrypt and decrypt messages
 - Known only by sender and receiver
 - Relatively fast
 - Challenges: exchanging keys with many people



© Copyright IBM Corporation 2013

Figure 8-4. Symmetric key encryption

WE4013.0

Notes:

The disadvantage of symmetric keys is that the same key is needed for encryption and decryption, and both parties must have the same keys.

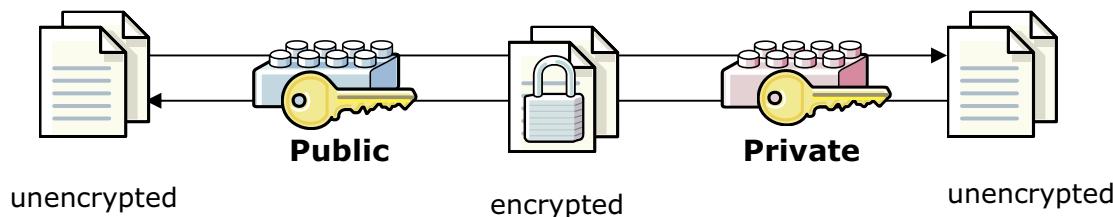
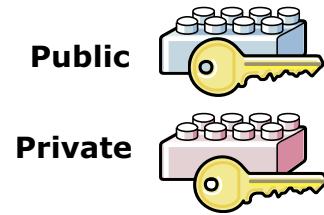
Typical symmetric algorithms are:

- DES (DEA)
- Triple DES (TDEA)
- AES
- RC2
- RC4
- IDEA

Asymmetric key encryption

Public key cryptography

- Two keys that are cryptographically related:
 - Public key (can share with everyone)
 - Private key (must never be shared; possession is proof)
- Keys are asymmetric:
 - Given message is encrypted with one key and decrypted with another
 - Symmetric, secret key technology uses the same key for encryption and decryption



© Copyright IBM Corporation 2013

Figure 8-5. Asymmetric key encryption

WE4013.0

Notes:

With asymmetric key encryption, the encryption and decryption keys are different.

The private key is mathematically linked to the public key.

Modern day public-key cryptographic systems are designed so that it is computationally infeasible to derive the private key from the public key.

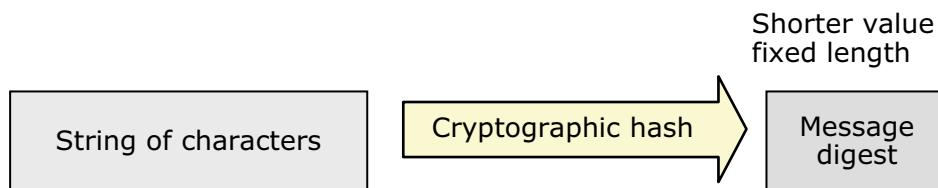
Mathematically, either the private key or the public key can be used to encrypt, and the other key is used to decrypt. If you are using PKI, then, by definition, the public key is used to encrypt. (In cryptography, RSA stands for Rivest, Shamir, and Adleman, who first publicly described this algorithm for public-key cryptography.)

Typical asymmetric encryption algorithms are:

- DH (Diffie-Hellman)
- DSS (Digital Signature Standard)
- RSA encryption algorithm (PKCS)

Security problem 2: Message integrity

- How do you know that anyone looked at the message and changed it?
 - Use a cryptographic hash
- A *cryptographic hash* function is an algorithm that transforms a string of characters into a shorter number of a fixed length
 - This value is called the message digest
 - If anyone tampers with the message, it results in a different message digest or hash number
- Purposefully creating two separate messages that create the same hash code is difficult



© Copyright IBM Corporation 2013

Figure 8-6. Security problem 2: Message integrity

WE4013.0

Notes:

The cryptographic hash computes a message digest or message authentication code (MAC), which is unique to that message.

It is computationally infeasible to find two messages that hash to the same thing.

A change in a cryptographic hash results in a change to the hash number.

Common hash functions are:

- MD5
- SHA1

Security problem 3: Nonrepudiation

- How do you know who the party on the other end is?
 - Use digital signatures
- With digital signatures, you can authenticate who sent the message
 - Provided within a digital certificate
 - Incorporates **asymmetric keys** and **cryptographic hash functions**
- The digital signature is:
 - Encrypted with the sender's private key
 - Verified when the certificate is checked

© Copyright IBM Corporation 2013

Figure 8-7. Security problem 3: Nonrepudiation

WE4013.0

Notes:

Digital signature

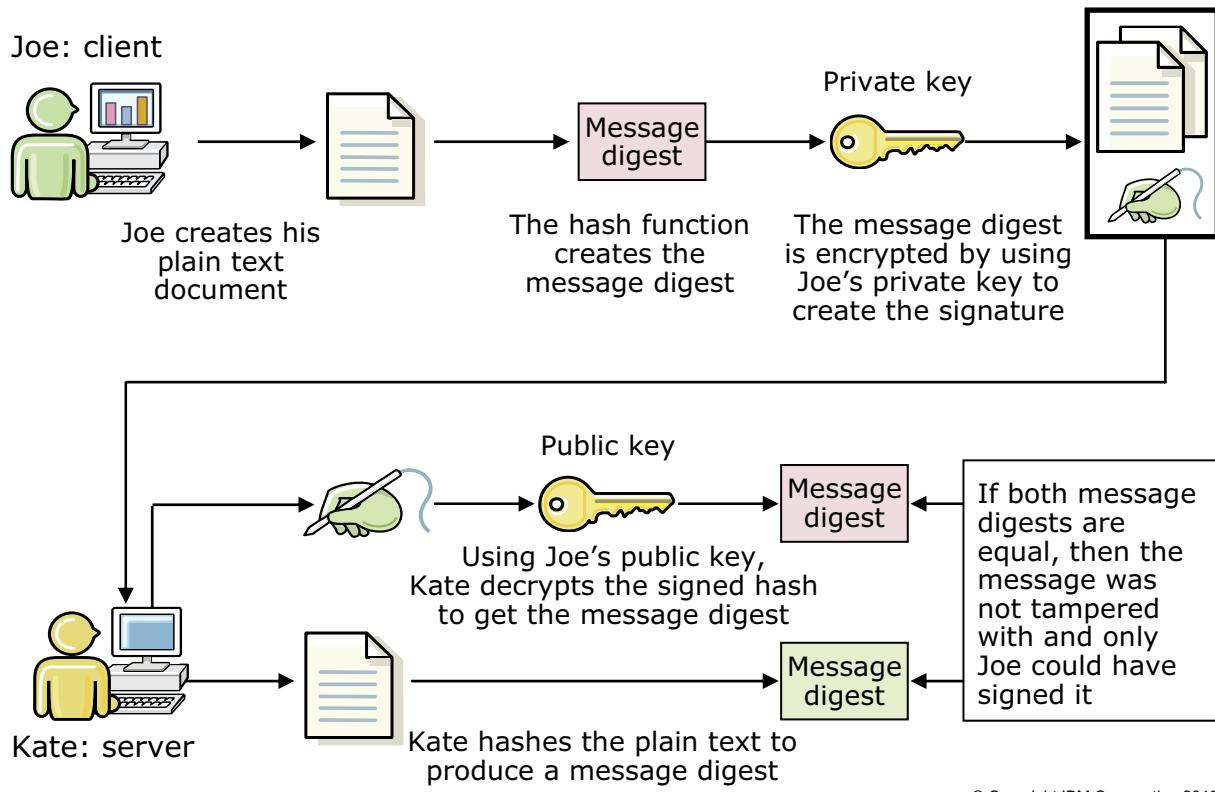


Figure 8-8. Digital signature

WE4013.0

Notes:

1. Joe creates a message.
2. The message is then hashed to create the message digest.
3. The message digest is then encrypted by using Joe's private key. The digital signature is created.
4. The message is then sent, along with the signature. The signed message is encrypted.
5. Kate receives the message and two processes are run against the signed message (after decryption, if necessary).
6. The received signed hash is decrypted by Joe's public key; a message digest (hash number) is created.
7. The received message is also hashed again by the cryptographic hash algorithm; another message digest (hash number) is created.
8. If these two hash numbers are equal, then someone has not tampered with the message.

Security problems: Solved

- Message confidentiality: how do you keep anyone from seeing your message?
 - Symmetric and asymmetric keys can encrypt and decrypt messages
- Message integrity: how do you know that anyone intercepted the message and tampered with it?
 - Cryptographic hash algorithms: these algorithms can be used to detect whether a message is tampered with
- Nonrepudiation: how do you verify the senders and authenticate that they are who they say they are?
 - Digital signatures: by using a public key, it can be determined that the supposed originator sent the message

© Copyright IBM Corporation 2013

Figure 8-9. Security problems: Solved

WE4013.0

Notes:

Digital certificates

- The problem with public-private key pairs is that they do not identify anyone
 - Given a message encrypted (or signed) by using a private key, you can determine which public key goes with it, but so what?
- The solution is digital certificates
 - A special **public key**
 - Contains your identity information in the form of a distinguished name
- A digital certificate contains:
 - The name of the certificate holder
 - A serial number
 - Validity dates
 - A copy of the public key of the certificate holder
 - A **digital signature** to verify that the certificate is not altered since the issuer signed it
 - An indication of the issuer of the certificate: “the trusted signer”
- A digital certificate does **not** contain the **private key** although the private key and certificate together are often referred to as “the certificate”

© Copyright IBM Corporation 2013

Figure 8-10. Digital certificates

WE4013.0

Notes:

A digital certificate is a data structure that is used in a public key system to bind a particular, authenticated individual to a particular public key.

A certificate might be internally created and distributed, and the company would be its own CA (self-signed).



Distribution problem

- First step is to create public-private key pairs to communicate securely
- Second step is to create a digital certificate to send to the party with whom you want to communicate
 - Digital certificate contains your digital signature with public key
- Problem: how to send your certificate to everyone with whom you want to communicate?
 - You also need signed certificate with a public key from everyone with whom you want to communicate
- Solution: create a certificate authority (CA) to resolve this issue
 - Instead of issuing self-signed digital certificates to everyone, use a CA certificate to sign your digital certificate
 - Parties validate digital signature of CA certificate that signed your digital certificate

© Copyright IBM Corporation 2013

Figure 8-11. Distribution problem

WE4013.0

Notes:

You need to download all of the common CA digital certificates to verify a certificate that is signed by a CA.

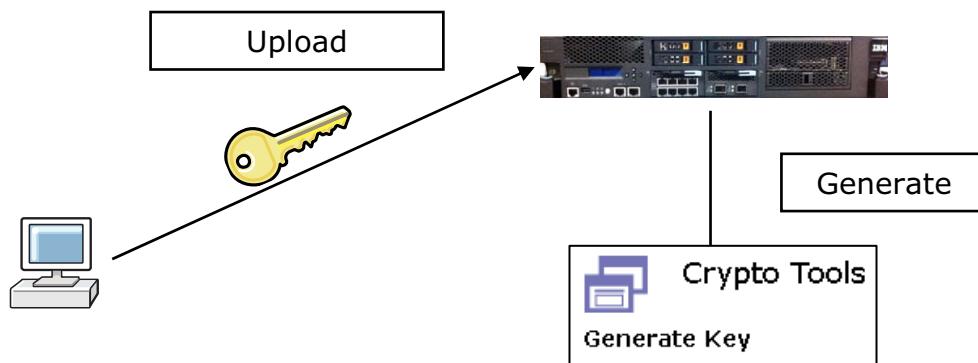
Imagine that every business entity had to create, digitally sign, and send out a certificate to each person who wants to use its service. This scenario is a distribution nightmare.

The CA is the issuer of many certificates.

DataPower cryptographic tools

Two methods for creating a private cryptographic key and self-signed digital certificate:

- Generated on-board using the DataPower crypto tools
- Uploading key files to the DataPower appliance
 - Supports Java Key Store



© Copyright IBM Corporation 2013

Figure 8-12. DataPower cryptographic tools

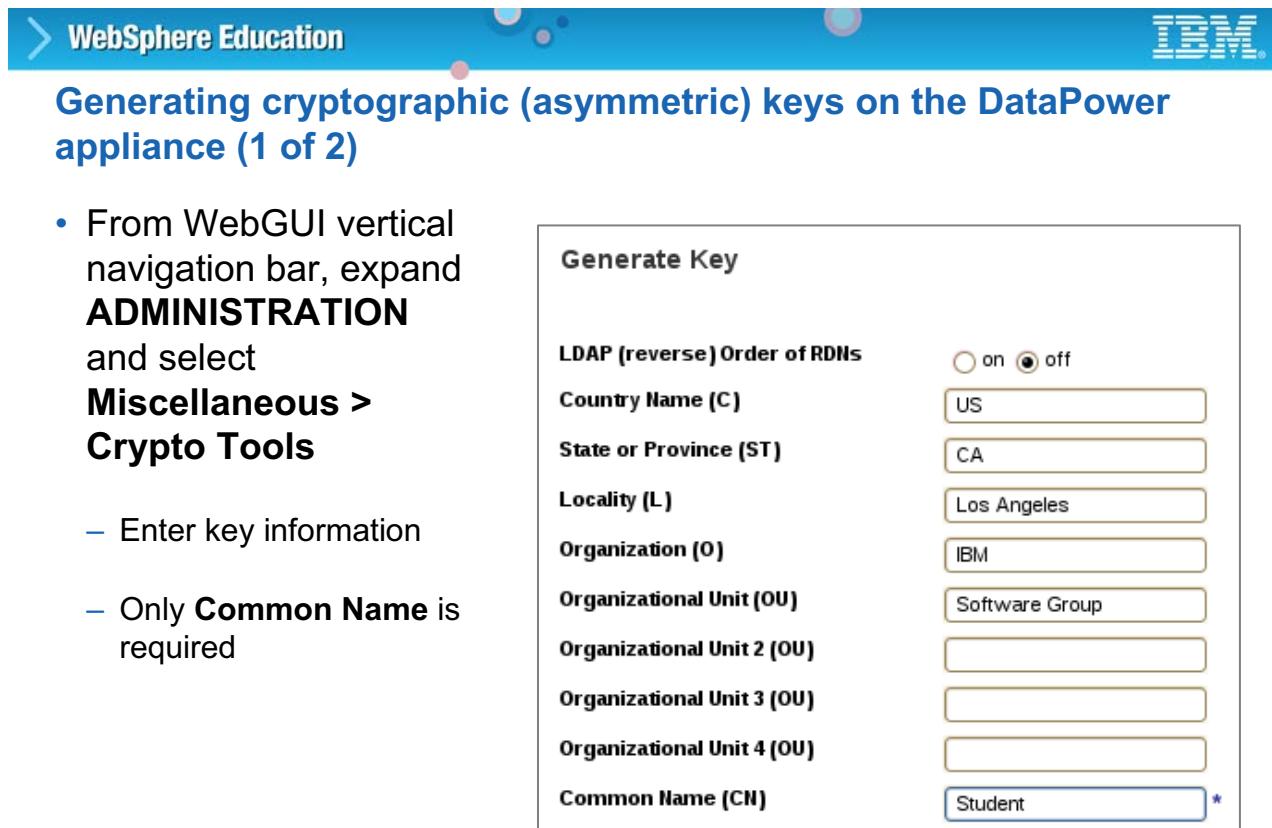
WE4013.0

Notes:

A self-signed certificate implies that there is no third-party certificate authority that 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.

The appliance supports the uploading of files from a Java keystore (JKS) to the appliance flash.



WebSphere Education

Generating cryptographic (asymmetric) keys on the DataPower appliance (1 of 2)

- From WebGUI vertical navigation bar, expand **ADMINISTRATION** and select **Miscellaneous > Crypto Tools**
 - Enter key information
 - Only **Common Name** is required

Generate Key

LDAP (reverse) Order of RDNs on off

| | |
|-----------------------------------|----------------|
| Country Name (C) | US |
| State or Province (ST) | CA |
| Locality (L) | Los Angeles |
| Organization (O) | IBM |
| Organizational Unit (OU) | Software Group |
| Organizational Unit 2 (OU) | |
| Organizational Unit 3 (OU) | |
| Organizational Unit 4 (OU) | |
| Common Name (CN) | Student * |

© Copyright IBM Corporation 2013

Figure 8-13. Generating cryptographic (asymmetric) keys on the DataPower appliance (1 of 2)

WE4013.0

Notes:

The files to submit to a certificate authority are created by default.

The fields from **Country Name** down to **Common Name** 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 DataPower appliance to workstation
 - Except when **Export Private Key** is selected
 - Also exported to `temporary`: directory
- The entered object name is used to represent the key and certificate object
- Click **Generate Key** to generate the key and certificate

The screenshot shows a configuration dialog box with the following settings:

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

© Copyright IBM Corporation 2013

Figure 8-14. Generating crypto (asymmetric) keys on board (2 of 2)

WE4013.0

Notes:

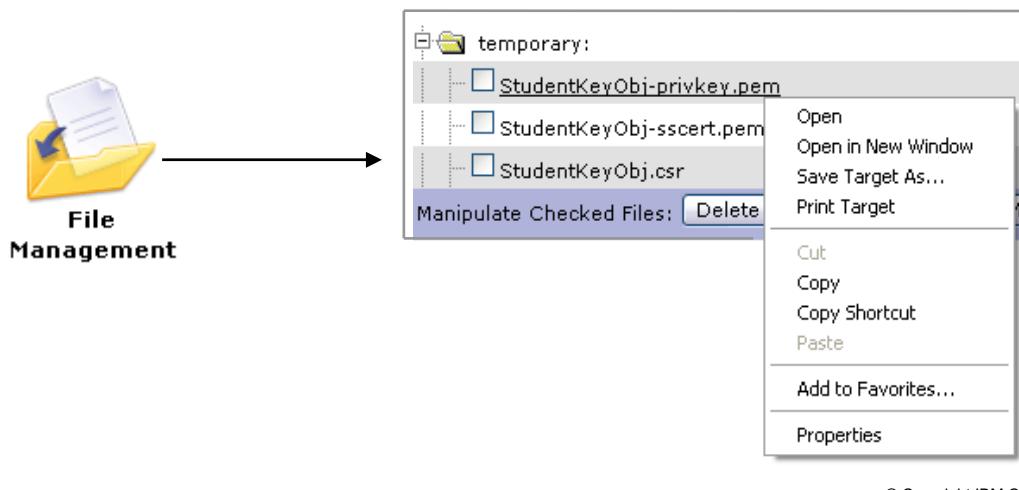
The password for the key file is generated.

Select **on** for **Generate Self-Signed Certificate** to generate a self-signed certificate for the key.

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

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, select the **File Management** icon
- Right-click the file to **Save Target As**



© Copyright IBM Corporation 2013

Figure 8-15. Download keys from temporary storage

WE4013.0

Notes:

The appliance has on-board memory where it stores files. These files are organized in directories. Each directory has its own associated permissions and visibility.



Keys and certificates are objects

- The generated key and certificate are accessed by using object names
 - Entered in object name field when generating key
 - Another level of security by providing indirection reference to the file

**Crypto
key**
 ↓
**Private
key file**

Configure Crypto Certificate

Main

Crypto Certificate : AliceCert [up]

| | | View Log View St |
|-------------------------|---|--------------------------|
| | | Apply Cancel Delete Undo |
| Admin State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled | |
| File Name | cert:/// <input type="button" value="Fetch..."/> * <input type="button" value="Details..."/> <input type="button" value="Upload..."/> | |
| Password | <input type="password"/> | |
| Confirm Password | <input type="password"/> | |
| 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 2013

Figure 8-16. Keys and certificates are objects

WE4013.0

Notes:

The page that is shown in this slide can be accessed from the vertical navigation bar, by selecting **Objects > Crypto > Crypto Key**.

Selecting **Password Alias** to be **on** means that the password entered for key is a password alias.



Crypto shared secret (symmetric) key

- Use key file generate a secret key object:
- From vertical navigation bar, select **Objects > Crypto > Crypto Shared Secret Key**
 - The “Crypto Shared Secret Key” page allows you to create a secret key object for the symmetric key
 - Provides more level of security by providing indirection reference to file

The screenshot shows a dialog box titled "Crypto Key". It contains the following fields:

- Name:** DPEduKey (marked with a required asterisk *)
- Administrative State:** A radio button group where "enabled" is selected.
- File Name:** A dropdown menu showing "cert:///" and a file selection field containing "dpedu.p12". To the right are "Upload..." and "Fetch..." buttons.

At the bottom right of the dialog box is the copyright notice: © Copyright IBM Corporation 2013.

Figure 8-17. Crypto shared secret (symmetric) key

WE4013.0

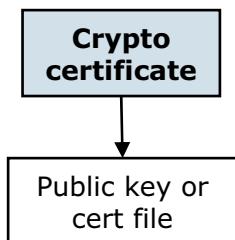
Notes:

A secret key is generated using symmetric key encryption.

The key file can be uploaded from this page.

Crypto certificate

- Create certificate object from key file
 - From vertical navigation bar, select **Objects > Crypto > Crypto Certificate**
 - Provides higher level of security by providing indirect reference to file



Crypto Certificate

Apply **Cancel**

| | | |
|-------------------------|---|---|
| Name | <input type="text"/> | * |
| Admin State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled | |
| File Name | cert: <input type="button" value="..."/> (none) | |
| Password | <input type="password"/> | |
| Confirm Password | <input type="password"/> | |
| 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 2013

Figure 8-18. Crypto certificate

WE4013.0

Notes:

An object that is created on this page is used to create a crypto identification credential that is discussed on the next slide.

Selecting **Password Alias** to be **on** means that the password entered for key is a password alias.

Ignore Expiration Dates controls whether the appliance enforces the certificate validity date.

Certificates can also be uploaded using this page if they do not exist on the DataPower appliance.

DataPower allows you to create certificate objects using certificates that are retrieved from z/OS. To do so, you would specify a directory of `saf-cert://`. In the file name field, you would reference a pre-existing **z/OS NSS Client** object (**Objects > ZOS Configurations > z/OS NSS Client**), and the necessary label name.

Certificates exist in a trust chain

- A certificate trust chain is a linked path of certificates from a certificate to a trusted certificate authority (CA)
 - A certificate that a CA signs can be assumed to have some level of validation

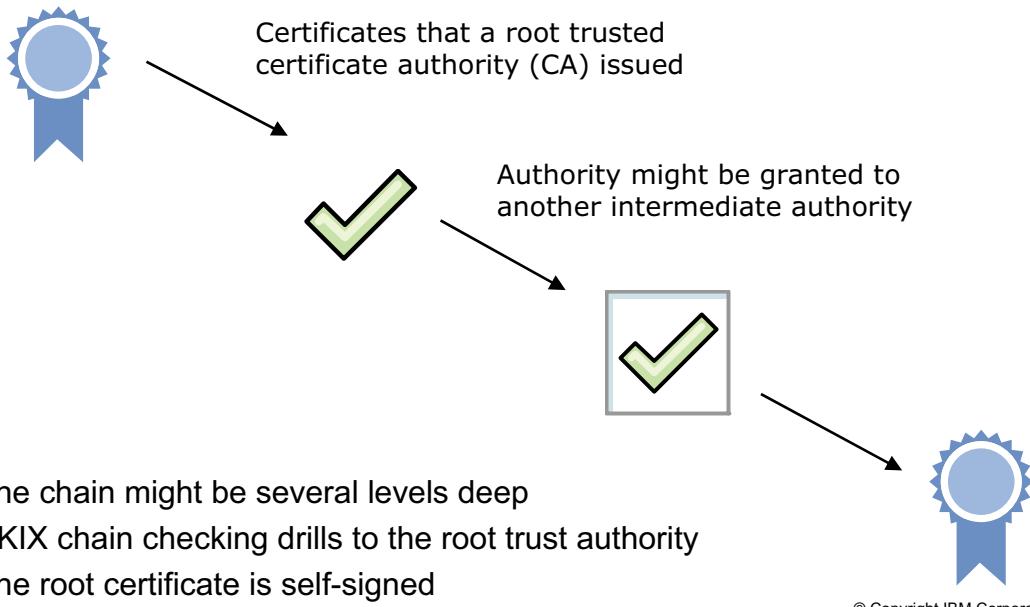


Figure 8-19. Certificates exist in a trust chain

WE4013.0

Notes:

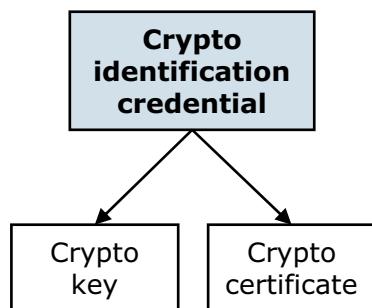
Cryptographic certificates exist in a trust chain; that is, certificates are issued by a root trusted certificate authority. This trusted root may then grant the authority to issue certificates to an intermediate authority, which then issues the certificate that is used in the field.

PKIX Chain Checking drills to the trusted root authority to establish a complete trust chain. If the complete chain is not trusted, then the presented certificate is not trusted.

Intermediate CA certificates may be necessary if the root trusted certificate is not trusted. Additional intermediate certificates may be required if that particular intermediate certificate is not trusted.

Crypto identification credential

- Create a crypto identification credential
 - Consists of crypto key object and crypto certificate object
 - Contains public (certificate) and private key pair that is used for SSL authentication
 - From vertical navigation bar, select **Objects > Crypto > Crypto Identification Credentials**



Crypto Identification Credentials

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

© Copyright IBM Corporation 2013

Figure 8-20. Crypto identification credential

WE4013.0

Notes:

Enter a name for the crypto identification credential.

In the **Crypto Key** field, select the Crypto Key object from the drop-down list. You can use the **+** and **...** buttons to create or edit a crypto key object.

In the **Certificate** field, select a certificate object from the drop-down list. You can use the **+** and **...** buttons to create or edit a certificate object.

If they are available, specify the **Intermediate CA Certificates** by clicking the **Add** button. A trust chain that consists of one or more certificate authority (CA) certificates is established.

You can also create a crypto identification credential by selecting **Keys and Certification Management > Identification Credentials** from the Control Panel.



Crypto validation credential

- Used to validate the authenticity of certificates and digital signatures
 - Who vouches for the certificate and that you are who you say you are?
- The appliance consists of certificates from:
 - Common public certificates that are stored in pubcert: directory
 - Certificates that are imported onto the appliance
 - Certificates that are generated on the appliance

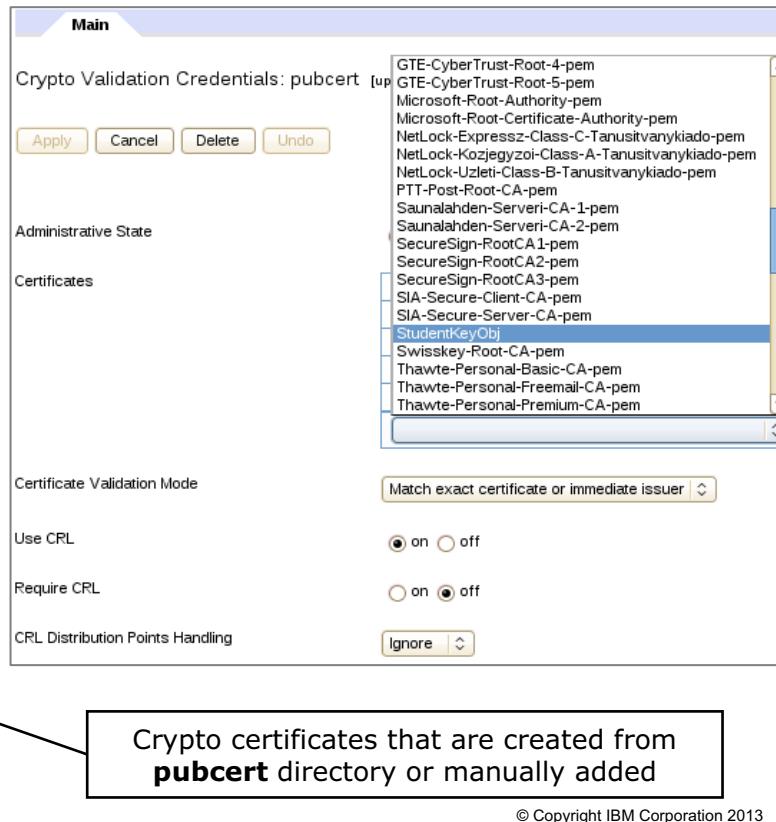
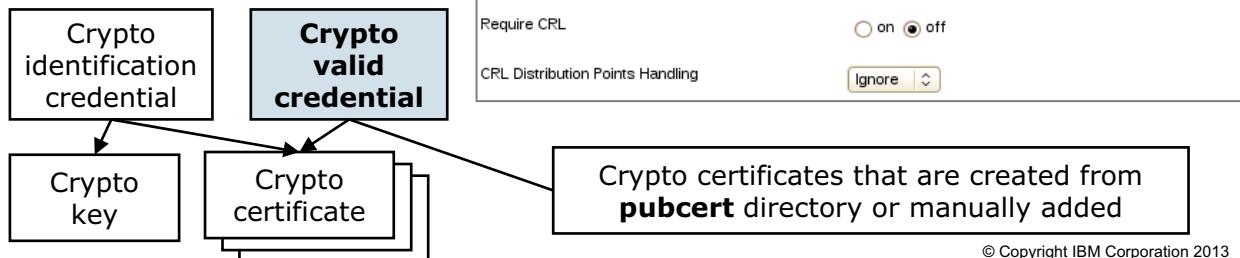


Figure 8-21. Crypto validation credential

WE4013.0

Notes:

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 ValCred from pubcert:** button on the Configure Crypto Validation Credentials 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 ones that you have added or imported. The **Create ValCred from pubcert:** button is available when clicking the Crypto Validation Credentials page.

The certification validation mode specifies how to validate the presented certificate.

Two options are available:

1. Match exact certificates or immediate issuer: The certificate that is presented or the immediate issuer of the certificate must be available on the appliance.

2. Full certificate chain checking (PKIX): The certificate that is presented and any intermediate certificate that is chained back to the root certificate must be trusted. This trust applies to all uses of the validation credential.

The **Use CRLs** field is used to check whether certificates in the trust chain are monitored for expiration.

Crypto profile

- Identifies profiles that can be used in SSL connections
 - SSL server profile has an **identification credential** (private and public key pair), and maybe a **validation credential** (certificate chain that validates the presented certificate)
 - SSL client profile has a **validation credential**

```

graph TD
    CP[Crypto profile] --> CIC[Crypto identification credential]
    CP --> CV[Crypto valid credential]
    CIC --> CK[Crypto key]
    CV --> CC[Crypto certificate]
  
```

© Copyright IBM Corporation 2013

Figure 8-22. Crypto profile

WE4013.0

Notes:

SSL is covered later.

The **Ciphers** property refers to the cipher suites supported by this profile; it indicates encryption strength, hashing algorithm, and key-encryption algorithms.

The **Options** property allows you to specify support for SSL and TLS protocols.

The **Send Client CA List** property allows you to specify whether the SSL server sends the client CA list during a request for the client certificate.



Import and export crypto objects

- Export certificate objects to a file
 - File is exported to **temporary**: directory on appliance
- Crypto objects reference certificates file
 - Eliminates need to create object for certificate
- **Import Crypto Object** brings in exported certificate objects
 - File can be in another directory, or uploaded

The screenshot shows a web-based interface for managing crypto objects. At the top, there are three tabs: 'Generate Key', 'Export Crypto Object' (which is highlighted in blue), and 'Import Crypto Object'. Below the tabs, the title 'Export Crypto Object' is displayed. There are three input fields: 'Object Type' (set to 'certificate'), 'Object Name' (empty), and 'Output File Name' (empty). At the bottom of the form is a blue 'Export Crypto Object' button.

© Copyright IBM Corporation 2013

Figure 8-23. Import and export crypto objects

WE4013.0

Notes:

This page is accessed from the vertical navigation bar by selecting **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 of the exported crypto object must be typed exactly.
- For an imported crypto object, a password alias can be supplied if the password is not entered.

WebSphere Education

Uploading keys

- From the vertical navigation bar, select **Objects > Crypto > Crypto Key**

The screenshot shows the 'Crypto Key' configuration page. On the left, there are fields for Name (DPEduKey), Administrative State (enabled), File Name (cert:///dpedu.p12), Password, and Password Alias. On the right, there is a 'File Management' dialog box titled 'File Management' with the sub-tittle 'Upload File to Directory cert:'. It has two radio button options: 'Source: File' (selected) and 'Java Key Store (Requires Sun JRE 1.4.2 or better)'. Below these are fields for 'File to upload:' and 'Save as:', both with browse buttons. At the bottom are 'Upload' and 'Cancel' buttons, with 'Upload' being highlighted with a red box. An arrow points from the 'Upload...' button on the main page to the 'Upload' button in the dialog box.

- On “Crypto Key” page, click **Upload** button to upload key file from:
 - Keystore file
 - Java Key Store

© Copyright IBM Corporation 2013

Figure 8-24. Uploading keys

WE4013.0

Notes:

Selecting the **Java keystore** radio button opens a new window with a Java applet. You must have a JRE V1.4.2 or higher installed in Internet Explorer to view the applet.

A keystore file can be generated using the `Java keytool` command.

Java keytool command

- IBM JDK includes a keytool utility to generate a keystore
 - A keystore is a database of private keys and an X.509 certificate chain, where the first certificate in the chain contains the public key
 - Users can create their own public-private key and self-signed certificate
 - The **keytool** command has the following syntax:


```
keytool -genkey {-alias alias} {-keyalg keyalg}
{-keysize keysize} {-sigalg sigalg} [-dname dname]
[-keypass keypass] {-validity valDays} {-storetype storetype}
{-keystore keystore} [-storepass storepass] [-provider
provider_class_name] {-v} {-Jjavaoption}
```
 - Example **keytool** command:


```
keytool -genkey -v -alias dpedu -keypass dpadmin -keystore dpedu.p12
-storepass dpadmin -storetype "pkcs12" -dname "cn=dpedu, o=ibm, c=ca"
-keyalg "RSA"
```
 - Generates a keystore file that is called **dpedu.p12** that contains a public-private key pair
 - Generates a self-signed certificate that contains the public key and entity name with values given by **-dname** flag

© Copyright IBM Corporation 2013

Figure 8-25. Java keytool command

WE4013.0

Notes:

The default keystore is implemented as a file. Private keys are protected with passwords.

The X.509 standard describes how the information is contained within a certificate and the format of that information.

For more information about the **keytool** command, see
<http://download.oracle.com/javase/1.5.0/docs/tooldocs/windows/keytool.html>

The default keystore implementation is a Java keystore (JKS). You can specify other formats by using the **-storetype** flag.

Certificates can expire or get revoked

- Certificates are valid only for a certain 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



Valid until 01-11-2012



- Certificates can also be revoked by issuing authority
- The appliance can check certificate revocation lists (CRL) for revoked certificates



© Copyright IBM Corporation 2013

Figure 8-26. Certificates can expire or get revoked

WE4013.0

Notes:

Warnings posted by the certificate monitor are posted to the system log and checked.

Expired certificates are not trusted.

Certificate revocation list (CRL) retrieval

- A certificate revocation list (CRL) is a list of certificates which are revoked and are no longer valid
 - Need to periodically check the validity of certificates
- Set up a CRL list from vertical navigation bar by selecting **Objects > Crypto Configuration > CRL Retrieval**
 - Click the **CRL Update Policy** tab to configure a CRL update policy

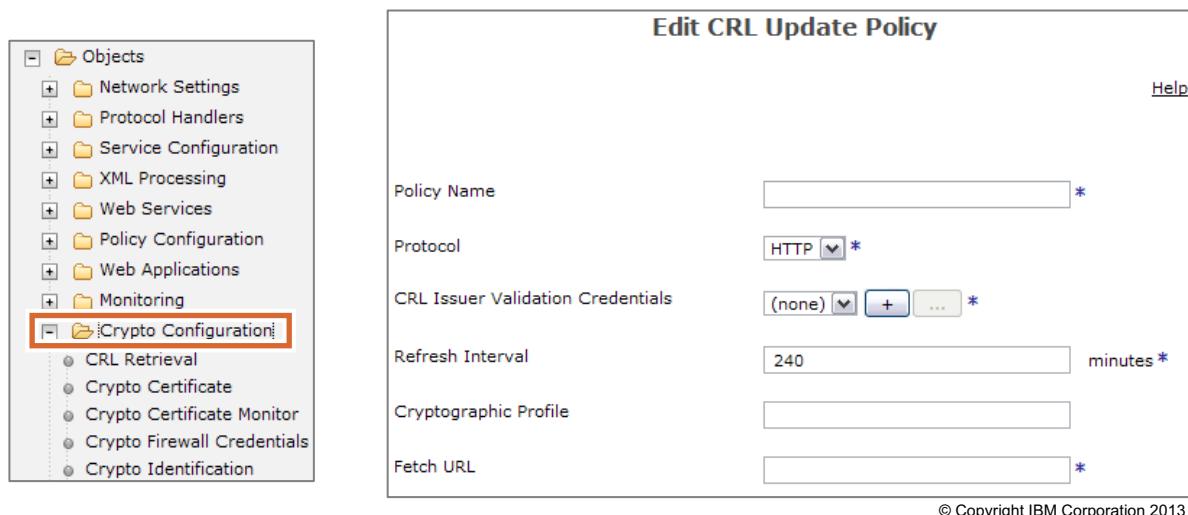


Figure 8-27. Certificate revocation list (CRL) retrieval

WE4013.0

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 and check for validity using the selected **CRL Issuer Validation Credential** object.

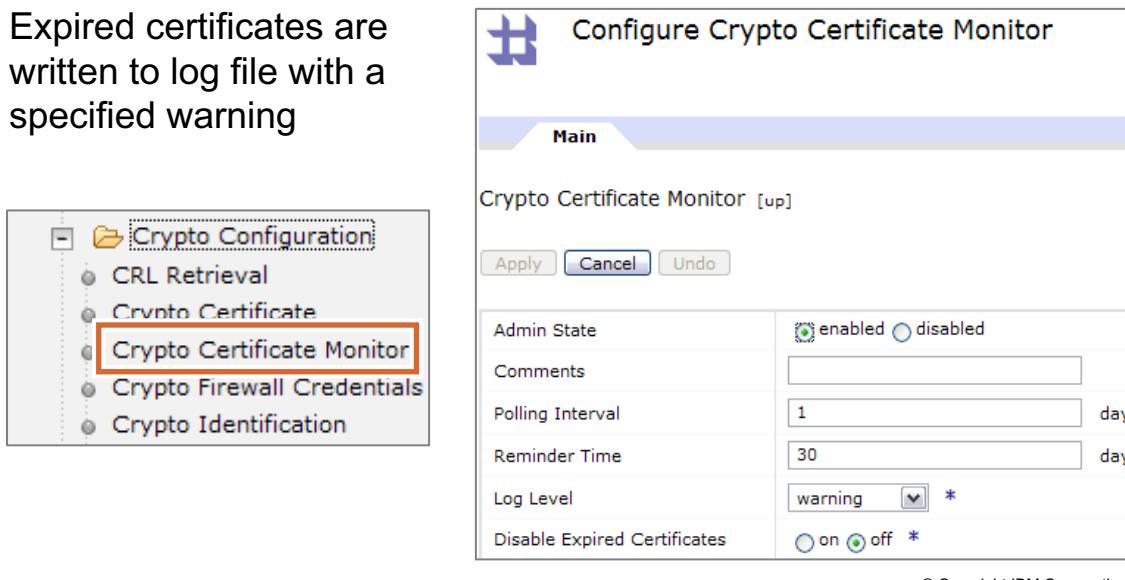
The protocol is either **http** or **ldap**. Appropriate fields display to support the protocol.

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

Another way to verify whether a certificate has been revoked is to use OCSP (Online Certificate Status Protocol). DataPower provides extension functions that use OCSP.

Crypto certification monitor

- Periodic task that runs on the appliance that checks expiration date of certificates
- Configure a **Crypto Certification Monitor** from vertical navigation bar by selecting **Objects > Crypto > Crypto Certificate Monitor**
- Expired certificates are written to log file with a specified warning



© Copyright IBM Corporation 2013

Figure 8-28. Crypto certification monitor

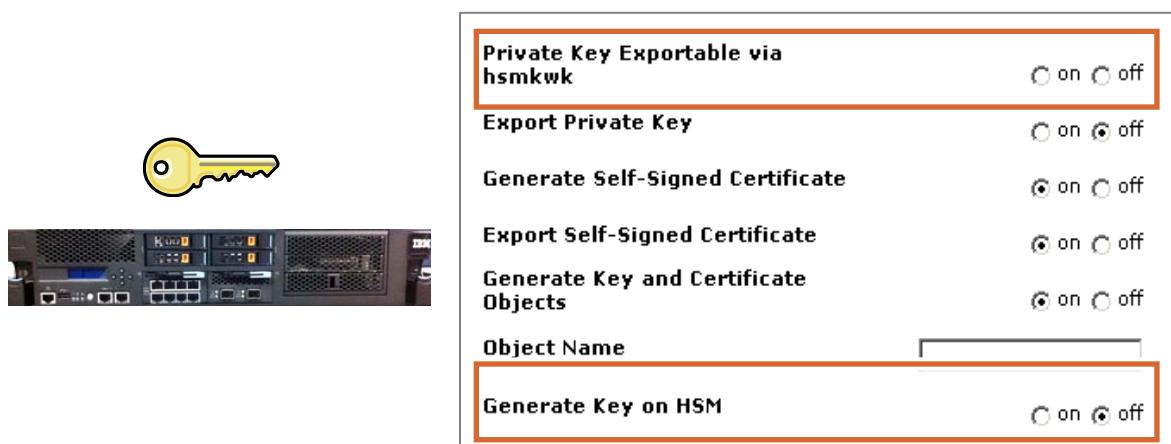
WE4013.0

Notes:

The polling interval specifies the frequency that certification expiration dates are checked. Remember, **time** refers to the number of days before the certification expiration event is written to the log file.

Hardware security module (HSM)

- Appliances with hardware security module (HSM) hardware that is installed can export or import private keys
 - Appliance where key is exported or imported must also have HSM hardware installed
- DataPower supports FIPS 140-2 level 2 and level 3 security



© Copyright IBM Corporation 2013

Figure 8-29. Hardware security module (HSM)

WE4013.0

Notes:

HSM is a piece of hardware with associated software and firmware that can perform a number of security functions. When you order DataPower, 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 this functionality through HSM hardware.

To export private keys on HSM hardware, the **Private Key Exportable via hsmkwk** must be selected from the "Crypto Tools" page. The HSM options appear only if you have an HSM installed.

Unit summary

Having completed this unit, you should be able to:

- Generate cryptographic keys by using the WebSphere DataPower tools
- Create a cryptographic identification credential object that contains a matching public and private key
- Create a cryptographic validation credential to validate certificates
- Set up certificate monitoring to ensure that certificates are up-to-date

© Copyright IBM Corporation 2013

Figure 8-30. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. Match the corresponding asymmetric and symmetric key encryptions:

| Description | Definition |
|-------------------|--|
| 1. Symmetric key | A. Uses two different keys for encrypting and decrypting |
| 2. Asymmetric key | B. Uses the same key |

2. What does a digital certificate contain?
- A. Your name, a serial number, expiration dates, a copy of the certificate holders public key, a digital signature
 - B. Your name, credit card number, expiration date, pin code
 - C. Digitalized company certificate with verification turned on
3. True or False: Keys that are generated on-board cannot be exported.

© Copyright IBM Corporation 2013

Figure 8-31. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. Match the corresponding asymmetric and symmetric key encryptions:

| Description | Definition |
|-------------------|--|
| 1. Symmetric key | A. Uses two different keys for encrypting and decrypting |
| 2. Asymmetric key | B. Uses the same key |

2. A. What does a digital certificate contain?

- ✓ A. Your name, a serial number, expiration dates, a copy of the certificate holders public key, a digital signature
- B. Your name, credit card number, expiration date, pin code
- C. Digitalized company certificate with verification turned on

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.

© Copyright IBM Corporation 2013

Figure 8-32. Checkpoint answers

WE4013.0

Notes:

Exercise 6



Creating cryptographic objects

© Copyright IBM Corporation 2013
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0.1

Figure 8-33. Exercise

WE4013.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

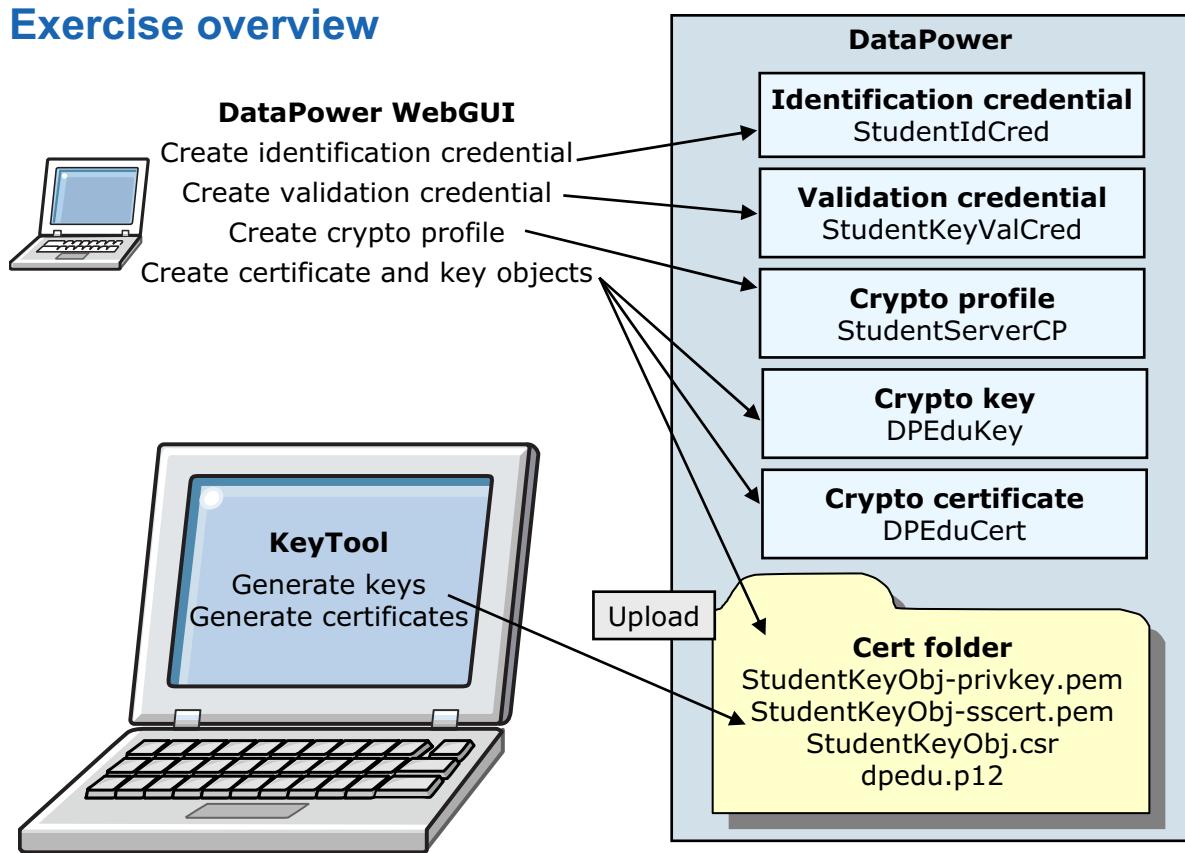
© Copyright IBM Corporation 2013

Figure 8-34. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 8-35. Exercise overview

WE4013.0

Notes:

Unit 9. Securing connections by using SSL

What this unit is about

This unit describes how to secure connections by using SSL to and from the DataPower appliance.

What you should be able to do

After completing this unit, you should be able to:

- Configure the WebSphere DataPower appliance to communicate by using SSL
- Associate an SSL proxy profile with keys and certificates
- Configure a user agent to select an SSL proxy profile that is associated with a URL pattern

How you will check your progress

- Checkpoint
- Exercise 7: Configuring SSL on DataPower services



Unit objectives

After completing this unit, you should be able to:

- Configure the DataPower appliance to communicate by using SSL
- Associate an SSL proxy profile with keys and certificates
- Configure a user agent to select an SSL proxy profile that is associated with a URL pattern

© Copyright IBM Corporation 2013

Figure 9-1. Unit objectives

WE4013.0

Notes:

What is SSL?

- SSL is Secure Sockets Layer, a set of cryptographic protocols that provides message security and integrity
 - Netscape Communications developed SSL
 - SSL Version 3.0 is the latest
- Transport Layer Security (TLS) is an IETF standards upgrade of SSL 3.0
 - Minor differences between SSL 3.0 and TLS 1.0
 - TLS 1.1 and 1.2 are available
- Generally, references to **SSL** also apply to **TLS**

© Copyright IBM Corporation 2013

Figure 9-2. What is SSL?

WE4013.0

Notes:

IETF is Internet Engineering Task Force: www.ietf.org.



SSL features

SSL provides:

- Message confidentiality
 - Uses asymmetric and symmetric key encryption
 - Uses a handshake when initiating contact
 - The handshake establishes a session key and encryption algorithm between both parties before any messages are sent
- Message integrity
 - Uses the combination of shared secret key and cryptographic hash function
 - Ensures that the contents of any messages are not modified
- Mutual authentication
 - Server always authenticates to client
 - Client optionally authenticates to server
 - Occurs during handshake

© Copyright IBM Corporation 2013

Figure 9-3. SSL features

WE4013.0

Notes:

A shared secret key is a symmetrical key.

SSL terminology

CipherSpec is a combination of:

- A cryptographic hash function
 - Is used to create the message digest or message authentication code (MAC)
- Encryption method algorithm
- Encryption method algorithm + hash function = CipherSpec

Cipher suite is a combination of:

- CipherSpec
- Authentication-key exchange algorithm
- CipherSpec + authentication-key exchange = cipher suite

© Copyright IBM Corporation 2013

Figure 9-4. SSL terminology

WE4013.0

Notes:

SSL handshake

- The handshake is the technique that is used to establish a secure communication link between an SSL client and an SSL server
- During handshake process, SSL will:
 - Negotiate the level of SSL to use
 - Decide on a cipher suite that both parties can use
 - Authenticate the server and (optionally) the client
 - Build a secret key that is used for this session only
- The SSL client initiates the handshake, sending a “hello” message to a target service, which becomes an SSL server

© Copyright IBM Corporation 2013

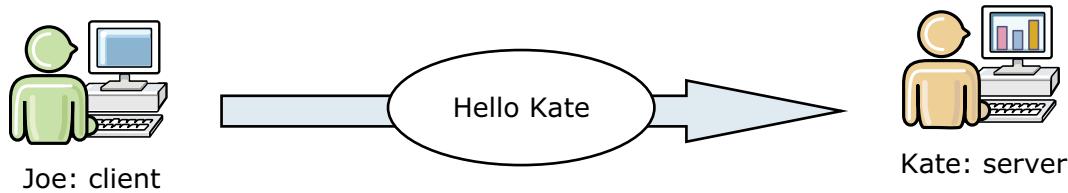
Figure 9-5. SSL handshake

WE4013.0

Notes:

SSL handshake: Client hello

- Joe (client) sends Kate (server) a `hello` message
 - The initial `hello` message contains a list of cipher suites that Joe (client) can use
- Joe is considered the SSL client since he initiated communication



© Copyright IBM Corporation 2013

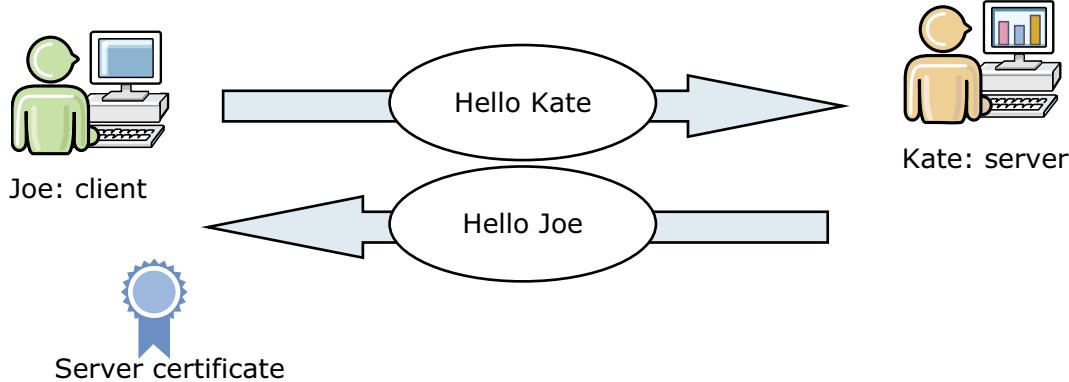
Figure 9-6. SSL handshake: Client hello

WE4013.0

Notes:

SSL handshake: Server hello

- Kate (the SSL server) responds with a **Hello**
 - Response contains the cipher suite that Kate selected from the list Joe sent her
- Kate also sends Joe a signed digital certificate that contains her public key
 - Joe needs a signed certificate in a keystore that he can use to verify Kate's signature



© Copyright IBM Corporation 2013

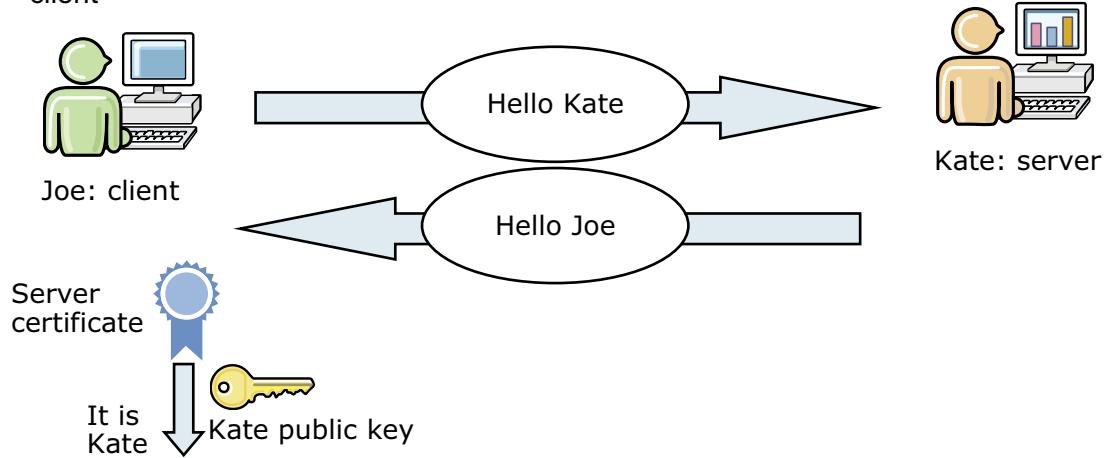
Figure 9-7. SSL handshake: Server hello

WE4013.0

Notes:

SSL handshake: Verify server certificate

- Receive Kate's public key from certificate
- Check server certificate for:
 - Certificate expiration
 - Verification of the certificate signature against a signed certificate in a keystore
 - Check certification revocation list to see whether the certificate might no longer be trusted
- If client authentication is being used, then Kate would request a digital certificate from the client



© Copyright IBM Corporation 2013

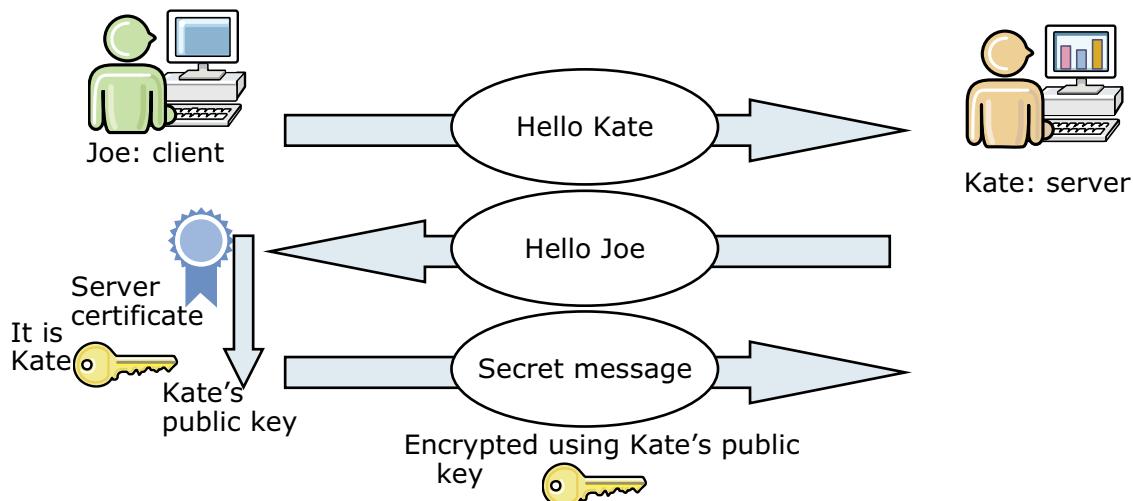
Figure 9-8. SSL handshake: Verify server certificate

WE4013.0

Notes:

SSL handshake: Client key exchange

- The client builds and sends the server a *secret message* that is encrypted by using Kate's (server) public key



© Copyright IBM Corporation 2013

Figure 9-9. SSL handshake: Client key exchange

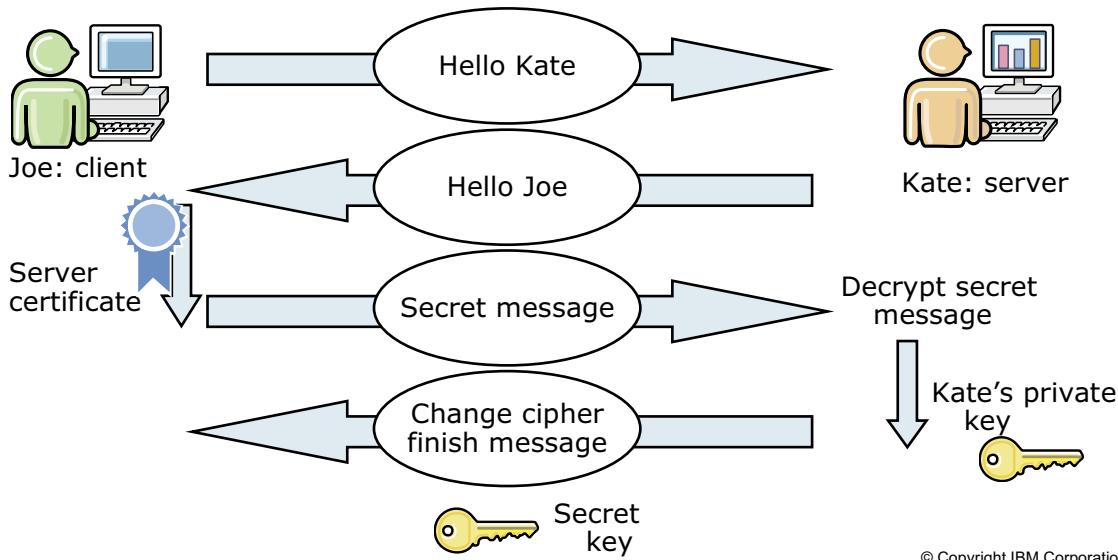
WE4013.0

Notes:

The secret message becomes a seed for calculating the symmetric key that is used for data exchange.

SSL handshake: Reply with secret key

- Kate (server) decrypts *secret message* by using her private key
- Both the client and the server generate a *secret key* by using the premaster secret in the *secret message* and random data that is generated during initial hello handshakes
 - Messages are encrypted by using the symmetric *secret key*



© Copyright IBM Corporation 2013

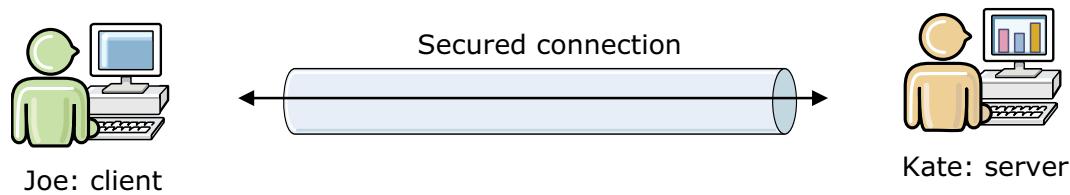
Figure 9-10. SSL handshake: Reply with secret key.

WE4013.0

Notes:

Securing the connection by using the SSL handshake

- Joe (SSL client) and Kate (SSL server) now agree upon:
 - Cipher suite
 - CipherSpec
 - Exchanged information to create a symmetric secret key
- Messages are sent inside a secure connection



© Copyright IBM Corporation 2013

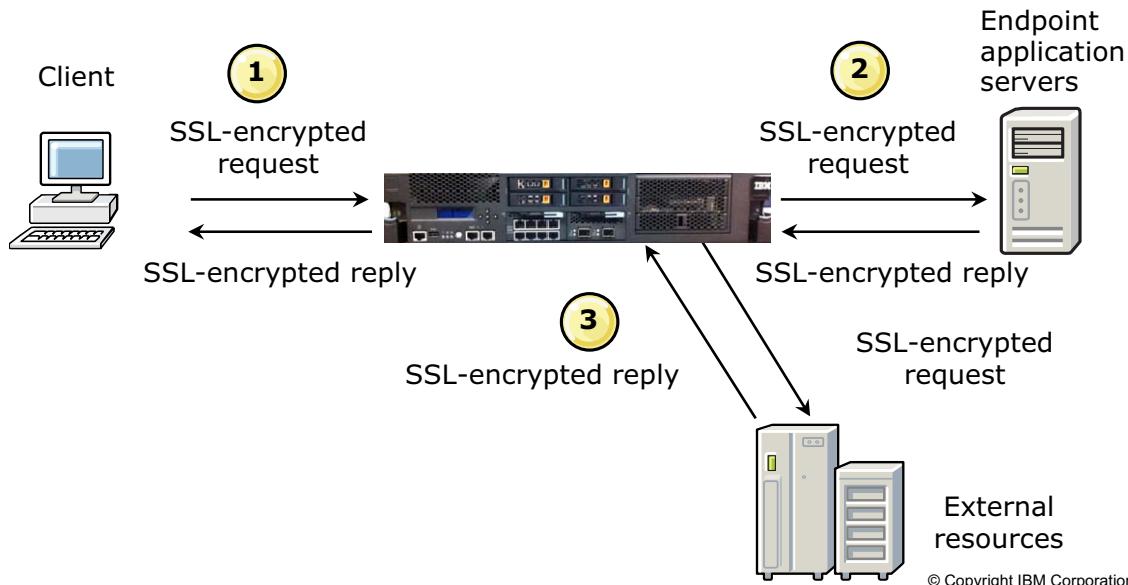
Figure 9-11. Securing the connection by using the SSL handshake

WE4013.0

Notes:

DataPower support for SSL

- DataPower appliance supports SSL:
 - From remote client to appliance
 - From appliance to external application server
 - From appliance to external resource such as authentication server



© Copyright IBM Corporation 2013

Figure 9-12. DataPower support for SSL

WE4013.0

Notes:



SSL proxy profile

- An SSL proxy profile:
 - Is a container of one or two crypto profiles, depending on SSL direction
 - Specifies SSL session caching and timeout
 - Might force client authentication
- SSL direction
 - Forward (client): acting as SSL client
 - Reverse (server): acting as SSL server
 - Two-Way: proxy profile supports SSL in both directions
- Automatically created when not using front side handlers
 - XSL proxy
 - XML firewall

SSL Proxy Profile: Two_way_demo [up]

| | |
|---------------------------------|---|
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| SSL Direction | <input type="button" value="Two-Way"/> * |
| Forward (Client) Crypto Profile | <input type="button" value="StudentClientCP"/> |
| Reverse (Server) Crypto Profile | <input type="button" value="StudentServerCP"/> |
| Server-side Session Caching | <input checked="" type="radio"/> on <input type="radio"/> off |

© Copyright IBM Corporation 2013

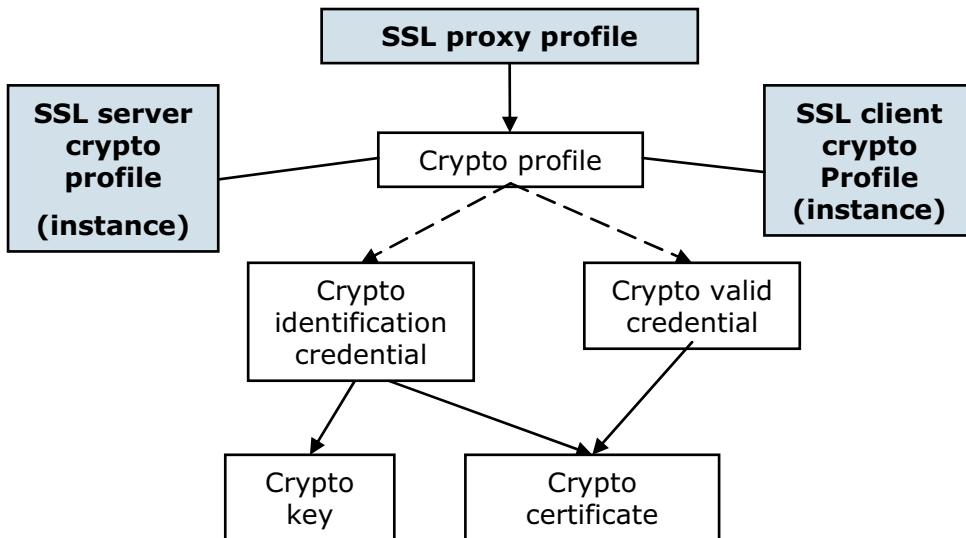
Figure 9-13. SSL proxy profile

WE4013.0

Notes:

SSL proxy profile: Crypto objects relationship

- Relationship between SSL proxy profile and crypto objects
 - Server: appliance is server, receives request to participate in SSL
 - Client: appliance is client, initiates request for SSL (`client_Hello`)



© Copyright IBM Corporation 2013

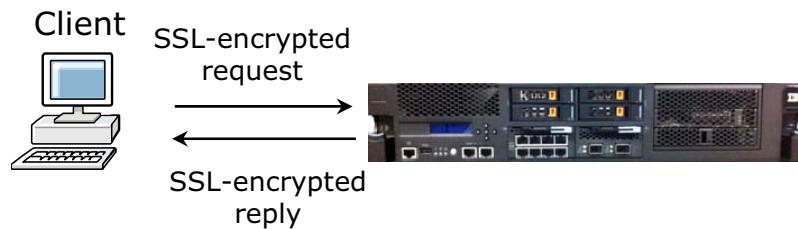
Figure 9-14. SSL proxy profile: Crypto objects relationship

WE4013.0

Notes:

Securing connections from client to appliance

- To set up SSL between client and appliance, you need to perform the following tasks:
 - DataPower appliance needs to supply a cryptographic certificate
 - Matching private key for certificate is maintained within appliance (ID credential)
 - Configure an **SSL proxy profile** and an **SSL server crypto profile** with cryptographic objects by linking to certificate-key pair
- Client validates the certificate that the appliance presents, which is often included in certificate chain (server-only authentication)
 - Appliance *might* request a certificate from client and validate (mutual authentication)
 - Appliance might use certificate authority (CA) certificates (there are many in `pubcert:` directory) to validate client certificates



© Copyright IBM Corporation 2013

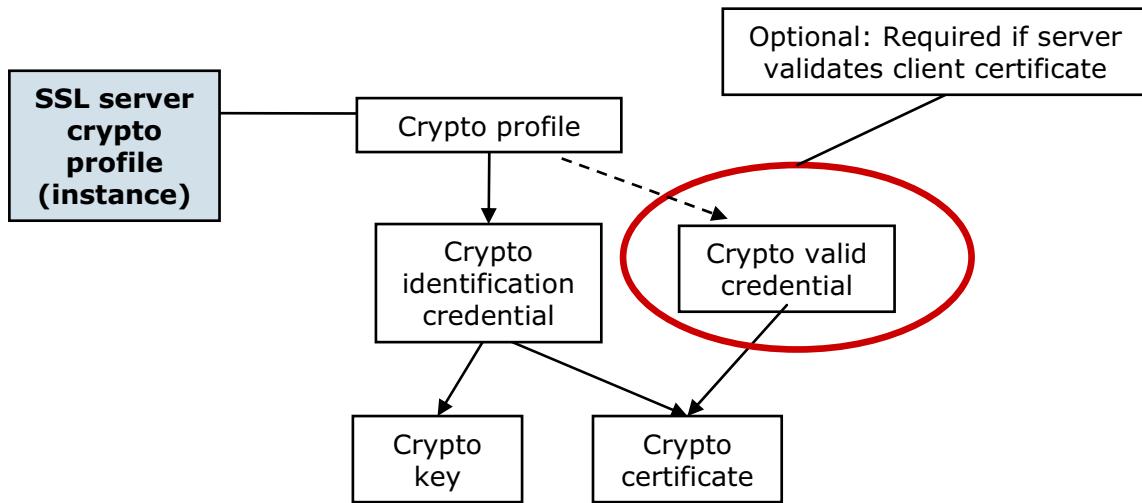
Figure 9-15. Securing connections from client to appliance

WE4013.0

Notes:

Step 1: Appliance supplies cryptographic certificate

- Create the crypto identification credential
 - Connects private key and certificate
- Might need crypto validation credential
 - Lists certificates that can validate the client certificate



© Copyright IBM Corporation 2013

Figure 9-16. Step 1: Appliance supplies cryptographic certificate

WE4013.0

Notes:



Step 2: SSL server crypto profile: non-FSH

- When the service does not use front side handlers
- Configure an SSL **server crypto profile** with cryptographic objects that link to the certificate-key pair
 - From the “Configure XML Firewall” page, select a **Server Crypto Profile** from the drop-down list, or create a new one
- SSL proxy profile** object that points to this crypto profile is automatically generated

The screenshot shows a configuration interface for a 'Front End'. It includes fields for 'Local IP Address' (0.0.0.0) and 'Port Number' (6081). A red box highlights the 'Reverse (Server) Crypto Profile' section, which contains a dropdown menu set to 'StudentServerCP' and three buttons: a downward arrow, a plus sign, and an ellipsis.

© Copyright IBM Corporation 2013

Figure 9-17. Step 2: SSL server crypto profile: non-FSH

WE4013.0

Notes:

An XSL proxy is configured in a similar fashion.



Step 2: SSL server crypto profile: FSH

- Front side handlers (FSH) point to an SSL proxy profile
 - Multi-protocol gateway
 - Web service proxy
- From the SSL Proxy field, create an **SSL proxy profile**

- Within the SSL proxy profile, create the server crypto profile

SSL Proxy Profile: EastAddressSSLProfile [up]

Administrative State: enabled disabled

SSL Direction: Reverse *

Reverse (Server) Crypto Profile: StudentServerCP +

© Copyright IBM Corporation 2013

Figure 9-18. Step 2: SSL server crypto profile: FSH

WE4013.0

Notes:



If you do not have an SSL server crypto profile...

Crypto Profile

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

- Create a profile:
 - Click the plus (+) button (new) for a Reverse (Server) Crypto Profile field
 - **Objects > Crypto Configuration > Crypto Profile**
- Validation Credentials are needed only if doing client authentication (mutual or two-way)

© Copyright IBM Corporation 2013

Figure 9-19. If you do not have an SSL server crypto profile...

WE4013.0

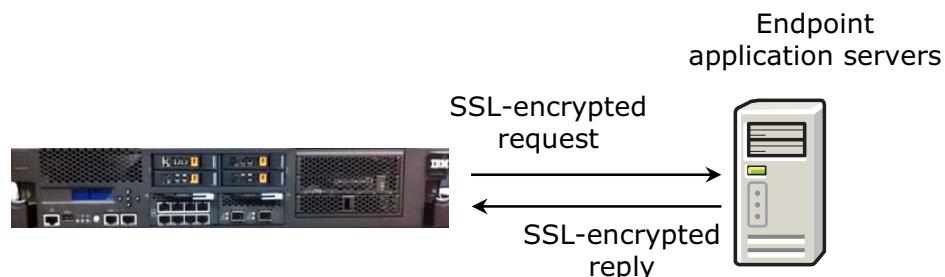
Notes:

The full default Ciphers field is HIGH:MEDIUM: !aNULL: !eNULL:@STRENGTH. This string means the SSL server supports AES or 3DES encryption as a primary choice, 128 bit RC2 and 128-bit RC4 encryption as a second choice, requires authentication, requires encryption, and the cipher list is sorted in order of encryption algorithm key length.

Securing the connection from appliance to external application server

To set up SSL between the appliance and an external application server, you need to perform the following actions:

- The DataPower appliance needs to validate the certificate that is supplied by the external applications server
 - The list of certificates that are used to validate is stored on the appliance
 - The application server or service contains a matching private key for the certificate
- Configure an SSL client crypto profile with cryptographic objects that link to validation credentials



© Copyright IBM Corporation 2013

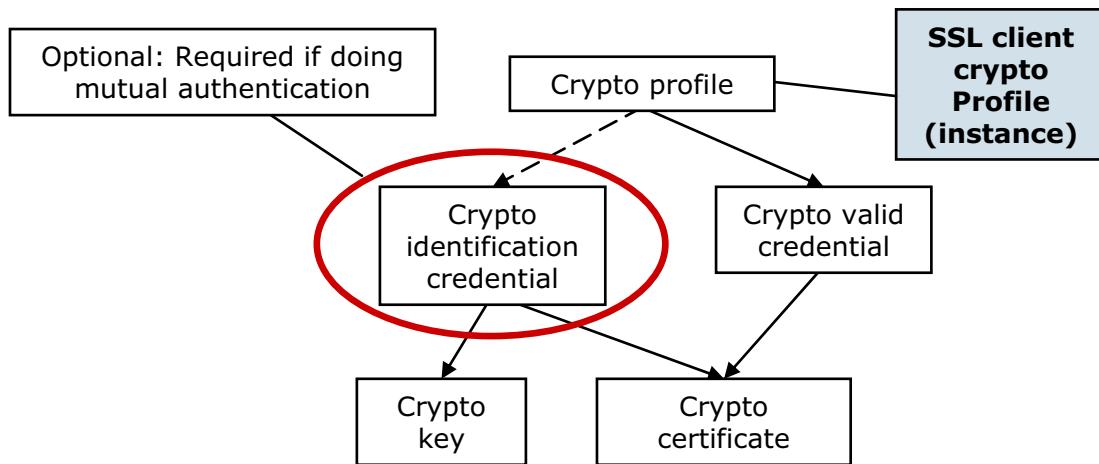
Figure 9-20. Securing the connection from appliance to external application server

WE4013.0

Notes:

Step 1: Appliance validates presented certificate

- Create the crypto validation credential
 - Lists certificates that can validate the client certificate
- Need identification credential also if doing mutual authentication



© Copyright IBM Corporation 2013

Figure 9-21. Step 1: Appliance validates presented certificate

WE4013.0

Notes:

Step 2: Configuring an SSL client crypto profile

- Configure an SSL client crypto profile that validates the presented certificate
- From the “Configure XML Firewall” or the “Configure Multi-Protocol Gateway” page, select an **SSL Client Crypto Profile** from the drop-down list



- For a web service proxy:
 - Go to the **Proxy Settings** tab, select (or create) an SSL proxy profile
 - Within the SSL proxy profile object, select (or create) a client crypto profile



- Within the SSL proxy profile object, select (or create) a client crypto profile



© Copyright IBM Corporation 2013

Figure 9-22. Step 2: Configuring an SSL client crypto profile

WE4013.0

Notes:



If you do not have an SSL client crypto profile...

Crypto Profile: StudentClientCP [up]

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

| | |
|----------------------------|--|
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Identification Credentials | (none) + ... |
| Validation Credentials | StudentKeyValCred + ... |
| Ciphers | HIGH:MEDIUM::aNULL::eNULL:@STRENGTH |
| Options | <input checked="" type="checkbox"/> Enable default settings <input type="checkbox"/> Disable SSL version 2 <input type="checkbox"/> Disable SSL version 3 <input type="checkbox"/> Disable TLS version 1 <input type="checkbox"/> Permit insecure SSL renegotiation to a legacy SSL client <small>*</small> |
| Send Client CA List | <input type="radio"/> on <input checked="" type="radio"/> off |

- Create a profile:
 - Click the plus (+) button (new) for a Forward (Client) Crypto Profile field
 - **Objects > Crypto Configuration > Crypto Profile**
- Identification Credential needed only if doing mutual authentication

© Copyright IBM Corporation 2013

Figure 9-23. If you do not have an SSL client crypto profile...

WE4013.0

Notes:

The full default Ciphers field is HIGH:MEDIUM: !aNULL: !eNULL:@STRENGTH. This string means the SSL server supports AES or 3DES encryption as a primary choice, 128 bit RC2 and 128-bit RC4 encryption as a second choice, requires authentication, requires encryption, and the cipher list is sorted in order of encryption algorithm key length.



SSL Proxy Profile list

- The list shows which 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
 - Objects > Crypto Configuration > SSL Proxy Profile**

Configure SSL Proxy Profile

[Refresh List](#)

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

[Add](#)

© Copyright IBM Corporation 2013

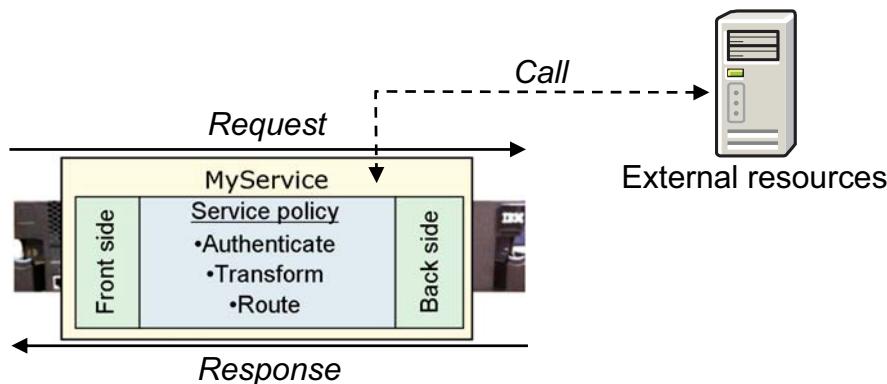
Figure 9-24. SSL Proxy Profile list

WE4013.0

Notes:

User agent

- User agent affects communications to external resources
 - A URL matching expression is used to associate a particular behavior
 - Associated with a service
- Many types of communication patterns can be affected
 - Each tab controls a particular behavior



© Copyright IBM Corporation 2013

Figure 9-25. User agent

WE4013.0

Notes:

These policies are frequently used when the url-open extension element is used within a style sheet to communicate with SSL-secured external resources.

Configuring a user agent

- The **default** XML manager object uses a **default** user agent
 - Alternatively, from the vertical navigation bar, select **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
 - Select the appropriate tab to specify a behavior policy
- There are many areas of communication that can have policies:
 - Proxy policy: specifies a URL match expression to forward to a remote address and port
 - Basic authentication 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
 - Public key authentication policy: associates a specific private key to use during public key authentication

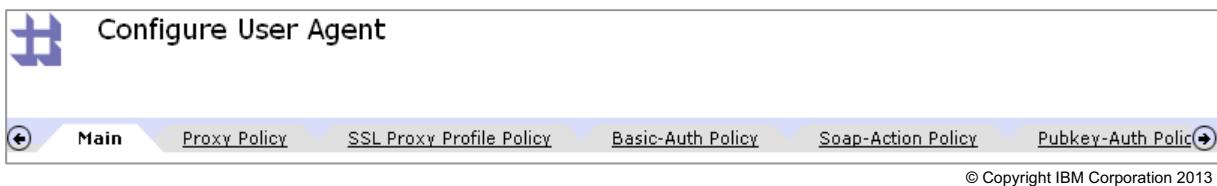


Figure 9-26. Configuring a user agent

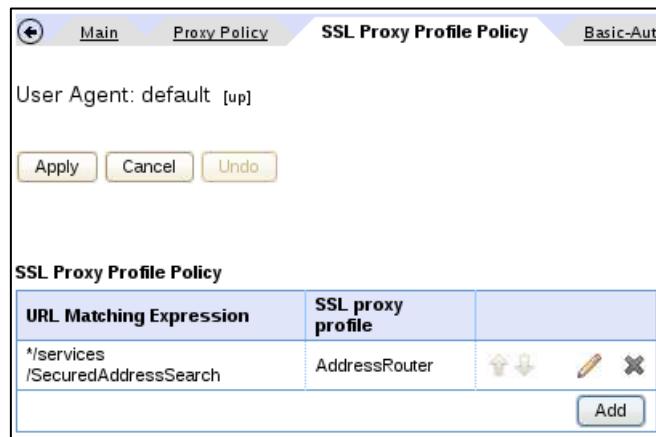
WE4013.0

Notes:



Creating a user agent configuration

- Configure a user agent to use an SSL proxy profile to communicate with an external resource
 - Select the **SSL Proxy Profile Policy** tab
 - Specify a URL match expression and a corresponding SSL proxy profile



© Copyright IBM Corporation 2013

Figure 9-27. Creating a user agent configuration

WE4013.0

Notes:

You might use this policy if a style sheet used a url-open extension element to call a web service that requires SSL. If the service sees a request that goes to the specified URL, it applies the SSL proxy profile that is listed.



Unit summary

Having completed this unit, you should be able to:

- Configure the DataPower appliance to communicate by using SSL
- Associate an SSL proxy profile with keys and certificates
- Configure a user agent to select an SSL proxy profile that is associated with a URL pattern

© Copyright IBM Corporation 2013

Figure 9-28. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. During handshake process, does SSL select process A or B?
 - A. Process A
 - Negotiate the level of SSL to use
 - Decide on a cipher suite that both parties can use
 - Authenticate the server and (optionally) the client
 - Build a secret key that is used for this session only
 - B. Process B
 - Negotiate the network communication protocol to use
 - Agree on which types of key to use
 - Create a random route that is untraceable
2. True or False: An SSL crypto profile identifies the crypto objects to use in SSL communication (identification credential or validation credential).
3. True or False: An SSL proxy profile references an SSL crypto profile for the client only.

© Copyright IBM Corporation 2013

Figure 9-29. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.

Checkpoint answers

1. A. During handshake process, does SSL select process A or B?

✓ A. **Process A**

- Negotiate the level of SSL to use
- Decide on a cipher suite that both parties can use
- Authenticate the server and (optionally) the client
- Build a secret key that is used for this session only

B. Process B

- Negotiate the network communication protocol to use
- Agree on which types of key to use
- Create a random route that is untraceable

2. **True.** An SSL crypto profile identifies the crypto objects to use in SSL communication (identification credential or validation credential).

3. **False.** The SSL proxy profile references an SSL crypto profile for the client and server.

© Copyright IBM Corporation 2013

Figure 9-30. Checkpoint answers

WE4013.0

Notes:

Exercise 7



Using SSL to secure connections

© Copyright IBM Corporation 2013

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

7.0.1

Figure 9-31. Exercise

WE4013.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

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

© Copyright IBM Corporation 2013

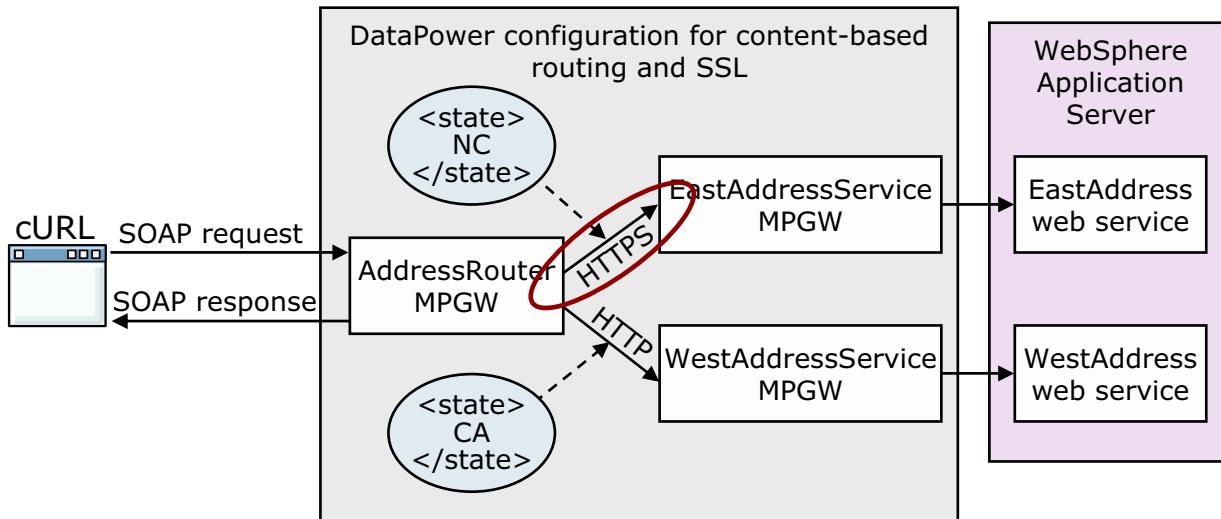
Figure 9-32. Exercise objectives

WE4013.0

Notes:

Exercise overview

- Create an SSL server configuration on the front side of the EastAddressSearch service
- Create an SSL client configuration on the back side of the AddressRouter service



© Copyright IBM Corporation 2013

Figure 9-33. Exercise overview

WE4013.0

Notes:

Unit 10. XML threat protection

What this unit is about

This unit covers the vulnerabilities that exist in XML messaging.

What you should be able to do

After completing this unit, you should be able to:

- Explain possible attack scenarios that are involved in XML-based applications
- Describe the various types of XML attacks
- Use the WebSphere DataPower SOA Appliance to protect against XML attacks

How you will check your progress

- Checkpoint
- Exercise 8: Protect against XML threats

Unit objectives

After completing this unit, you should be able to:

- Explain possible attack scenarios that are involved in XML-based applications
- Describe the various types of XML attacks
- Use the WebSphere DataPower SOA Appliance to protect against XML attacks

© Copyright IBM Corporation 2013

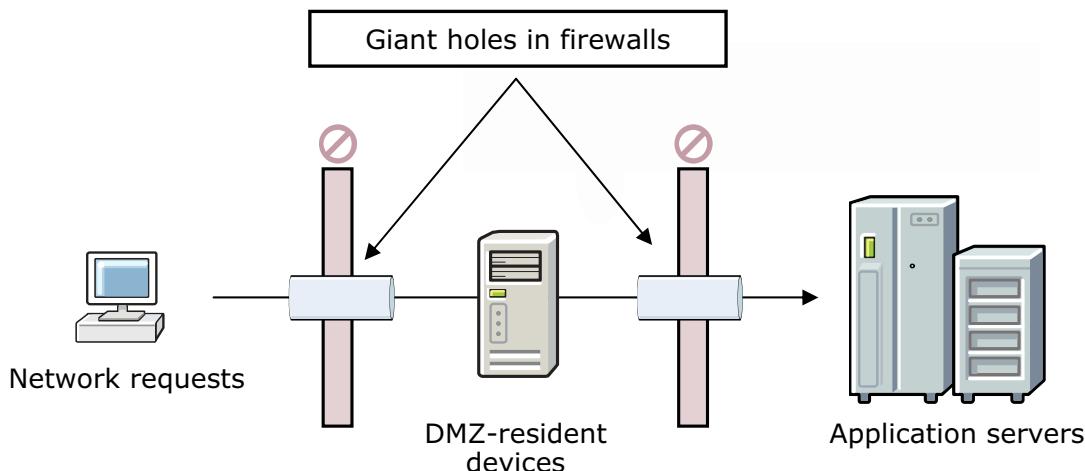
Figure 10-1. Unit objectives

WE4013.0

Notes:

What are the security concerns?

- XML-based web services easily expose back-end systems to customers and partners
- Traditional security devices do not secure XML and SOAP traffic



© Copyright IBM Corporation 2013

Figure 10-2. What are the security concerns?

WE4013.0

Notes:

Web services are based on SOAP, XML, and HTTP. Many of the messages by these services and protocols are sent to port 80, so they pass through the firewall.

These protocols gained widespread attention because of their simplicity and ease of use. However, along with their security, a new class of issues and problems is introduced. This new class of problems is known collectively as XML threats.

Traditional systems and exposure

- The “firewall-safe” feature of SOAP/XML can easily be used to launch XML attacks
 - XML validation is typically “off” for performance reasons
- No rate-limiting functions exist either with the product or built into the architecture
 - Lack of traffic throttling, classification, and screening
 - It is a good practice to place these functions up front in the architecture
 - Filter to block potentially malicious messages before they are allowed to flow to the back-end systems
 - Otherwise, damage would already be initiated, introducing system stability and availability issues
- Traditional edge devices are not smart enough to check for these attacks

© Copyright IBM Corporation 2013

Figure 10-3. Traditional systems and exposure

WE4013.0

Notes:

An XML firewall can do:

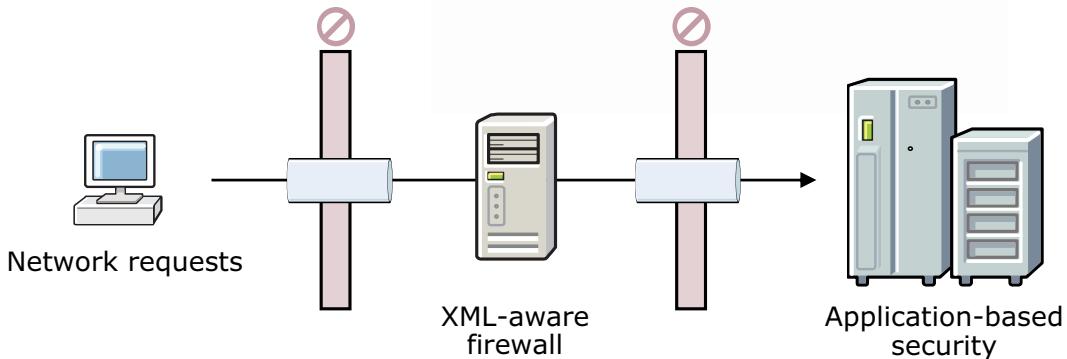
- XML denial of service
- Unauthorized access
- Data integrity and confidentiality
- System compromise

Rate-limiting functions: In computer networks, rate limits are used to control traffic that is sent or received on a network interface. Traffic that is less than or equal to the specified rate is sent, whereas traffic that exceeds the rate is dropped or delayed. A device or mechanism that implements rate limits are called a rate limiter. Rate limiting is done by defining and enforcing a policy that encapsulates the parameters within which rate limit is administered.

Edge devices: The EoN (Edge of Network), is the part of the network topology where traffic enters or leaves a corporate network.

Addressing the security concerns

- Multiple levels of defense: “Defense in depth” strategy
- First level
 - XML security gateway for enhanced security, scalability, and simplicity
- Second level
 - Application server for more processing



© Copyright IBM Corporation 2013

Figure 10-4. Addressing the security concerns

WE4013.0

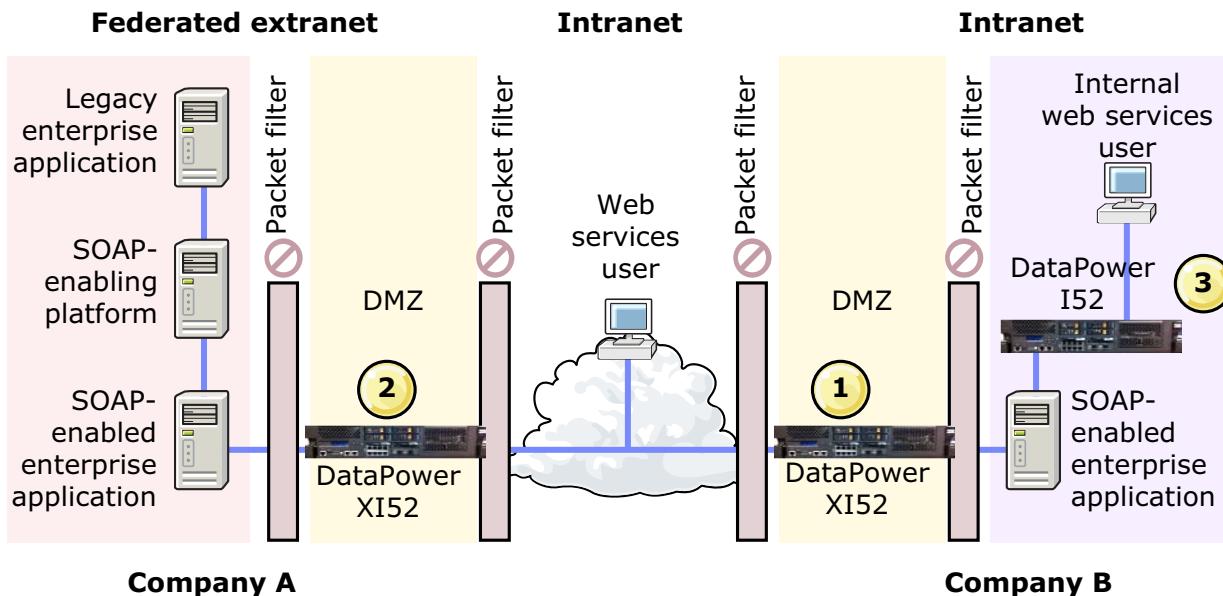
Notes:

Web services security can be enforced at two levels:

- First level of defense: XML Security Gateway
 - Performance: at least 10 times more improvement over software
 - Scalability: can minimize the number of servers
 - Manageability: fewer enforcement points simplify configuration
 - Availability: XDoS checking protects application and web servers
- Second level of defense: web services application
 - Manageability: can integrate with container-based security
 - Security: business-specific security is embedded within an application

Three high-level deployment patterns

- Three typical deployment patterns:



© Copyright IBM Corporation 2013

Figure 10-5. Three high-level deployment patterns

WE4013.0

Notes:

- XS40 in the DMZ of company A.** It is set up with an XML firewall. Schema validation is enabled to offload the application server. XML threats settings are configured to valid sizes. It uses AAA to check the client ID and authority.
- XS40 deployed in the DMZ of company B.** It is also set up with an XML firewall to verify outgoing requests. Outbound AAA verifies the ID and authority. Company B and company A implemented a federated extranet (perhaps by Tivoli Federated Identity Manager). XS40 injects SAML assertions and attributes into outbound messages.
- XS40 deployed in the intranet of company A.** Since the requests are coming from within, company A decided to apply different security rules. An XML firewall is defined to allow only specific IP addresses access to the internal application.

Four types of XML attacks

- XML denial of service (XDoS)
 - Slowing down or disabling a web service so that service requests are hampered or denied
- Unauthorized access
 - Gaining unauthorized access to a web service or its data
- Data integrity and confidentiality
 - Data integrity attacks of web service requests, responses, or underlying databases
- System compromise
 - Corrupting the web service itself or the servers that host it

© Copyright IBM Corporation 2013

Figure 10-6. Four types of XML attacks

WE4013.0

Notes:

These attacks are covered in detail in subsequent slides.

XML denial of service (XDoS): Single-message attacks

- Jumbo payloads
 - Sending an excessively large XML message to exhaust memory and processor capacity on the target system
- Recursive elements
 - XML messages that can be used to force recursive entity expansion or other repeated processing to exhaust server resources
- Mega tags
 - Otherwise valid XML messages have excessively long element names; might lead to buffer overruns
- Coercive parsing
 - XML messages are constructed to be difficult to parse and consume the resources of the machine
- Public key DoS
 - Forcing resource exhaustion on the recipient by using the asymmetric nature of public key operations
 - Transmitting a message with many long-key-lengths, computationally expensive digital signatures

© Copyright IBM Corporation 2013

Figure 10-7. XML denial of service (XDoS): Single-message attacks

WE4013.0

Notes:

XML denial of service (XDoS): Multiple-message attacks

- XML flood
 - Sending thousands of useless messages per second to tie up a web service
 - This attack can be combined with a replay attack to bypass authentication, for example, and single message XDoS to increase its impact
- Resource hijack
 - Sending messages that lock or reserve resources on the target server as part of a never-completed transaction
 - For example, messages that intentionally force lock contention on resources or similar situations

© Copyright IBM Corporation 2013

Figure 10-8. XML denial of service (XDoS): Multiple-message attacks

WE4013.0

Notes:

Unauthorized access attacks

- Dictionary attacks
 - Guessing the password of a valid user by a brute force search through dictionary words
- Falsified messages
 - Faking that a message is from a valid user
 - Using an intermediary to gain a valid message and then modifying it to send a different message
- Replay attack
 - Resending a previously valid message for malicious effect
 - Possibly where only parts of the message (such as the security token) are replayed

© Copyright IBM Corporation 2013

Figure 10-9. Unauthorized access attacks

WE4013.0

Notes:

Data integrity and confidentiality attacks

- Message tampering
 - Modifying parts of a request or response; most dangerous when undetected
- Data tampering
 - Using weaknesses in the access control mechanism that permits the attacker to make unauthorized calls to the web service to alter data
- Message snooping
 - A direct attack on data privacy by examining all or part of the content of a message
- XPath or XSLT injection
 - Injection of expressions into the application logic
- SQL injection
 - Modifying SQL in XML to obtain more data than the service was designed to return
- WSDL enumeration
 - Examining the services that are listed in the WSDL to guess at and gain access to unlisted services
- Routing detour
 - Using SOAP routing headers to access internal web services

© Copyright IBM Corporation 2013

Figure 10-10. Data integrity and confidentiality attacks

WE4013.0

Notes:

System compromise attacks

- Malicious include causes a web service to:
 - Include invalid external data in output
 - Return privileged files from the server file system
- For example, by embedding “file:” URLs to return UNIX password files or other privileged data to the attacker
- Memory space breach
 - Accomplished by using stack overflow, buffer overrun, or heap error
 - Allows execution of arbitrary code that is supplied by the attacker with permission of host processes
- XML encapsulation
 - Embedded system command in XML payload, for example, by the CDATA tag
- XML virus (X-Virus)
 - Using SOAP with attachments or other attachment mechanisms to transmit malicious executable code, such as viruses or worms

© Copyright IBM Corporation 2013

Figure 10-11. System compromise attacks

WE4013.0

Notes:



XML parser limits

- In the “Configure XML Manager” page, select the **XML Parser** tab
- This choice enhances security and stability by protecting against DoS attacks and runaway data
- XML parser limits:
 - Impose limits on various characteristics of XML documents that the device parses
- Parser limits are assigned to an XML manager, and any service that manager supports inherits these limits
 - Note: Service-specific settings can override these inherited properties

XML Manager: default [up]

Apply **Cancel** **Undo**

These settings do not impact resource allocation, and are used only as part of your application's processing logic.

| | |
|----------------------------------|----------|
| XML Bytes Scanned | 4194304 |
| XML Element Depth | 512 |
| XML Attribute Count | 128 |
| XML Maximum Node Size | 33554432 |
| XML Maximum Distinct Prefixes | 0 |
| XML Maximum Distinct Namespaces | 0 |
| XML Maximum Distinct Local Names | 0 |
| XML External Reference Handling | Forbid |

© Copyright IBM Corporation 2013

Figure 10-12. XML parser limits

WE4013.0

Notes:

You can get to the list of XML Managers by going to **Objects > XML Processing > XML Manager**.

Parser limits:

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

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

Configure XML Firewall

XML Threat Protection

General Advanced Stylesheet Params Headers Monitors XML Threat Protection

Apply Cancel Delete Clone Export View Log View Status Show Probe Validate Conformance Help

XML Firewall Service status: [up]

Miscellaneous XML Threat Protection Configuration

This page lets you configure the device for protection against the following threats:

- Single Message XML Denial of Service (XDoS) Protection
- Multiple Message XML Denial of Service (MMXDoS) Protection
- Padding Oracle Protection
- Message Tampering Protection
- SQL Injection Protection
- Protocol Threat Protection
- XML Virus (X-Virus) Protection
- Dictionary Attack Protection

- Maximize protection against XML-based threats
- XML threat protection types:
 - Single message XML denial of service (XDoS) protection
 - Multiple message XML denial of service (MMXDoS) protection
 - Message tampering protection
 - Protocol threat protection
 - XML virus (X-Virus) protection
 - Dictionary attack protection

© Copyright IBM Corporation 2013

Figure 10-13. XML threat protection

WE4013.0

Notes:

Using a coordinated set of objects and firewall configuration objects, it is possible to maximize firewall protection against XML threats (malicious attempts to disrupt service by the firewall or the back-end server).



XML threat protection: Single message XDoS

- XML threat protection settings:

Single Message XML Denial of Service (XDoS) Protection

| | |
|---|---|
| Max. Message Size | <input type="text" value="0"/> KB |
| Override XML Manager parser limits <input checked="" type="radio"/> on <input type="radio"/> off | |
| Max. XML Attribute Count | <input type="text" value="128"/> * |
| Max. XML Bytes Scanned | <input type="text" value="4194304"/> bytes * |
| Max. XML Element Depth | <input type="text" value="512"/> * |
| Max. XML Node Size | <input type="text" value="33554432"/> bytes * |
| XML Maximum Distinct Prefixes | <input type="text" value="0"/> * |
| XML Maximum Distinct Namespaces | <input type="text" value="0"/> * |
| XML Maximum Distinct Local Names | <input type="text" value="0"/> * |
| Attachment Byte Count Limit | <input type="text" value="2000000000"/> bytes * |
| Attachment Package Byte Count Limit | <input type="text" value="0"/> bytes * |
| Recursive Entity Protection <input checked="" type="radio"/> on <input type="radio"/> off | |

© Copyright IBM Corporation 2013

Figure 10-14. XML threat protection: Single message XDoS

WE4013.0

Notes:

These settings provide protection against a single malicious XML message. A number of the parameters that are used to provide this protection are set in the XML Manager. This page offers the opportunity to override the XML Manager with firewall-level settings (go to **Override XML Manager parser limits** and select the **on** or the **off** radio button).

- **Max. Message Size:** the maximum allowed size, in KB, of any provided message. The range is 0–2097151. The default is 0, which means that no limit is enforced.
- **Override XML Manager parser limits:** When left at the default of **Off**, the parser limits set in the XML Manager that is used by this firewall remain in effect.
- **Max XML Attribute Count:** an integer value that limits the number of attributes for any provided element.
- **Max XML Bytes Scanned:** This limits the number of bytes contained in any provided XML message. A value of 0 enforces no limit.
- **Max XML Element Depth:** This limits the depth of nested elements in an XML message.

- **Max. XML Node Size:** This limits the size of any one XML node. The minimum value that allowed is 1. Note: This value might be larger than the **Max. XML Bytes Scanned** value, but the limit on the total number of bytes scanned takes precedence.
- **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 URLs in a document.
- **XML Maximum Distinct Local Names:** defines the maximum number of distinct XML local names in a document.
- **Attachment Byte Count Limit:** This limits the size, in bytes, of any single attachment to the message. Enter 0 to enforce no limit. This property setting is not available in the XML Manager parser limits.
- **Attachment Package Byte Count Limit:** defines the maximum number of bytes allowed for all parts of an attachment package, including the root part. This property setting is not available in the XML Manager parser limits.
- **Recursive Entity Protection:** always enabled.



XML threat protection: Multiple message XDoS

- Protects against a denial of service attack with multiple XML messages sent to a service
- XML threat protection settings:
 - Max. Duration for a Request
 - Interval for Measuring Request Rate from Host
 - Max. Request Rate from Host
 - Interval for Measuring Request Rate for Firewall
 - Max. Request Rate for Firewall
 - Block Interval
 - Log Level

Multiple Message XML Denial of Service (MMXDoS) Protection

Enabling MMXDoS will create duration and count monitors and attach them to this firewall.

| | |
|---|---|
| Enable MMXDoS Protection | <input checked="" type="radio"/> on <input type="radio"/> off |
| Max. Duration for a Request | <input type="text"/> msec * |
| Interval for Measuring Request Rate from Host | <input type="text" value="1000"/> msec * |
| Max. Request Rate from Host | <input type="text"/> messages/interval * |
| Interval for Measuring Request Rate for Firewall | <input type="text" value="1000"/> msec * |
| Max. Request Rate for Firewall | <input type="text"/> messages/interval * |
| Block Interval | <input type="text" value="0"/> msec * |
| Log Level | <input type="text" value="error"/> * |

© Copyright IBM Corporation 2013

Figure 10-15. XML threat protection: Multiple message XDoS

WE4013.0

Notes:

- Max. Duration for a Request:** indicates the maximum number of milliseconds allowed for processing any one request.
- Interval for Measuring Request from Host:** an integer in milliseconds, used for measuring the rate of requests from any provided host. The default is 1000, and it measures requests per second.
- Max. Request Rate from Host:** an integer that sets the maximum number of requests that can be received, within the interval period, from any one host.
- Interval for Measuring Request Rate for Firewall:** an integer that sets the interval, in milliseconds, for measuring the request rate for the entire firewall.
- Max Request Rate for Firewall:** the maximum number of requests that can be received, within the interval period, by the firewall.
- Block Interval:** an integer that sets the time, in milliseconds, for which the firewall blocks access after one of the other thresholds are reached.

- **Log Level:** the level at which log messages generate those threat protection thresholds. When a threshold is reached, the firewall generates a log message.



XML threat protection: Protocol threats

- Protocol threat protection prevents messages that are formatted according to unauthorized protocols from passing through the firewall
- Protocol Threat Protection** settings:
 - Request Type
 - Response Type
 - Request HTTP Version
 - Response HTTP Version

Protocol Threat Protection

Request HTTP Version

Response HTTP Version

© Copyright IBM Corporation 2013

Figure 10-16. XML threat protection: Protocol threats

WE4013.0

Notes:

For the Web Service Proxy service, the **Protocol Threat Protection** section is different. It asks you to characterize the client traffic type and the server traffic type as either Non-XML, pass-through, SOAP, or XML. SOAP is the default.

XML threat protection: XML virus

- Viruses are typically contained in message attachments
- First, specify how to handle request/response attachments
- If you allow attachments, you can use these actions in the processing policy to invoke an external virus scan service:
 - Filter action
 - Results action
 - Anti-virus action (V3.6.1)

XML Virus (X-Virus) Protection:

Select a firewall processing policy which includes a Filter action that uses the Processing Control File “store:///Virus-ScanAttachments.xsl”. This Filter action must specify an Output context name (for example, “attachments”) and must also employ a style sheet parameter named “{http://www.datapower.com/param/config}SendTo” with the value set to the URL of your virus scanner.

You might elect to add a Filter action as described to an existing or to a new processing policy. Use the Firewall Policy inputs under “Message Tampering Protection” to select, edit, or create the wanted processing policy.

© Copyright IBM Corporation 2013

Figure 10-17. XML threat protection: XML virus

WE4013.0

Notes:

These settings provide protection against viruses that typically flow as message attachments.

You can use either a **Results action** or **Filter action** to call the external virus scanner service.

For the **Filter** action approach, use the style sheet
`store:///Virus-ScanAttachment.xsl`, with a style sheet parameter that contains the URL of the virus scanner.

With V3.6.1, there is a new **Anti-Virus** action that can be used to specify interaction with an external virus that scans service. The **Results** action and **Filter** action approaches continue to work.

In all cases, communication with the external virus that scans service is done by using ICAP (Internet Content Adaptation Protocol).

This slide shows the section as seen in the Web Service Proxy service. An XML firewall service does not list the "Requests with attachments" choices, as those options are already available on the **General** tab.

XML threat protection: Dictionary attack

- Repeatedly denied requests for access detect a dictionary attack.
 - A message count monitor is employed to provide dictionary attack protection
- A service can monitor access requests through a AAA action that is activated on every request for a service
 - When the count of rejected access requests reaches a certain level, the service can send a notification and even deny service for a certain period

© Copyright IBM Corporation 2013

Figure 10-18. XML threat protection: Dictionary attack

WE4013.0

Notes:

The Web Service Proxy service does not have a tab to specify associated monitors, such as the count monitor mentioned here. In the page for a Web Service Proxy service, this section also allows you to specify count and duration monitor objects.

Dictionary Attack Protection

Dictionary attacks that are detected by repeatedly denied requests for access, which is typically a visible symptom of someone that probes for data dictionary definitions to use. The firewall can monitor access requests through an AAA (Authentication and Authorization Action) that is activated on every request for service. When the count of rejected access requests reaches a certain level, the Firewall can send notification and even deny service for a time.

To create this protection, it is necessary to create a Count Monitor object which has its Measure property that is set to "xpath". You can invoke the User page to create a new Count Monitor by clicking "+" alongside the Count Monitor inputs.

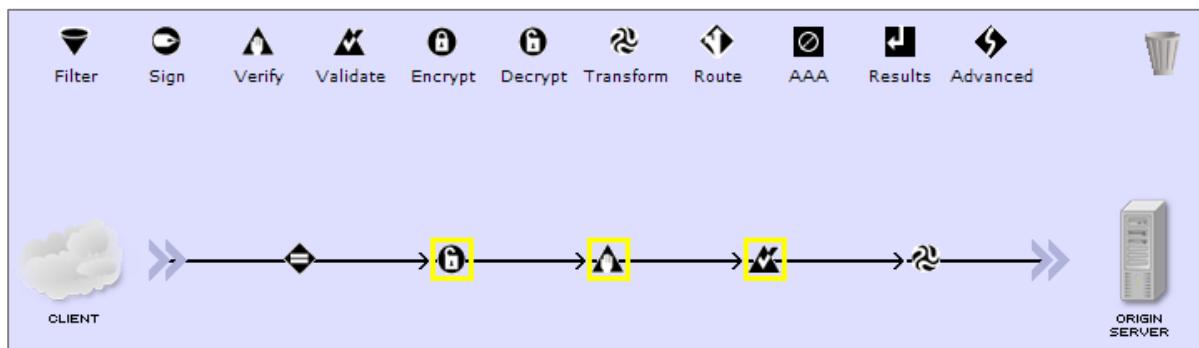
This Count Monitor must then be identified an AAA action as the Rejected Counter. This action must be part of the Firewall.

Policy that is identified for this firewall. You can add this action to the current policy by clicking the "..." button alongside the policy input under "Message Tampering." Then, drag an AAA icon onto the processing line and double-click the icon.

Finally, the count monitor created for this purpose must be listed as one of the Count Monitors that is associated with this firewall. Use the Count Monitors inputs on the Monitors tab to accomplish this task.

Message tampering

- Message tampering protection can employ:
 - **Encrypt** and **decrypt** the message to hide the content
 - **Sign** and **verify** the message to ensure message integrity
 - **Validate** the schema of the message content for proper structure



© Copyright IBM Corporation 2013

Figure 10-19. Message tampering

WE4013.0

Notes:

Message tampering protection employs schema validation that is done on submitted messages. This action is done to prevent messages that are altered and no longer pass validation from reaching the back-end server endpoints.

You can also add a **Verify** action to check incoming digitally signed documents.

Your policy can use a **Sign** action to digitally sign a document that leaves the service so that the receiver can check against the signature upon receipt.

SQL injection attack

- A hacking technique that attempts to pass SQL commands through a web application or web service for execution by a back-end database
- Enabled by application code that does not properly screen SQL statement data that are received from a client or web form input (for example, SQL keywords, statements, comments)
- Example: database query from a web form
 - Expected: `SELECT id FROM logins WHERE userName='Tom'`
 - Actual: `SELECT id FROM logins WHERE userName='Tom' OR 'y'='y'`
- Example: building a database query from a web form
 - Expected: `SELECT product FROM products WHERE pName LIKE '%Chairs'`
 - Actual: `SELECT product FROM products WHERE pName LIKE '%Chairs' UNION SELECT username FROM dba_users WHERE username like '%'`

© Copyright IBM Corporation 2013

Figure 10-20. SQL injection attack

WE4013.0

Notes:

The problem occurs because the application code does not properly filter the SQL strings that come in from the client, either as fields from a web page form, or as data in an XML message.

The underlined text indicates the text that is being entered in the web form or sent in the XML document. The "expected" is what the developer expected to receive, and the "actual" is what the hacker sent.

In first example, the developer expects just a single user name to be received. The hacker entered the underlined code, which includes an OR that always returns true. The effect is to return all IDs from the table.

The second example is supposed to return a list of all selected products. The hacker adds a union of another table, which returns all database users in the application. The resulting table contains the selected product rows, also all of the database users. Clearly, this result is not what the developer intended.

The typical exposure is when:

1. Input parameters are not properly screened, allowing one of two attacks:
 - a. Additional SQL keywords cause the boolean condition to always return a result, such as the entire list of user names and passwords.
 - b. Additional SQL statements are slipped in. If the interface allows read/write access, the attacker can add their own user name and password.
2. The two dashes (--) are appended to the end of the parameter to comment out the rest of the original SQL statement.

SQL injection is one of the most common application layer attacks. An attacker passes a string input to an application in hopes of manipulating the SQL statement to their advantage. The complexity of the attack involves by an SQL statement that might be unknown to the attacker. Open source applications and commercial applications that are delivered with source code are more susceptible, since an attacker can find potentially vulnerable statements before an attack.



SQL injection attack protection

- DataPower SQL injection attack protection uses a style-sheet-based approach to filter out potentially risky SQL strings
 - Add a **Filter** action to a firewall processing policy
 - Configure the processing control file to use `store:///SQL-Injection-Filter.xsl`
 - The style sheet sets a parameter named `{http://www.datapower.com/param/config}SQLPatternFile` to `store:///SQL-Injection-Patterns.xml`
 - You can customize it to point to a custom SQL injection pattern file



© Copyright IBM Corporation 2013

Figure 10-21. SQL injection attack protection

WE4013.0

Notes:

In a web service scenario, the SQL query that is entered is included inside a web service request that is used to do a database operation.

Under the **Advanced** tab in the "Configure Filter Action" page, you can specify the `SQL-Injection-Filter.xsl` style sheet, and the SQL injection pattern file. The SQL injection pattern file defaults to `SQL-Injection-Patterns.xml`, but you can change it to any other pattern file you want.



Unit summary

Having completed this unit, you should be able to:

- Explain possible attack scenarios that are involved in XML-based applications
- Describe the various types of XML attacks
- Use the WebSphere DataPower SOA Appliance to protect against XML attacks

© Copyright IBM Corporation 2013

Figure 10-22. Unit summary

WE4013.0

Notes:



Checkpoint questions

1. What are the four types of XML threats?
 - A. XML denial of service (XDoS),
 - B. Unauthorized access
 - C. Data integrity and confidentiality
 - D. System compromise
 - E. Phishing
2. True or False: XML virus protection is a periodic virus scan utility that the DataPower SOA appliance offers.
3. True or False: The **Validate** action is sufficient to protect against a message tamper.

© Copyright IBM Corporation 2013

Figure 10-23. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. A, B, C, and D. What are the four types of XML threats?
 - ✓ A. XML denial of service (XDoS),
 - ✓ B. Unauthorized access
 - ✓ C. Data integrity and confidentiality
 - ✓ D. System compromise
 - E. Phishing
2. False.
3. False. You also need the appropriate use of the **Sign** and **Verify** actions.

© Copyright IBM Corporation 2013

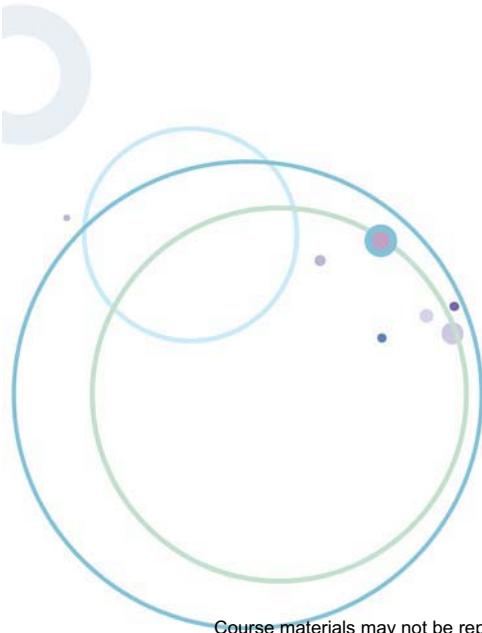
Figure 10-24. Checkpoint answers

WE4013.0

Notes:

Exercise 8

Protecting against XML threats



© Copyright IBM Corporation 2013

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

7.0.1

Figure 10-25. Exercise

WE4013.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

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

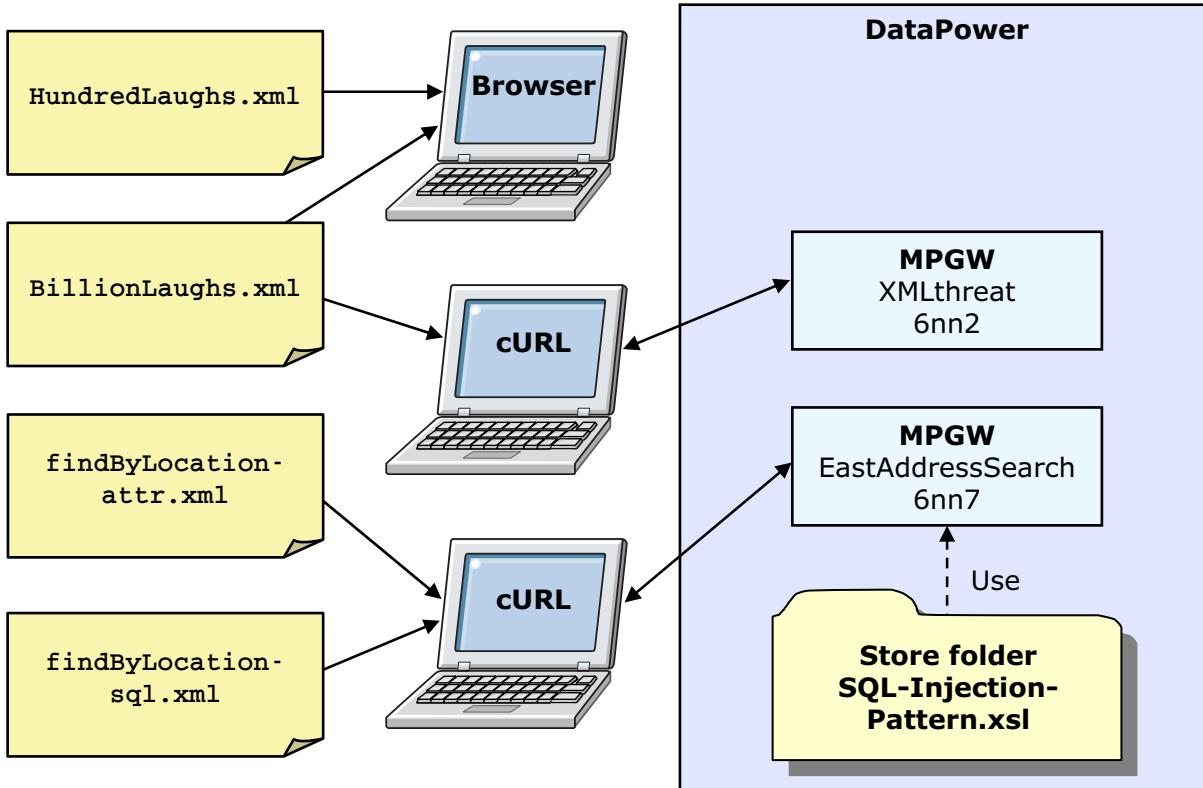
© Copyright IBM Corporation 2013

Figure 10-26. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 10-27. Exercise overview

WE4013.0

Notes:

Unit 11. Web service proxy service

What this unit is about

This unit describes the web service proxy service and its role in an XML-aware web-services-based network. You learn the configuration steps that are required to create and manage a web services proxy. You also learn advanced web service configuration steps, such as proxy-level security, SOAPAction policy, and web service endpoint.

What you should be able to do

After completing this unit, you should be able to:

- Describe the web service proxy architecture
- List and explain the configuration steps that are needed to create a web service proxy
- Create and configure a web service proxy policy at various levels of the Web Services Description Language (WSDL) file

How you will check your progress

- Checkpoint
- Exercise 9: Configuring a web service proxy

Unit objectives

After completing this unit, you should be able to:

- Describe the web service proxy architecture
- List and explain the configuration steps that are needed to create a web service proxy
- Create and configure a web service proxy policy at various levels of the Web Services Description Language (WSDL) file

© Copyright IBM Corporation 2013

Figure 11-1. Unit objectives

WE4013.0

Notes:

Web service proxy overview

- A web service proxy is a middleware component that exists between the client and the web service
 - Decouples web service client from the actual web service
 - Hides web service endpoint address from client
 - Flexibility to change back-end address without affecting client code
 - Can virtualize multiple web services into a single client-facing web service
 - Performs security, validation, and transformation on a request or response to offload these tasks from the back end web service
- The XG45/XI52/XI50 blade DataPower appliances allow you to create a web service proxy to accelerate and mediate communication between a client and a web service
 - Rules that are associated with different parts of a WSDL interface
 - Supports multiple WSDL documents
 - Fine-grained policy control
 - Built-in service-level monitoring (SLM) capabilities

© Copyright IBM Corporation 2013

Figure 11-2. Web service proxy overview

WE4013.0

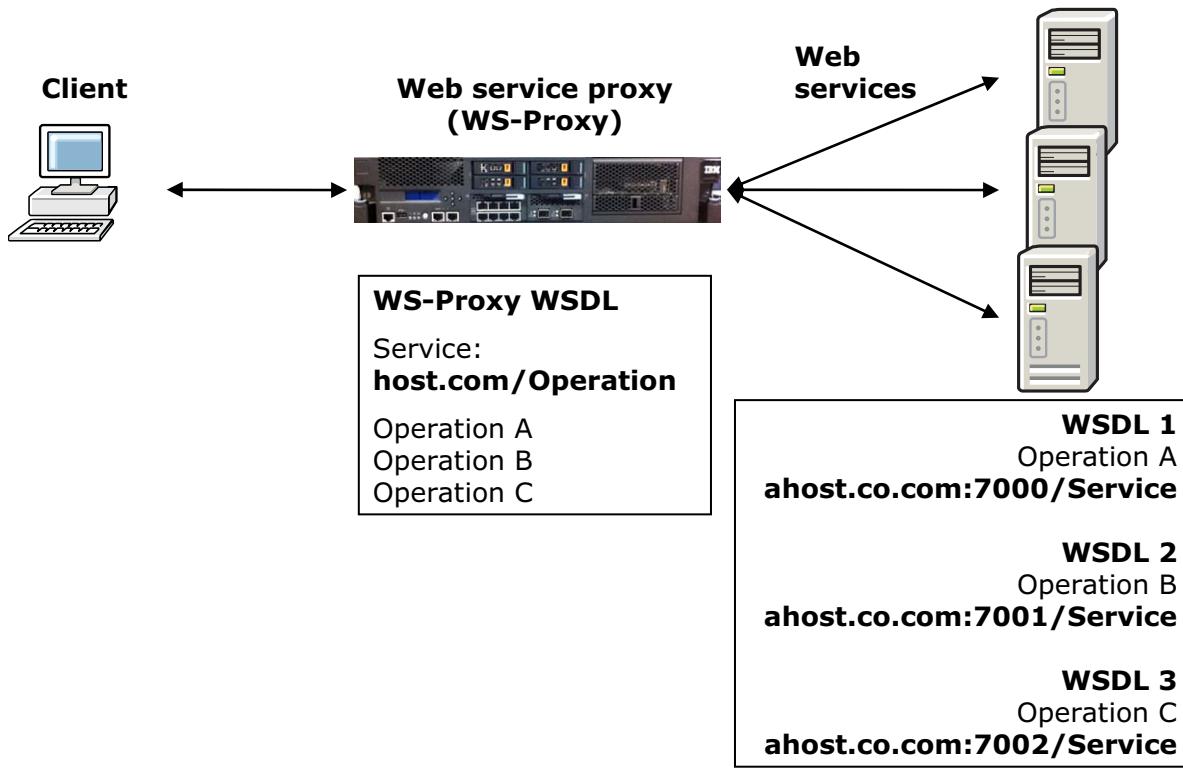
Notes:

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, since all requests pass through the appliance.



Web service virtualization



© Copyright IBM Corporation 2013

Figure 11-3. Web service virtualization

WE4013.0

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 simply by loading a WSDL document
 - Clients now connect directly to web service proxy and not the back-end service
- Creates processing policy with rules and actions at fine-grained level
 - Rules at a proxy, service, port, or operation level can process request and response messages
- Automatic schema validation of request, response, and fault messages (user policy)
 - User does not need to create a processing policy for this validation
- Service virtualization can occur in real time
 - Web service proxy can update the proxy WSDL automatically when underlying WSDL is updated
 - Integrates with WebSphere Service Registry and Repository and UDDI registries
- Can enforce policy and monitor performance of services
 - Multiple appliance support for virtual services

© Copyright IBM Corporation 2013

Figure 11-4. Web service proxy benefits

WE4013.0

Notes:

A **user policy** allows you to schema validate request, response, and fault messages. It is automatically created when you create a web service proxy.

The web service proxy is built on top of the XML firewall. Therefore, it provides all of the functionality of an XML firewall, such as encryption, validation, AAA, and more.

UDDI is a service repository that is used to search for WSDL files of a service.

Creating a WSDL cache policy enables the proxy WSDL file to be updated automatically when the underlying WSDL changes.

You can create an SLM peer group to share SLM data and enforce SLM policy between multiple DataPower appliances.

The screenshot shows the 'Web service proxy configuration tabs (1 of 2)' section. At the top left is a blue header bar with the 'WebSphere Education' logo and the IBM logo at the top right. Below the header is a title 'Web service proxy configuration tabs (1 of 2)'. A sidebar on the left contains a square icon and the text 'Configure Web Service Proxy'. The main area has a light gray background with a horizontal navigation bar containing several tabs: 'WSDL files' (highlighted in blue), 'SLM Policy', 'Services', 'Policy', 'SLA Policy Details', 'Proxy Settings', 'Advanced Proxy Settings', and 'Headers/Params'. The 'WSDL files' tab is currently active.

- WSDL files
 - Uploads or associates a WSDL document with a web service proxy
 - Configures proxy and remote URI (address, port) of services that are contained in WSDL document
- SLM Policy
 - Monitors and shapes traffic that enters the web service proxy
- Services
 - Listing of services that are defined in each WSDL document
 - Can publish services to UDDI registry
- Policy
 - Configures a web service proxy policy
- SLA Policy Details
 - View WSDL attachments that relate to service level agreement
- Proxy Settings
 - Specifies method of forwarding to service, security, XML manager, and HTTP settings

© Copyright IBM Corporation 2013

Figure 11-5. Web service proxy configuration tabs (1 of 2)

WE4013.0

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, 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 'Configure Web Service Proxy' page. At the top left is the 'WebSphere Education' logo. At the top right is the 'IBM' logo. Below the header is a title 'Web service proxy configuration tabs (2 of 2)'. Underneath the title is a section titled 'Configure Web Service Proxy' with a blue square icon. Below this are several tabs: 'Proxy Settings' (selected), 'Advanced Proxy Settings', 'Headers/Params', 'WS-Addressing', 'WS-ReliableMessaging', 'Monitors', 'XML Threat Protection', and a small 'Help' icon.

- Advanced Proxy Settings
 - Configures advance connection settings
- Headers/Params
 - Add or remove HTTP headers and pass style sheet parameters
- WS-Addressing
 - Specifies the WS-Addressing mode for this service
- WS-ReliableMessaging
 - Toggle to enable WS-ReliableMessaging (deprecated)
- Monitors
 - Identifies any message monitors associated to this service
- XML Threat Protection
 - Provides protection against XML threats

© Copyright IBM Corporation 2013

Figure 11-6. Web service proxy configuration tabs (2 of 2)

WE4013.0

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 DataPower WebGUI Control Panel
or
 - Using vertical navigation bar, select **Services > Web Service Proxy > New Web Service Proxy**
3. Upload the WSDL document and add it to the web service proxy
4. Configure endpoint of services that are defined in WSDL document
 - Define both proxy URI and endpoint URI for each service in WSDL document
5. Specify a service policy that consists of rules for the web service (optional)

© Copyright IBM Corporation 2013

Figure 11-7. Web service proxy basic configuration steps

WE4013.0

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:

- 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.
- Select the SOAP action policy to specify how to consume messages with a SOAPAction header.
- 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 2013

Figure 11-8. Step 1: Obtain WSDL document

WE4013.0

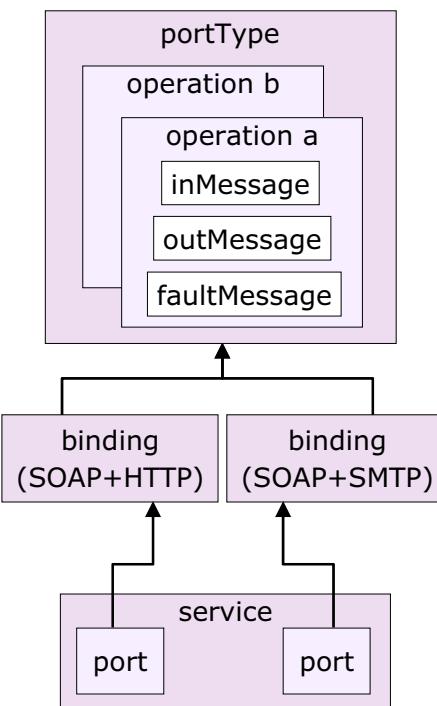
Notes:

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 2013

Figure 11-9. WSDL structure

WE4013.0

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



Configure Web Service Proxy

| Web Service Proxy Name | Op-State | Logs | Type | Req-Type | Back Side URL | Resp-Type |
|------------------------|----------|------|------------------|----------|---------------|-----------|
| AddressSearchProxy | up | | static-from-wsdl | soap | NA | soap |

Add

- Using the vertical navigation bar, select **Services > Web Service Proxy > New Web Service Proxy**
- You are first prompted for the name of the web service proxy

Web Service Proxy Name

Create Web Service Proxy

© Copyright IBM Corporation 2013

Figure 11-10. Step 2: Creating a web service proxy

WE4013.0

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, select **Objects > Service Configuration > Web Service Proxy**
 - All configuration options are available

A screenshot of the "Configure Web Service Proxy" interface. The left sidebar shows a tree view with categories like Status, Services, Network, Administration, and Objects. Under Objects, there are Network Settings, Protocol Handlers, and Service Configuration, which is expanded to show options like HTTP Service, Multi-Protocol Gateway, SSL Proxy Service, TCP Proxy Service, UDDI Subscription, and Web Application Firewall. The "Web Service Proxy" option is highlighted with a red box. The main panel title is "Configure Web Service Proxy" with a gear icon. Below it, the sub-tab "Main" is selected. The central area displays "Web Service Proxy: AddressSearchProxy [up]". It includes buttons for Apply, Cancel, Delete, and Undo. To the right are links for Export, View Log, View Status, and Show P. Below these are fields for "Administrative State" (with radio buttons for enabled and disabled, where enabled is selected), "Comments" (a text input field), and "Service Priority" (a dropdown menu). At the bottom right is a copyright notice: "© Copyright IBM Corporation 2013".

Figure 11-11. An alternative: Web service proxy object editor

WE4013.0

Notes:

The WSDL cache policy and some attachment processing specifications are example configurations that are only possible by using this editor.



Step 3: Add WSDL document to web service proxy

Configure Web Service Proxy

WSDL files SLM Policy Services Policy SLA Policy Details Pro

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 2013

Figure 11-12. Step 3: Add WSDL document to web service proxy

WE4013.0

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)

| Web Service Proxy WSDLs | | | |
|---|---|--|---|
| AddressSearchService - AddressSearch | | | |
| <input checked="" type="button"/> Local | | | |
| Local Endpoint Handler | | URI | |
| <input type="button"/> AddressSearchFSH <input type="button"/> + <input type="button"/> ... | | <input type="text" value="/EastAddressSearch"/> | |
| <input checked="" type="button"/> Remote | | | |
| Protocol <input type="button"/> HTTP | Remote Endpoint Host <input type="text" value="transit01.com"/> | Port <input type="text" value="9080"/> | Remote URI <input type="text" value="/EastAddress/services/AddressSearch"/> |
| <input type="checkbox"/> Published | <input checked="" type="checkbox"/> Use Local | | |

© Copyright IBM Corporation 2013

Figure 11-13. Step 4: Configure WSDL endpoint

WE4013.0

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. See the DataPower Information Center for further details.



Step 5: Configure local endpoint handler

- A Local Endpoint Handler (a front side handler) is used to determine the IP address, port, and protocol
- Click the plus sign (+) to create a new local endpoint handler object
 - Specify the appliance local IP address and port number to listen for requests
 - Can also restrict access based on HTTP attributes

Create a New:

- FTP Poller Front Side Handler
- NFS Poller Front Side Handler
- SFTP Poller Front Side Handler
- ...
- HTTP Front Side Handler**
- ...
- IMS Connect Handler
- TIBCO EMS Front Side Handler
- WebSphere JMS Front Side Handler
- MQFTE Front Side Handler
- MQ Front Side Handler
- SFTP Server Front Side Handler
- Stateless Raw XML Handler
- Stateful Raw XML Handler

AddressSearchService - Address Search

Local

Local Endpoint Handler

http_fsh_east_address_search

http_fsh_east_address_search

HTTP Front Side Handler

Name

Administrative State enabled disabled

Comments

Local IP Address

Port Number

HTTP Version to Client

© Copyright IBM Corporation 2013

Figure 11-14. Step 5: Configure local endpoint handler

WE4013.0

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 fashion.



Step 6: Add the WSDL to the service

| AddressSearchService - AddressSearch | | | |
|--------------------------------------|--------------------|--|-------------|
| Local | | | |
| Local Endpoint Handler | URI | Binding (Suffix) | Edit/Remove |
| AddressSearchFSH | /EastAddressSearch | <input checked="" type="checkbox"/> SOAP 1.1 <input type="checkbox"/> SOAP 1.2 HTTP GET | Add |

→

| AddressSearchService - AddressSearch | | | |
|--------------------------------------|--------------------|--|-------------|
| Local | | | |
| Local Endpoint Handler | URI | Binding (Suffix) | Edit/Remove |
| AddressSearchFSH | /EastAddressSearch | SOAP 1.1() | Edit Remove |
| (none) | | <input checked="" type="checkbox"/> SOAP 1.1 <input type="checkbox"/> SOAP 1.2 HTTP GET | Add |

Remote

| Protocol | Remote Endpoint Host | Port | Remote URI |
|----------|----------------------|------|--------------------------------|
| HTTP | myserver.ibm.com | 9080 | /EastAddress/services/AddressS |

Published Use Local

Click **Add** to add the customized WSDL information to the service; then click **Next**

Next **Cancel**

© Copyright IBM Corporation 2013

Figure 11-15. Step 6: Add the WSDL to the service

WE4013.0

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
- Additional options, such as adding another WSDL, are now available

The screenshot shows the 'Configure Web Service Proxy' interface. At the top, there's a navigation bar with tabs: WSDL files, SLM Policy, Services, Policy, SLA Policy Details, Proxy Settings, Advanced Proxy Settings, Headers/Params, and Help. The 'WSDL files' tab is selected. Below the navigation bar, the 'Web Service Proxy Name' field contains 'AddressSearchProxy'. Underneath it are buttons for Apply, Cancel, Delete, and Refresh. To the right, there are links for Export, View Log, View Status, View Operations, Show Probe, Validate Conformance, and Help. A large arrow points from the text 'Edit WSDL or Subscription' in the list above to the 'Edit WSDL or Subscription' link in the toolbar below. The 'WSDLs' section shows a table with one row:

| WSDL Source Location | Endpoint Handler Summary | WSDL Status | WS-I BP Status | Action |
|---------------------------------|--------------------------|-------------|----------------|--------|
| local:///EastAddressSearch.wsdl | 1 up / 1 configured | Okay | Okay | Remove |

At the bottom right, there's a copyright notice: © Copyright IBM Corporation 2013.

Figure 11-16. Initial WSDL completed

WE4013.0

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 Registry and Repository (WSRR)
- Subscribe to a WSRR Saved Search
 - Can automatically “push” changes to the web service proxy

The screenshot shows the 'Configure Web Service Proxy' interface. At the top, there is a navigation bar with tabs: WSDL files, SLM Policy, Services, Policy, SLA Policy Details, Proxy Settings, Advanced Proxy Settings, Headers/Params, and a help icon. Below the navigation bar, there is a search bar labeled 'Web Service Proxy Name [up]' containing the text 'AddressSearchProxy'. Underneath the search bar are buttons for Apply, Cancel, Delete, and Refresh. To the right of these buttons are links for Export, View Log, View Status, View Operations, Show Probe, Validate Conformance, and Help. The main area is titled 'WSDLs' and contains a table. The table has columns: WSDL Source Location, Endpoint Handler Summary, WSDL Status, WS-I BP Status, and Action. There is one row in the table with the following data: 'local:///EastAddressSearch.wsdl', '1 up / 1 configured', 'Okay', 'Okay', and a 'Remove' link. At the bottom of the page, there is a copyright notice: '© Copyright IBM Corporation 2013'.

Figure 11-17. Other ways to get a WSDL

WE4013.0

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 the **Publish to UDDI** button 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 the active tab), Policy, SLA Policy Details, Proxy Settings, and Advanced. Below the navigation bar, the main area has a title "Web Service Proxy Name [up]" with a field containing "AddressSearchProxy" and a mandatory indicator (*). Below this are buttons for Apply, Cancel, Delete, and Refresh. A link labeled "Export" is also present. The next section is titled "Services" and contains a table with one row. The table has columns for "Service" and "AddressSearchService". In the "Service" column, there is a "Publish to UDDI" button. The table also has a "WSDL Name: EastAddressSearch.wsdl" header row.

© Copyright IBM Corporation 2013

Figure 11-18. View WSDL services

WE4013.0

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** button opens another window that lists the operations that are defined in the WSDLs exposed by this service.

Retrieve the client WSDL from the service

- You can retrieve the client-facing WSDL from the service
 - Edit the endpoint handler to allow HTTP GET
 - Enter:
`http://myDPappliance.com:6999/EastAddressSearch?wsdl`
 - 6999: The port number of the web service proxy
 - /EastAddressSearch: the local or client URI to invoke web service
- Returned WSDL contains the IP address or service port for the appliance as the WSDL <address location= >
 - Original


```
<wsdl:port binding="..." name="AddressSearch">
<address location="http://training.ibm.com:9080/
  EastAddress/services/AddressSearch" /> </wsdl:port>
```
 - Retrieved by ?wsdl


```
<wsdl:port binding="..." name="AddressSearch">
<address
  location="http://192.168.10.41:6999/EastAddressSearch" />
</wsdl:port>
```

| | |
|------------------------------|---|
| Allowed Methods and Versions | <input checked="" type="checkbox"/> HTTP/1.0 <input checked="" type="checkbox"/> HTTP/1.1 <input checked="" type="checkbox"/> POST <input checked="" type="checkbox"/> GET <input checked="" type="checkbox"/> PUT <input type="checkbox"/> HEAD <input type="checkbox"/> OPTIONS |
|------------------------------|---|

© Copyright IBM Corporation 2013

Figure 11-19. Retrieve the client WSDL from the service

WE4013.0

Notes:

The original WSDL used in defining the WS Proxy contains a location that is no longer correct for the web service that is proxied by this service.

When you append ?wsdl to the URL that the client uses to access the web service on the appliance, the appliance returns a WSDL with:

- The appliance IP address
- The service port
- The URI that the client uses to access the web service

These values are not the values in the original WSDL.

Modifying the location in the client WSDL

- The WSDL retrieved from the web service proxy using `?wsdl` by default places the IP address and port for the appliance in the “location”

```
<wsdl:port binding="..." name="AddressSearch">
  <address
    location="http://192.168.10.41:6999/EastAddressSearch" />
</wsdl:port>
```

- You can specify a different host name or port to place in the WSDL
 - Clear **Use Local** to enter your own values
 - Now retrieved by `?wsdl`

```
<wsdl:port binding="..." name="AddressSearch">
  <address
    location="http://myDPappliance.com:6999/EastAddressSearch"
  " /> </wsdl:port>
```

© Copyright IBM Corporation 2013

Figure 11-20. Modifying the location in the client WSDL

WE4013.0

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.



Step 7: Configuring web service proxy policy (optional)

WSDL files SLM Policy Services Policy SLA Policy Details Proxy Settings Advanced Proxy Settings Headers/Para

Web Service Proxy Name [up] AddressSearchProxy *

Apply Cancel Delete Refresh Export | View Log | View Status | View Show Probe | Validate Conform

Policy

Use this pane to define the processing policies to implement at various levels in the WSDL hierarchy.

WSDL Policy Tree Representation

Show portType and binding nodes

Define the policies to apply in the tree.

proxy: AddressSearchProxy

- WS-Policy (default) WS-I Conformance (none) Priority Normal
 - Processing Rules** Request rules:1, Response rules:1
- + wsa1: westAddressSearch.wsdl ✓
 - WS-Policy (default) WS-I Conformance (none) Priority Normal
 - Processing Rules**
- + wsa1: eastAddressSearch.wsdl ✓
 - WS-Policy (default) WS-I Conformance (none) Priority Normal
 - Processing Rules**

- Click the **Policy** tab to view web service proxy policy
 - The web service proxy policy consists of rules
 - Rules can be executed at a specific level per request or response
- Select **Processing Rules** to open policy editor for rules at that level

© Copyright IBM Corporation 2013

Figure 11-21. Step 7: Configuring web service proxy policy (optional)

WE4013.0

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.

proxy: addressSearchProxy

- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules (Request rules:1 ,Response rules:1)

wsdl: EastAddressSearch.wsdl

- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules

Policy Configuration

Define the processing rules and the actions to perform against requests and responses and the processing for error conditions.

Rule: Rule Name: addressSearchProxy_default_request-rule Rule Direction: Client to Server

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

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

CLIENT → [] → [] → []

Configured Rules

| Order | Rule Name | Direction | Actions |
|-------|--|------------------|----------------------|
| 1 | addressSearchProxy_default_request-rule | Client to Server | Filter, SLM, Results |
| 2 | addressSearchProxy_default_response-rule | Server to Client | Filter, Results |

• View the rule configuration at any level by clicking the **Processing Rules** button
 • Expand/collapse the levels by clicking the +/- icon
 • That opens a policy editor section for rules at the selected level

• Rules are configured in the policy editor as done in the MPGW or XML firewall service
 • Shown are the default rules that are provided at the proxy level of the WSDL

- Request rule that enables an SLM policy
- Reply rule that just passes the message through

© Copyright IBM Corporation 2013

Figure 11-22. Configure web service proxy policy rule

WE4013.0

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 **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

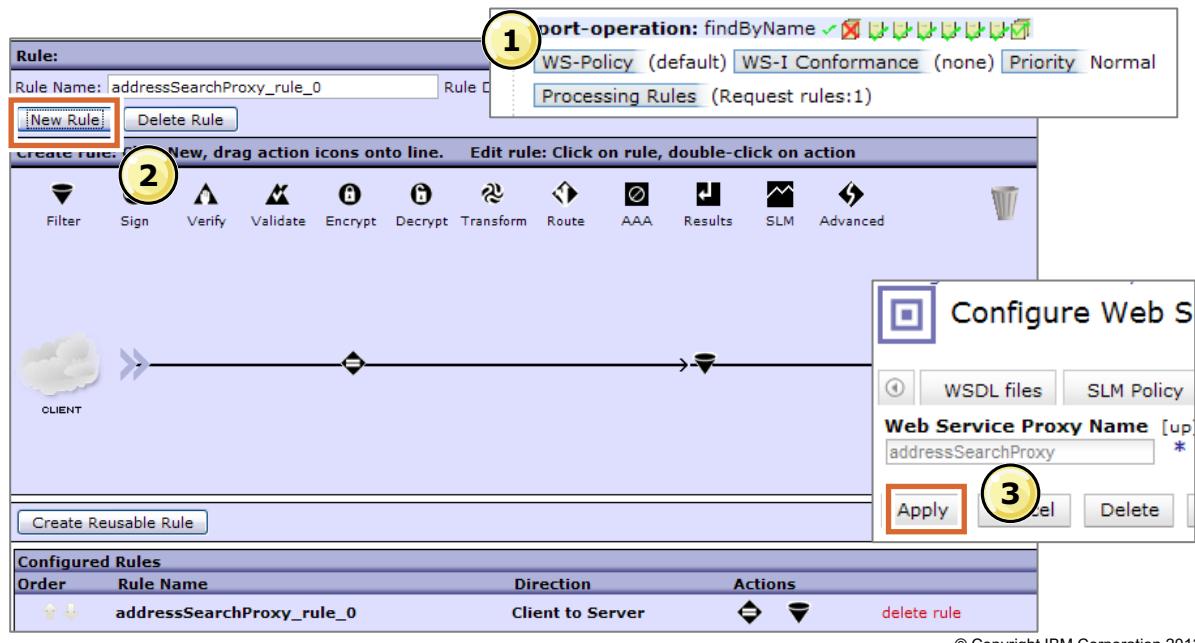


Figure 11-23. Adding a rule

WE4013.0

Notes:

The number and type of rules that are defined at that level in the WSDL are displayed next to the **Processing Rules** button.

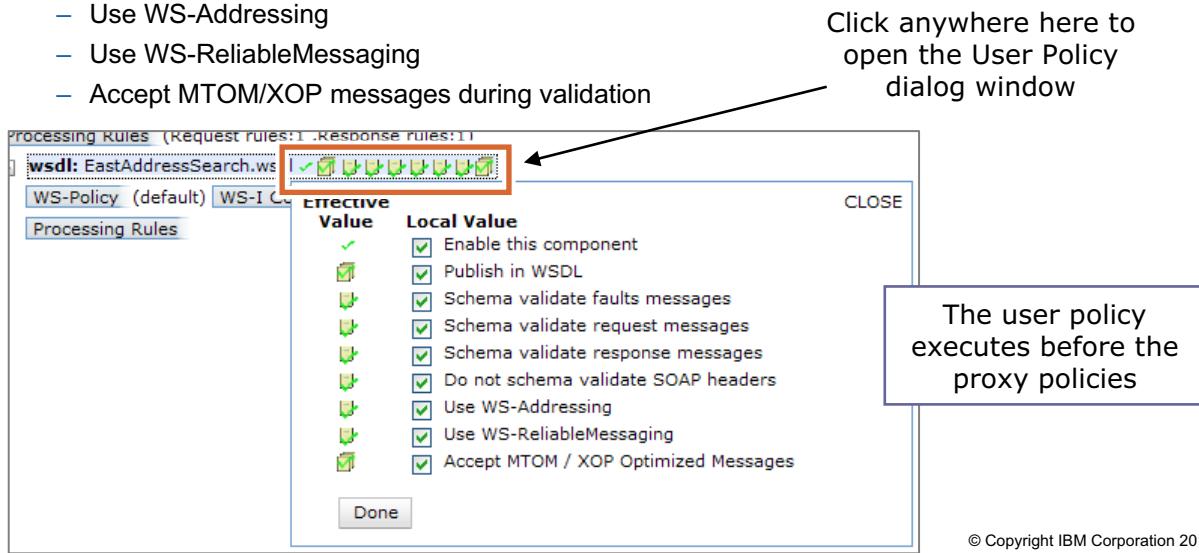
You create and configure the actions in the rule as you normally would.

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 2013

Figure 11-24. Default validation (user policies)

WE4013.0

Notes:

Click any of the icons at each level to view the user policy pop-up dialog.

The first check mark enables the component. Each option that is shown in the pop-up dialog maps to an icon with a green check mark or red X.

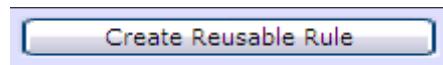
Each policy level contains a user policy that can be enabled or disabled.

The web service proxy policy and user policy are separate from each other; the user policy is executed before the web service proxy policy.



Create reusable rule

- The rule configuration area allows you to select a set of actions to be invoked as a reusable rule
 - Click the **Create Reusable Rule** button.
 - Draw a box around the actions to include in the reusable rule.
 - Click **Apply**. A gray rectangle appears around the reusable rule in the rule configuration area and generates a new rule name for the new reusable rule.
 - Use the **Advanced – Call Processing Rule** action to reuse the rule.



© Copyright IBM Corporation 2013

Figure 11-25. Create reusable rule

WE4013.0

Notes:

Reusable rules are useful for applying a common set of actions at many levels of the web service proxy. Additional actions can be added before or after the reusable rules. It allows you to more easily manage a set of actions that repeat across many levels of the web service proxy.

Reusable rules can be defined in the other service type processing policies, such as an XML firewall policy.

Advanced web service proxy configuration

Additional options and tabs available for advanced web services proxy configuration:

- Proxy settings
 - Web service proxy type
 - Security settings (AAA, cryptographic key)
 - SOAP Action Policy
 - XML Manager
- Advanced proxy settings
 - HTTP connection settings
- Headers, parameters
 - Adds or removes HTTP headers and passes style sheet parameters
- WS-Addressing
 - Indicates support for WS-Addressing for front end or back end
- WS-ReliableMessaging
 - Indicates whether to use WS-ReliableMessaging
- XML threat protection
 - Provides protection against XML threats

© Copyright IBM Corporation 2013

Figure 11-26. Advanced web service proxy configuration

WE4013.0

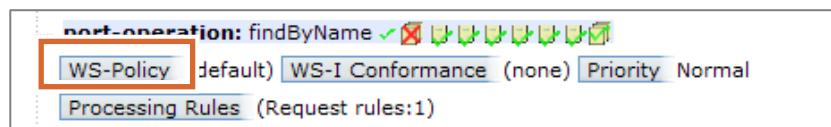
Notes:

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/Params**, **WS-Addressing**, and **WS-ReliableMessaging** tabs.

The XML threat protection settings are explained in the XML threat protection presentation.

WS-Policy

- WS-Policy is a specification that defines metadata to enable interoperability between web service consumers and web service providers
- The WS-Policy specifications enable organizations to automate their service governance models by creating a concrete instance of web service governance
- Behaviors:
 - Parse WSDL with policy elements already included in the WSDL and recognize standardized policy “domains” (WS-Security Policy, WS-ReliableMessaging Policy)
 - DataPower supports retrieving WSDL by using WebSphere Service Registry and Repository queries
 - DataPower supports retrieving WSDL by using a UDDI interface



© Copyright IBM Corporation 2013

Figure 11-27. WS-Policy

WE4013.0

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 11-28. Conformance policy

WE4013.0

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 style sheet; the firmware does not automatically provide it.

Conformance policy object

- Profiles:** profiles against which conformance is checked
- Ignored Requirements:** conformance requirements to ignore
- Corrective Stylesheets:** XSL sheets are invoked after conformance analysis
- Record Report:** the degree of nonconformance that causes a report to be recorded
- Reject non-conforming messages:** the degree of nonconformance that causes a rejected message
- Use analysis as a result:** deliver conformance analysis report as an action result

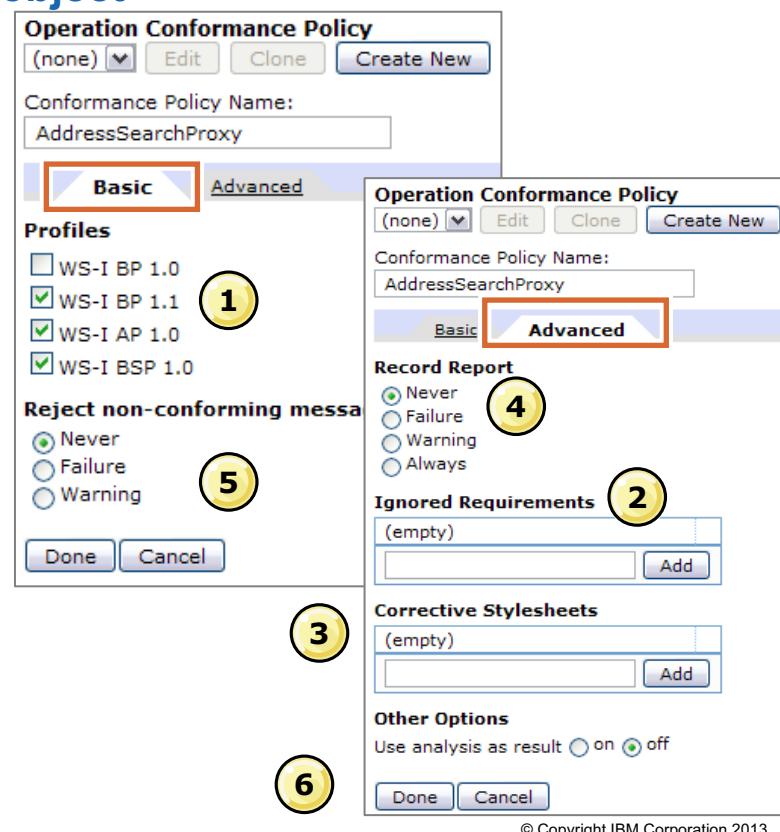


Figure 11-29. Conformance policy object

WE4013.0

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

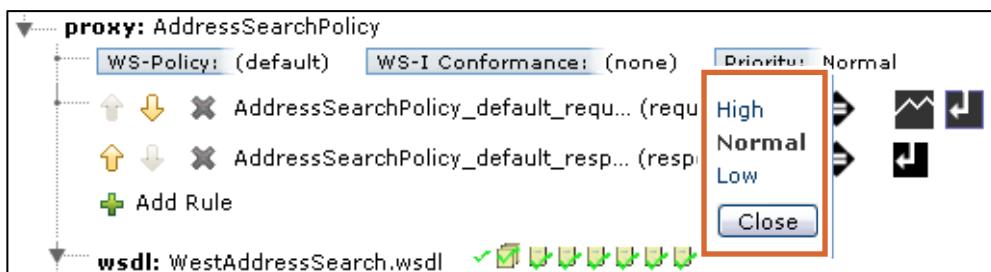
Reject nonconforming messages include:

- 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
 - Low: receives lower than normal priority
 - Normal: (default) receives normal priority

© Copyright IBM Corporation 2013

Figure 11-30. Service priority

WE4013.0

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**
 - Static Backend**: web service proxy forwards to single back end server
 - Dynamic Backend**: web service proxy determines back-end server during service processing
 - Static from WSDL (default)**: Service section in the WSDL file determines the back end server

The screenshot shows the 'Proxy Settings' tab in the WebSphere Administration console. The 'Web Service Proxy Name' is set to 'AddressSearchProxy'. In the 'General Configuration' section, the 'Type' dropdown is highlighted with a red box and contains three options: 'Dynamic Backend', 'Static Backend', and 'Static from WSDL', with 'Static from WSDL' selected. Other sections visible include 'Comments', 'XML Manager' (set to 'default'), and 'Authorization AAA Policy' (set to '(none)').

Figure 11-31. Proxy settings (1 of 4)

WE4013.0

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 2013

Figure 11-32. Proxy settings (2 of 4)

WE4013.0

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, since a typical schema validation does not check for these elements.

An example SOAP message with encrypted payload might look like:

```
<SOAP:Body>
<EncryptedData ...>
```

Using a cryptographic key ensures that a message can pass automatic validation by decrypting the message payload before validation. The entire message must be encrypted, not only fields within the message.

EncryptedKeySHA1 Cache Lifetime is the cache lifetime for the decrypted generated key. Setting the value to 0 means that the decrypted generated key is not cached.

Preserve EncryptedKey Chain, if it is on, outputs the chain of elements that the decrypted Encrypted Data uses, such as xenc:EncryptedKey, wsc:DerivedKeyToken. Otherwise, all

xenc:EncryptedKey elements are removed after decryption, and even some of the encrypted data might not be decrypted successfully.

Decrypt with Key from EncryptedData: In scenarios in which the key is inside an EncryptedData element (such as “encrypted SAML Assertion”), the decrypt action cannot locate the key to decrypt the corresponding EncryptedData elements. Select **on** to enable the decrypt action to attempt decryption with the key that is inside the EncryptedData element.

The **Client Principal** field contains the full name of the client principal when the web service proxy must automatically decrypt encrypted requests. Use this property when the encryption uses a Kerberos session key, or a key that was derived from the session key.

In a similar fashion, the **Server Principal** field specifies the full name of the server principal when the web service proxy must automatically decrypt encrypted responses.



Proxy settings (3 of 4)

- **Kerberos Keytab**
 - Selects the Kerberos keytab file that contains the principals
- **SOAP Action Policy:** validates messages that contain a SOAPAction HTTP header
 - **Lax:** validates messages with empty SOAPAction HTTP header or empty string within SOAPAction HTTP header
 - **Off:** SOAPAction HTTP header is ignored
 - **Strict:** message must contain exact match of SOAPAction header that is specified in WSDL file
- **Monitor via Web Services Management Agent:** allows for autonomous monitoring of this service by a WS-Management agent

The screenshot shows a configuration interface for a web service proxy. It includes fields for selecting a Kerberos Keytab (with '(none)' selected), setting the SOAP Action Policy (with 'Lax' selected), and enabling monitoring via a Web Services Management Agent (with 'on' selected).

| Kerberos Keytab | | |
|--|----------------------------------|------------------------------------|
| (none) | <input type="button" value="+"/> | <input type="button" value="..."/> |
| SOAP Action Policy | | |
| <input checked="" type="radio"/> Lax | <input type="radio"/> Off | <input type="radio"/> Strict |
| Monitor via Web Services Management Agent | | |
| <input checked="" type="radio"/> on | <input type="radio"/> off | |

© Copyright IBM Corporation 2013

Figure 11-33. Proxy settings (3 of 4)

WE4013.0

Notes:

Select the Kerberos Keytab object that contains the principals for the **Kerberos Keytab** list. The web service proxy uses these principals to automatically decrypt encrypted requests and responses.

The WSDL file for a service defines the value that a SOAPAction header must contain for a SOAP request. The SOAPAction header is defined in the HTTP header, not the SOAP header.

The **SOAP Action Policy** setting specifies how to validate messages with a SOAPAction HTTP header.

A WS-Management agent can monitor the web service proxy without any monitors that are defined on the Monitors tab.



Proxy settings (4 of 4)

- **XML Manager:** assigns an XML manager to the web service proxy
- **Authorization AAA Policy:** selects or creates a AAA policy to apply to all service endpoints configured for this web service proxy
 - AAA policy can also be applied at a fine-grained level in the **Policy** tab

The screenshot shows the 'Proxy Settings' tab in the WebSphere Studio Application Developer interface. The 'Web Service Proxy Name' is set to 'AddressSearchProxy'. Under 'General Configuration', there are sections for 'Comments' (empty) and 'Type' (radio buttons for 'Dynamic Backend', 'Static Backend', and 'Static from WSDL', with 'Static from WSDL' selected). To the right, two sections are highlighted with a red box: 'XML Manager' (set to 'default') and 'Authorization AAA Policy' (set to '(none)').

© Copyright IBM Corporation 2013

Figure 11-34. Proxy settings (4 of 4)

WE4013.0

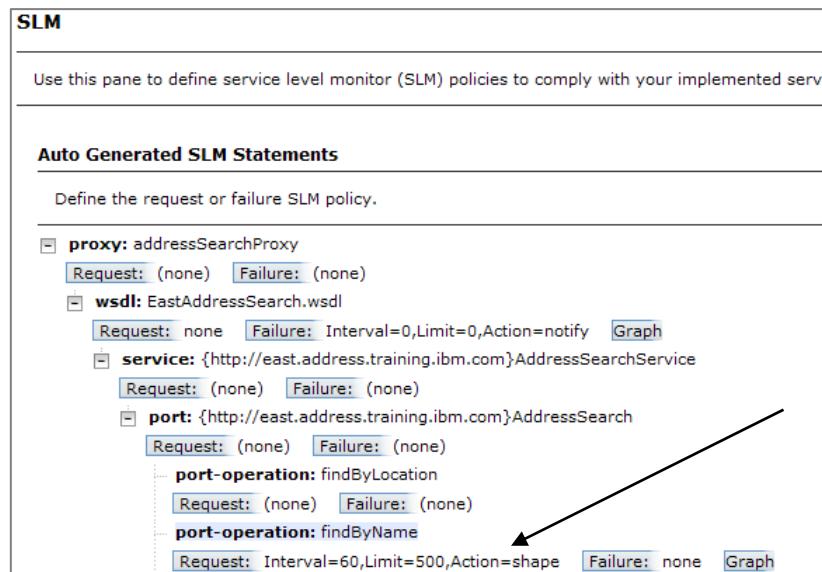
Notes:

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 entering the web service proxy using the **Throttle** and **Shape** action
 - Can view graph to see results of the traffic



© Copyright IBM Corporation 2013

Figure 11-35. Web service proxy: SLM Policy tab

WE4013.0

Notes:

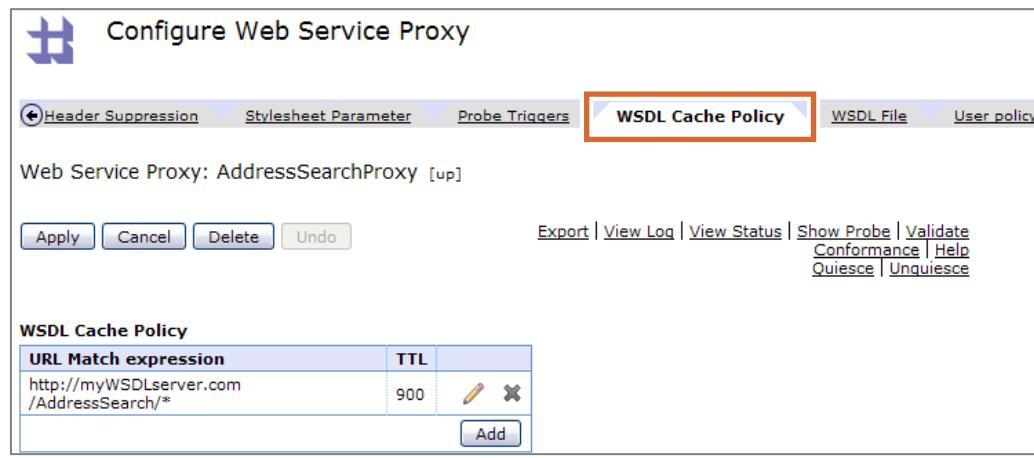
Under **Request**, you can count the number of transactions that occur with a specific **interval** (in seconds). If the transaction **limit** is exceeded, you can specify an **action** to:

- Notify:** Generate a log message when the transaction limit is exceeded.
- Throttle:** Additional transactions above the limit are rejected, and log messages are generated.
- Shape:** The first 2500 transactions in excess of the maximum transaction rate are queued for later transmission, and subsequent transactions in excess of the 2500 limit are rejected. Log messages are generated.

Under **Failure**, you can specify the same information as **Request**, except that these settings apply to error messages.

WSDL cache policy

- Create a WSDL cache policy to update the WSDL proxy with changes from underlying WSDL file
 - Scheduled poll of underlying WSDL
 - If changes are detected, then the proxy WSDL is automatically updated
- Option is available only from **Objects** view:
 - Select **Objects > Service Configuration > Web Service Proxy**; then click an existing web service proxy
 - Use the arrows at the top to select the **WSDL Cache Policy** tab



© Copyright IBM Corporation 2013

Figure 11-36. WSDL cache policy

WE4013.0

Notes:

Click **Add** to create a WSDL cache policy.

The URL match expression is used to match the URL of the WSDL file (that is, its location).

Time to Live (TTL) is expressed in seconds. It specifies how long the current WSDL file exists until it is automatically refreshed when a corresponding URL match expression is matched.

The WSDL file might exist on an external server.



Troubleshooting a web service proxy

Check active web service operations by using **Status > Web Service > Web Services Operations**

| WSProxy | Index | Interface | Port | URL | SOAP Action | SOAP Body | Status | |
|--------------------|-------|--------------|------|--------------------|-------------|--|------------|---------|
| AddressSearchProxy | 0 | 172.16.78.44 | 6975 | /EastAddressSearch | | {http://east.address.training.ibm.com}retrieveAll | Registered | http:// |
| AddressSearchProxy | 1 | 172.16.78.44 | 6975 | /EastAddressSearch | | {http://east.address.training.ibm.com}findByLocation | Registered | http:// |
| AddressSearchProxy | 2 | 172.16.78.44 | 6975 | /EastAddressSearch | | {http://east.address.training.ibm.com}findByName | Registered | http:// |
| AddressSearchProxy | 3 | 172.16.78.44 | 6975 | /WestAddressSearch | | {http://west.address.training.ibm.com}retrieveAll | Registered | http:// |
| AddressSearchProxy | 4 | 172.16.78.44 | 6975 | /WestAddressSearch | | {http://west.address.training.ibm.com}findByLocation | Registered | http:// |

List of validation checks for web service proxy

- Request
 - Web service proxy active and listening on port
 - Verify that client submitted correct URI
 - Web service proxy received request (system log, probe)
 - SOAPAction header must agree with operation name in SOAP body
 - Passed automatic schema validation (user policy)
 - Back end service active and available (system log)
 - Request that is transmitted to correct back end URL (system log)
- Response
 - Response that is received from back end service (system log)
 - Response passed automatic schema validation (user policy)
 - Response that transmitted completely to client (system log, probe)

© Copyright IBM Corporation 2013

Figure 11-37. Troubleshooting a web service proxy

WE4013.0

Notes:

The default error messages that the web service proxy returns are intentionally vague so that no clues are provided to an intruder who is trying to compromise the system. For example:

```

HTTP/1.0 500 Error
X-Backside-Transport: FAIL
Connection: close
Content-Type: text/xml
<?xml version='1.0' ?>
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/envelope/'>
<env:Body>
<env:Fault>
<faultcode>General</faultcode>
<faultstring>Internal Error</faultstring>
</env:Fault>
</env:Body>
</env:Envelope>
```



Unit summary

Having completed this unit, you should be able to:

- Describe the web service proxy architecture
- List and explain the configuration steps that are needed to create a web service proxy
- Create and configure a web service proxy policy at various levels of the Web Services Description Language (WSDL) file

© Copyright IBM Corporation 2013

Figure 11-38. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. True or False: A web service proxy and SLM policy can be defined at a fine-grained level.
2. Which of the following levels can be configured with a web service proxy policy?
 - A. Proxy
 - B. Message
 - C. Service
 - D. Port
3. True or False: A WSDL must be uploaded onto the appliance when creating a web service proxy.
4. List the three options under the SOAPAction policy:
 - A. lax: This option validates messages with an empty SOAPAction HTTP header or an empty string within the SOAPAction HTTP header
 - B. strict: The message must contain an exact match of the SOAPAction header that is provided in the WSDL file
 - C. off: The SOAPAction HTTP header is ignored
 - D. lazy: The SOAPAction allows all messages through

© Copyright IBM Corporation 2013

Figure 11-39. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

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

Checkpoint answers

1. **True.** A web service proxy and SLM policy can be defined at a fine-grained level.
2. **A, C, and D.** Which of the following levels can be configured with a web service proxy policy?
 - ✓ A. Proxy
 - B. Message
 - ✓ C. Service
 - ✓ D. Port
3. **False.** A WSDL can be retrieved from a subscription.
4. **A, B, and C.** 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 message through

© Copyright IBM Corporation 2013

Figure 11-40. Checkpoint answers

WE4013.0

Notes:

Exercise 9



Configuring a web service proxy

© Copyright IBM Corporation 2013

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

7.0.1

Figure 11-41. Exercise 9

WE4013.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Configure a web service proxy to virtualize an existing set of web services
- Create a policy within the web service proxy

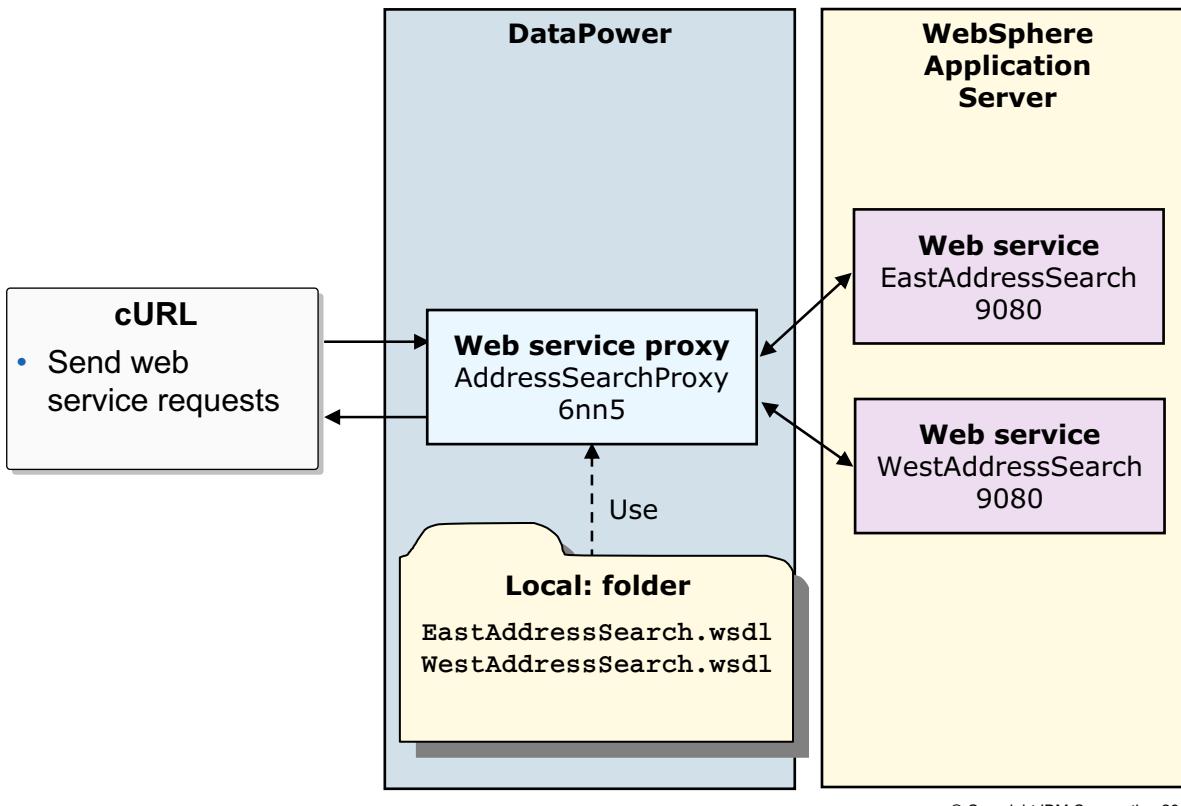
© Copyright IBM Corporation 2013

Figure 11-42. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 11-43. Exercise overview

WE4013.0

Notes:

Unit 12. XML and web services security overview

What this unit is about

This unit describes the features of the web services security specification. This specification provides message level security to ensure message confidentiality and integrity by using XML encryption and XML signatures. You learn how to use the DataPower device to encrypt and decrypt, and to sign and verify messages.

What you should be able to do

After completing this unit, you should be able to:

- Describe the features of the WS-Security specification
- Enable message confidentiality by using XML encryption
- Provide message integrity by using XML signatures

How you will check your progress

- Checkpoint
- Exercise 10: Web service encryption and digital signatures

Unit objectives

After completing this unit, you should be able to:

- Describe the features of the WS-Security specification
- Enable message confidentiality by using XML encryption
- Provide message integrity by using XML signatures

© Copyright IBM Corporation 2013

Figure 12-1. Unit objectives

WE4013.0

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 2013

Figure 12-2. Review of basic security terminology

WE4013.0

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.

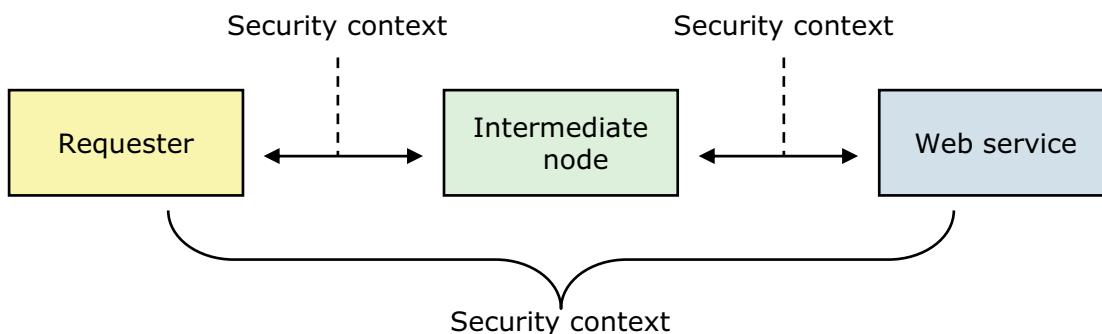
Confidentiality ensures that only authorized parties have access to protected resources. The effect of confidentiality is to keep private data or resources secret. This quality is often implemented through the encryption of data, in which only authorized parties have the means of making obscured data 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 2013

Figure 12-3. Web services security

WE4013.0

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
 - Username token profile
 - X.509 token profile
 - Kerberos token profile
 - SAML token profile: Security Assertion Markup Language
 - REL token profile: Rights Expression Language
- Confidentiality (XML encryption)
 - Process for encrypting data and representing the result in XML
- Integrity (XML signature)
 - Digitally sign the SOAP XML document, providing integrity and signer authentication
- XML canonicalization
 - Normalizes XML document
 - Ensures that two semantically equivalent XML documents contain the same octet stream

© Copyright IBM Corporation 2013

Figure 12-4. Components of WS-Security

WE4013.0

Notes:

An XML digital signature is based on the W3C recommendation specification for XML-signature syntax and processing. See: <http://www.w3.org/TR/xmldsig-core/>

XML encryption is based on the W3C recommendation for XML encryption syntax and processing. 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 2013

Figure 12-5. Specifying security in SOAP messages

WE4013.0

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 2013

Figure 12-6. Scenario 1: Ensure confidentiality with XML encryption

WE4013.0

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 2013

Figure 12-7. XML encryption and WS-Security

WE4013.0

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 2013

Figure 12-8. DataPower support for XML encryption

WE4013.0

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
- Recipient Certificate: the certificate that is used to encrypt the encryption key
- One Ephemeral Key: causes all encryption in this step to use the same ephemeral key

© Copyright IBM Corporation 2013

Figure 12-9. Encrypt action

WE4013.0

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:

- 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 the message.

The WS-Security standard puts the signature and key information in the WS-Security header of the SOAP message. This standard adds no elements to the body of the message, and therefore does not violate the underlying schema.

Standard XML encryption was originally designed to handle any XML message, including those messages that are not formatted to the SOAP specification. It puts the signature and

key information in the body of the message, thus adding more elements to the body of the message.

The DataPower SOA appliance supports both methods of encryption. The appliance can use either standard for full message or partial encryption.

The following message types are supported:

- **SOAP message**: an encrypted SOAP document
- **Raw XML document**: an encrypted XML document (it cannot be used with WSSec encryption)
- **Selected elements (field-level)**: a partially encrypted SOAP document

The following options are in the **Message and Attachment Handling** menu:

- **Attachments only**: Only the attachments of the message are encrypted.
- **Message only**: Only the message (root part) is encrypted.
- **Message and attachments**: The 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

Message 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 2013

Figure 12-10. Decrypt action

WE4013.0

Notes:

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



Field-level encryption and decryption

- Performs field-level encryption and decryption on messages
 - Under **Message Type**, select the **Selected Elements (Field-Level)** radio button
 - Create a **Document Crypto Map** with an XPath expression of the fields to encrypt or decrypt

The screenshot shows two interface components related to field-level encryption and decryption:

- Document Crypto Map Dialog:** This window is titled "Document Crypto Map : Name_DCM [up]". It contains fields for "Admin State" (enabled), "Comments", "Operation" (set to "Decrypt"), and an "XPath Expression" input box containing the value: `/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByName']/*[local-name()='EncryptedData']`. There are also buttons for "Apply", "Cancel", "Undo", and links for "Export", "View Log", "View Status", and "Help".
- Message Type Configuration Panel:** This panel has a "Message Type" section with three radio button options: "Entire Message/Document" (unchecked), "Selected Elements (Field-Level)" (checked), and "Advanced" (unchecked). Below this is a "Document Crypto Map" section with a dropdown menu set to "Name_DCM" and a "Buttons" row containing a plus sign (+) and a three-dot ellipsis (...).

© Copyright IBM Corporation 2013

Figure 12-11. Field-level encryption and decryption

WE4013.0

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 the **XPath Tool** button 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

Select XPath Expression for something in an XML File

Select sample XML document

| | | |
|--|---|---|
| URL of Sample XML Document | local:/// <input type="button" value="Edit..."/> <input type="button" value="View..."/> * | findByLocation.xml <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> |
| Namespace Handling | <input checked="" type="radio"/> local <input type="radio"/> prefix <input type="radio"/> uri | |
| Click on element or attribute name or value to select an XPath expression | | |
| XPath * <pre>/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']/*[namespace-uri()='http://address-training.ibm.com' and local-name()='findByLocation']/state</pre> | | |

© Copyright IBM Corporation 2013

Figure 12-12. XPath tool

WE4013.0

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:** Compares only the local name (element name), ignoring the namespace.
- prefix:** 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:** Compares the local name and namespace URI.

Sample encrypted SOAP message

```

<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header>
        <wsse:Security soapenv:mustUnderstand="1">
            <xenc:EncryptedKey>
                ...
                </xenc:EncryptedKey>
            </wsse:Security>
        </soapenv:Header>
        <soapenv:Body>
            <q0:findByName>
                <xenc:EncryptedData>
                    <EncryptionMethod />
                    <CipherData>
                        <CipherValue>
                            ...
                            </CipherValue>
                        </CipherData>
                    </xenc:EncryptedData>
                </q0:findByName>
            </soapenv:Body>
        </soapenv:Envelope>
    
```

The diagram illustrates the structure of an encrypted SOAP message. It shows the XML code for an envelope, header, and body. In the header, there is a security block containing an encrypted key. In the body, there is a 'findByName' operation which contains an encrypted data block. The entire body is grouped by a large brace labeled 'Field-level XML encryption'. Within the body, the 'EncryptedData' element is grouped by a smaller brace labeled 'Key that is used to encrypt message'.

© Copyright IBM Corporation 2013

Figure 12-13. Sample encrypted SOAP message

WE4013.0

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 guarantee that message is not changed
 - Mismatch between decrypted hash (from signature) and computed hash (from cleartext) indicates that message is modified
 - Signatures provide strong indication of identity
 - Only holder of private key can create signature that matches the enclosed certificate (if asymmetric)
- XML signature standard provides a schema for storing digital signature information within XML messages
 - Does not describe how to digest and sign messages
 - Supports symmetric and asymmetric signing key
- The recipient validates the signature by repeating the same steps that are used to generate a digital signature
 - Compute message digest of received message
 - Obtain original message digest by decrypting signature from received message by using certificate, which holds public key
 - Compare computed message digest against original message digest

© Copyright IBM Corporation 2013

Figure 12-14. Scenario 2: Ensure integrity with XML signatures

WE4013.0

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, 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.

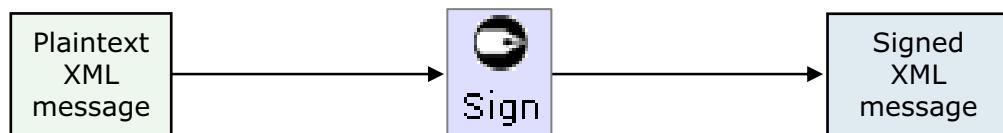
Consider the following example:

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

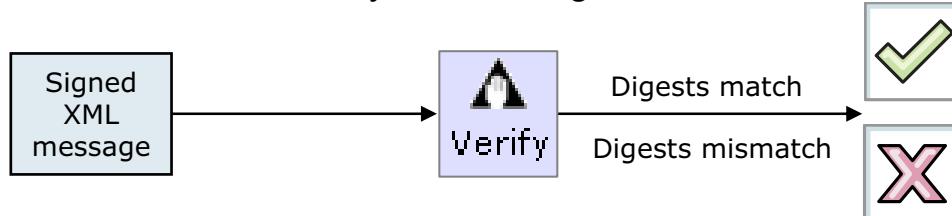
You can sign the value within the symbol element, the entire symbol element (including the value), or a group of elements within transaction-info.

DataPower support for XML signature

- Apply XML signature to a message by defining a document processing rule that contains:
 - **Sign** action: digitally signs a message
 - **Verify** action: verifies the digital signature in the message
- Acts as sender to sign message that is sent to the server



- Acts as receiver to verify the message that the client sent



© Copyright IBM Corporation 2013

Figure 12-15. DataPower support for XML signature

WE4013.0

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 signature in message
 - Message Type**
 - Key:** crypto key object that is used to sign message
 - Certificate:** crypto certificate object that is associated with crypto key object

| Sign | |
|----------------------------|---|
| Envelope Method | <input type="radio"/> Enveloped Method <input type="radio"/> Enveloping Method <input type="radio"/> SOAPSec Method <input checked="" type="radio"/> WSSec Method <input type="radio"/> Advanced * |
| Message Type | <input type="radio"/> SOAP Message <input type="radio"/> SOAP With Attachments <input type="radio"/> Raw XML Document <input type="radio"/> Selected Elements (Field-Level) <input type="radio"/> Advanced * |
| Asynchronous | <input type="radio"/> on <input checked="" type="radio"/> off |
| WS-Security Version | 1.0 <input type="button" value="Save"/> |
| Use Asymmetric Key | <input checked="" type="radio"/> on <input type="radio"/> off <input type="button" value="Save"/> |
| Signing Algorithm | rsa <input type="button" value="Save"/> |
| Key | (none) <input type="button" value="Save"/> |
| Certificate | (none) <input type="button" value="Save"/> |

© Copyright IBM Corporation 2013

Figure 12-16. Sign action

WE4013.0

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 itself. The object, or its content, is identified by using a reference through a URI fragment identifier or transform (not considered a good idea).
- SOAPSec Method:** The signature is included in a SOAP header entry.
- WSSec Method:** The signature is included in a WS-Security security header.

If an Envelope Method of **WSSec Method** and a Message Type of either **SOAP Message** or **SOAP With Attachments** are selected, then the page shows a **Use Asymmetric Key** option. If the choice is:

- **On**, then the Signing Algorithm shows asymmetric choices
- **Off**, then the Signing Algorithm shows symmetric HMAC choices



Verify action

- The **Verify** action verifies a digital signature
- Signature Verification Type
 - Use asymmetric only, symmetric only, or either
- Optional Signer Certificate
 - Used instead of passed certificate
- Validation credential object
 - One or more certificate objects that are used to validate the signer certificate

Verify

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

© Copyright IBM Corporation 2013

Figure 12-17. Verify action

WE4013.0

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 * |
| XSL style sheet | store://I	verify.xsl Upload... Fetch... Edit Stylesheet Summary: Verify RSA, DSA or HMAC signatures. |
| Asynchronous | <input type="radio"/> on <input checked="" type="radio"/> off |
| Output Type | Default |
| Signature Verification Type | RSA/DSA Signatures <input type="checkbox"/> Save |
| Optional Signer Certificate | <input type="text"/> <input type="checkbox"/> Save |
| Validation Credential | (none) <input type="button" value="+"/> <input type="button" value="..."/> <input type="checkbox"/> Save |
| Check Timestamp | <input checked="" type="radio"/> on <input type="radio"/> off <input type="checkbox"/> Save |
| Check Timestamp Created | <input type="radio"/> on <input checked="" type="radio"/> off <input type="checkbox"/> Save |
| Check Timestamp Expiration | <input checked="" type="radio"/> on <input type="radio"/> off <input type="checkbox"/> Save |
| Timestamp Expiration Override Period | 0 sec <input type="checkbox"/> Save |
| (empty) | |

© Copyright IBM Corporation 2013

Figure 12-18. Verify action: Advanced tab

WE4013.0

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 the **Selected Elements (Field-Level)** radio button
 - Create a Document Crypto Map with an XPath expression of the fields to sign



© Copyright IBM Corporation 2013

Figure 12-19. Field-level message signature and verification

WE4013.0

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>

```

XML
signature

© Copyright IBM Corporation 2013

Figure 12-20. Sample signed SOAP message

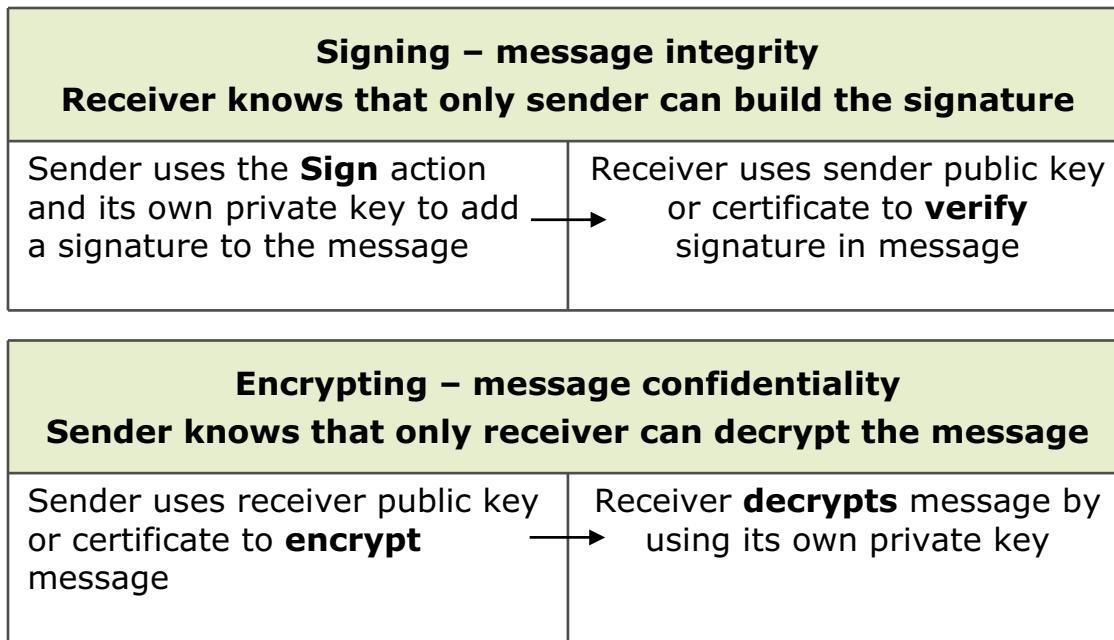
WE4013.0

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



Sign message, then encrypt, for best confidentiality and integrity

© Copyright IBM Corporation 2013

Figure 12-21. Summary of security and keys

WE4013.0

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 signatures

© Copyright IBM Corporation 2013

Figure 12-22. Unit summary

WE4013.0

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 the message and field levels, but sign and verify occur at the message level only.
3. True or False: The validation credential object validates the signer certificate, which is the public key that is used to generate the digital signature. This certificate is usually included in the message, but an alternative certificate can be specified in the **Signer Certificate** field.

© Copyright IBM Corporation 2013

Figure 12-23. Checkpoint questions

WE4013.0

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.** 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 2013

Figure 12-24. Checkpoint answers

WE4013.0

Notes:

Exercise 10



Web service encryption and digital signatures

© Copyright IBM Corporation 2013

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

7.0.1

Figure 12-25. Exercise 10

WE4013.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Create a multi-protocol gateway to generate a message with XML encryption
- Create a multi-protocol gateway to generate a message with an XML digital signature
- Perform field-level encryption and decryption on XML messages
- Create a rule to decrypt messages and verify digital signatures that are contained in a message within a web service proxy policy

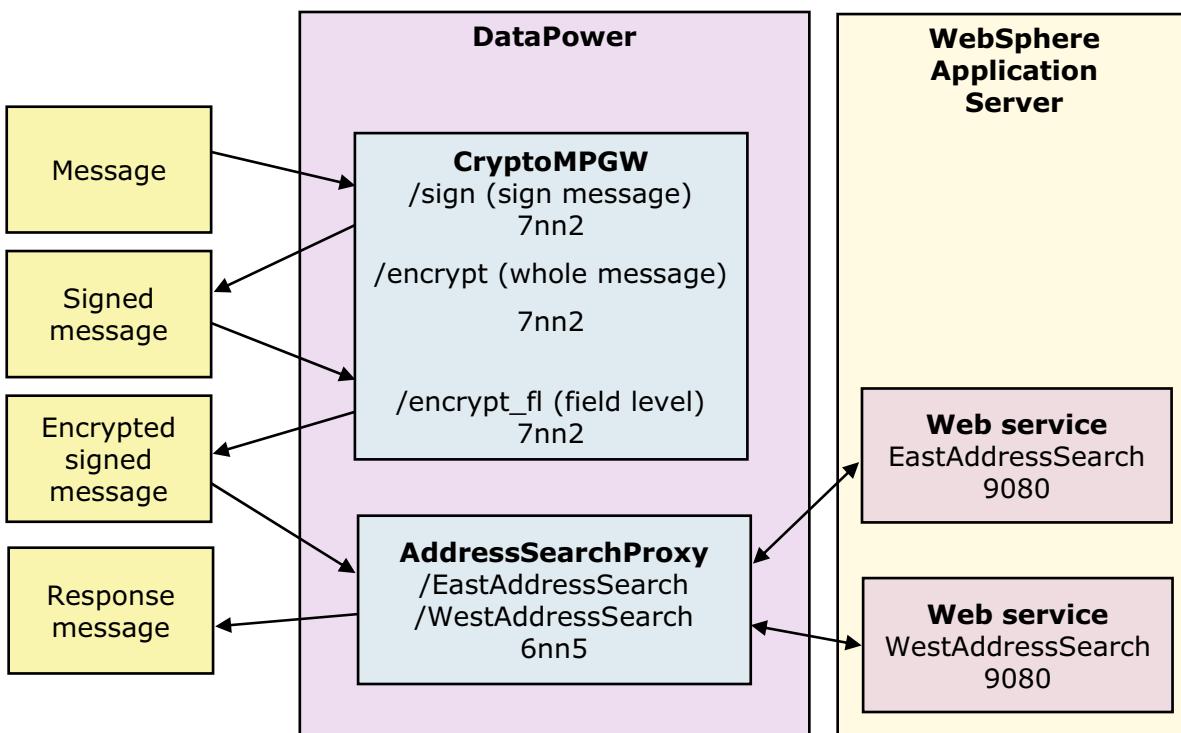
© Copyright IBM Corporation 2013

Figure 12-26. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 12-27. Exercise overview

WE4013.0

Notes:

Unit 13. Authentication, authorization, and auditing (AAA)

What this unit is about

This unit describes the authentication, authorization, and auditing (AAA) framework within the XI50 and XS40 IBM WebSphere DataPower SOA Appliances. These three facets of security both monitor and restrict access to resources.

What you should be able to do

After completing this unit, you should be able to:

- Describe the AAA framework within the WebSphere DataPower SOA Appliance
- Explain the purpose of each step in an access control policy
- Authenticate and authorize web service requests with:
 - WS-Security username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

How you will check your progress

- Checkpoint
- Exercise 11: Web service authentication and authorization

Unit objectives

After completing this unit, you should be able to:

- Describe the AAA framework within the WebSphere DataPower SOA Appliance
- Explain the purpose of each step in an access control policy
- Authenticate and authorize web service requests with:
 - WS-Security Username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

© Copyright IBM Corporation 2013

Figure 13-1. Unit objectives

WE4013.0

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 2013

Figure 13-2. Authentication, authorization, and auditing

WE4013.0

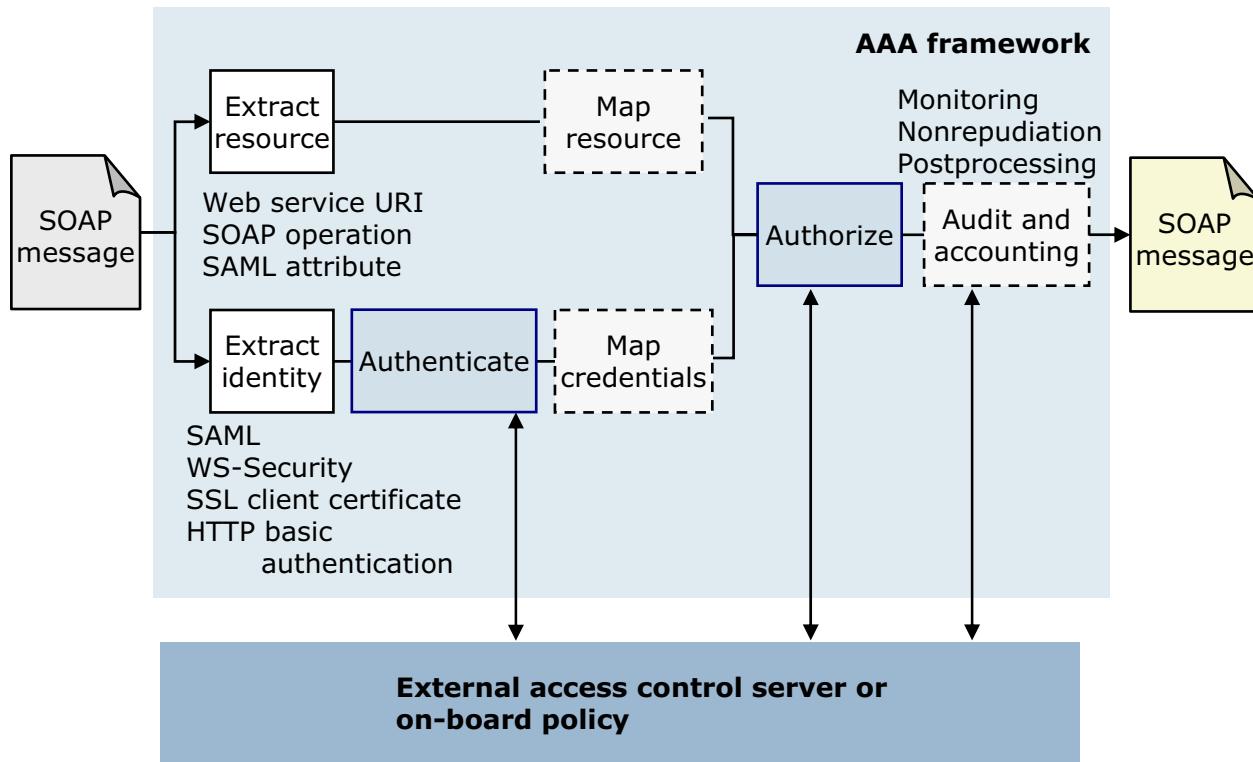
Notes:

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 2013

Figure 13-3. Authentication and authorization framework

WE4013.0

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.

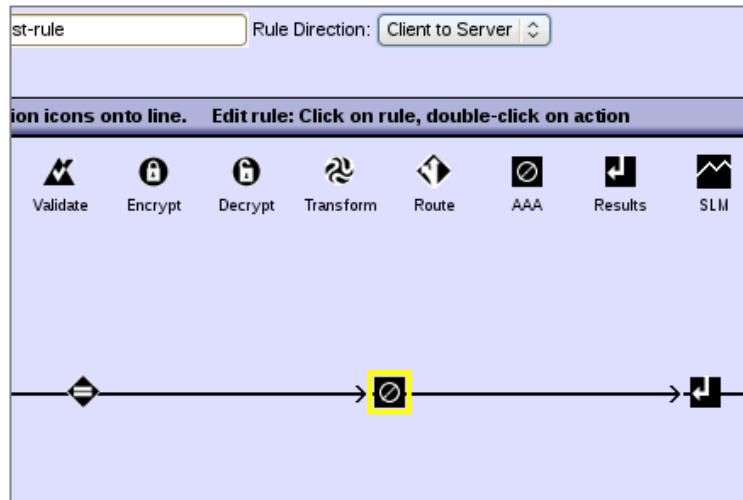
The various tokens that can be generated are:

- 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 2013

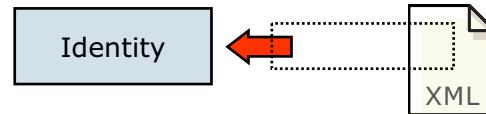
Figure 13-4. AAA action and access control policy

WE4013.0

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 2013

Figure 13-5. How to define an access control policy (1 of 2)

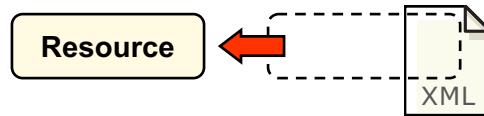
WE4013.0

Notes:

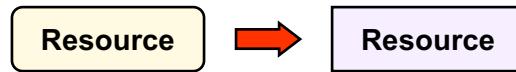
The access control policy steps relate directly to the processing stages within the AAA framework. In the first step, the policy defines how the framework retrieves information about the client identity. The framework can treat the requested URL, the client IP address, the HTTP header, or any part of 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)

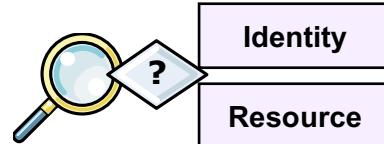
4. Define resource extraction methods



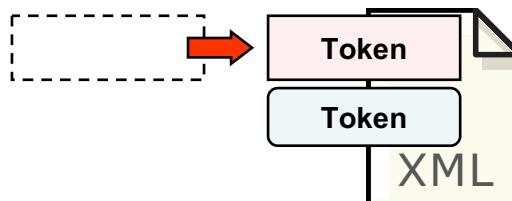
5. Map requested resources (optional)



6. Define the authorization method



7. Specify postprocessing actions (optional)



© Copyright IBM Corporation 2013

Figure 13-6. How to define an access control policy (2 of 2)

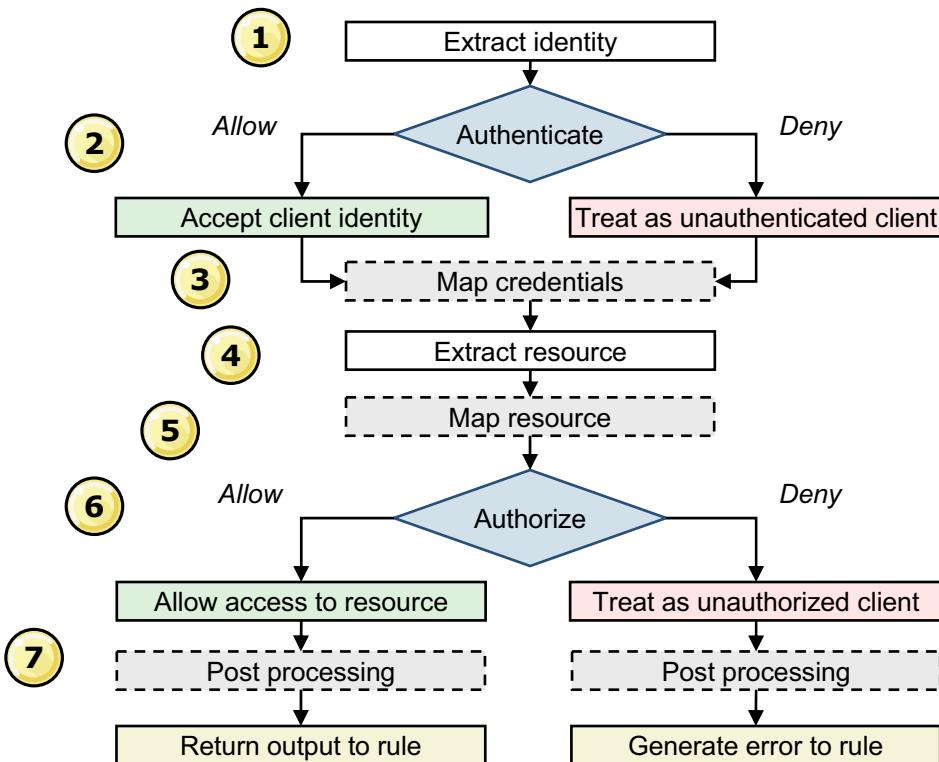
WE4013.0

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 2013

Figure 13-7. Access control policy processing

WE4013.0

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
 - The requested resource is the web service operation
 - Allows any authenticated client access to the web service operation

© Copyright IBM Corporation 2013

Figure 13-8. Scenario 1: Authorize authenticated clients

WE4013.0

Notes:

In this scenario, the client includes a WS-Security username token with a password or password digest as a proof of identity. As a good practice, clients send plaintext tokens, such as the WS-Security username token, within a secure channel, such as an SSL connection.

The access control policy on the DataPower SOA appliance verifies the user name and password against a built-in user list. It assumes that all authenticated users have full access to any resource protected by the policy.

Scenario 1: Sample SOAP request message

```
<?xml version="1.0" encoding="UTF-8">
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsse="http://...wssecurity-secext-1.0.xsd"
    xmlns:q0="http://east.address.training.ibm.com">
    <soap:Header>
        <wsse:Security>
            <wsse:UsernameToken>
                <wsse: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 2013

Figure 13-9. Scenario 1: Sample SOAP request message

WE4013.0

Notes:

The WS-Security username token provides a basic method to transport user credentials to a web service. The password field can be in plaintext, or the Secure Hash Algorithm (SHA1) can hash it. Since SHA1 is a well-known algorithm, a hashed password provides a minimal level of security by obfuscating the password. Messages with these identity credentials are sent only over a secure connection.

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

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.

Scenario 1: Identify the client

1. Create a AAA policy object on the DataPower SOA appliance
2. Extract the client's identity by the **Password-carrying UsernameToken Element from WS-Security Header** option
3. For the authentication method, **Use DataPower AAA Info File**
 - Specify the name of the AAA information file in the **URL** field
4. Leave the identity mapping method at **none**

AAA Policy Name: AddressAdmin

Define how to extract a user's identity from an incoming request.

HTTP Authentication Header
 Password-carrying UsernameToken Element from WS-Security Header
 Derived-key UsernameToken Element from WS-Security Header
 BinarySecurityToken Element from WS-Security Header

Define how to authenticate the user.

Method

Accept a SAML Assertion with a Valid Signature
 Accept an LTPA token
 Bind to Specified LDAP Server
 Contact a SAML Server for a SAML Authentication Statement
 Contact a WS-Trust Server for a WS-Trust Token
 Contact ClearTrust Server
 Contact Netegrity SiteMinder
 Contact NSS for SAF Authentication
 Contact Tivoli Access Manager
 Custom Template
 Pass Identity Token to the Authorize Step
 Retrieve SAML Assertions Corresponding to a SAML Browser Art
 Use an Established WS-SecureConversation Security Context
 Use certificate from BinarySecurityToken
 Use DataPower AAA Info File
 Use specified RADIUS Server

| | |
|-----|--|
| URL | <input type="text" value="store:///"/> AAAInfo.xml + ... Upload... |
|-----|--|

© Copyright IBM Corporation 2013

Figure 13-10. Scenario 1: Identify the client

WE4013.0

Notes:

Scenario 1: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method
 - The name of the child element in the SOAP body of the request is the request element name
6. Leave the resource mapping method at **None**
7. For the authorization method, allow any request from an authenticated client to proceed

Resource Identification Methods

- URL Sent to Back End
- URL Sent by Client
- URI of TopLevel Element in the Message
- Local Name of Request Element
- HTTP Operation (GET/POST)
- XPath Expression
- Processing Metadata
- *

Define how to map resources.

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

Define how to authorize a request.

| | |
|---------------|--|
| Method | <input type="radio"/> AAA Info File <input checked="" type="radio"/> Allow Any Authenticated Client <input type="radio"/> Always Allow <input type="radio"/> Check for Membership in an LDAP Group <input type="radio"/> Contact ClearTrust Server <input type="radio"/> Contact Netegrity SiteMinder <input type="radio"/> Contact NSS for SAF Authorization <input type="radio"/> Contact Tivoli Access Manager |
|---------------|--|

© Copyright IBM Corporation 2013

Figure 13-11. Scenario 1: Authorize access to resources

WE4013.0

Notes:



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 2013

Figure 13-12. Scenario 2: Security token conversion

WE4013.0

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 2013

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

WE4013.0

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. Since this encoding method is known, the client uses an SSL connection to keep the contents of this message private.

Base64 is a binary-to-text encoding scheme by printable (mostly alphanumeric) characters. As a MIME content transfer encoding, it is used to encode binary data into email messages.

In the HTTP basic authentication scheme, the user name and password are concatenated with a colon (:) before it is encoded byBase64. For example, the user name “Alice” and the password “ond3mand” become “Alice:ond3mand”. In Base64 encoding, the user name and password string is QWxpxY2U6b25kM21hbmQ= .

 WebSphere Education 

Scenario 2: Identify the client

1. Create a AAA policy object on the DataPower SOA appliance

2. Extract the client's identity by the **HTTP Authentication header** option

- The value within the Authorization HTTP header represents the HTTP authentication header

3. For the authentication method, specify **Pass Identity Token** to the Authorize Step

4. Leave the identity mapping method at **none**

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

| |
|---|
| <input type="radio"/> Contact Tivoli Access Manager |
| <input type="radio"/> Custom Template |
| <input checked="" type="radio"/> Pass Identity Token to the Authorize Step |
| <input type="radio"/> Retrieve SAML Assertions Corresponding to a SAML Browser Artifact |

| | |
|---------------------------------------|---|
| Define how to map credentials. | |
| Method | <input type="text" value="none"/> * |

© Copyright IBM Corporation 2013

Figure 13-14. Scenario 2: Identify the client

WE4013.0

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

| | | | | | | | | |
|---|--|--|-------------------------|--|--|--|------------------------------|----------------------|
| Resource Identification Methods | | <input type="checkbox"/> URL Sent to Back End <input type="checkbox"/> URL Sent by Client <input type="checkbox"/> URI of TopLevel Element in the Message <input checked="" type="checkbox"/> Local Name of Request Element <input type="checkbox"/> HTTP Operation (GET/POST) <input type="checkbox"/> XPath Expression <input type="checkbox"/> Processing Metadata * | | | | | | |
| Define how to authorize a request. <table border="1"> <tr> <td>Method</td> <td> <input type="radio"/> AAA Info File <input type="radio"/> Allow Any Authenticated Client <input checked="" type="radio"/> Always Allow <input type="radio"/> Check for Membership in an LDAP G <input type="radio"/> Contact ClearTrust Server </td> </tr> </table> | | | Method | <input type="radio"/> AAA Info File <input type="radio"/> Allow Any Authenticated Client <input checked="" type="radio"/> Always Allow <input type="radio"/> Check for Membership in an LDAP G <input type="radio"/> Contact ClearTrust Server | | | | |
| Method | <input type="radio"/> AAA Info File <input type="radio"/> Allow Any Authenticated Client <input checked="" type="radio"/> Always Allow <input type="radio"/> Check for Membership in an LDAP G <input type="radio"/> Contact ClearTrust Server | | | | | | | |
| Choose any post processing. <table border="1"> <tr> <td>Include Password</td> <td> <input checked="" type="radio"/> on <input type="radio"/> off </td> </tr> <tr> <td>WS-Security UsernameToken Password Type</td> <td> <input type="radio"/> Digest <input type="radio"/> Basic </td> </tr> <tr> <td>Actor/Role Identifier</td> <td> <input type="text"/> </td> </tr> </table> | | | Include Password | <input checked="" type="radio"/> on <input type="radio"/> off | WS-Security UsernameToken Password Type | <input type="radio"/> Digest <input type="radio"/> Basic | Actor/Role Identifier | <input type="text"/> |
| Include Password | <input checked="" type="radio"/> on <input type="radio"/> off | | | | | | | |
| WS-Security UsernameToken Password Type | <input type="radio"/> Digest <input type="radio"/> Basic | | | | | | | |
| Actor/Role Identifier | <input type="text"/> | | | | | | | |

© Copyright IBM Corporation 2013

Figure 13-15. Scenario 2: Authorize access to resources

WE4013.0

Notes:

The **Post Processing Enabled** setting applies a custom style sheet to the outgoing request message. The setting does not require enablement for a built-in post processing step, such as adding a WS-Security username token.

The slide does not show the setting of the “Add WS-Security username Token” to **on**. When this option is set, the page repaints to include the fields shown (“Include Password” and the others).

Scenario 3: Multiple identity extraction methods

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - Uses either a WS-Security UsernameToken element or a BinarySecurityToken element from the WS-Security header to determine the client's identity
 - Verifies the identity of the client
 - The requested resource is the web service operation
 - Allows any authenticated client access to the web service operation

© Copyright IBM Corporation 2013

Figure 13-16. Scenario 3: Multiple identity extraction methods

WE4013.0

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 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 Specified LDAP Server**
 - The LDAP directory server provides an external list of authenticated users
4. Leave the identity mapping method at **none**

| |
|---|
| <input type="checkbox"/> HTTP's Authentication Header |
| <input checked="" type="checkbox"/> Password-carrying UsernameToken Element from WS-Security Header |
| <input type="checkbox"/> Derived-key UsernameToken Element from WS-Security Header |
| <input checked="" type="checkbox"/> BinarySecurityToken Element from WS-Security Header |
| <input type="checkbox"/> WS-SecureConversation Identifier |
| <input type="checkbox"/> WS-Trust Base or Supporting Token |
| <input type="checkbox"/> Kerberos AP-REQ from WS-Security Header |
| <input type="checkbox"/> Kerberos AP-REQ from SPNEGO Token |

| | |
|---------------|---|
| Method | <input type="radio"/> Use DataPower AAA Info File <input checked="" type="radio"/> Bind to Specified LDAP Server <input type="radio"/> Contact Tivoli Access Manager <input type="radio"/> Contact Netegrity SiteMinder <input type="radio"/> Use specified RADIUS Server |
|---------------|---|

| | |
|---------------------------------------|---|
| Define how to map credentials. | |
| Method | <input type="text" value="none"/> * ▼ |

© Copyright IBM Corporation 2013

Figure 13-17. Scenario 3: Identify the client

WE4013.0

Notes:



Scenario 3: Authorize access to resources

5. Select **Local Name of Request Element** as the resource extraction method

- The name of the child element in the SOAP body of the request is the request element name

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

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

Define how to map resources.

| | | |
|---------------|--|-------------------------------------|
| Method | <input style="border: 1px solid black; padding: 2px 10px; margin-right: 5px;" type="button" value="none"/> * | * |
|---------------|--|-------------------------------------|

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

Define how to authorize a request.

| | |
|---------------|--|
| Method | <input type="radio"/> AAA Info File <input checked="" type="radio"/> Allow Any Authenticated Client <input type="radio"/> Always Allow <input type="radio"/> Check for Membership in an LDAP G <input type="radio"/> Contact ClearTrust Server |
|---------------|--|

© Copyright IBM Corporation 2013

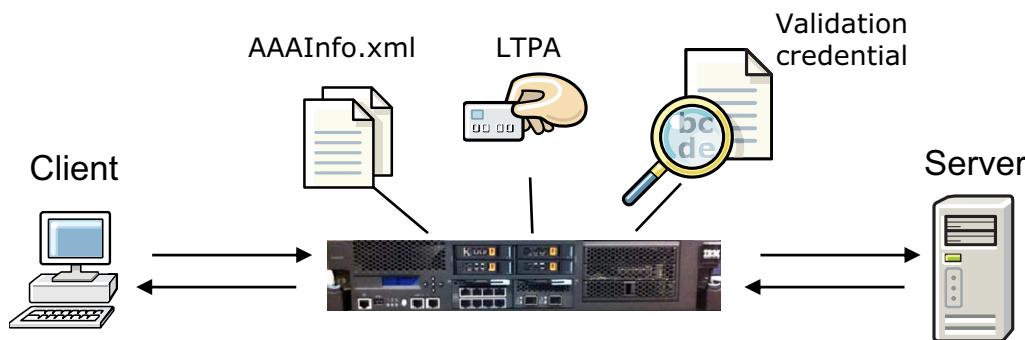
Figure 13-18. Scenario 3: Authorize access to resources

WE4013.0

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 2013

Figure 13-19. Internal access control resources

WE4013.0

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 DataPower AAA Info File**
 - Select an existing XML file or create a AAA file

Use DataPower AAA Info File
 Use specified RADIUS Server
 Validate a Kerberos AP-REQ for the Correct Server Principal
 Validate the Signer Certificate for a Digitally Signed Message.
 Validate the SSL Certificate from the Connection Peer
 *

URL

store:/// *

© Copyright IBM Corporation 2013

Figure 13-20. AAA XML file

WE4013.0

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 2013

Figure 13-21. Example AAA XML file

WE4013.0

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 2013

Figure 13-22. Lightweight Third Party Authentication

WE4013.0

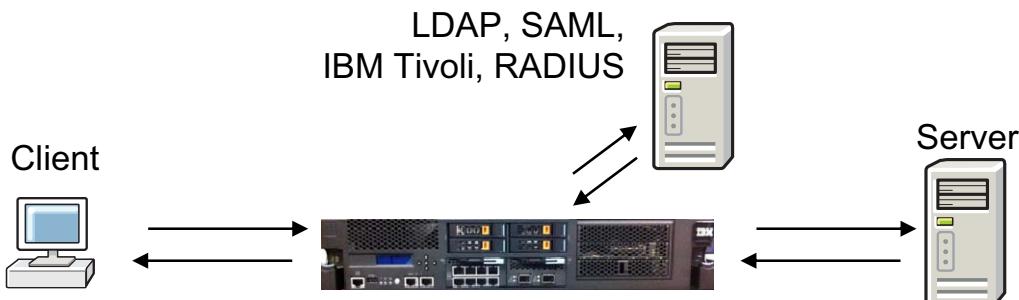
Notes:

For more information, see the article **IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6**, which includes a section on using the LTPA token:

http://www.ibm.com/developerworks/websphere/library/techarticles/0911_rasmussen/0911_rasmussen.html

The following IBM developerWorks article provides an overview of a single sign-on scenario with IBM WebSphere and Lotus Domino servers, which LTPA tokens facilitate:
<http://www.ibm.com/developerworks/ibm/library/it-0101art2/>

External access control resource



- Delegates the authentication and authorization task to an external security system
- The authentication and authorization tasks can be delegated to the same system or to separate systems
 - For example, an LDAP directory tracks client identities, while IBM Tivoli Access Manager determines whether the client has access to the specified resource
 - The **map credentials** and **map resource** steps convert the security token to match the input that the authorization step requires

© Copyright IBM Corporation 2013

Figure 13-23. External access control resource

WE4013.0

Notes:

It is also possible to do authentication and authorization on an IBM Tivoli Access Manager system. Tivoli 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. Consult the WebGUI guide for a full list of security products and specifications that are supported.

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)** acts as 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 2013

Figure 13-24. Lightweight Directory Access Protocol

WE4013.0

Notes:

The next unit covers configuring AAA by LDAP.

Security Assertion Markup Language

- SAML provides an XML-based framework for exchanging authentication, authorization, and attribute assertions between the entities
 - This language 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 2013

Figure 13-25. Security Assertion Markup Language

WE4013.0

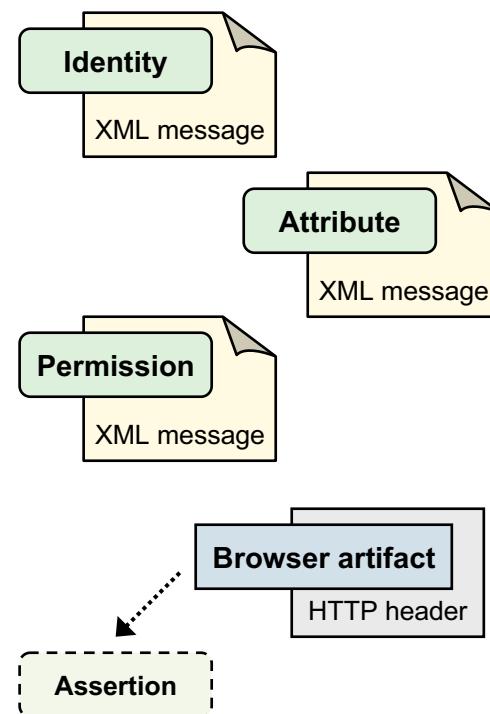
Notes:

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 itself. One common use for assertions is single sign-on: after a security server authenticates a client, a SAML authentication statement is tagged to the client request. Subsequent systems that process the request need only to trust the assertion instead of authenticating the client again.

Types of SAML assertions

- Three main types of XML-based SAML assertions exist:
 - Authentication** assertions represent the identity of the specified subject that is verified by another entity
 - Attribute** assertions represent any attributes that are associated with the specified subject
 - Authorization** decision assertions represent whether the specified subject is granted or denied access to a specified resource
- In addition, the HTTP binding provides a non-XML reference:
 - A **SAML artifact** that is embedded in the URL query string provides a reference to an actual SAML assertion that is stored in a remote site



© Copyright IBM Corporation 2013

Figure 13-26. Types of SAML assertions

WE4013.0

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 2013

Figure 13-27. Scenario 4: Authorize valid SAML assertions

WE4013.0

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 in 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 2013

Figure 13-28. Scenario 4: SAML authentication statement (1 of 2)

WE4013.0

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 2013

Figure 13-29. Scenario 4: SAML authentication statement (2 of 2)

WE4013.0

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 2013

Figure 13-30. Scenario 4: SAML attribute statement

WE4013.0

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 the client

1. Create a AAA policy object on the DataPower SOA appliance
2. Extract the client's identity by the **Name from SAML authentication assertion** option
3. For the authentication method, select **Accept a SAML Assertion with a Valid Signature**
 - Specify the validation credential for the SAML signature
4. Leave the identity mapping method at **none**

| Identification Methods | <input type="checkbox"/> Name from SAML attribute assertion <input checked="" type="checkbox"/> Name from SAML authentication assertion <input type="checkbox"/> SAML artifact <input type="checkbox"/> Client IP address <input type="checkbox"/> Subject DN from certificate in the message's signature |
|------------------------|---|
|------------------------|---|

| Define how to authenticate the user. | |
|--------------------------------------|---|
| Method | <input checked="" type="radio"/> Accept a SAML Assertion with a Valid Signature <input type="radio"/> Accept an LTPA token <input type="radio"/> Bind to Specified LDAP Server <input type="radio"/> Contact a SAML Server for a SAML Authentication |

| SAML Signature Validation Credentials | <input type="text" value="pubcert"/> <input type="button" value="..."/> |
|---------------------------------------|---|
|---------------------------------------|---|

© Copyright IBM Corporation 2013

Figure 13-31. Scenario 4: Identify the client

WE4013.0

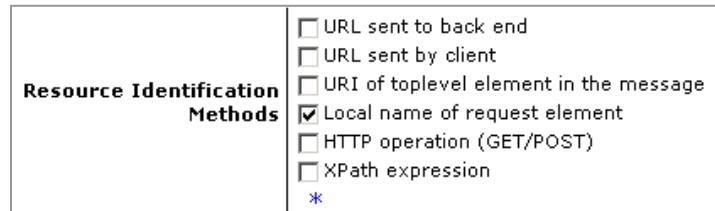
Notes:

To verify the signature of the SAML assertion, the access control policy needs the validation credential.

Scenario 4: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method
 - The name of the child element in the SOAP body of the request is the request element name
6. For the authorization method, **Use SAML Attributes from Authentication**
 - Set the SAML attribute that matches type as **Must match at least one name and value**
7. Select **SAML Attributes** from the authentication method page



| Define how to authorize a request. | |
|------------------------------------|---|
| Method | <input type="radio"/> Generate a SAML Authorization Query <input type="radio"/> Generate a SAML Attribute Query <input checked="" type="radio"/> Use SAML Attributes from Authentication <input type="radio"/> XPath <input type="radio"/> Any <input type="radio"/> All <input checked="" type="radio"/> Any-Value <input type="radio"/> All-Values |
| Type | |

SAML Attributes

[Back](#) |
 [Next](#) |
 [Advanced](#) |
 [Cancel](#)

© Copyright IBM Corporation 2013

Figure 13-32. Scenario 4: Authorize access to resources

WE4013.0

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. In 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 | http://address.trainir |
| Local Name | EastAddressSearch |
| Attribute Value | Query |

Submit **Cancel**

Configure an Access Control Policy

SAML Attributes

| Namespace URI | Local Name | Attribute Value |
|---------------------------------|-------------------|-----------------|
| http://address.training.ibm.com | EastAddressSearch | Query |

© Copyright IBM Corporation 2013

Figure 13-33. Scenario 4: Match SAML attributes

WE4013.0

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 2013

Figure 13-34. Access control policy by SAML information

WE4013.0

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 WebSphere DataPower SOA Appliance
- Explain the purpose of each step in an access control policy
- Authenticate and authorize web service requests with:
 - WS-Security Username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

© Copyright IBM Corporation 2013

Figure 13-35. Unit summary

WE4013.0

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 allows an access control policy to use different methods for authorization. The authorization step can use tokens that were not part of the original request message.
3. True or False: The postprocessing step in an access control policy adds more information to the outgoing request message or transforms the message itself.

© Copyright IBM Corporation 2013

Figure 13-36. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. **True.** 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.** Mapping credentials and requested resources allows an access control policy to use different methods for authorization. The authorization step can use tokens that were not part of the original request message.
3. **True.** The postprocessing step in an access control policy adds more information to the outgoing request message or transforms the message itself.

© Copyright IBM Corporation 2013

Figure 13-37. Checkpoint answers

WE4013.0

Notes:



Exercise 11



Web service authentication and authorization

© Copyright IBM Corporation 2013

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

7.0.1

Figure 13-38. Exercise 11

WE4013.0

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

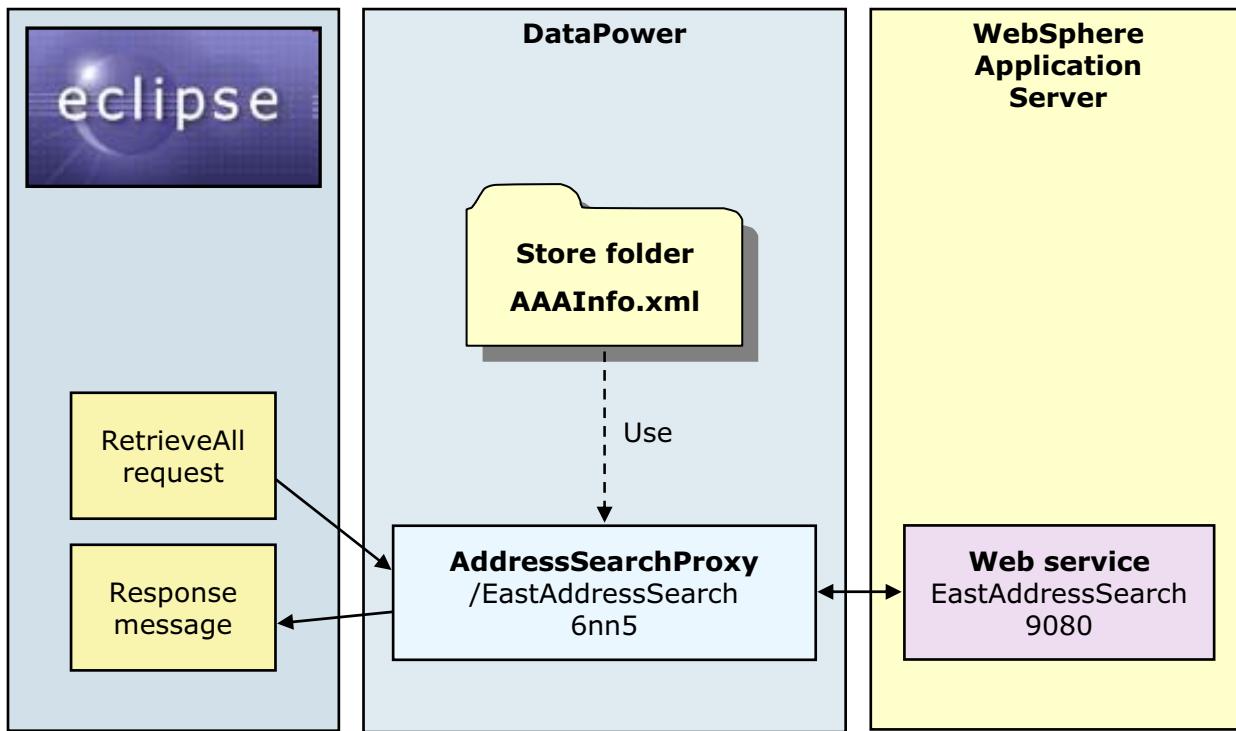
© Copyright IBM Corporation 2013

Figure 13-39. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 13-40. Exercise overview

WE4013.0

Notes:

Unit 14. Monitoring objects

What this unit is about

This unit shows you how to configure monitors to measure traffic volume and system latency.

What you should be able to do

After completing this unit, you should be able to:

- Identify messages that should be monitored
- Configure a message count monitor
- Set up a message duration monitor

How you will check your progress

- Checkpoint
- Lab exercise 12: Creating message counter monitors for a AAA policy

Unit objectives

After completing this unit, you should be able to:

- Identify messages that should be monitored
- Configure a message count monitor
- Set up a message duration monitor

© Copyright IBM Corporation 2013

Figure 14-1. Unit objectives

WE4013.0

Notes:

Message monitors

- Monitors allow constant feedback on the processing of messages that flow through the appliance
 - Define message set (that is, which messages to monitor)
 - Specify a count-based or response time-based threshold
 - Impose administrative controls when the message set exceeds configured thresholds
- Message count monitors:
 - Increment a counter every time a message that matches the message set passes through a service
- Message duration monitors:
 - Increment a counter every time a message that matches the message set has a response time that exceeds a configured amount of time

© Copyright IBM Corporation 2013

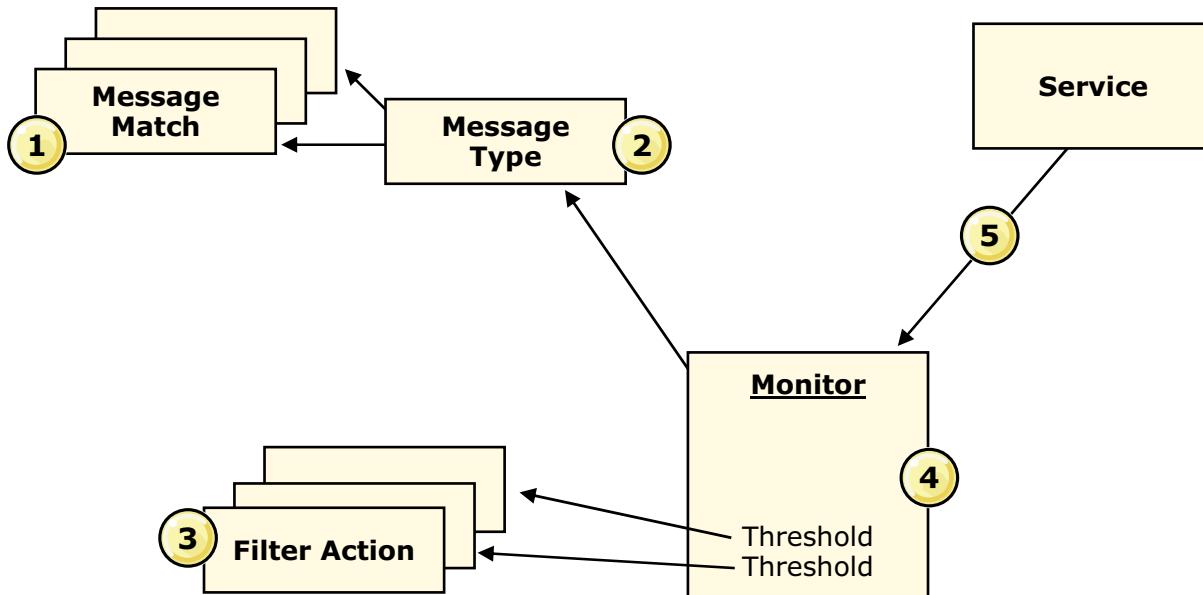
Figure 14-2. Message monitors

WE4013.0

Notes:

The message type object is what you use to specify the message set.

Monitor objects



© Copyright IBM Corporation 2013

Figure 14-3. Monitor objects

WE4013.0

Notes:

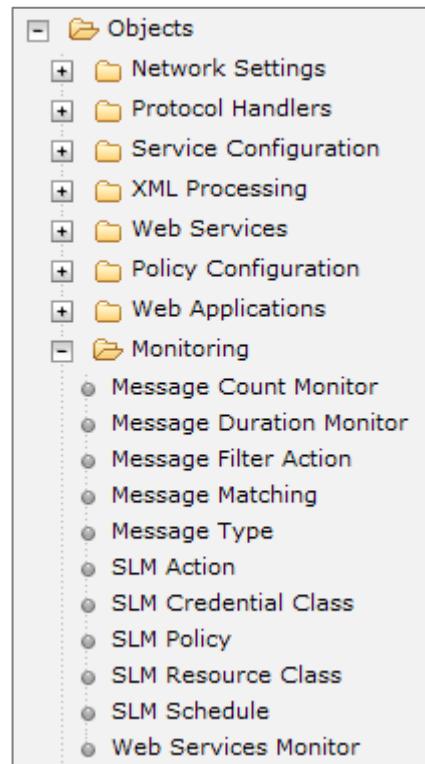
1. Message Match: an object that is used to identify a particular message stream
2. Message Type: an aggregate object that holds one or more message match objects
3. Message Filter Action: an object that specifies the administrative action that is taken when a threshold is exceeded
4. Message Monitor: an object that holds the definitions for the threshold values, associated message types, and associated **Filter** actions
5. Service Association: a service that must identify any message monitors watching the message traffic through the service

Filter policies might be simple (generation of a logging message), or stricter, which can result in a temporary denial of service to the offending message type.

You can define compound policies that activate an initial customary response when a message type exceeds a low threshold value, and activate stricter sanctions when the same message exceeds a higher threshold value.

Defining monitor objects

- From the vertical navigation bar, select **Objects > Monitoring**
- You can define:
 - Bottom-up: start at lowest level object; the easiest way to create reusable monitor objects
 - Top-down: create a message monitor, and create the lower-level objects from within that object
- Message Matching, Message Type, and Message Filter Action objects are defined the same way, regardless of which type of message monitor is used



© Copyright IBM Corporation 2013

Figure 14-4. Defining monitor objects

WE4013.0

Notes:

You can even start the creation of a message monitor from within the service that uses it.



Step 1: Specifying particular traffic to monitor

- From the vertical navigation bar, select **Objects > Monitoring > Message Matching > Add**
- Configure a message that matches an object to:
 - Specify an IP address range to include or exclude in the traffic definition
 - Limit the traffic definition to a single HTTP method
 - Specify a URL set to be included in the traffic definition
 - Indicate HTTP headers and values to include or exclude (separate tabs)

The screenshot shows the 'Configure Message Matching' dialog box with the 'Main' tab selected. The title bar says 'Configure Message Matching'. Below it is a toolbar with 'Apply', 'Cancel', 'Delete', and 'Undo' buttons. The main area has tabs for 'Main', 'HTTP Headers', and 'Excluded HTTP Headers', with 'Main' being active. A message matching rule is listed: 'Message Matching: retrieveAllCounter-message-matching [up]'. The configuration fields include:

| | |
|-----------------------|---|
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments | Text input field |
| IP Addresses | Text input field |
| Excluded IP Addresses | Text input field |
| HTTP Method | Dropdown menu set to 'any' |
| Request URL | Text input field containing '*' (wildcard) |

At the bottom right of the dialog box is the copyright notice: © Copyright IBM Corporation 2013.

Figure 14-5. Step 1: Specifying particular traffic to monitor

WE4013.0

Notes:



Step 1: Matching on HTTP headers

- Use the **HTTP Headers** tab to specify HTTP-based criteria for inclusion to the traffic definition
 - Specify HTTP header name-value pairs included in the traffic definition
- Use the **excluded HTTP headers** to specify exclusion from the traffic definition
 - Specify HTTP header name-value pairs to be excluded in the traffic definition

The screenshot shows a software interface titled 'HTTP Headers' with tabs for 'Main', 'HTTP Headers', and 'Excl'. Below the tabs, the text 'Message Matching: retrieveAllCount' is displayed. At the bottom are buttons for 'Apply', 'Cancel', 'Delete', and 'Undo'. The main area is titled 'HTTP Headers' and contains a table with two rows. The first row has columns for 'Name' (DataPower) and 'Value Match' (X*). The second row is empty. There are edit and delete icons next to the first row, and an 'Add' button at the bottom right.

| Name | Value Match |
|-----------|-------------|
| DataPower | X* |
| | |

© Copyright IBM Corporation 2013

Figure 14-6. Step 1: Matching on HTTP headers

WE4013.0

Notes:

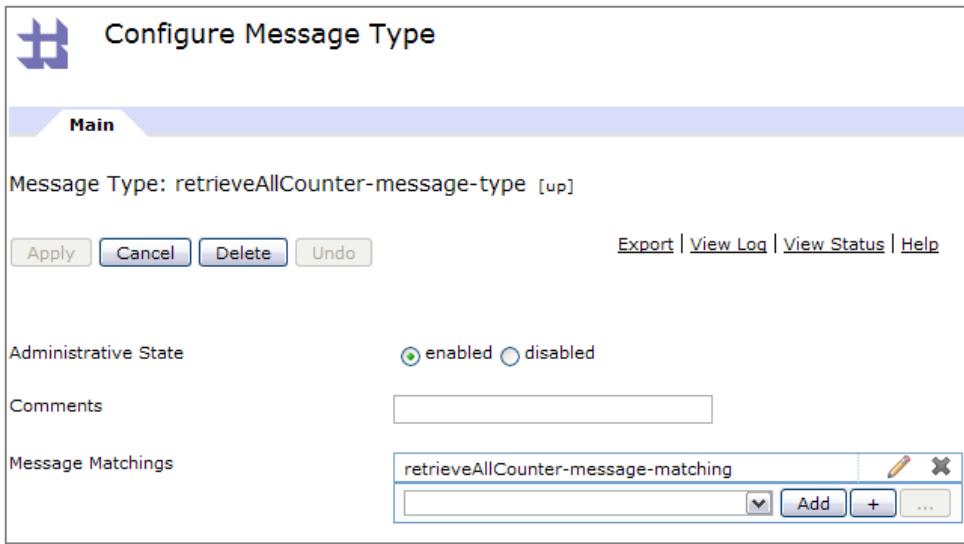
Messages are also matched by HTTP header values or by messages that do not contain certain HTTP header values.

For example, a match might be constructed to match only HTTP header values with the name `DataPower` and the value `x*` (the asterisk (*) is a wildcard).



Step 2: Message type configuration

- The *message type* object is a collection of one or more *message matching* objects
- Message type objects make it possible to combine various message matching objects into one type
- Each message type can use a different combination of message matching objects



The screenshot shows the 'Configure Message Type' dialog box. At the top, there's a 'Main' tab and a message type identifier 'Message Type: retrieveAllCounter-message-type [up]'. Below the tabs are buttons for 'Apply', 'Cancel', 'Delete', and 'Undo', along with links for 'Export', 'View Log', 'View Status', and 'Help'. The main area contains fields for 'Administrative State' (set to 'enabled') and 'Comments' (an empty text input). A 'Message Matchings' section displays a single entry: 'retrieveAllCounter-message-matching'. This entry has a delete icon (X) and a toolbar with icons for edit (pencil), add (plus), and more options (ellipsis). The footer of the dialog box includes a copyright notice: '© Copyright IBM Corporation 2013'.

Figure 14-7. Step 2: Message type configuration

WE4013.0

Notes:

Step 3: Message Filter Action configuration

What occurs when a threshold is exceeded:

- **Type:** specifies sanction type
 - **Notify:** adds a log entry
 - **Reject:** rejects over-threshold messages for the Block Interval, adds a log entry
 - **Shape:** over-threshold messages are buffered and processed at a rate that does not exceed the threshold, adds a log entry
 - Buffer-overflow messages are rejected
- **Log priority**
 - Log messages that are generated when threshold values are exceeded are written at this priority

The screenshot shows the 'Configure Message Filter Action' dialog box. At the top, it says 'Configure Message Filter Action' and has tabs for 'Main' and 'Advanced'. Below that, it shows 'Message Filter Action: retrieveAllCounter-monitor-action [up]'. There are buttons for 'Apply', 'Cancel', 'Delete', and 'Undo'. Under 'Administrative State', there are radio buttons for 'enabled' (selected) and 'disabled'. The 'Comments' field is empty. The 'Type' field is set to 'Reject *' (highlighted with a red box). The 'Log Priority' field is set to 'emergency' (highlighted with a red box). The 'Block Interval' field contains the value '60000'.

© Copyright IBM Corporation 2013

Figure 14-8. Step 3: Message Filter Action configuration

WE4013.0

Notes:

The sanction type of **reject** is also referred to as “throttle.”

Block interval: Meaningful only when the type is **reject** or **shape**. It specifies the duration of service denial. The default value of zero indicates that while over-threshold messages are dropped, no service denial penalty is imposed.

Log targets do not include messages that a monitor generates unless the log target is configured with an event subscription class of **monitor** or **all**. Also, the log priority must be equal to or lower than the value set on this page.

Step 4C: Message Count Monitor configuration

- Select OBJECTS > Monitoring > Message Count Monitor

- Message Type:**

- Use the values list to select the message type that is monitored by the current message count monitor

- Measure:**

- Use the values list to specify the event that increments the message count monitor
- Requests:** receipt of a client request of the monitored message type
- Responses:** receipt of a server response of the monitored message type
- XPath:** a style sheet programmatically increments the counter
- Errors:** receipt of an HTTP error response, or execution of error rule

The screenshot shows the 'Configure Message Count Monitor' dialog box. At the top, there are tabs for 'Main' and 'Thresholds/Filters'. Below the tabs, the title is 'Message Count Monitor: retrieveAllCounter [up]'. There are several configuration fields:

- Administrative State:** A radio button group with 'enabled' selected.
- Comments:** An input field containing a placeholder.
- Message Type:** A dropdown menu set to 'retrieveAllCounter-message-type' with a '+' button and a '...' button.
- Measure:** A dropdown menu set to 'XPath' with a '*' validation character.
- Source:** A dropdown menu set to 'All' with a '*' validation character.
- Header:** An input field containing 'X-Client-IP'.
- Maximum Distinct Sources:** An input field containing '10000' with a '*' validation character.

 A red box highlights the 'Message Type' dropdown. At the bottom right, there is a copyright notice: '© Copyright IBM Corporation 2013'.

Figure 14-9. Step 4C: Message Count Monitor configuration

WE4013.0

Notes:

Source: This message count monitor aggregates IP address information. This property is meaningful only if the monitored message type contains inclusive or exclusive IP address criteria.

- All:** IP address information is gathered and reported for the aggregate address range.
- Each IP:** IP address information is gathered and reported for individual IP addresses (up to a maximum of 10000) within the address range.
- IP from Header:** Specifies that source address monitoring and information gathering is individualized for all IP addresses within the address range. The value of the HTTP header that is specified in the **Header** field determines the IP address.

Header: The name of the HTTP header that is read to determine the value of the source IP address.

For the **Measure** choice of **XPath**, the `dp:increment-integer` extension function increments the counter.

Step 4C: Thresholds/Filters for count monitor

- In the “Configure Message Count Monitor” page, select the **Thresholds/Filters** tab
 - Specify the threshold value that triggers a **Filter** action
 - Select the **Filter** action to execute upon exceeding that threshold
- Interval:** the measurement interval in milliseconds
 - Interval works with rate limit and burst limit to define the conditions that activate a **Filter** action
- Rate Limit:** threshold that is expressed in number of messages
- Burst Limit:** the allowed burst value
- Action:** select the **Filter** action to trigger when the monitored message type exceeds threshold values
- Multiple threshold or filter settings can be specified

Edit Thresholds/Filters [Help](#)

| | |
|-------------|--|
| Name | <input type="text" value="retrieveAllCounter-count-monitor-f"/> * |
| Interval | <input type="text" value="10000"/> msec * |
| Rate Limit | <input type="text" value="2"/> messages * |
| Burst Limit | <input type="text" value="4"/> messages * |
| Action | <input type="text" value="retrieveAllCounter-monitor-action"/> + ... * |

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

© Copyright IBM Corporation 2013

Figure 14-10. Step 4C: Thresholds/Filters for count monitor

WE4013.0

Notes:

The following example imposes a **Filter** action when the monitored message type exceeds 50 transactions per second, but also allows a short-term burst of up to 100 transactions per second:

```
< interval = 1000, rate limit = 50, burst limit = 100 >
```



Step 4D: Message Duration Monitor configuration

- Select Objects > Monitoring > Message Duration Monitor
- **Message Type**
 - Use the values list to select the message type monitored
- **Measure:** the values list to identify which portion of the transaction cycle to measure
 - **Messages**
 - **Responses**
 - **Requests**
 - **Server**

The screenshot shows the 'Main' tab of a configuration dialog for a 'Message Duration Monitor'. The monitor is named 'retrieveAll10 [up]'. The 'Administrative State' is set to 'enabled'. The 'Comments' field is empty. The 'Message Type' dropdown is set to 'retrieveAllCounter-message-type' with a '+' button and an asterisk (*) indicating it's a required field. The 'Measure' dropdown is set to 'Messages' with a dropdown arrow and an asterisk (*) indicating it's a required field. At the bottom are 'Apply', 'Cancel', 'Delete', and 'Undo' buttons.

© Copyright IBM Corporation 2013

Figure 14-11. Step 4D: Message Duration Monitor configuration

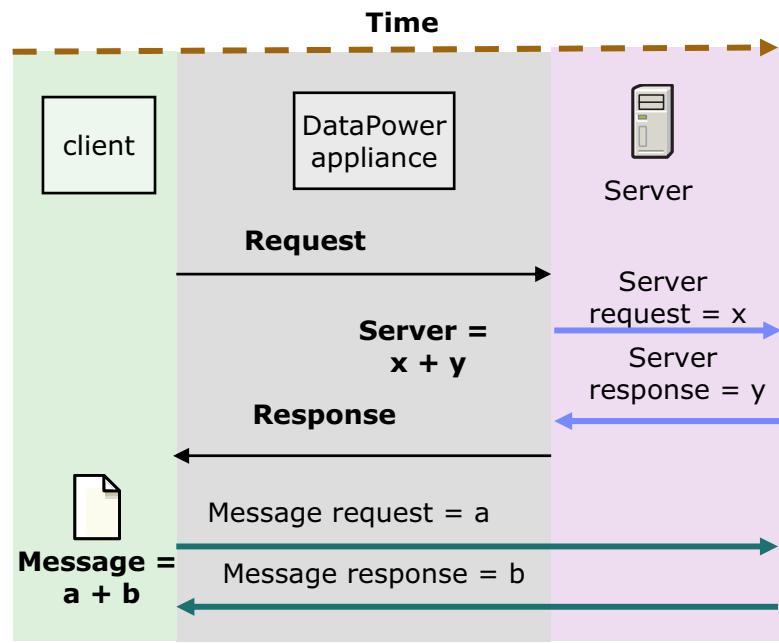
WE4013.0

Notes:

The **Measure** value list items are defined in the next slide.

Step 4D: The transaction lifecycle

- Request
 - Time that is required to service requests by the appliance
 - Scope: internal
- Response
 - Time that is required to process responses from the server
 - Scope: internal
- Server
 - Time that the back-end server requires to process requests
 - Scope: external
- Messages
 - Time that is required to process a client request, including the back-end server processing time
 - Scope: internal and external



© Copyright IBM Corporation 2013

Figure 14-12. Step 4D: The transaction life cycle

WE4013.0

Notes:

Each item represents a portion of the client request to server response “round trip” timeline. The **requests** and **response** values are internal to the appliance. The **requests** measure the time that is required to do filtering, cryptography, and transformation on client requests. The **responses** measure the time that is required to do similar processing on the server responses.

The **server** and **messages** values deal with external processing, that is, the processing that the back-end servers perform. **Server** measures the actual server processing time (plus network latency), while **messages approximate** the sum of requests, servers, and responses.



Step 4D: Thresholds/Filters for duration monitor

- In the “Configure Message Duration Monitor” page, select the **Thresholds/Filters** tab
 - Specify the threshold value that triggers a **Filter** action
 - Select the **Filter** action to execute upon exceeding that threshold
- **Value:** response time threshold value (in milliseconds)
 - Works with the **Measure** field
- Example:
 - Measure = messages
 - Value = 100
 - If time from receipt of request until transmission of response exceeds 100 ms, counter is incremented
- **Action:** select the **Filter** action to trigger when the monitored message type exceeds threshold values
- Multiple threshold/filter settings can be specified

© Copyright IBM Corporation 2013

Figure 14-13. Step 4D: Thresholds/Filters for duration monitor

WE4013.0

Notes:

< Measure = messages, Value = 100 >

Impose administrative sanctions when the **average** interval between the receipt of a client request and the transmission of the associated server response back to the client exceeds 100 milliseconds (the **Value** field). (In this rule, the associated server response refers to the “messages” **Measure** field on the **Main** tab.)

Step 5: Service-monitor association example

- Both message count monitors and message duration monitors can be associated to a service
- Previously defined monitors, which you can add, are in the drop-down list
- You can create new monitors by clicking the plus sign (+) button

The screenshot shows a configuration interface for a Multi-Protocol Gateway. At the top, there are tabs: General, Advanced, Stylesheet Params, Headers, and Monitors. The Monitors tab is selected and highlighted with a red border. Below the tabs are buttons for Apply, Cancel, and Delete, and links for Export, View Log, and View Status. A status message says "Multi-Protocol Gateway status: [up]". The main area is titled "Monitors" and contains three sections: "Message Count Monitor" (empty), "Message Duration Monitor" (empty), and "Gateway Service Level Monitors" (empty). Each section has a "retrieveAll-Counter" dropdown, an "Add" button, a plus sign (+) button, and a "...". Below these sections is a "Monitors Evaluation Method" section with a dropdown set to "Terminate at First Throttle".

© Copyright IBM Corporation 2013

Figure 14-14. Step 5: Service-monitor association example

WE4013.0

Notes:

The figure assumes that a message count monitor called **Count100** exists.

There is one more type of monitor listed on this page: service level monitor. This monitor is actually the older web service monitor (mentioned on the next page), not the newer service level monitor.

Other types of monitors

- Service level monitor (SLM)
 - WSDL-based, operation-specific (in web service proxy)
 - Can customize to specific requested resources and requesters
 - Can apply different thresholds at different times
 - Monitors counts, errors, and time
 - Actions are **Log**, **Reject**, and **Shape**
- Web services monitor
 - WSDL-based, identifies a specific endpoint
 - Monitors front-end errors and transaction rates
 - Actions are **Log** and **Reject**

© Copyright IBM Corporation 2013

Figure 14-15. Other types of monitors

WE4013.0

Notes:

SLM provides the finest-grained control.

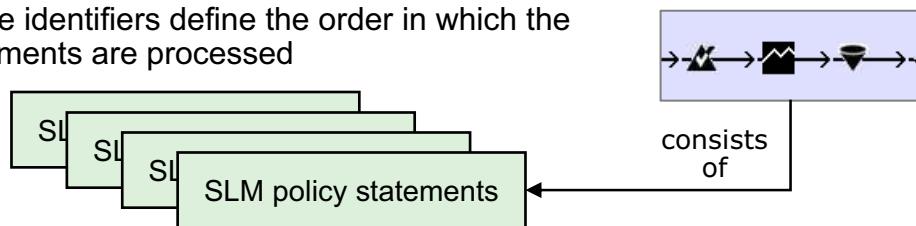
SLM is covered in a later unit.

The web services monitor requires that statistics be enabled on the appliance. It is called “gateway service level monitor” in a multi-protocol gateway.

Other types of monitors: Service level monitors

Service level monitor (SLM)

- SLM policy actions offer the most fine-grained and sophisticated approach for setting up SLM in the DataPower device
 - Currently these actions can be used in the web service proxy and the multiprotocol gateway services
- An SLM action consists of one to many policy statements
 - Every statement has a numerical identifier
 - These identifiers define the order in which the statements are processed



A basic property of an SLM policy is the mode in which the contained policy statements should be handled

- The following modes are available:
 - Processing of all statements of a policy
 - Processing of statements until the first action (notify, shape, reject) is taken
 - Processing of statements until a reject occurs

© Copyright IBM Corporation 2013

Figure 14-16. Other types of monitors: Service level monitors

WE4013.0

Notes:

SLM provides the finest-grained control.

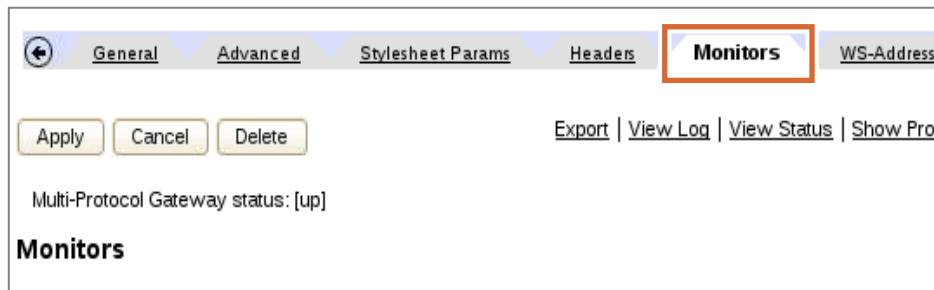
A later unit covers SLM in more detail.

Web services monitor requires that statistics be enabled on the appliance. It is called “gateway service level monitor” in a multi-protocol gateway.



Other types of monitors: Web services monitors

- Web services monitor
 - Web service monitors are configured by providing the URL of a Web Services Definition Language (WSDL)
 - The main difference to message monitors is that a web service endpoint to monitor can be specified
 - Web service monitors are also configured on the Monitor tab of the DataPower WebGUI with basic and simple configuration options



- Contrary to web service monitors, SLM policy actions are a more sophisticated and flexible way of monitoring web services

© Copyright IBM Corporation 2013

Figure 14-17. Other types of monitors: Web services monitors

WE4013.0

Notes:

SLM provides the finest-grained control.

SLM is covered in a later unit.

Web services monitor requires that statistics be enabled on the appliance. It is called “gateway service level monitor” in a multi-protocol gateway.

Contrary to web service monitors, SLM policy actions are a more sophisticated and flexible way of monitoring web services (see IBM Red piece: *IBM WebSphere DataPower SOA Appliances Part IV: Management and Governance*, page 71: <http://www.redbooks.ibm.com/abstracts/redp4366.html?Open>).

Monitor types that are supported by a service

- The types of monitors you can associate with a service depend on what type of service it is

| Service type | Message monitors | Service level monitors | Web service monitors | Notes |
|--------------------------|------------------|------------------------|----------------------|---|
| XSL proxy | ✓ | | | |
| XML firewall | ✓ | | ✓ | |
| Web service proxy | ✓ | ✓ | | (Msg) Monitor tab in OBJECTS view |
| Multi-protocol gateway | ✓ | ✓ | ✓ | |
| Web application firewall | ✓ | | | Message count monitor as part of the error policy |

© Copyright IBM Corporation 2013

Figure 14-18. Monitor types that are supported by a service

WE4013.0

Notes:

The DataPower device has the following types of monitors:

- Message monitors (count monitors, duration monitors)
- Web service monitors
- SLM policy action

Message monitors and web service monitors are applied directly to the DataPower service object. When creating DataPower services, message monitors and web service monitors are added by using the Monitors tab available when configuring the object. Service level monitor (SLM) policies are configured as an action that is part of a DataPower services processing policy.

Note: In the DataPower WebGUI, you can see the full set of options that are available for a DataPower service. These options are available only when you access the service that is using the objects menu in the left navigation bar of the DataPower interface.

Service level monitors are implemented as a policy. The policy is specified by using an SLM action in a processing policy. SLM actions are supported in the WS Proxy and multi-protocol gateway policy editors.

The WS Proxy has an **SLM** tab on the main edit page to help in coding the SLM policy. To specify message monitors, you must go the OBJECTS view of the proxy to find the Monitor tab.

Web application firewalls support a message count monitor only as part of its error policy.



Unit summary

Having completed this unit, you should be able to:

- Identify messages that should be monitored
- Configure a message count monitor
- Set up a message duration monitor

© Copyright IBM Corporation 2013

Figure 14-19. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. What are the two types of message monitors?
 - A. Message count monitor
 - B. Message duration monitor
 - C. Message size monitor
 - D. Message payload monitor

2. Select the proper **Message Filter** action definition:
 - A. **Message Filter** action: A **Filter** action filters the message payload to the dev/null bucket.
 - B. **Message Filter** action: A **Filter** action specifies the administrative actions that are taken in response to the overuse of system or network resources by a monitored message type.
 - C. **Message Filter** action: A **Message Filter** action strips off the message header as defined in the action header section. A check sum is then added to the end of the message with the specific **Message Filter** hash algorithm.

© Copyright IBM Corporation 2013

Figure 14-20. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

1.

2.

Checkpoint answers

1. A and B. What are the two types of message monitors?
 - ✓ A. Message count monitor
 - ✓ B. Message duration monitor
 - C. Message size monitor
 - D. Message payload monitor

2. B. Select the proper **Message Filter** action definition:
 - A. **Message Filter** action: A **Filter** action filters the message payload to the dev/null bucket.
 - ✓ B. **Message Filter** action: A **Filter** action specifies the administrative actions that are taken in response to the overuse of system or network resources by a monitored message type.
 - C. **Message Filter** action: A **Message Filter** action strips off the message header as defined in the action header section. A check sum is then added to the end of the message with the specific **Message Filter** hash algorithm.

© Copyright IBM Corporation 2013

Figure 14-21. Checkpoint answers

WE4013.0

Notes:

Exercise 12



Creating message counter monitors
for a AAA policy

© Copyright IBM Corporation 2013

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

7.0.1

Figure 14-22. Exercise 12

WE4013.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Create message count monitors
- Specify message count monitors as the “authorized counter” and the “rejected counter” in a AAA policy

© Copyright IBM Corporation 2013

Figure 14-23. Exercise objectives

WE4013.0

Notes:

Unit 15. Configuring LDAP by using AAA

What this unit is about

This unit describes how to authenticate and authorize users by using LDAP in a AAA policy. You learn basic LDAP concepts and constructs. You also learn how to configure LDAP in a AAA policy to connect to a directory service.

What you should be able to do

After completing this unit, you should be able to:

- Describe the fundamentals of configuring the Lightweight Directory Access Protocol (LDAP) and deploying directory services
- Authenticate and authorize user credentials by using LDAP from a AAA policy

How you will check your progress

- Checkpoint
- Exercise 13: Creating a AAA policy by using LDAP

Unit objectives

After completing this unit, you should be able to:

- Describe the fundamentals of configuring the Lightweight Directory Access Protocol (LDAP) and deploying directory services
- Authenticate and authorize user credentials by using LDAP from a AAA policy

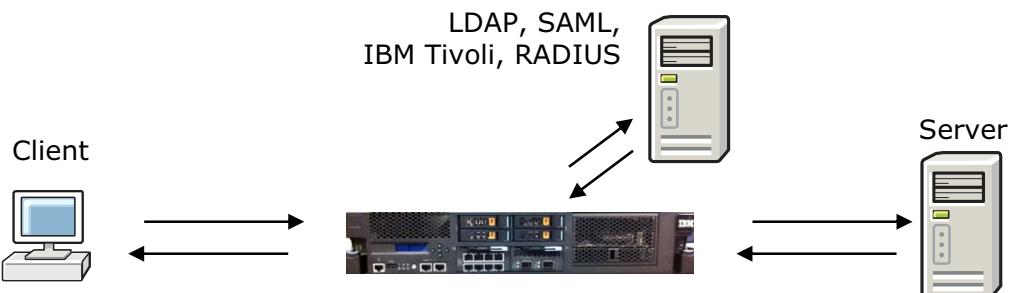
© Copyright IBM Corporation 2013

Figure 15-1. Unit objectives

WE4013.0

Notes:

External access control resource



- Delegate the authentication and authorization task to an external security system
- The authentication and authorization tasks can be delegated to the same system or to separate systems
 - For example, an LDAP directory tracks client identities while IBM Tivoli Access Manager determines whether the client has access to the specified resource
 - The **map credentials** and **map resource** steps convert the security token to match the input that the authorization step requires

© Copyright IBM Corporation 2013

Figure 15-2. External access control resource

WE4013.0

Notes:

It is also possible to authenticate and authorize on an IBM Tivoli Access Manager system.

The list of external access controls on this slide is merely an example. Consult the WebGUI guide for a full list of security products and specifications that are supported.

You can also use RACF on a z/OS system. The NSS client object in DataPower enables integration with RACF on the z/OS Communications Server.

For OAuth, the underlying AAA policy might use external resources for authentication and authorization.

Lightweight Directory Access Protocol

- LDAP is a networking protocol for communicating with directory services over TCP/IP
 - The LDAP protocol allows the storage and retrieval of information about people, groups, or objects from a centralized X.500 directory server
 - Based on the client/server model of computing
 - **X.500** enables information to be organized and queried, by using LDAP, from multiple web servers by using various attributes
 - **LDAP** reduces system resources by including only a functional subset of the original X.500 Directory Access Protocol (DAP)

© Copyright IBM Corporation 2013

Figure 15-3. Lightweight Directory Access Protocol

WE4013.0

Notes:

LDAP is an optimized data repository that is targeted towards applications that require many read operations.

- All modern LDAP implementations are V3.
- LDAP V3 is defined in RFC 2251.
- LDAP is a binary protocol.

Directory services

- The telecommunications industry drove the creation of a directory services standard, X.500
 - The X.500 standard organizes data in a hierarchical manner
 - Accessing an X.500 directory requires the full OSI protocol layer stack
- LDAP was created as a lightweight alternative for accessing the X.500 standard
 - Requires only TCP/IP instead of the entire OSI protocol layer stack

© Copyright IBM Corporation 2013

Figure 15-4. Directory services

WE4013.0

Notes:

The past experiences of the telecommunications industry in managing phone directories helped to develop the X.500 standard.

The LDAP standard does not define the directory format. It refers to the LDAP protocol, which defines how to obtain information from a directory service.

Directories

- Information about users, groups, and objects is collected into a special database that is called a **directory**
 - A directory contains entries
 - Entries are listed in order that is based on a key
 - Each entry describes an object or contains a list of attributes
- Example: telephone book
 - Entries are phone numbers that are ordered according to name
 - Each person entry contains other information, such as the address
- Storing information about objects allows for greater flexibility
 - An object stores typed information about itself
 - Example: server = object
 - The *server* entry contains information about its host address, port, building location, and more
 - Use case: execute a query to search for servers in a specific building

© Copyright IBM Corporation 2013

Figure 15-5. Directories

WE4013.0

Notes:

The concept of a directory is similar to that of a phone book, which allows you to search by names or businesses.

An **object class** can be defined to specify typed attributes and relationships with other object classes.

Objects can contain subclasses, which is similar to the concept of inheritance in object-oriented design. The root object in every directory service is called **top** with an attribute **objectClass**.

Every attribute has a type that stores one or more values.

Attributes can have behaviors that are associated with them when they are searched. For example, when searching for telephone numbers, hyphens are ignored.

Common LDAP attributes

| Attribute, alias | Attribute syntax | Description | Example |
|----------------------------|------------------|-----------------------------------|---------------------------|
| commonName, cn | cis | Common name of an entry | John |
| surname, sn | cis | Surname (family name) of a person | Smith |
| TelephoneNumber | tel | Telephone number | 555-555-1111 |
| organizationalUnitName, ou | cis | Name of organizational unit | WebSphere |
| owner | dn | DN of person who owns the entry | cn=John Smith, o=IBM,c=ca |
| organization, o | cis | Name of organization | IBM |
| jpegPhoto | bin | Photographic image in JPEG format | Photograph of John Smith |

© Copyright IBM Corporation 2013

Figure 15-6. Common LDAP attributes

WE4013.0

Notes:

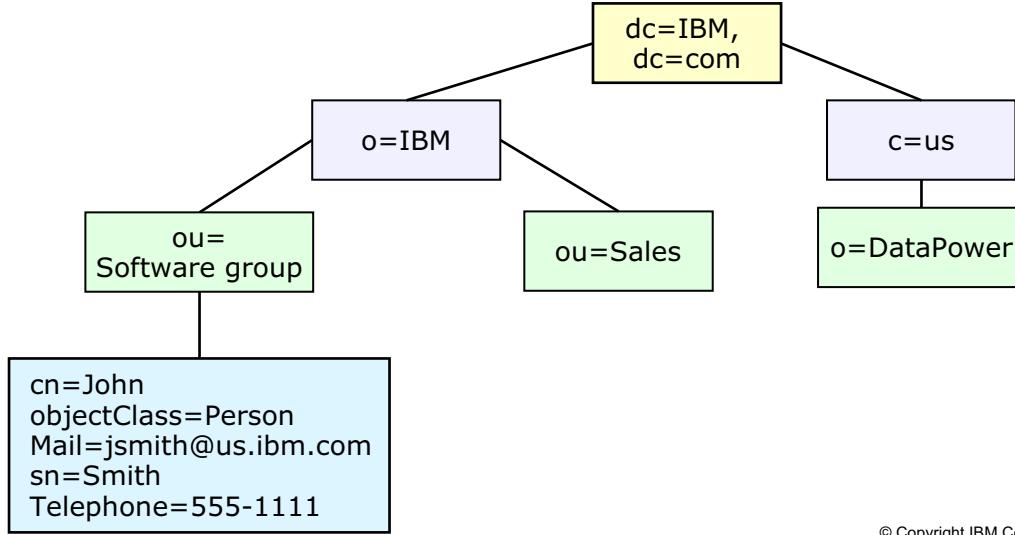
These attributes are used when creating an LDAP query URL. They are also attributes associated to directory entries stored in a directory service.

The attribute syntax can specify:

- **cis**: case-ignore string; not case-sensitive
- **tel**: telephone number; spaces and dash are ignored
- **dn**: distinguished name
- **bin**: binary information

Directory services structure

- The directory service organizes a collection of objects (directory entries) in a tree structure that is called the **directory information tree (DIT)**
 - The **distinguished name (DN)** defines the key of the entry
 - Each node in the tree is called a **relative distinguished name (RDN)**
 - A **DN** consists of a set of **RDNs**



© Copyright IBM Corporation 2013

Figure 15-7. Directory services structure

WE4013.0

Notes:

Several methods are available for organizing directories.

For example, directories can be organized according to organization and then split into different organizational units, or directories can be organized according to country and split into the various organizations.

LDAP operations

- Each LDAP message contains the operation (bind, update, delete) that the client requested
- Authentication:
 - A bind operation authenticates the client by sending the client's distinguished name and password in cleartext
 - Uses an SSL connection to keep LDAP queries secret
 - An anonymous bind resets the connection to an anonymous state
 - Default access rights
- Search
 - Search: specifies criteria that are used to return matching entries
 - Compare: uses the DN and attribute name-value pairs to check whether the DN entry contains name-value pairs for that attribute
- Update consists of add, delete, and modify operations
 - Add: inserts new entries into the directory
 - Delete: removes only leaf nodes from the directory
 - Modify: add, update, or remove attributes or attribute values for an entry

© Copyright IBM Corporation 2013

Figure 15-8. LDAP operations

WE4013.0

Notes:

The default port for LDAP is 389, and 636 is for LDAP over SSL.

There is no password, and DN is required for an anonymous bind.

DataPower LDAP load balancer groups can use anonymous binds to check the health of an LDAP server.

The search operation contains many parameters to specify the criteria for searching entries in a directory service. The most often used parameters are:

- baseObject: DN of the entry to start the search
- scope: select either baseObject (named entry), singleLevel (children of named entry), or sub (entry subtree that starts at the base DN)
- filter: criterion for select entry by using attribute name-value pairs

The **modifyDN** is another update operation that allows you to move a subtree of entries into a new location.



LDAP URL

- The format for an LDAP URL is: `ldap://<host>:<port>/<path>` where `<path>` has the form
`<dn>[?<attributes>[?<scope>?<filter>]]`
 - `ldap://` is the protocol
 - `host` and `port` represent the LDAP server host address and port number
 - `<dn>` represents the distinguished name to search
 - `attributes` is a comma-separated list of attributes
 - `scope` defines where and what objects to return
 - `filter` specifies the object class by using the `objectClass` attribute
- Example:
`ldap://example.com:389/cn=John Smith?middleName`
 - Returns the *middle name* attribute entry for *John Smith*

© Copyright IBM Corporation 2013

Figure 15-9. LDAP URL

WE4013.0

Notes:

- If no port number is specified in the LDAP URL, the standard LDAP port number (389) is used.
- If no attributes are specified, the search returns all attributes.
- If no search scope is specified, the search is restricted to the base entry.
- If no filter is specified, the directory uses the default filter (`objectclass=*`).

Directory services implementations

- IBM Tivoli Directory Server:

<http://www.ibm.com/software/tivoli/products/directory-server/>

- IBM LDAP V3 implementation
- Uses IBM DB2 as the back-end for LDAP transaction integrity, performance, and backup and restore
- Supports clustering of LDAP servers

- IBM Lotus Domino:

<http://www.ibm.com/software/lotus/products/domino/>

- Messaging and collaboration system
- Domino server directory support uses LDAP

- IBM Tivoli Directory Integrator:

<http://www.ibm.com/software/tivoli/products/directory-integrator/>

- Metadata synchronization product
- Handles LDAP transactions from LDAP clients

- OpenLDAP: <http://www.openldap.org>

- Open source LDAP V3 directory server
- Stand-alone LDAP server and replication server

© Copyright IBM Corporation 2013

Figure 15-10. Directory services implementations

WE4013.0

Notes:

Example scenario

- A client sends an HTTP message that uses basic authentication to the XML firewall, which uses LDAP to authenticate the identity

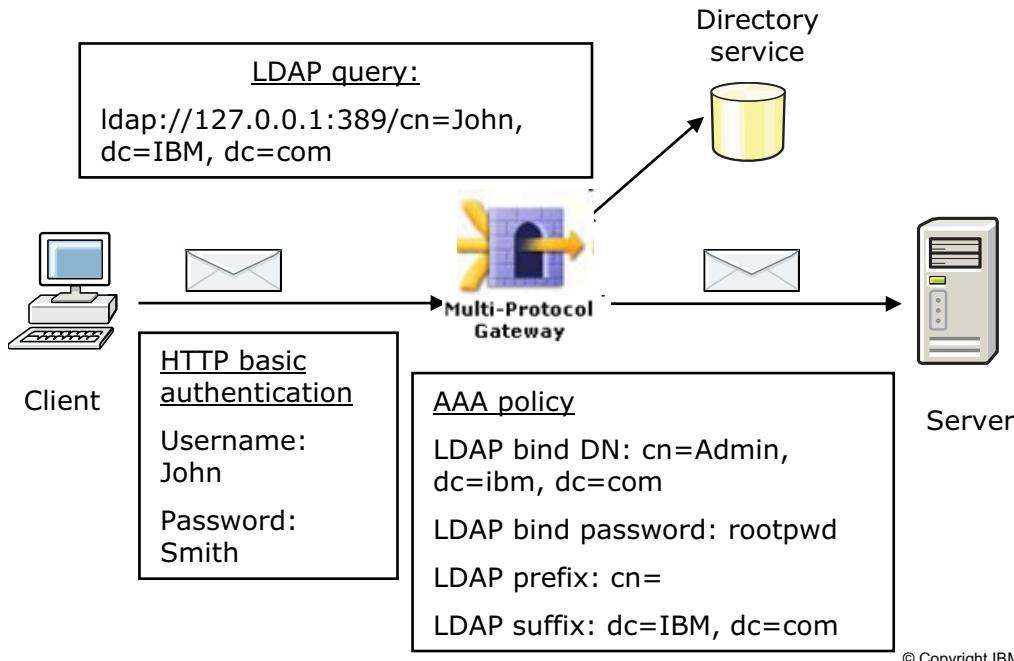


Figure 15-11. Example scenario

WE4013.0

Notes:

The LDAP bind DN and password are used to authenticate into the directory service. As soon as the client is authenticated, the user name and password that were sent as HTTP basic authentication are used to build the LDAP query.

The password that the client sent through HTTP basic authentication is not part of the LDAP query, but it is still passed in the request. In the directory service, the **userPassword** field is used to authenticate the user within the RDN entry.

The user name part of the HTTP basic authentication message is substituted into the common name for the LDAP query.

Use LDAP to authenticate the client

1. Set Bind to Specified LDAP Server as the authentication method
2. Bind to the LDAP server specified in the Host and Port settings or the Load Balancer Group
3. Set the Bind DN and Bind Password for an LDAP query
4. Use the Search Attribute fields to verify the password digest from a WS-Security Username Token
5. Use the Prefix and 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**

| Define how to authenticate the user. | |
|--------------------------------------|--|
| 1 | Method LDAP Load Balancer Group <input type="radio"/> Accept an LTPA token <input checked="" type="radio"/> Bind to Specified LDAP Server (none) <input type="button" value="+"/> <input type="button" value="..."/> |
| 2 | Host 389 |
| 3 | SSL Proxy Profile (none) <input type="button" value="+"/> <input type="button" value="..."/> |
| 4 | LDAP Bind DN LDAP Bind Password LDAP Search Attribute userPassword LDAP Version v2 LDAP Search for DN <input checked="" type="radio"/> on <input type="radio"/> off LDAP Prefix cn= LDAP Suffix dc=ibm,dc=com |
| 5 | |

© Copyright IBM Corporation 2013

Figure 15-12. Use LDAP to authenticate the client

WE4013.0

Notes:

For the **Bind to Specified LDAP Server** authentication method, specify the LDAP prefix and suffix that are used for the LDAP query. The prefix, extracted identity, and suffix together form the distinguished name that is used in the search.

The LDAP suffix value that is used in the slide uses a naming scheme that is based on domain names with dc attributes. For example, the domain **ibm.com** is represented as **dc=ibm, dc=com**.

To access a cluster of LDAP servers, create a load balancer group with the address for each LDAP server. For an individual LDAP server, specify the host address and port number.

In either scenario, send the query over an SSL connection by configuring an SSL proxy profile.

Set the **LDAP Bind DN** and **LDAP Bind Password** that are used in the bind operation (authentication) to the LDAP server.

If the client uses a WS-Security username token and a password digest, the system cannot confirm whether the password is valid by using only a simple LDAP query. The system must retrieve the password field that is stored within the LDAP entry. In this scenario, set the **LDAP Search Attribute** to: userPassword

The directory service includes an entry with an attribute called **userPassword** that contains the user password.

Use LDAP to authorize the client

1. Bind to the LDAP server specified in the **Host** and **Port** settings
2. Select or create an **SSL Proxy Profile**
3. Specify the **Group DN** of which the identity is a member
4. Set the **Bind DN**, **Bind Password** for an LDAP query
5. Use the **Load Balancer Group** to specify a cluster of LDAP servers
6. The **LDAP Group Attribute** is a string used to check for membership in the **Group DN** of the identity
7. The **LDAP Search Scope** and **LDAP Search Filter** are used to refine the search in an LDAP query

| | |
|-----------------------------------|------------------------|
| 1 Host | example.com |
| 2 Port | 389 |
| 3 SSL Proxy Profile | (none) |
| 4 Group DN | cn=grpDP,dc=ibm,dc=com |
| 5 LDAP Bind DN | |
| 6 LDAP Bind Password | |
| 7 LDAP Load Balancer Group | |
| LDAP Group Attribute | member |
| LDAP Version | v3 |
| LDAP Search Scope | subtree |
| LDAP Search Filter | (objectClass=*) |

© Copyright IBM Corporation 2013

Figure 15-13. Use LDAP to authorize the client

WE4013.0

Notes:

The identity that is checked for membership is the DN that is specified in the LDAP authenticate step (LDAP prefix and suffix).

The **LDAP Group Attribute** member is one of the default attribute types in an RDN entry.

The **LDAP Search Scope** specifies the depth of the search by using:

- **Base**: The search matches only the input itself.
- **One-level**: The search matches the search input and any object one level below.
- **Subtree**: The default. It specifies that the search matches the input and any descendants.

The **objectClass** parameter is the `<filter>` part of the LDAP URL. If no filter is specified, the default (`objectClass=*`) is used to return all entries by the search.

Unit summary

Having completed this unit, you should be able to:

- Describe the fundamentals of configuring the Lightweight Directory Access Protocol (LDAP) and deploying directory services
- Authenticate and authorize user credentials by using LDAP from a AAA policy

© Copyright IBM Corporation 2013

Figure 15-14. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. True or False: In LDAP, a collection of objects is organized in a directory structure.
2. True or False: In the AAA policy, the LDAP bind DN and LDAP bind password are configured.
3. True or False: LDAP is a choice for only the authentication step in a AAA policy.

© Copyright IBM Corporation 2013

Figure 15-15. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. **True.** With LDAP, a collection of objects is organized in a directory structure.
2. **True.** In the AAA policy, the LDAP bind DN and LDAP bind password are configured.
3. **False.** LDAP is a choice for both the authentication step and the authorization step in a AAA policy.

© Copyright IBM Corporation 2013

Figure 15-16. Checkpoint answers

WE4013.0

Notes:

Exercise 13



Creating a AAA policy by using
LDAP

© Copyright IBM Corporation 2013
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0.1

Figure 15-17. Exercise 13

WE4013.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Add entries to the IBM Tivoli Directory Server LDAP server
- Authenticate and authorize users on an LDAP server by configuring a AAA policy

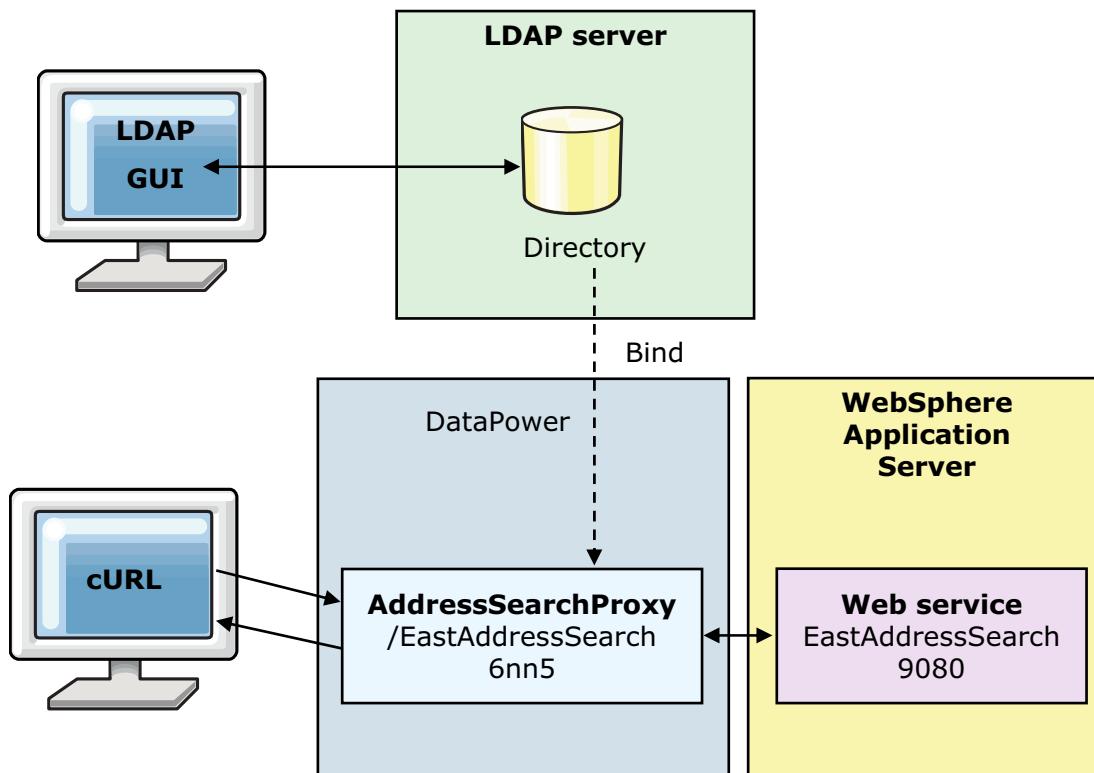
© Copyright IBM Corporation 2013

Figure 15-18. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure 15-19. Exercise overview

WE4013.0

Notes:

Unit 16. OAuth overview and DataPower implementation

What this unit is about

This unit introduces OAuth technology and its DataPower implementation.

What you should be able to do

After completing this unit, you should be able to:

- Describe what OAuth is
- Describe why OAuth is useful in security scenarios
- Describe the OAuth 2-legged and 3-legged design pattern
- Explain the role that a DataPower appliance performs in an OAuth architecture
- Describe the OAuth configuration options on DataPower: the Web Token Service, the AAA action, the OAuth client, and an OAuth group

How you will check your progress

- Checkpoint

References

<http://pic.dhe.ibm.com/infocenter/wsdatap/v5r0m0/index.jsp>

Unit objectives

After completing this unit, you should be able to:

- Describe what OAuth is
- Describe why OAuth is useful in security scenarios
- Describe the OAuth 2-legged and 3-legged design pattern
- Explain the role that a DataPower appliance performs in an OAuth architecture
- Describe the OAuth configuration options on DataPower: the Web Token Service, the AAA action, the OAuth Client, and an OAuth Group

© Copyright IBM Corporation 2013

Figure 16-1. Unit objectives

WE4013.0

Notes:

What is OAuth?

- **Open standard for authorization**
- Resource owners allow third-party access to the resource **without** sharing their credentials
- Two versions of OAuth, 1.0a and 2.0, are available
- Version 1.0a relies on cryptographic operation
 - Therefore, 1.0a does not need SSL/TSL for transport protection
 - However, the complexity is increased
- Version 2.0 is an IETF “standard” which relies heavily on SSL/TLS
 - The complexity that is required in version 1.0a is eliminated by using version 2.0
 - Version 2.0 is not finalized, but version 2.0 is stable
- There are two types of OAuth patterns, 2-legged and 3-legged
- 3-legged:
 - 3-legged OAuth is the traffic and data pattern that OAuth is designed to solve
 - [OAuth 2.0] grant type: authorization code, implicit grant
- 2-legged:
 - Traditional client <-> server pattern
 - [OAuth 2.0] grant type: resource owner password, client credential

© Copyright IBM Corporation 2013

Figure 16-2. What is OAuth?

WE4013.0

Notes:

OAuth allows users to share their private resources (for example, photos, videos, and contact lists) stored on one site with another site without having to hand out their credentials. They typically supply user name and password tokens instead. Each token grants access to a specific site (for example, a video editing site) for specific resources (for example, just videos from a specific album) and for a defined duration (for example, the next 2 hours). The process allows users to grant a third-party site access to their information stored with another service provider, without sharing their access permissions or the full extent of their data.

Why OAuth?

- OAuth provides a process that decreases the need to share user credentials with a third party
- OAuth provides a method that allows a third party to access a resource on behalf of the user

© Copyright IBM Corporation 2013

Figure 16-3. Why OAuth?

WE4013.0

Notes:

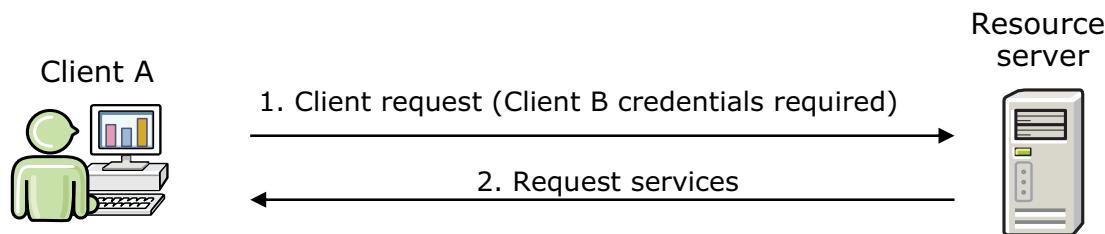
Why OAuth?

Many proprietary web authorization protocols have emerged over the years: Google's AuthSub, AOL's OpenAuth, Yahoo's BBAuth, Upcoming API, Amazon Web Services API, to name a few. OAuth integrates the commonalities and adopts the good practices of these other web authorization protocols into a single open standard.

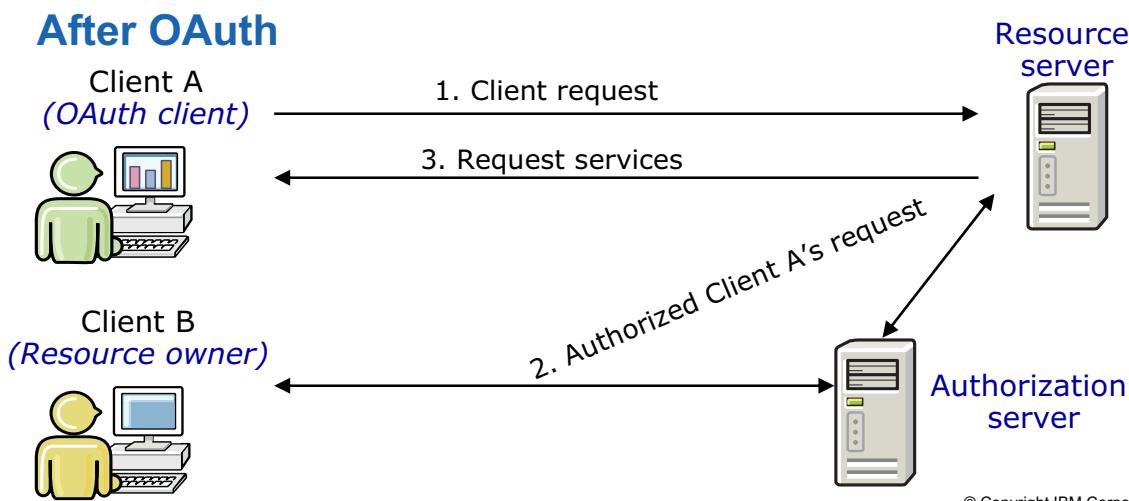
There are other reasons for using OAuth authorization:

- Compatible with existing authorization methods
- Flexibility to adjust to security needs of different sites
- Extensible through different signing algorithms
- Designed to work with mobile devices and desktop applications

Before OAuth



After OAuth



© Copyright IBM Corporation 2013

Figure 16-4. Before OAuth

WE4013.0

Notes:

Before OAuth, if Client A requested information from a server as Client B, Client A would require Client B's credentials. The process requires the sharing of credentials.

When using OAuth, Client B's credentials are housed within the authorization server. Therefore, Client A needs permission only to use Client B's credentials. Client A is granted permission without ever receiving the actual credentials.

The four actors in an OAuth security scenario include;

1. OAuth client
2. Resource owner
3. Authorization server
4. Resource server

3-legged OAuth is the pattern OAuth was designed for

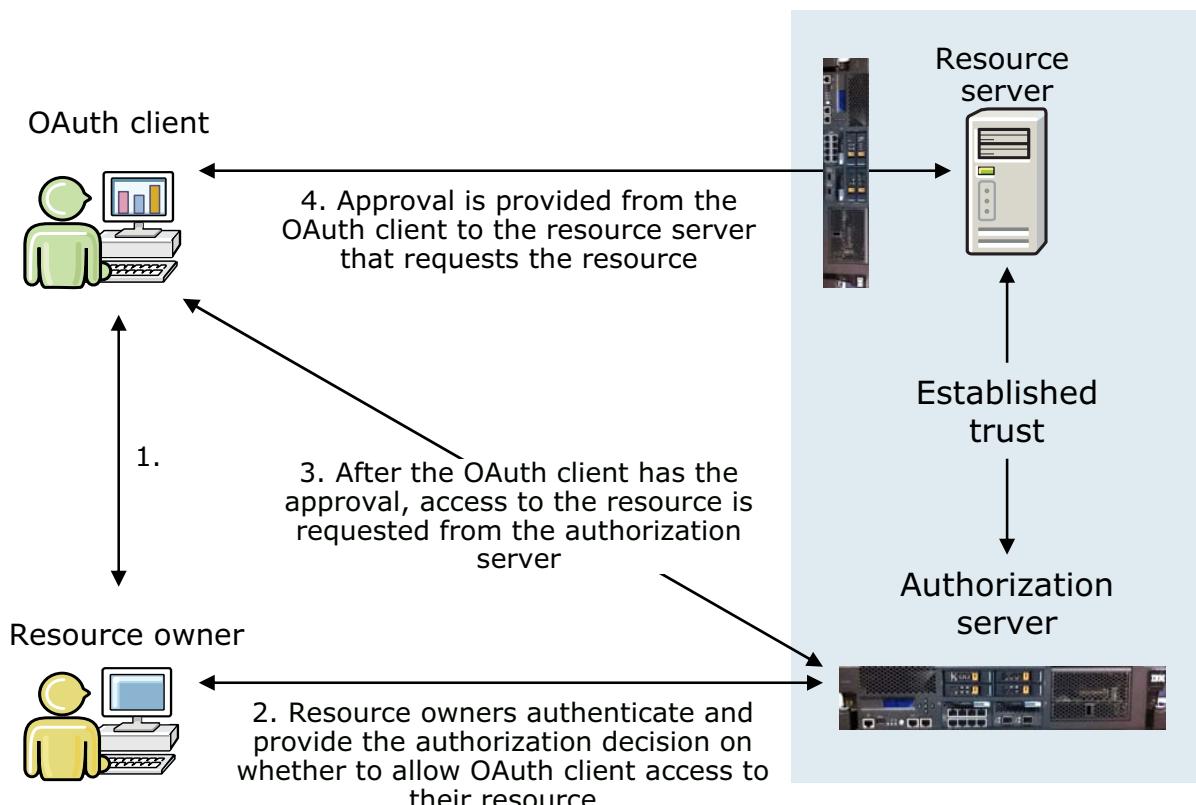


Figure 16-5. 3-legged OAuth is the pattern OAuth was designed for

WE4013.0

Notes:

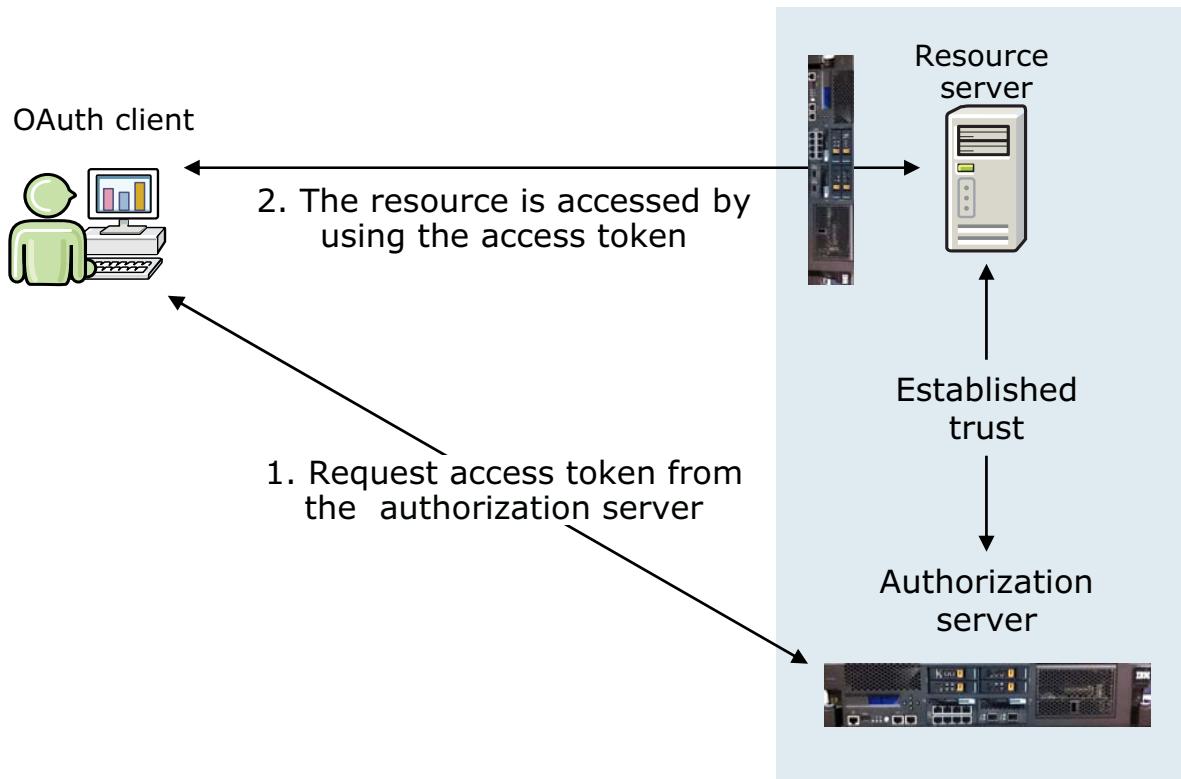
3-legged OAuth describes the scenario for which OAuth was originally developed: a resource owner wants to give a client access to a server without sharing the resource owner credentials (that is, user name and password). A typical example is a user (resource owner) who wants to give a third-party application (client) access to the resource owner's Twitter account (server).

On a conceptual level, it works in the following way:

- Client signed up to the server and got the client credentials (also known as “consumer key and secret”) ahead of time.
- User wants to give the client access to the user protected resources on the server.
- Client retrieves the temporary credentials (also known as “request token”) from the server.
- Client redirects the resource owner to the server.
- Resource owner grants the client access to the resource owner’s protected resources on the server.

- Server redirects the user back to the client.
- Client uses the temporary credentials to retrieve the token credentials (also known as “access token”) from the server.
- Client uses the token credentials to access the protected resources on the server.

2-legged OAuth is a traditional client/server pattern



© Copyright IBM Corporation 2013

Figure 16-6. 2-legged OAuth is a traditional client/server pattern

WE4013.0

Notes:

2-legged OAuth describes a typical client/server scenario, without any user involvement. An example for such a scenario might be a local Twitter client application that accesses your Twitter account.

On a conceptual level, 2-legged OAuth consists of the first and last steps of 3-legged OAuth:

- Client signed up to the server and got the client credentials (also known as “consumer key and secret”).
- Client uses the client credentials (and empty token credentials) to access the protected resources on the server.

OAuth support in DataPower (version 5.0+) (1 of 2)

- The role DataPower plays in OAuth
 - Authorization server (there is no back-end server or loopback): authorization endpoint, token endpoint
 - Enforcement point for resource server
- OAuth version 2.0 (draft 2-23)
- Token format: bearer
 - Self-contained
 - Token is cryptographically protected with shared secret (32 bytes)
- OAuth client that is supported:
 - confidential
 - client_secret
- Grant type that is supported:
 - Authorization code (3-legged)
 - Resource owner password credentials (2-legged)
 - Client credentials (2-legged)

© Copyright IBM Corporation 2013

Figure 16-7. OAuth support in DataPower (version 5.0+) (1 of 2)

WE4013.0

Notes:

DataPower participates in an OAuth configuration by being the authorization endpoint and the token endpoint. DataPower can also serve as the enforcement point for a resource server.

OAuth support in DataPower (version 5.0+) (2 of 2)

- Extension points for customization
- OAuth support that is specified in AAA policy
 - AAA policy is in either Web Token Service (authorization server endpoints) or multi-protocol gateway (authorization server endpoints and enforcement point for resource servers)
 - List of supported OAuth clients is configured in EI step of AAA
 - All remaining steps in AAA are related to the resource owner
- Support clustering in AO
- The appliance is **not** an OAuth client
- DataPower implementation of OAuth includes
 - Web Token Service
 - AAA policy
 - OAuth Client and OAuth Client Group

© Copyright IBM Corporation 2013

Figure 16-8. OAuth support in DataPower (version 5.0+) (2 of 2)

WE4013.0

Notes:

OAuth support for DataPower is tied to the AAA action. OAuth configuration occurs when a AAA is configured.

OAuth is supported in a clustered Application Optimization (AO) DataPower environment.

DataPower offers three components to implement OAuth. These three ways include:

- Web Token Service configuration
- AAA action configuration
- OAuth Client and OAuth Client Group configuration.



Web Token Service

- Utility service that provides authorization server endpoints (authorization and token)
 - Contains a AAA policy
- WTS can be accessed by going to
 - Objects > Service Configuration > Web Token Service**
 - Services > Web Token Service > Edit Web Token Service**

The screenshot shows the 'Control Panel' interface. On the left, there is a search bar and a navigation tree with the following structure:

- Status
- Services
 - XML Firewall
 - Web Service Proxy
 - Web Application Firewall
 - Web Token Service
 - Edit Web Token Service
 - New Web Token Service
- VSI Services

To the right, a modal window titled 'Create a Web Token Service' is open. It contains the following fields:

- Web Token Service Type:** OAuth 2.0
- Web Token Service Name:** OAuthTokenServcieTest*

© Copyright IBM Corporation 2013

Figure 16-9. Web Token Service

WE4013.0

Notes:

WTS configuration has three simple tabs: Main, Advanced, and Probe Settings.

It has support for HTTP and HTTPS front side handlers and allows configuration of a processing policy.

The WTS wizard eases the configuration of the WTS to be an OAuth authorization server.

It takes inputs and auto-creates the processing policy at the time of commit.

The processing policy contains the required actions (like http-convert) to configure an OAuth token service.

The administrator can still modify the processing policy that the wizard creates.

The WTS wizard eases the configuration of form-based login for ResourceOwner.



AAA: Extract identity and extract resource

The screenshot shows the 'Configure an Access Control Policy' interface for an 'OAuth' policy named 'OAuth_Test'. It highlights the 'Extract Identity' step.

OAuth Client Group:

- HTML Forms-based Authentication
- OAuth
- *
- (none)
- +
- ...

Buttons: Back, Next, Cancel

Resource Identification Methods:

- Local Name of Request Element
- HTTP Operation (GET/POST)
- XPath Expression
- Processing Metadata
- *

Processing Metadata Items:

- oauth-scope-metadata
- +
- ...

© Copyright IBM Corporation 2013

Figure 16-10. AAA: Extract identity and extract resource

WE4013.0

Notes:

To apply OAuth protocol support, check the OAuth box. An OAuth client group contains a list of the OAuth clients that this AAA policy supports.

Use the processing metadata as the resource to authorize the access, such as variables and protocol headers. One can define only the metadata items with this object to return. If this object is not selected for the Extract Resource method, all the metadata items applicable for the current processing rule are returned.

Configure OAuth Client Group

- To support the OAuth 2.0 protocol, a AAA policy requires the configuration of an OAuth client group
- An OAuth client group contains the configured OAuth clients from which the DataPower appliance accepts requests

The screenshot shows the 'OAuth Client Group' configuration dialog. At the top are 'Apply' and 'Cancel' buttons. Below is a 'Name' field with a required asterisk. Under 'Administrative State', 'enabled' is selected. The 'Comments' field is empty. The 'Customized OAuth' checkbox is unchecked. In the 'OAuth Role' section, both 'Authorization Server' and 'Resource Server' checkboxes are checked. The 'Client' section shows a list labeled '(empty)' with a 'Add' button and other controls.

© Copyright IBM Corporation 2013

Figure 16-11. Configure OAuth client group

WE4013.0

Notes:

To support the OAuth 2.0 protocol, a AAA policy requires the configuration of an OAuth client group. An OAuth client group contains the configured OAuth clients that the DataPower appliance accepts requests from.

When creating an OAuth client group for a AAA policy, the OAuth client group consists of one or more OAuth clients with the **same** OAuth roles.

The customized OAuth indicates whether the configuration is for a customized OAuth client group. This property is mutually exclusive to the OAuth Role property.

The OAuth Role identifies the roles of the clients in the group. This property is mutually exclusive to the Customized OAuth property.

If the Authorization Server check box is selected, the DataPower appliance acts as an authorization server.

If the Resource server check box is selected, the DataPower appliance acts as a resource server.

The client manages the group of OAuth clients. Use the controls to add or remove clients from the group.



Configure OAuth Client (1 of 2)

- An OAuth client object:
 - Represents the DataPower endpoint for an external OAuth client
 - Is the basic building block for an OAuth client group
- The following types of OAuth clients can be created:
 - Authorization server
 - Enforcement point for a resource server
 - Authorization server and an enforcement point for a resource server

OAuth Client

| | |
|------------------------------------|---|
| Name | <input type="text"/> |
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments | <input type="text"/> |
| Customized OAuth | <input type="checkbox"/> |
| OAuth Role | <input checked="" type="checkbox"/> Authorization Server <input checked="" type="checkbox"/> Resource Server * |
| Client Type | <input type="button" value="Confidential"/> |
| Supported Authorization Grant Type | <input checked="" type="checkbox"/> Authorization Code <input type="checkbox"/> Resource Owner Password Credential <input type="checkbox"/> Client Credentials * |

© Copyright IBM Corporation 2013

Figure 16-12. Configure OAuth Client (1 of 2)

WE4013.0

Notes:

An OAuth client is the basic building block for an OAuth client group. When you create an OAuth client, you define its role.

The following types of OAuth clients can be created:

- A client that acts as an authorization server
- A client that acts as an enforcement point for a resource server
- A client that acts as an authorization server and an enforcement point for a resource server

When creating an OAuth client, style sheets for customization can be used.

A fully customized OAuth client can be created. A customized OAuth client allows one to customize clients as an authorization server and, optionally, as the enforcement point for a resource server. To create a customized OAuth client as only the enforcement point for a resource server, create the client as only a resource server.

Customization uses style sheets. These style sheets must be in the `local:` or `store:` directory. The style sheets must define the following behaviors that depend on:

- How to verify requests; required when the client acts an authorization server
- For an authorization code grant type, how to issue and verify the authorization code; required when the client acts an authorization server
- For an access token, how to issue the access token; required when the client acts an authorization server
- For an access token, how to verify the access token; required when the client acts the enforcement point for a resource server

OAuth clients interact with protected resource in the following sequence:

1. The client requests access from the authorization server.
2. The authorization server grants access in the form of an access token.
3. The client presents the access token to access protected resources on the resource server.
4. The resource server provides access to the protected resource.

Configure OAuth Client (2 of 2)

- The Redirect URI defines the set of redirection URLs to exchange tokens for an authorization code
- By default, the resource owner is the user name from the extracted identity
- The Access Token Lifetime sets the lifetime for the access token in seconds

Generate Client Secret

Redirect URI *

Customized Scope Check

Scope *

Custom Resource Owner Handling

DataPower State Lifetime *

Authorization Grant Lifetime *

Access Token Lifetime *

Shared Secret *

Authorization Form

Additional OAuth Process

© Copyright IBM Corporation 2013

Figure 16-13. Configure OAuth Client (2 of 2)

WE4013.0

Notes:

The Generate Client Secret indicates whether to generate the client secret for the OAuth client. The specification refers to the client secret as `client_secret`.

- If you generate the passphrase, the passphrase becomes the client secret.
- If you do not generate the passphrase, you must explicitly define the client secret.

The Redirect URI defines the set of redirection URLs to exchange tokens for an authorization code. You define as many redirection URLs as the authorization process uses.

Redirection URLs help to detect malicious clients and prevent phishing attacks. The authorization server must have the registered redirection URLs before the authorization server can validate the authorization request from the client.

The Customized Scope Check indicates how to check the scope for authorization grants and access tokens.

- When checking the scope by using a custom style sheet, specify the location of the style sheet with the Scope Customized Process property. The style sheet must be in the local: or store: directory.
- When checking the scope with a PCRE, specify the expression with the Scope property.

A custom scope check should be used in the following situations:

- An authorization request where the OAuth client requests an authorization code
- An access request where the OAuth client requests an access token
- A resource request where the OAuth client requests a resource

The Scope specifies the PCRE to check the scope. The minimum length of the expression is 1 character. The maximum length of the expression is 1023 characters.

The Scope Customized Process specifies the location of the style sheet for a custom scope check. The style sheet must be in the local: or store: directory. The style sheet validates and sets the scope to check.

The Custom Resource Owner Handling indicates whether to use a style sheet to extract information about the resource owner. When using a custom style sheet to extract, use the Resource Owner Process property to specify the location of the style sheet. The style sheet must be in the local: or store: directory.

By default, the resource owner is the user name from the extracted identity. For custom handling, you must provide a style sheet that overrides information about the resource owner.

- For AAA identity extraction, the extraction method can be basic authentication or forms-based login.
- For custom handling, the style sheet overrides data about the resource owner with information from authentication.

The custom handling should be used in the following situations:

- When presenting the authorization form to the resource owner
- When issuing a code for an authorization code grant type
- When issuing an access token

The Resource Owner Process specifies the location of the style sheet to extract information about the resource owner. The file must be in the local: or store: directory.

The DataPower State Lifetime sets the operational duration in seconds for the local authorization page. Enter a value in the range 1–6000. The default value is 1200.

If the user does not submit the request before the duration elapses, the authorization request from the OAuth client is rejected. The location of the style sheet that defines the local authorization page and the error handling is set with the Authorization Form property.

The Authorization Grant Lifetime sets the lifetime for an authorization code in seconds. Enter a value in the range 1–600. The default value is 300.

An authorization code is the intermediary result of a successful authorization. The client uses authorization codes to obtain the access token. Instead of sending tokens to a client, clients receive authorization codes on their redirection URI. Each supported redirection URI for the client is defined with the Redirect URI property.

The Access Token Lifetime sets the lifetime for the access token in seconds. Enter a value in the range 1–2678400. The default value is 3600.

The Shared Secret assigns the shared secret key to protect tokens that use the OAuth protocol.

The Authorization Form specifies the location of the style sheet that submits the authorization request from the resource owner and handles errors. The file must be in the local: or store: directory. You can use the `OAuth-Generate-HTML.xsl` style sheet in the store: directory or copy this file to the local: directory and modify as needed.

The style sheet must be stored on the appliance in the local: or store: directory. The HTML authorization form remains operational for the duration that is defined with the DataPower State Lifetime property. If the user does not submit the request before the duration elapses, the authorization from the OAuth client is rejected.

The Additional OAuth Process specifies the location of the style sheet to process after generating a code, after generating an access token, or after generating an access token but before sending it to the resource server. The style sheet must be in the local: or store: directory.

Additional OAuth processing can be used in the following situations:

- An authorization request after successfully generating a code for an authorization code grant with the `authorization_request` operation. Processing returns a node set. This information becomes part of the query string and is returned to the OAuth client.
- An access request after successfully generating an access token with the `access_request` operation. Processing returns a node set. This information becomes part of the JSON object that contains the access token.
- A resource request after successfully verifying an access token but before sending the request to the resource server with the `resource_request` operation.

Unit summary

Having completed this unit, you should be able to:

- Describe what OAuth is
- Describe why OAuth is useful in security scenarios
- Describe the OAuth 2-legged and 3-legged design pattern
- Explain the role that a DataPower appliance performs in an OAuth architecture
- Describe the OAuth configuration options on DataPower: the Web Token Service, the AAA action, the OAuth Client, and an OAuth Group

© Copyright IBM Corporation 2013

Figure 16-14. Unit summary

WE4013.0

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: 3-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. 1-legged authentication
 - c. AAA action
 - d. SSL
 - e. OAuth Client 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 2013

Figure 16-15. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

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

Checkpoint answers

1. **True.** With OAuth, resource owners allow third-party access to the resource **without** sharing their credentials.
2. **True.** 3-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. 1-legged authentication
 - c. AAA action
 - d. SSL
 - e. OAuth Client and OAuth Client Group
4. **False:** OAuth configuration on DataPower **does** allow for the use of custom style sheets.

© Copyright IBM Corporation 2013

Figure 16-16. Checkpoint answers

WE4013.0

Notes:

Unit 17. Service level monitoring

What this unit is about

Service level management is the monitoring and management of message traffic that concerns quality of service (QoS) indicators such as throughput, response time, and availability. Within DataPower, service level monitoring (SLM) is a tool that helps support those activities. This unit defines the DataPower version of SLM and describes various ways to configure it.

What you should be able to do

After completing this unit, you should be able to:

- Identify the service level monitoring (SLM) functions that the WebSphere DataPower SOA Appliance provides
- Create an SLM policy object by using the WebGUI
- Create a custom SLM statement
- Use the SLM Policy tab in the web service proxy to create a basic SLM policy

How you will check your progress

- Checkpoint
- Exercise 14. Implementing an SLM monitor in a web service proxy

Unit objectives

After completing this unit, you should be able to:

- Identify the service level monitoring (SLM) functions that the WebSphere DataPower SOA 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 2013

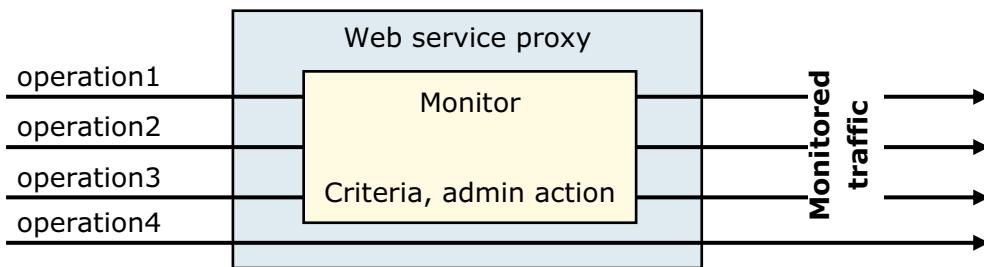
Figure 17-1. Unit objectives

WE4013.0

Notes:

Service level monitoring (SLM) in DataPower

- Concept of monitoring message traffic regarding a predefined set of criteria, and possibly managing the throughput
- Criteria can be traffic rate, client ID, target resource, time, 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 2013

Figure 17-2. Service level monitoring (SLM) in DataPower

WE4013.0

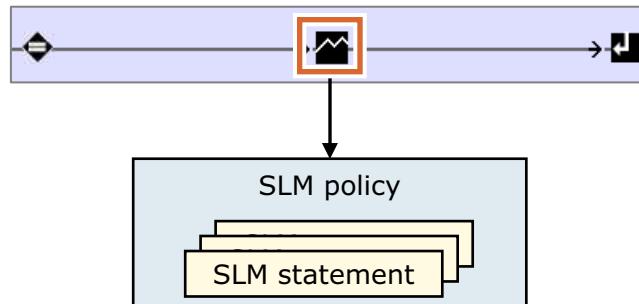
Notes:

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 (1 of 2)

- The presence of an **SLM processing action** in a rule enables the monitor



- The SLM action specifies an SLM policy object
- The **SLM policy** consists of one or more SLM statements
- An **SLM statement** defines the measurement criteria and administrative action
- A message is processed through the statements in order
 - If any thresholds are exceeded, the specified administrative actions are taken

© Copyright IBM Corporation 2013

Figure 17-3. The pieces of SLM (1 of 2)

WE4013.0

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 2013

Figure 17-4. Approaches to define SLM policies

WE4013.0

Notes:

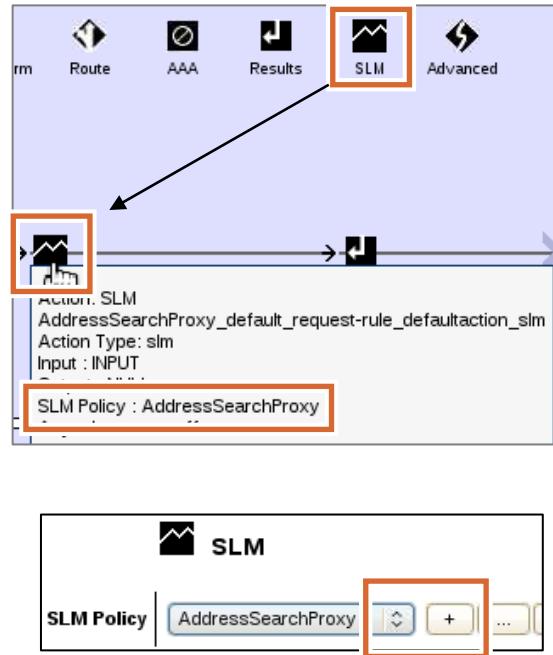
The first two approaches have been supported for many years.

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 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 a new one
- Without an SLM action and SLM policy, no monitoring occurs



© Copyright IBM Corporation 2013

Figure 17-5. Approach 1: Add an SLM action to a request rule

WE4013.0

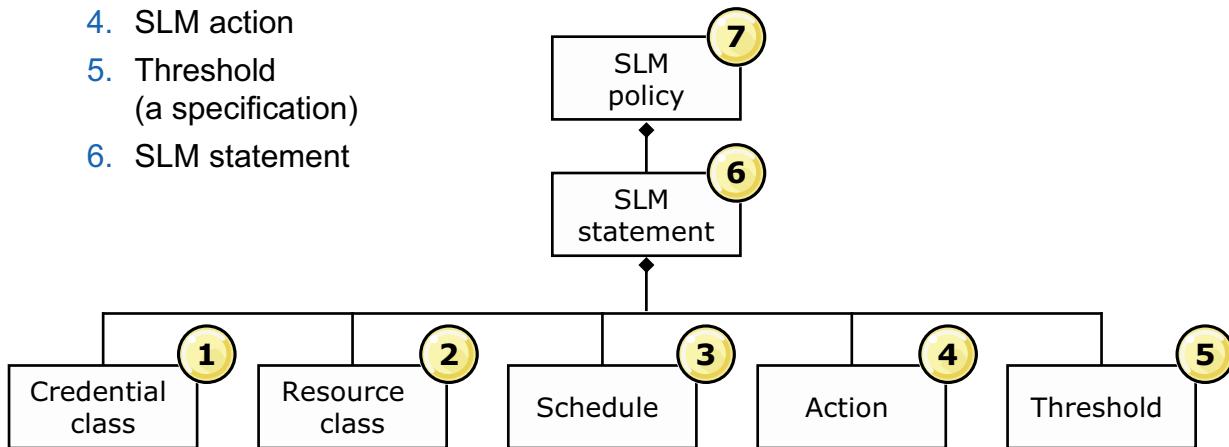
Notes:

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 (2 of 2)

- 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



© Copyright IBM Corporation 2013

Figure 17-6. The pieces of SLM (2 of 2)

WE4013.0

Notes:

A threshold is not a separate object. It is a specification within an SLM statement.

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
 - Select **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

Apply Cancel

Name *

Administrative State enabled disabled

Comments

Credential Type *

Match Type *

© Copyright IBM Corporation 2013

Figure 17-7. The SLM credential class

WE4013.0

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 **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. PCRE-style expressions (regular expressions) can also be selected.
- 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.

The SLM resource class

- Identifies a set of resources subject to an SLM policy statement
 - Select **Objects > Monitoring > SLM Resource Class** to define individually

- A resource class consists of:

- Resource Type:** specifies a method used to identify the resource
- Match Type:** specifies how a successful match is determined
- Resource Value:** values to match

Configure SLM Resource Class

Main

SLM Resource Class

Name *

Administrative State enabled disabled

Comments

Resource Type *

Match Type *

Apply Cancel

© Copyright IBM Corporation 2013

Figure 17-8. The SLM resource class

WE4013.0

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**. 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 <input type="checkbox"/> * |
| Match Type | Exact <input type="checkbox"/> * |
| Resource Value | <input type="text"/> {http://east.address.training.ibm.com}AddressSearch/findByName ↑ ↓ × <input type="button" value="Add"/> |

© Copyright IBM Corporation 2013

Figure 17-9. SLM resource class example

WE4013.0

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 SLM schedule

- Specifies a time period during which the associated SLM statement is enforced
 - Select Objects > Monitoring > SLM Schedule to define individually
- Schedule elements
 - Week Days**
 - Start Time**
 - Duration**
 - Start Date**
 - End Date**

© Copyright IBM Corporation 2013

Figure 17-10. The SLM schedule

WE4013.0

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 checked days.

The **Start Date** and **Stop Date** indicate which dates this schedule is in effect. The Stop Date is non-inclusive.

The SLM action

- When an SLM statement detects a threshold violation, an SLM action defines the response
 - Select Objects > Monitoring > SLM Action to define individually
- Action types
 - Notify:** creates log message when action is fired
 - Shape:** buffers request to meet traffic threshold up to limit; otherwise, it rejects
 - Throttle:** reject outright
- Three SLM actions are predefined
 - New SLM actions can be defined to change log priority of logged message

The first screenshot shows the 'SLM Action' configuration dialog. It has fields for Name (empty), Admin State (radio buttons for enabled and disabled, with 'enabled' selected), Comments (empty), Type (dropdown menu set to 'Log Only *'), and Log Priority (dropdown menu set to 'debug'). There are 'Apply' and 'Cancel' buttons at the top.

The second screenshot shows the 'Configure SLM Action' page. It features a 'Refresh List' button and a table listing three predefined SLM actions:

| Name | Status | Op-State | Logs | Admin State | Comments |
|----------|--------|----------|------|-------------|----------|
| notify | saved | up | 🔍 | enabled | |
| shape | saved | up | 🔍 | enabled | |
| throttle | saved | up | 🔍 | enabled | |

An 'Add' button is located at the bottom left of the table area.

© Copyright IBM Corporation 2013

Figure 17-11. The SLM action

WE4013.0

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.

Action types:

- Log only**
 - After the action is triggered, writes a log entry and continues to process subsequent transactions.
- Reject**
 - After the action is triggered, writes a log entry and drops traffic until the monitored entity is within conformance levels.
- Shape**

- After the action is triggered, writes a log entry. The next 2500 transactions are queued for later transmission when the monitored entity is within conformance levels.
- After 2500 transactions are queued, further transactions are rejected.

Do not confuse the **SLM action object** that is used within an SLM statement with the **SLM processing action** that is used in a processing rule to enable SLM monitoring.

WebSphere Education

SLM statement (1 of 2)

- An SLM statement can consist of:
 - Credential Class:** defines a possible client group subject to this SLM statement
 - Resource Class:** identifies a possible resource group subject to this SLM statement
 - Schedule:** time frame during which this SLM statement is enforced
 - Action:** administrative action (sanction) to take if threshold violated (required)
- SLM statements exist only within the SLM policy object

Edit Statement

| | |
|---------------------------|---|
| Identifier | 2 * |
| User Annotation | Auto generated |
| Credential Class | (none) <input type="button" value="+"/> <input type="button" value="..."/> |
| Resource Class | AddressSearchPolicy_port-operation_findByName <input type="button" value="+"/> <input type="button" value="..."/> |
| Schedule | (none) <input type="button" value="+"/> <input type="button" value="..."/> |
| SLM Action | shape <input type="button" value="+"/> <input type="button" value="..."/> * |
| Threshold Interval Length | 0 |
| Threshold Interval Type | Fixed <input type="button" value="..."/> |
| Threshold Algorithm | Greater Than <input type="button" value="..."/> |

© Copyright IBM Corporation 2013

Figure 17-12. SLM statement (1 of 2)

WE4013.0

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 Length:** length of measurement interval
- Threshold Interval Type:**
 - Fixed: a discrete block of time, for example, 8 a.m. to 9 a.m.
 - Moving: a moving window, for example, the last 60 minutes
- Threshold Algorithm:** greater than, less than, token bucket, high-low threshold
- Threshold Type:** count all, count errors, back-end latency, internal latency, total latency
- Threshold Level:** value that triggers the threshold

The screenshot shows a configuration dialog for SLM statement settings. The visible fields are:

- Threshold Interval Length: 0 Seconds
- Threshold Interval Type: Fixed
- Threshold Algorithm: Greater Than
- Threshold Type: Count All
- Threshold Level: 300
- Reporting Aggregation Interval: 0 Minutes
- Maximum Records Across Intervals: 5000 Records*
- Auto Generated by GUI: on (radio button selected)
- Maximum Credentials-Resource Combinations: 5000 Records*

At the bottom are 'Apply' and 'Cancel' buttons.

© Copyright IBM Corporation 2013

Figure 17-13. SLM statement (2 of 2)

WE4013.0

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 thresholding and when to stop in cases where those two values are not the same. The threshold level is the “high” starting point. The **High Low Release Level** (not shown) configures the “low” stopping point.

Threshold Type specifies how the **Threshold Level** is applied to the count.

Reporting Aggregation Interval is the base aggregation level in minutes for the reporting statistics. This property is independent of the thresholding interval.

Maximum Records Across Intervals is the total number of records for a reporting interval. A single reporting aggregation interval can contain multiple records; for example,

one record per resource or credential. This property allows you to define a maximum memory-consumption threshold. The default is 5000.

Auto Generated by GUI is a read-only property that, when on, indicates that the WebGUI created the statement is part of a default SLM configuration (**SLM Policy** tab in a web service proxy).

Maximum Credentials-Resource Combinations is the maximum number of records for the combination of credentials and resources. This property limits the maximum number of combinations and allows the setting of a maximum memory-consumption threshold. The default is 5000.



SLM policy: Main tab

- An SLM policy consists of one or more SLM statements and an evaluation method
 - Select **Objects > Monitoring > SLM Policy** to define individually
- Evaluation method: determines how the SLM policy evaluates the remaining SLM statements if a threshold is exceeded in the current SLM statement
 - Execute all statements**
 - Terminate at first action**
 - Terminate at first reject**

SLM Policy: AddressSearchProxy [up]

Administrative State: enabled disabled

Comments: [Text input field]

Evaluation Method: **Terminate at First Reject**

Peer Group: **(none)**

© Copyright IBM Corporation 2013

Figure 17-14. SLM policy: Main tab

WE4013.0

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

The screenshot shows a software interface for managing SLM policies. At the top, there are tabs for 'Main' and 'Statement'. Below the tabs, the title 'SLM Policy: AddressSearchProxy [up]' is displayed. A row of buttons includes 'Apply', 'Cancel', 'Delete', and 'Undo'. The main area is titled 'Statement' and contains a table with the following data:

| Identifier | User Annotation | Credential Class | Resource Class | Schedule | SLM Action | Thre Inter Leng |
|------------|----------------------|------------------|---|----------|------------|-----------------|
| 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 2013

Figure 17-15. SLM policy: Statements tab

WE4013.0

Notes:

Getting SLM statements into the Statement list

- Since SLM statements do not exist as separate objects, they cannot be selected from a drop-down list
- SLM statements are added to the list by:
 - Specifying SLM criteria on the SLM Policy tab of a web service proxy (auto-generates SLM statements)
 - Clicking **Add** beneath the list to create a custom SLM statement
- The first two statements are auto-generated
- The third statement is custom

| Hold Al | Threshold Algorithm | Threshold Type | Threshold Level | High-Low Release Level | Burst Limit | Reporting Aggregation Interval | Maximum Records Across Intervals | Auto Generated by GUI | Maximum Credentials-Resource Combinations | |
|---------|---------------------|----------------|-----------------|------------------------|-------------|--------------------------------|----------------------------------|-----------------------|---|------------|
| | Greater Than | Count All | 2 | 0 | 0 | 0 | 5000 | on | 5000 | |
| | Greater Than | Count All | 5 | 0 | 0 | 0 | 5000 | on | 5000 | |
| | Greater Than | Count All | 200 | 0 | 0 | 0 | 5000 | off | 5000 | |
| | | | | | | | | | | Add |

© Copyright IBM Corporation 2013

Figure 17-16. Getting SLM statements into the Statement list

WE4013.0

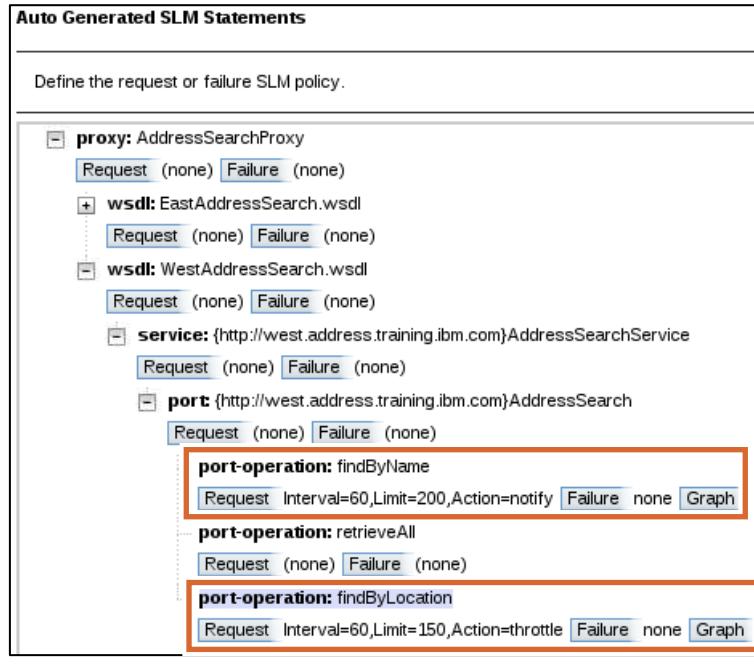
Notes:

This graphic is the right side of the WebGUI page from the previous slide.

 WebSphere Education 

Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2)

- Web service proxy has an **SLM Policy** tab to allow simple definitions of SLM monitoring criteria
- Specifying this criterion creates the auto-generated SLM statements
- SLM criteria can be uniquely specified at the different levels of the WSDL (proxy, wsdl, service, port, port-operation)
- Criteria can be set for successful transactions (Request) and errors (Failure)



© Copyright IBM Corporation 2013

Figure 17-17. Approach 2: Specify SLM criteria to the levels of the WSDL (1 of 2)

WE4013.0

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.



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** lists only custom SLM statements
- **Create New Statement** allows creation of a custom SLM statement

| | | | | | | |
|-----------------------------|---|----------------|----------|-----------------|----------------|--------|
| SLM Peers | Define the collection of SLM peers that monitor an SLA. | | | | | |
| Type | SLM Unicast * | | | | | |
| URL | (empty) Add | | | | | |
| SLM Statements | SLM Statements define custom SLM policies to monitor transactions that meet specific credential or resource criteria. | | | | | |
| ID | Credential Class | Resource Class | Schedule | Threshold Level | Threshold Type | Action |
| Create New Statement | | | | | | |

© Copyright IBM Corporation 2013

Figure 17-18. Approach 2: Specify SLM criteria to the levels of the WSDL (2 of 2)

WE4013.0

Notes:

This graphic is the lower part of the SLM Policy tab for a web service proxy.

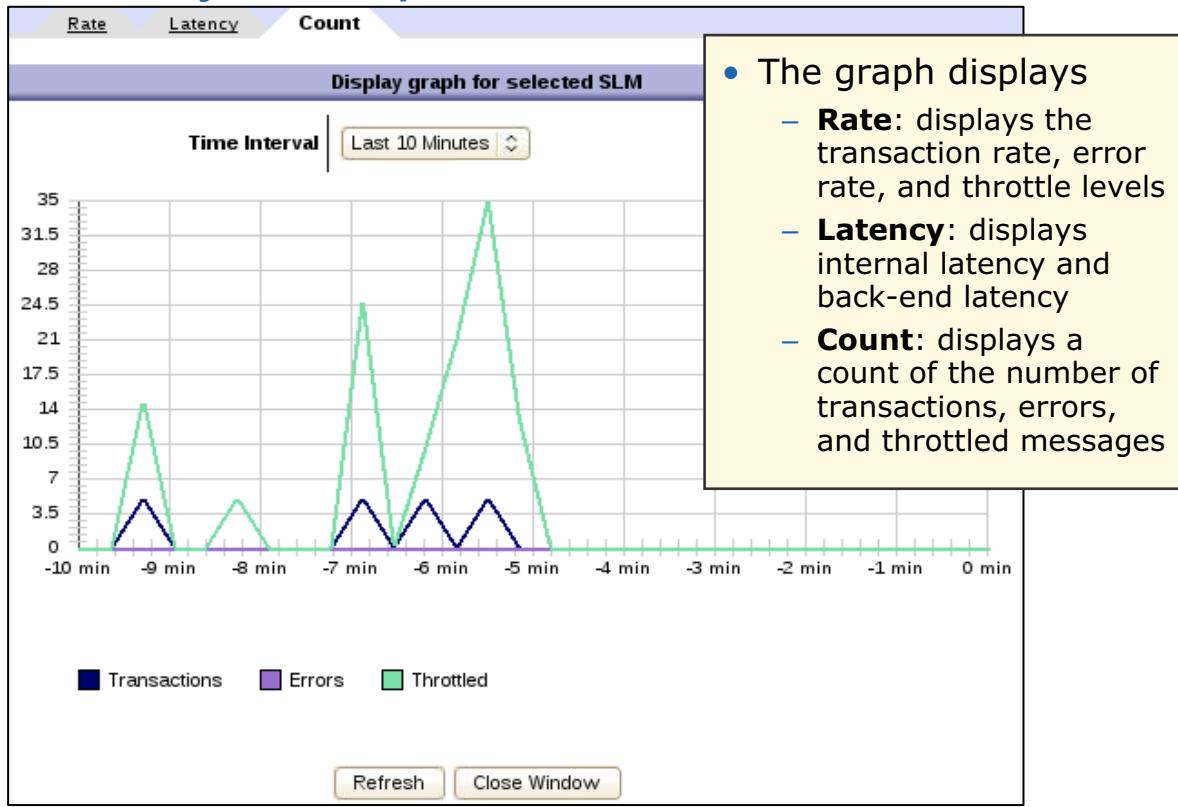
Configuring SLM peers is an administrative task.

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 repaints with a section that contains the same fields as exist in an SLM statement configuration page.

 WebSphere Education 

SLM Policy tab: Graphs



- The graph displays
 - **Rate**: displays the transaction rate, error rate, and throttle levels
 - **Latency**: displays internal latency and back-end latency
 - **Count**: displays a count of the number of transactions, errors, and throttled messages

© Copyright IBM Corporation 2013

Figure 17-19. SLM Policy tab: Graphs

WE4013.0

Notes:

The SLM graph is displayed by selecting the appropriate **Graph** radio button in the web service proxy **SLM Policy** tab.

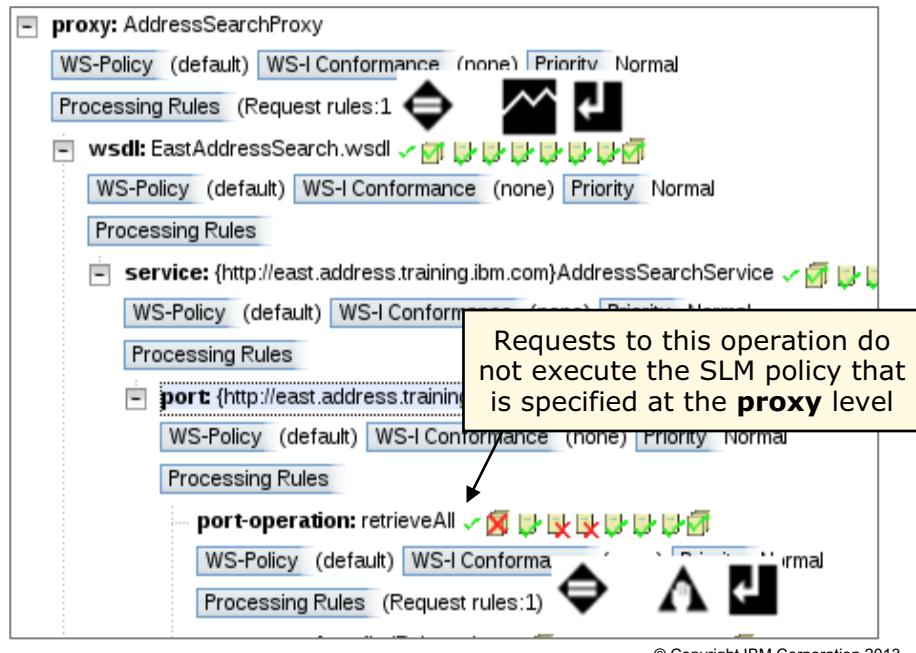
The possible time intervals are last 10 minutes, last 30 minutes, last hour, and last three hours.

This option is only for development time monitoring. For production monitoring, use software such as the IBM Tivoli Monitoring Family.

SLM action granularity

- A web service proxy or multi-protocol gateway service policy must explicitly define an **SLM action** in order for client requests to participate in service level monitoring

- The default web service proxy request policy contains an SLM action
- A fine-grained policy without an SLM action does not participate in service level monitoring



© Copyright IBM Corporation 2013

Figure 17-20. SLM action granularity

WE4013.0

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 service level monitoring (SLM) functions that the WebSphere DataPower SOA 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 2013

Figure 17-21. Unit summary

WE4013.0

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. Encryption, polymorphism, inheritance, objects, and class

2. Match the functionality to the **Reject** and **Shape** action types:

| Description | Definition |
|------------------|---|
| 1. Reject action | A. Log and drop traffic |
| 2. Shape action | B. Log, queue traffic to meet threshold, otherwise reject |

3. True or False: SLM monitors are implemented as part of a service policy.

© Copyright IBM Corporation 2013

Figure 17-22. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. A. 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. Encryption, polymorphism, inheritance, objects, and class
2. Match the functionality to the **Reject** and **Shape** action types:

| Description | Definition |
|------------------|--|
| 1. Reject action | <input type="checkbox"/> A. Log and drop traffic |
| 2. Shape action | <input type="checkbox"/> B. Log, queue traffic to meet threshold, otherwise reject |

3. True. SLM monitors are implemented as part of a service policy.

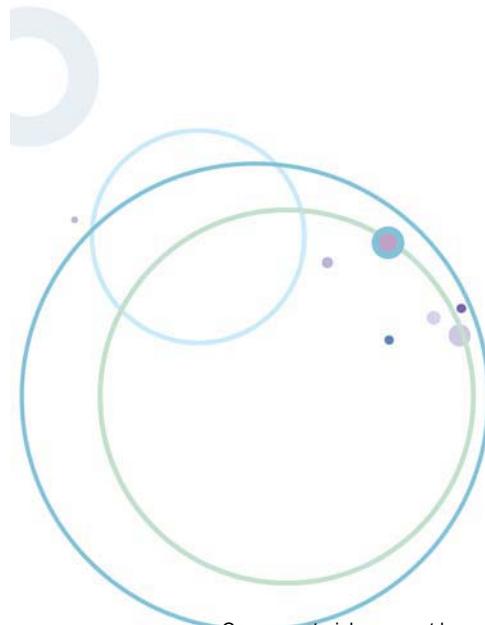
© Copyright IBM Corporation 2013

Figure 17-23. Checkpoint answers

WE4013.0

Notes:

Exercise 14



Implementing an SLM monitor in a web service proxy

© Copyright IBM Corporation 2013

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

7.0.1

Figure 17-24. Exercise 14

WE4013.0

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 2013

Figure 17-25. Exercise objectives

WE4013.0

Notes:

Unit 18. Integration with WebSphere MQ

What this unit is about

This unit describes how to configure the DataPower appliance to communicate with WebSphere MQ. You learn how to receive and put messages on WebSphere MQ queues. You also learn how DataPower manages transactions between WebSphere MQ queue managers.

What you should be able to do

After completing this unit, you should be able to:

- Create a multi-protocol gateway with a WebSphere MQ front-side handler
- Configure a WebSphere MQ back-end Uniform Resource Locator (URL)
- Manage transactionality between WebSphere MQ queue managers

How you will check your progress

- Checkpoint
- Exercise 15: Configuring a multi-protocol gateway service with WebSphere MQ



Unit objectives

After completing this unit, you should be able to:

- Create a multi-protocol gateway with a WebSphere MQ front-side handler
- Configure a WebSphere MQ back-end Uniform Resource Locator (URL)
- Manage transactionality between WebSphere MQ queue managers

© Copyright IBM Corporation 2013

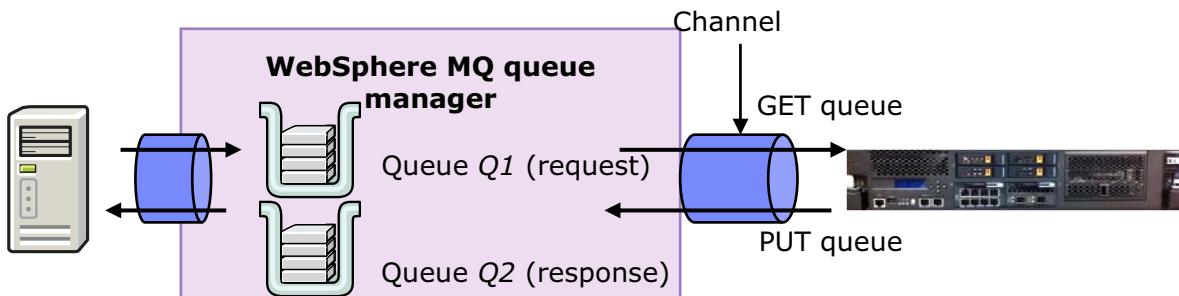
Figure 18-1. Unit objectives

WE4013.0

Notes:

WebSphere MQ fundamentals

- A **queue manager** manages a container for messages that are sent over a WebSphere MQ network
 - In a publish/subscribe model, **queues** represent a message destination for messages that are organized in FIFO order
 - Queue managers send messages over a communications link known as a **channel**
 - A WebSphere MQ client (such as the WebSphere MQ front side handler) must poll the queue manager for new messages
 - The queue manager itself does not initiate connections to the clients



© Copyright IBM Corporation 2013

Figure 18-2. WebSphere MQ fundamentals

WE4013.0

Notes:

IBM WebSphere MQ allows asynchronous message communication across a network. If HTTP communication is analogous to telephone calls, then message delivery over WebSphere MQ is analogous to a courier service. For point-to-point communications, messages are deposited in a queue and used by a service later. The queue manager maintains a set of queues in one node on the network. Separate queues store and forward request and response messages.

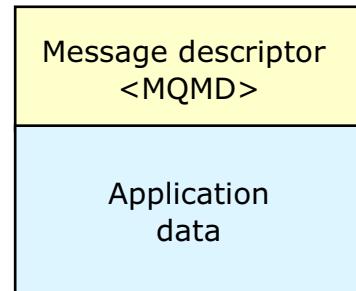
FIFO stands for first-in first-out. Queues mainly work in a FIFO fashion unless a special weighting for messages is implemented.

IBM WebSphere MQ does all network communications over a **channel**. More specifically, the software program that allows network communication between a WebSphere MQ client and the WebSphere MQ queue manager is known as a **client channel**. The channel is a program that runs on the same host as the WebSphere MQ queue manager that provides network connectivity, rather than the connection itself. If a client application is local to the queue manager, then a channel is not necessary, but it is allowed. Since the DataPower

appliance is always remote to the queue manager, communication is always over a channel.

WebSphere MQ message

- WebSphere MQ messages are divided into two parts:
 - Message descriptor: contains message ID and control information
 - Application data: message payload
- Data that are 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 2013

Figure 18-3. WebSphere MQ message

WE4013.0

Notes:

Controlling information within a WebSphere MQ message can include the message priority, reply queue name, correlation ID, and more.

Every message has a message identifier which is determined by the value of the field `MsgId` in its message descriptor. When an application puts a message on a queue, either the application can supply a message identifier, or it can ask the queue manager to generate a unique one.

The correlation identifier is normally used to provide an application with some means of matching a reply with the original message. In a reply message, therefore, the value of the `CorrelId` field is normally copied from the `MsgId` field of the original message.

Transactions

- A transaction is a sequence of operations that either commit or roll back their work
 - A transaction *rolls back* if any one of the operations in the transaction fails
 - A transaction *commits* if all the operations in the transaction succeed
- A **local unit of work** is defined as when only the queue manager resources are being updated
- A **global unit of work** is defined as when resources of other resource managers are also being updated

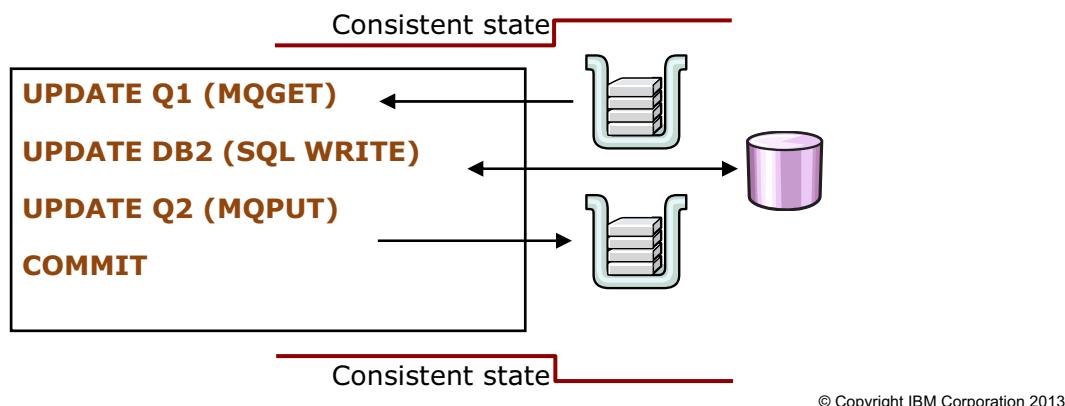


Figure 18-4. Transactions

WE4013.0

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 are in a consistent state from the point of view of the business and its applications.

Resource managers such as WebSphere MQ queue manager can participate in a global unit of work, which involves the processing of resources from multiple resource managers. A transaction manager is required to coordinate such a transaction. It uses the two-phase commit protocol, with a prepare and commit phase. The prepare phase ensures that all resources marked for commitment can be redistributed. The commit phase sends a request to all resource managers to commit their work. The standard interface that is used

between the transaction and resource manager is the X/Open XA interface. Global units of work are sometimes referred to as XA transactions.

DataPower support for WebSphere MQ

- The DataPower XI52 device can exchange messages with WebSphere MQ systems
 - DataPower XI52 provides an enhanced implementation of the IBM WebSphere MQ client
 - DataPower WebSphere MQ client configuration is performed by using the DataPower management interfaces
 - Supports both point-to-point and publish/subscribe modes
- Bridges disparate transport protocols, such as HTTP to WebSphere MQ
 - Messages originating within or outside of WebSphere MQ can flow easily to and from another WebSphere MQ messaging bus or other messaging system, such as HTTP or TIBCO EMS
- The multi-protocol gateway service allows for the implementation of multiple transport protocols by using:
 - Front side handlers
 - Back-end URL

© Copyright IBM Corporation 2013

Figure 18-5. DataPower support for WebSphere MQ

WE4013.0

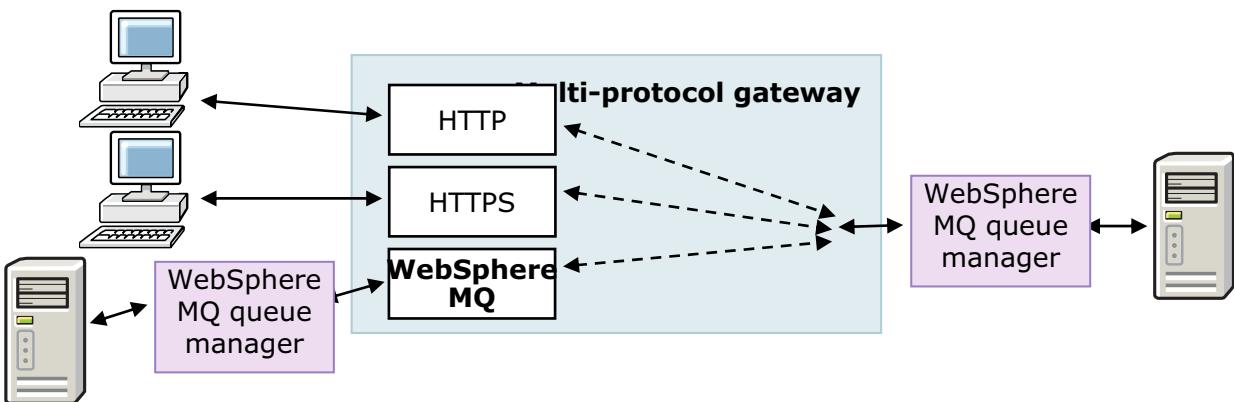
Notes:

The publish/subscribe mode is available with firmware V3.8.0, and WebSphere MQ V7.

Support “fire and forget” or one-way messaging by setting the reply queue to an empty string and the message type to **pass-through**.

Provide WebSphere MQ access

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



© Copyright IBM Corporation 2013

Figure 18-6. Provide WebSphere MQ access

WE4013.0

Notes:

Contact your IBM WebSphere MQ administrator for the host name, port, and queue names in your application. Setting up a WebSphere MQ queue manager is beyond the scope of this presentation.

The DataPower appliance can obtain responses that are associated to a request. The DataPower appliance polls the reply-to queue to find a correlated response message. The gateway examines the correlation ID value in the WebSphere MQ header of messages on the reply-to queue. When this ID is the same as the message ID assigned to the request, the gateway takes the message as the response.

If such a message is found, the multi-protocol gateway can again apply any configured processing policy actions to the response and returns the reply to the requesting HTTP client. This message includes error responses from the back-end application server. If no response is found, the MPGW generates an error to the front side client.

The web service proxy can also use the WebSphere MQ front side handler.

Step 1: Create an WebSphere MQ queue manager (1 of 2)

1. Create a WebSphere MQ queue manager object from the **control panel**
2. Provide the host name or IP address of the queue manager
 - If the port is not specified, the default value 1414 is used
3. Provide the **queue manager name** if it is different from the default
4. Provide an alternate **channel name** if necessary
5. Enter a user name that identifies the client (the WebSphere MQ queue manager object) to the queue manager

| Name | Value |
|-------------------------|---|
| Admin State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments | WebSphere MQ Queue Manager |
| Host Name | [REDACTED] (1414) |
| Queue Manager Name | WBRK6_DEFAULT_QUEUE_MANAGER |
| Channel Name | EastAddress.Channel |
| Channel Heartbeat value | 300 |
| User Name | MUSR_MQADMIN |
| Maximum Message Size | 1048576 |

© Copyright IBM Corporation 2013

Figure 18-7. Step 1: Create a WebSphere MQ queue manager (1 of 2)

WE4013.0

Notes:

A WebSphere MQ queue manager object allows a back side handler on the DataPower SOA appliance to access an IBM WebSphere MQ queue manager. The same object also allows WebSphere MQ queue managers to connect to a DataPower service through a front side handler.

The **Queue Manager Name** field is necessary only if a nondefault queue manager name is assigned to this queue manager.

SYSTEM.DEF.SVRCONN represents the default server connection channel.

The **user name** field is used to provide a plaintext string that identifies the client to the WebSphere MQ queue manager. You provide a user name with administrative permissions on the local operating system.

During the installation of WebSphere MQ, it creates a user in the local Windows registry that is called MUSR_MQADMIN in the WebSphere MQ user group, which has local OS administrative permissions. In Linux, the default user is “mqm”.

The default port number, if it is not specified in the host name field, is 1414.

Step 1: Create an WebSphere MQ queue manager (2 of 2)

6. Specify whether the WebSphere MQ queue manager participates in a transaction

- 0 (zero), undeliverable messages are silently discarded
- 1, the queue manager commits or rolls back the transaction

7. Set the total number of open connections

8. Specify the encryption key and type for an SSL connection

9. Configure an automatic retry interval to automatically reconnect to the queue manager

| | |
|--------------------------|---|
| Cache Timeout | <input type="text"/> seconds |
| Units of Work | <input type="text"/> 6 |
| Total Connection Limit | <input type="text"/> 7 |
| Initial Connections | <input type="text"/> 1 |
| SSL Key Repository | <input type="text"/> cert:/// (none) <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> |
| SSL Cipher Specification | <input type="text"/> None 8 |
| Convert Input | <input checked="" type="radio"/> on <input type="radio"/> off |
| Automatic Retry | <input checked="" type="radio"/> on <input type="radio"/> off 9 |
| Retry Interval | <input type="text"/> 1 seconds |
| Reporting Interval | <input type="text"/> 1 seconds |
| Alternate User | <input checked="" type="radio"/> on <input type="radio"/> off |
| Local Address | <input type="text"/> |
| XML Manager | <input type="text"/> default <input type="button" value="+"/> <input type="button" value="..."/> * |
| SSL proxy profile | <input type="text"/> (none) <input type="button" value="+"/> <input type="button" value="..."/> |

© Copyright IBM Corporation 2013

Figure 18-8. Step 1: Create a WebSphere MQ queue manager (2 of 2)

WE4013.0

Notes:

The **SSL Key Repository** specifies the location of the key database file. Set this field to **none** to disable SSL support.

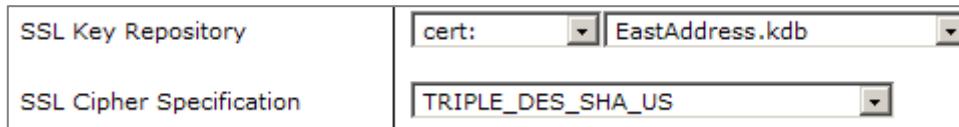
Set **Convert Input** to **on** to convert messages to the default CCSID (coded character set identifier) in use by the IBM WebSphere MQ queue manager. For example, the character encoding that is used on a zSeries system might not match an Intel Linux server that is based on Intel. The CCSID helps the system to interpret the message properly.

The **Local Address** field specifies the Ethernet address to use for the outbound message.

When it is turned on, the **Alternate User** field contains a flag inside a WebSphere MQ message. It allows WebSphere MQ to run a system exit call that calls on custom C/C++ code in WebSphere MQ.

Step 1: Use SSL in mutual authentication mode

- The WebSphere MQ queue manager can be configured to use SSL in mutual authentication mode with a remote WebSphere MQ queue manager
- Execute the following steps:
 - Configure the remote WebSphere MQ queue manager to use SSL
 - In DataPower, generate a self-signed certificate
 - DataPower generates the certificate key pair in the PEM format
 - Use an external application to convert the PEM format to pkcs12
 - This step is required since WebSphere MQ does not understand the PEM format
 - Import the converted certificate key into WebSphere MQ
 - Obtain the WebSphere MQ key database file and import it into DataPower and select it in the **SSL Key Repository** field



© Copyright IBM Corporation 2013

Figure 18-9. Step 1: Use SSL in mutual authentication mode

WE4013.0

Notes:

To set up SSL between the DataPower WebSphere MQ client and a WebSphere MQ queue manager, both parties exchange keys that are used during SSL communication.

The first step is to enable SSL for the WebSphere MQ queue manager. In DataPower, you generate the certificate key pair that is imported into WebSphere MQ. DataPower generates certificates in the PEM format, which WebSphere MQ does not support. You convert the PEM format into the pkcs12 format. You can use the OpenSSL tool to convert between certificate formats.

When the certificate key pair is converted, you import it into the WebSphere MQ key database. Finally, you export the WebSphere MQ key database and import it into DataPower. When the key database is imported, you can select it in the **SSL Key Repository** field.

For more information, see the technote “Configuring DataPower WebSphere MQ client to use SSL in mutual authentication mode” at:

<http://www.ibm.com/support/docview.wss?&fdoc=aimwdp&rs=2362&uid=swg21260155>

Step 2: Add a WebSphere MQ front side handler

1. Open the multi-protocol gateway from the previous scenario
2. Create a **WebSphere MQ Front Side Handler** to accept requests from a WebSphere MQ system
3. Select the queue manager object that was defined in the previous step
4. Specify the queue for the request messages (Get Queue) and the response messages (Put Queue)
5. Configure the character code set identifier (CCSI) for the messages on the queue

The screenshot shows the 'Create a New' dialog for adding a WebSphere MQ front side handler. The 'General' tab is selected. The 'Queue Manager' field is highlighted with a yellow circle containing the number 2. The 'Get Queue' field is highlighted with a yellow circle containing the number 3. The 'Put Queue' field is highlighted with a yellow circle containing the number 4. The 'CCSI' field is highlighted with a yellow circle containing the number 5. The 'Name' field is empty. The 'Comments' field is also empty. The 'Administrative State' field is set to 'Enabled'. The 'Polling Interval' is set to 30. The 'Get Message Options' is set to 4097. The 'Retrieval Backout Settings' is set to 'on'. The 'Use Queue Manager in URL' is set to 'on'. The 'CCSI' value is 0.

Create a New:

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

General

Name

Administrative State

Comments

Queue Manager

Get Queue

Put Queue

The number of concurrent MQ connections

Get Message Options

Polling Interval

Retrieve Backout Settings

Use Queue Manager in URL

CCSI

AddressQM *

ADDRESS_REQ *

ADDRESS_RESP

1

4097

30

on off

on off

0

Example for point-to-point mode

© Copyright IBM Corporation 2013

Figure 18-10. Step 2: Add a WebSphere MQ front side handler

WE4013.0

Notes:

The administrator on the WebSphere MQ queue manager defines the names of the Get and Put queues.

The publish/subscribe mode uses different fields on the web page.

There is an option to support an asynchronous Put operation (Async Put on or off).



Step 3: Configure an WebSphere MQ back-end transport

1. Click the **MQHelper** button in the **Back side settings**
2. Select a new or existing queue manager object
3. Set the **URI** that identifies the service on the final back-end destination
4. Specify the request and response queues
5. Set **Transactionality** to **on** if the queues participate in a unit of work
6. Enable the **UserIdentifier** header field, if a processing action is added an identifier
7. Set **ReplyToQ** to **on** to copy the reply ObjectName in MQOD to ReplyToQ in MQMD

The screenshot shows the URL Builder interface with numbered steps:

1. Backend URL: http://9.65.242.185:9080/EastAddr*
2. MQHelper button (highlighted)
3. Queue Manager: AddressQM
4. URI: stAddress/services/AddressSearch
5. RequestQueue: SEARCH_REQ
6. Transactionality: on (radio button selected)
7. User Identifier: on (radio button selected)
8. ReplyToQ: on (radio button selected)
9. Build URL button

Example for point-to-point mode

© Copyright IBM Corporation 2013

Figure 18-11. Step 3: Configure a WebSphere MQ back-end transport

WE4013.0

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 WebSphere MQ back-end transport to add a value to the user identifier header field. This setting adds `PMO=2052` to the URL. The actual header value itself must be set by header injection or another processing action.

The back-end URL for a WebSphere MQ uses a URI syntax specific to DataPower. For example, the settings in the slide would create a URL of:

dpmq://AddressQM/EastAddress/services/
AddressSearch?RequestQueue=SEARCH_REQ;ReplyQueue=SEARCH_RESP



Publish/subscribe: WebSphere MQ front side handler support

1. Rather than **Get Queue** and **Put Queue** fields, a Topic String needs to be entered
2. Messages are published to a Topic String
3. Subscribers are delivered messages that were published to the Topic Strings to which they are subscribed
4. Specify **Subscription Name** for durable subscription
5. If response is needed, specify **Publish Topic String**
6. If both Get Queue and Subscribe Topic String are present, the Get Queue overrides
7. If both Put Queue field and Publish Topic String are present, the Put Queue overrides

The screenshot shows a user interface titled "Publish and Subscribe". It contains three input fields: "Subscribe Topic String" with an empty text box, "Subscription Name" with an empty text box, and "Publish Topic String" with an empty text box. The entire interface is contained within a light gray rectangular frame.

© Copyright IBM Corporation 2013

Figure 18-12. Publish/subscribe: WebSphere MQ front side handler support

WE4013.0

Notes:

Publish/subscribe applies to WebSphere MQ V7.



Publish/subscribe: WebSphere MQ back-end transport support

1. Use the **MQHelper** button in the **Back side settings** as before
2. A queue manager object 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 2013

Figure 18-13. Publish/subscribe: WebSphere MQ back-end transport support

WE4013.0

Notes:

The helper constructs a DataPower WebSphere MQ URL like:

`dpmq://QM/?SubscribeTopicString=yyyy;SubscriptionName=zzz`

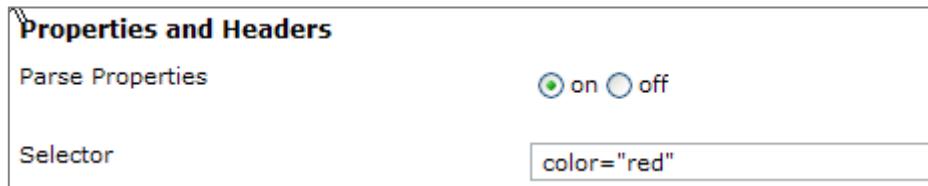
In the MQHelper dialog:

- If both the RequestQueue and PublishTopicString are entered, the RequestQueue overrides.
- If both the ReplyQueue and SubscribeTopicString are entered, the ReplyQueue overrides.
- If the WebSphere MQ URL is entered manually, whatever is entered the latest in the URL string overrides.



Message properties

- Name-value pairs that are associated with the message (WebSphere MQ V7)
- Allow DataPower awareness of properties by enabling parsing in WebSphere MQ front side handler
- Messages can be selectively retrieved by specifying an SQL92-style statement as a “message selector”



- Message properties can be manipulated in a style sheet within a Transform Binary action

© Copyright IBM Corporation 2013

Figure 18-14. Message properties

WE4013.0

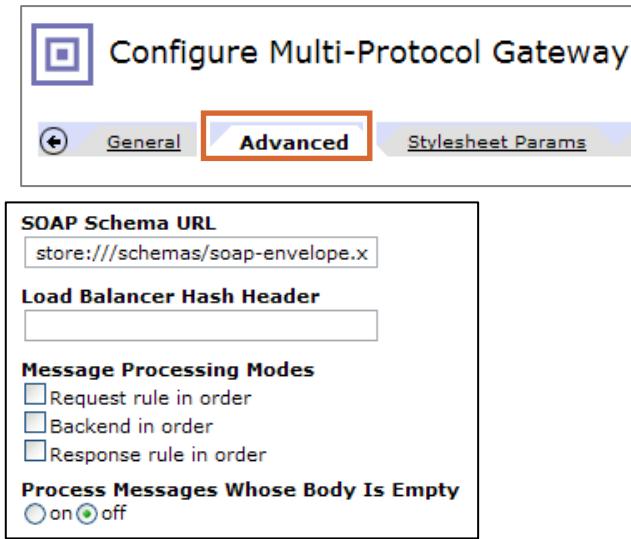
Notes:

Messages are retrieved from the queue only if the message property satisfies the selector statement.

A selector can also be specified in the WebSphere MQ URL.

Ordered processing of WebSphere MQ messages

- Enforces serial processing of WebSphere MQ messages:
 - Retrieves from the front side queue and is presented to the request rule
 - Exits the request rule and is sent to the back side queue
 - Retrieves from the back side queue and is presented to the response rule
- The message order is the sequence in which messages are pulled from the front side request queue
- Messages are buffered, if needed, to maintain order
- Messages exiting a response rule might get buffered
 - Messages sent to the front-end queue are always delivered in the order that they are pulled from the back side queue



© Copyright IBM Corporation 2013

Figure 18-15. Ordered processing of WebSphere MQ messages

WE4013.0

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 back-end in order serial processing. In that case, the appliance buffers messages so that it sends messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Back-end in order: Enforces the serial processing of messages that are delivered to the back-end request queue. If necessary, the appliance buffers messages to complete the request rule that processes out of order. It also delivers messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Response rule in order: Enforces serial processing of messages for actions in the response rule. The appliance initiates and completes response rule processing for

messages in the order in which they were pulled from the back-end reply queue. The appliance starts the response rule for the second message in the reply queue only after it completes the processing of the first message. The appliance always buffers messages so that it sends messages to the front-end reply queue in the same order in which they were pulled from the back-end reply queue.

Controlling backout of WebSphere MQ messages

- WebSphere MQ queue manager object
 - **Units of Work** must be 1
 - **Backout Threshold** indicates the number of reprocessing attempts per message
 - **Backout Queue Name** indicates the queue where messages are placed after they are attempted to the threshold limit

| | |
|--------------------|---|
| Units of Work | <input type="text" value="1"/> |
| Automatic Backout | <input checked="" type="radio"/> on <input type="radio"/> off |
| Backout Threshold | <input type="text" value="3"/> |
| Backout Queue Name | <input type="text" value="EastAddressQueueError99"/> |

- WebSphere MQ front side handler
 - Setting **Retrieve Backout Settings** to **on** causes the appliance to retrieve the backout settings from the actual WebSphere MQ server

| | |
|--|---|
| <input type="button" value="Retrieve Backout Settings"/> | <input checked="" type="radio"/> on <input type="radio"/> off |
|--|---|

© Copyright IBM Corporation 2013

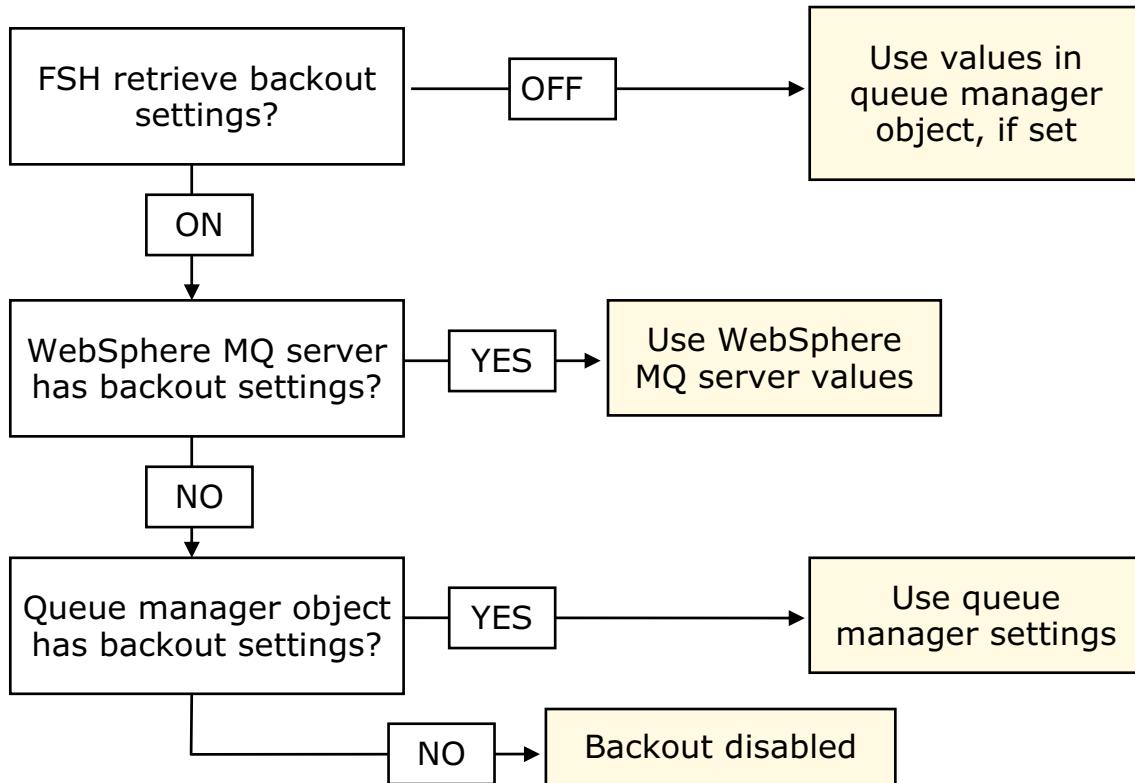
Figure 18-16. Controlling backout of WebSphere MQ messages

WE4013.0

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 2013

Figure 18-17. Decision tree for the backout settings

WE4013.0

Notes:

WebSphere MQ header action in service policy

- In policy editor:
Advanced > MQ Header
- Enables manipulation of WebSphere MQ headers without requiring a style sheet
- Allows modification of MQMD request headers, MQMD response headers, queue manager and reply queue for response, message retrieval by message ID, or correlation ID

MQ Header

| | |
|-------------------------------------|---|
| Asynchronous | <input type="radio"/> on <input checked="" type="radio"/> off |
| MQ Processing Type | <input checked="" type="radio"/> request <input type="radio"/> response <input type="checkbox"/> Save |
| MQ Request Header Processing | MQMD for PUT <input type="checkbox"/> Save |
| Override Existing MQMD | <input type="radio"/> on <input checked="" type="radio"/> off <input type="checkbox"/> Save |
| Message Id | [Text Box] |
| Correlation Id | [Text Box] |
| Character Set Id | [Text Box] |
| Format Name | [Text Box] |
| ReplyToQ | [Text Box] |
| ReplyToQMgr | [Text Box] |

© Copyright IBM Corporation 2013

Figure 18-18. WebSphere MQ header action in service policy

WE4013.0

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 a WebSphere MQ header action

- Retrieve a response message by WebSphere MQ message ID or WebSphere MQ correlation ID
 - Retrieve either value from a DataPower variable and enter it in the appropriate entry field
 - The field with an entry determines which ID is used for retrieval
- Modify MQMD request message headers
 - Selectively override any of the listed headers, or replace them with new and default headers
- Modify MQMD response message headers
 - Selectively override any of the listed headers, or replace them with new and default headers
- Change the queue manager for response messages
 - Modify the destination reply queue manager that is defined in the response header
- Change the reply queue for response messages
 - Modify the destination reply queue that is defined in the response header

© Copyright IBM Corporation 2013

Figure 18-19. Typical uses of a WebSphere MQ header action

WE4013.0

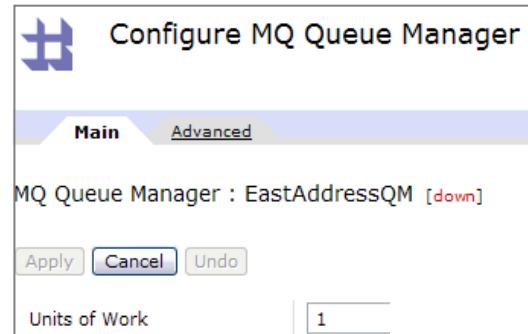
Notes:



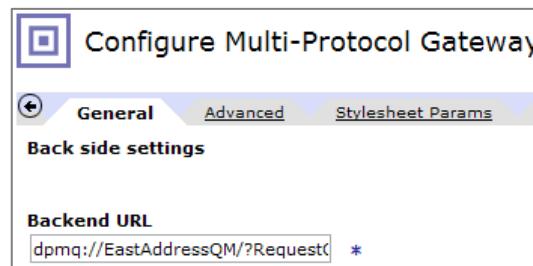
Transactions and WebSphere MQ

DataPower allows control of transactions at both ends:

- Front side
 - **Units of Work** parameter in WebSphere MQ Queue Manager object



- Back side
 - **Backend URL:**
sync query parameter,
transactional query parameter



© Copyright IBM Corporation 2013

Figure 18-20. Transactions and WebSphere MQ

WE4013.0

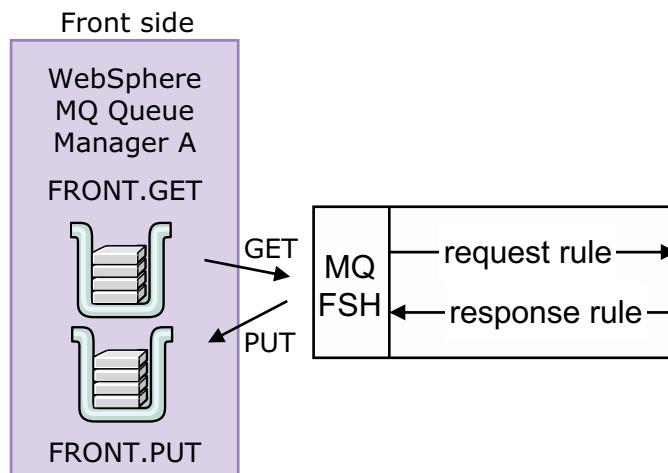
Notes:

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 WebSphere MQ queue managers.

WebSphere MQ front side transactions



- The WebSphere MQ front side handler (MQFSH) gets the message from FRONT.GET
 - If Units of Work = 1, then the transaction with Queue Manager A begins
- When the response rule completes, the MQFSH puts the message to FRONT.PUT
 - If Units of Work = 1, then the transaction with Queue Manager A ends

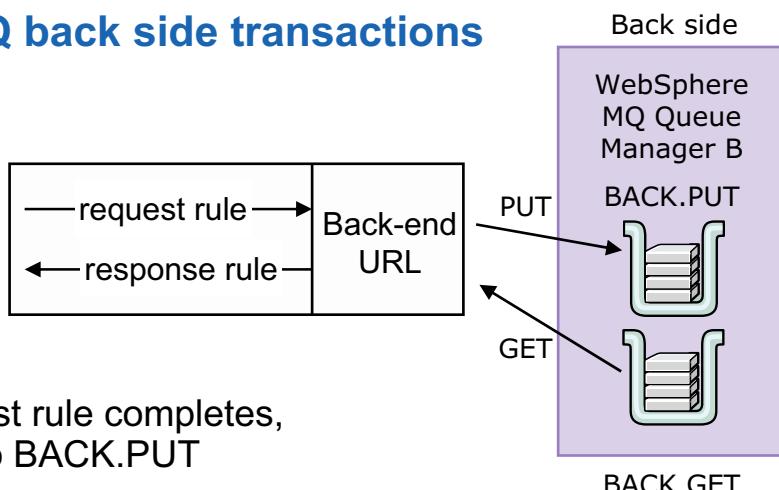
© Copyright IBM Corporation 2013

Figure 18-21. WebSphere MQ front side transactions

WE4013.0

Notes:

WebSphere MQ back side transactions



- When the request rule completes, MQPUT is put to BACK.PUT
 - If Sync = true, MQCOMMIT is then sent
 - Message is available on BACK.PUT and visible to back side WebSphere MQ application
- Service retrieves response message from BACK.GET
 - If Transactional = true, then the transaction with Queue Manager B starts
- Response rule completes
- FSH writes message to FRONT.PUT
 - If Transactional = true, then the transaction with Queue Manager B ends

© Copyright IBM Corporation 2013

Figure 18-22. WebSphere MQ back side transactions

WE4013.0

Notes:

Sync=true is necessary if Queue Manager B is using transactions.

Sync = true|false cannot be set by the WebSphere MQ URL Builder. You must edit the back-end URL field to add the Sync parameter.

WebSphere MQ DataPower URL

- The WebSphere MQ DataPower URL is of the following form:
`dpmq://QueueManager/URI?RequestQueue=PUTQ;ReplyQueue=GETQ;Sync=true;Transactional=true;PMO=2048`
 - QueueManager: name of the WebSphere MQ queue manager object
 - URI: string to include in the URL
 - RequestQueue: name of the queue where messages are sent
 - ReplyQueue: name of the queue to poll for messages
 - Transactional (true or false): used to enforce transactional units of communication with back side queue managers
 - Sync (true or false): used to send MQCOMMIT to back side queue manager after MQPUT
 - PMO: set options on the MQPUT call
- The url-open extension element supports the WebSphere MQ DataPower URL
 - A style sheet can GET and PUT to queues

© Copyright IBM Corporation 2013

Figure 18-23. WebSphere MQ DataPower URL

WE4013.0

Notes:

Existing MQMD headers can be accessed from the variable:

`var://context/contextname/_extension/header/MQMD`

WebSphere MQ queue manager Group object

- Provides failover of WebSphere MQ queue manager objects
 - Consists of a single primary WebSphere MQ queue manager and many backup WebSphere MQ queue managers
- If the primary queue manager becomes unavailable, the DataPower appliance can attempt to place the message on one of the backup WebSphere MQ queue managers

MQ Queue Manager Group

| | |
|--------------------------|--|
| Name | <input type="text" value="EastAddressQMGroup"/> * |
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments | <input type="text"/> |
| Primary MQ Queue Manager | <input type="text" value="EastAddressQM"/> <input type="button" value="..."/> * |
| Backup MQ Queue Managers | <input type="text" value="AddressQM"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="text" value="AddressQM"/> <input type="button" value="Add"/> <input type="button" value="..."/> |

© Copyright IBM Corporation 2013

Figure 18-24. WebSphere MQ queue manager Group object

WE4013.0

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 a WebSphere MQ front-side handler
- Configure a WebSphere MQ back-end Uniform Resource Locator (URL)
- Manage transactionality between WebSphere MQ queue managers

© Copyright IBM Corporation 2013

Figure 18-25. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. True or False: WebSphere MQ support is only available on the multi-protocol gateway.
2. True or False: The DataPower MQ client implementation supports one-way messaging.
3. Match the definitions between local and global units of work:

| Description | Definition |
|-------------------------|---|
| 1. Local units of work | A. Involves the updating of resources on multiple resource or queue managers |
| 2. Global units of work | B. Involves updating resources of a single resource or queue manager C. DataPower supports |

© Copyright IBM Corporation 2013

Figure 18-26. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. **False.** The Web Service Proxy can also use an MQ front side handler.
2. **True.** The DataPower MQ client implementation supports one-way messaging.
3. Match the definitions between local and global units of work.

| Description | Definition |
|-------------------------|---|
| 1. Local units of work | A. Involves the updating of resources on multiple resource or queue managers |
| 2. Global units of work | B. Involves updating resources of a single resource or queue manager C. DataPower supports |

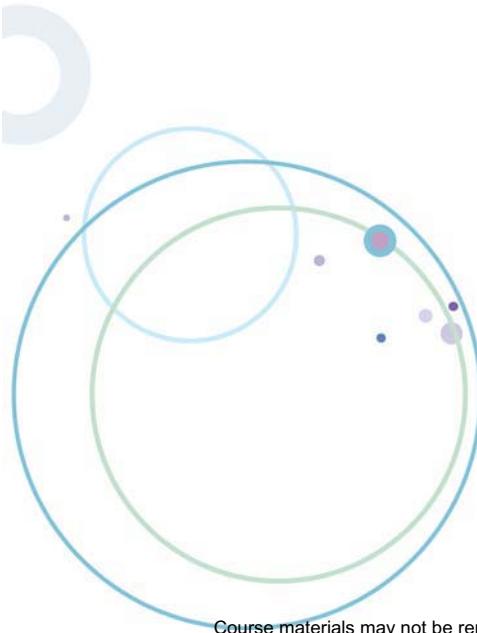
© Copyright IBM Corporation 2013

Figure 18-27. Checkpoint answers

WE4013.0

Notes:

Exercise 15



Configuring a multi-protocol gateway service with WebSphere MQ

© Copyright IBM Corporation 2013

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

7.0.1

Figure 18-28. Exercise 15

WE4013.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Create a WebSphere MQ front-side handler (FSH) that gets messages from a queue and puts responses on a queue
- Send messages from a multi-protocol gateway service to a queue in WebSphere MQ in a fire-and-forget messaging pattern
- Configure transactionality between WebSphere DataPower and WebSphere MQ when errors occur during message processing

© Copyright IBM Corporation 2013

Figure 18-29. Exercise objectives

WE4013.0

Notes:

Exercise overview

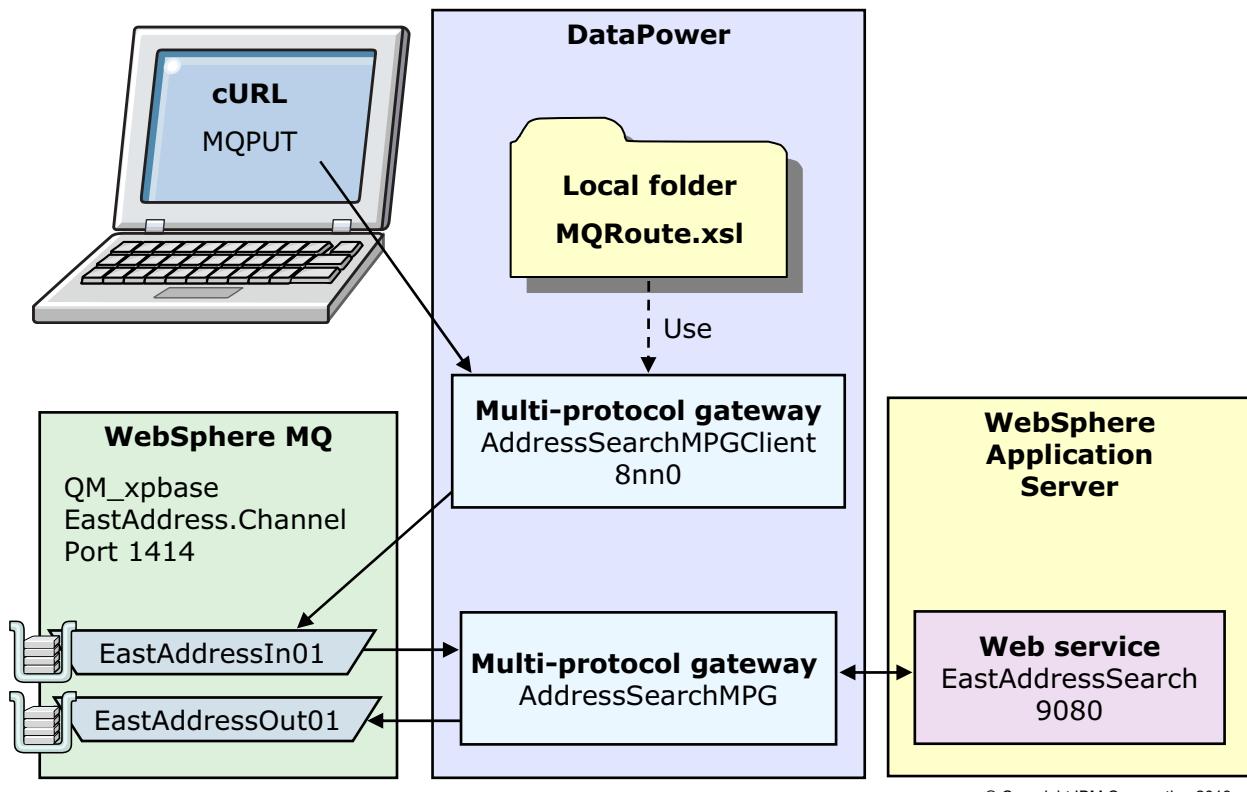


Figure 18-30. Exercise overview

WE4013.0

Notes:

Unit 19. DataPower and WebSphere JMS

What this unit is about

This unit describes how to configure a WebSphere JMS front-side handler and WebSphere JMS back-end URL to connect to the default messaging provider in WebSphere Application Server V7.

What you should be able to do

After completing this unit, you should be able to:

- Configure a WebSphere JMS front-side handler to send JMS messages to the default messaging provider in WebSphere Application Server V7
- Configure a back-end WebSphere JMS URL to communicate with the default messaging provider in WebSphere Application Server V7
- Describe the components of the service integration bus on WebSphere Application Server V7

How you will check your progress

- Checkpoint
- Appendix Exercise B: Configuring WebSphere JMS (optional)

Unit objectives

After completing this unit, you should be able to:

- Configure a WebSphere JMS front-side handler to send JMS messages to the default messaging provider in WebSphere Application Server V7
- Configure a back-end WebSphere JMS URL to communicate with the default messaging provider in WebSphere Application Server V7
- Describe the components of the service integration bus on WebSphere Application Server V7

© Copyright IBM Corporation 2013

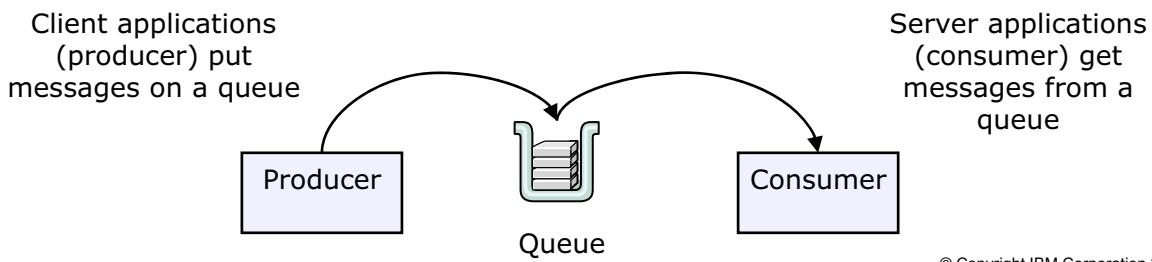
Figure 19-1. Unit objectives

WE4013.0

Notes:

Messaging middleware

- Messaging middleware acts as a broker to provide asynchronous delivery of data between applications
 - Acts as an intermediary between the producer and consumer of the message
 - Ensures reliable message delivery
 - WebSphere MQ is an IBM messaging middleware product
 - WebSphere Application Server includes messaging middleware
 - Enterprise Message Service (EMS) is the TIBCO messaging middleware product
- Messages are sent to destinations and stored until delivered
 - The messaging middleware manages delivery
 - Consumer and producer do not need to be active at the same time



© Copyright IBM Corporation 2013

Figure 19-2. Messaging middleware

WE4013.0

Notes:

There are many different types of middleware: transaction processing monitors, remote procedure calls, and object request brokers can all be considered middleware.

Messaging middleware stores, routes, and manages messages for delivery. Imagine a courier service that allows you to drop off a package for delivery. That company ensures that your package is delivered before a certain time. If the other party is not available, it holds the package for later delivery or pickup. Messages are routed through the services of messaging middleware in this manner.

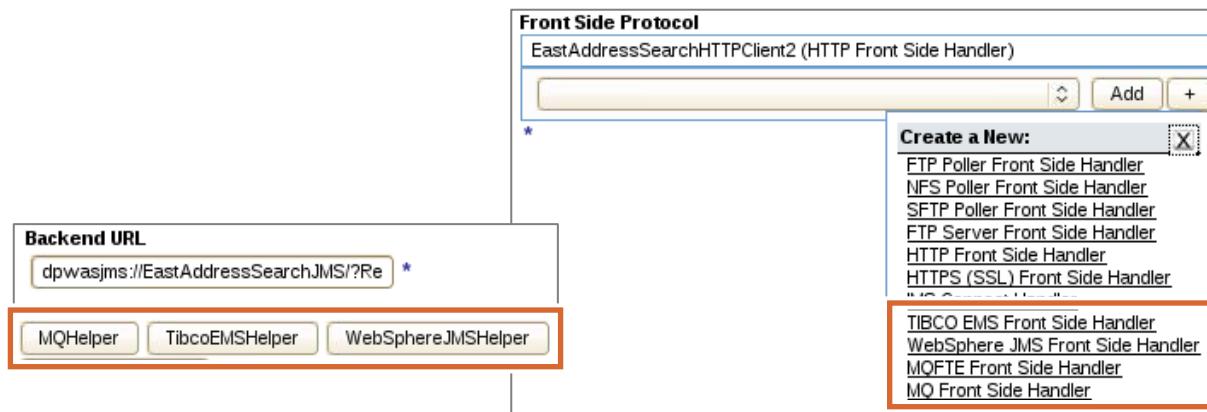
The application that produces the messages is known as the producer, and the application that receives the messages is known as the consumer.

Messages can remain on the queue for some length of time.



DataPower support of messaging middleware

- Message queueing
 - WebSphere MQ
- Java Message Service (JMS)
 - WebSphere Application Server
 - TIBCO Enterprise Message Service (EMS)



© Copyright IBM Corporation 2013

Figure 19-3. DataPower support of messaging middleware

WE4013.0

Notes:

DataPower uses WebSphere MQ to support the MQ model.

It supports the JMS model by using two of the implementations: WebSphere Application Server and TIBCO EMS.

No other implementations of JMS are supported currently.

The support is at both the front side and the back end. It is also supported within style sheets. The url-open extension element can call the MQ and JMS implementations from within a style sheet.

Java Message Service (JMS)

- Java Message Service (JMS) is a Java API for accessing messaging middleware
- JMS provides:
 - Programming interface: does not provide actual runtime
 - Vendor-neutral and standard approach to use messaging middleware, such as WebSphere MQ
- A messaging middleware that supports JMS is known as a JMS provider
- Java Enterprise Edition requires that applications servers:
 - Include a JMS provider
 - Support JMS to access messaging middleware
- DataPower does not contain a Java run time, so it cannot connect directly to a JMS resource
 - Uses other protocols to connect to the Java resource
 - Customized approach available for only WebSphere Application Server and TIBCO
 - Supports JMS messages within WebSphere MQ messages

© Copyright IBM Corporation 2013

Figure 19-4. Java Message Service (JMS)

WE4013.0

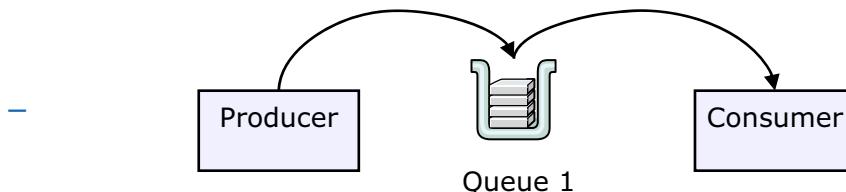
Notes:

Before JMS, you needed to use a proprietary client API to access messaging middleware. WebSphere Application Server V7 is a Java EE V5-compliant server that also provides a pure Java JMS V1.1 provider.

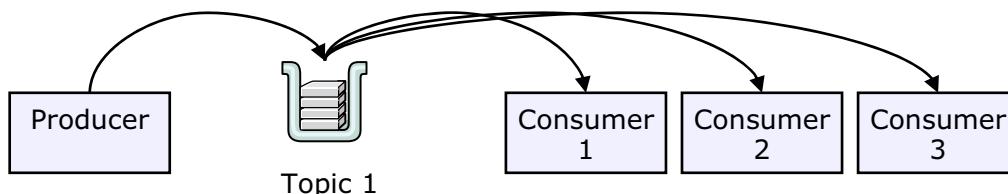
JMS models

JMS provides two messaging models:

- Point-to-point
 - Only one consumer might receive a particular message



- Publish/subscribe
 - Many-to-many
 - Consumers register to receive messages on a topic



© Copyright IBM Corporation 2013

Figure 19-5. JMS models

WE4013.0

Notes:

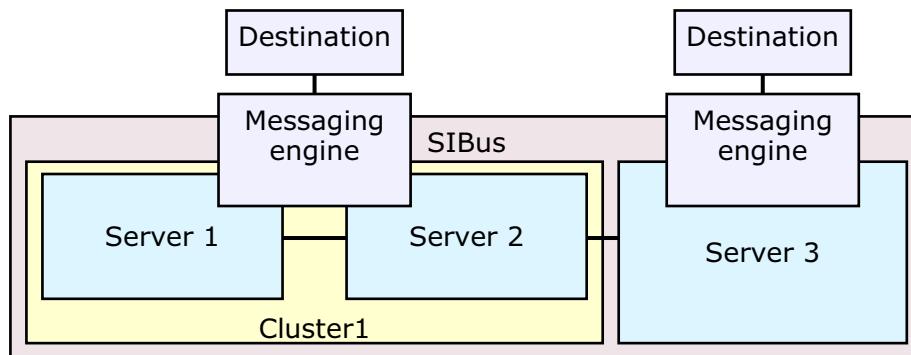
In the point-to-point messaging domain, it is not necessary for the message consumer and producer to be available at the same time. If either one is unavailable, the JMS queue keeps the message.

The publish/subscribe model defines one or more message consumers that subscribe to a particular topic. The JMS sender transmits a message to a particular topic.

Use **durable subscriptions** to deliver messages even if the message consumer is temporarily unavailable. This option stores the message in the topic, ready to be retrieved when the consumer becomes available.

WebSphere service integration bus (SIBus)

- Communication infrastructure that provides service integration through synchronous and asynchronous messaging; is part of WebSphere Application Server
- SIBus is an interconnection of bus members
 - Bus member is either a server cluster or server
 - Each bus member has an associated messaging engine.
 - Destinations (queue, topic) are linked to a messaging engine.
 - Logical construct
- When SIBus is used for JMS applications, it is referred to as a messaging bus



© Copyright IBM Corporation 2013

Figure 19-6. WebSphere service integration bus (SIBus)

WE4013.0

Notes:

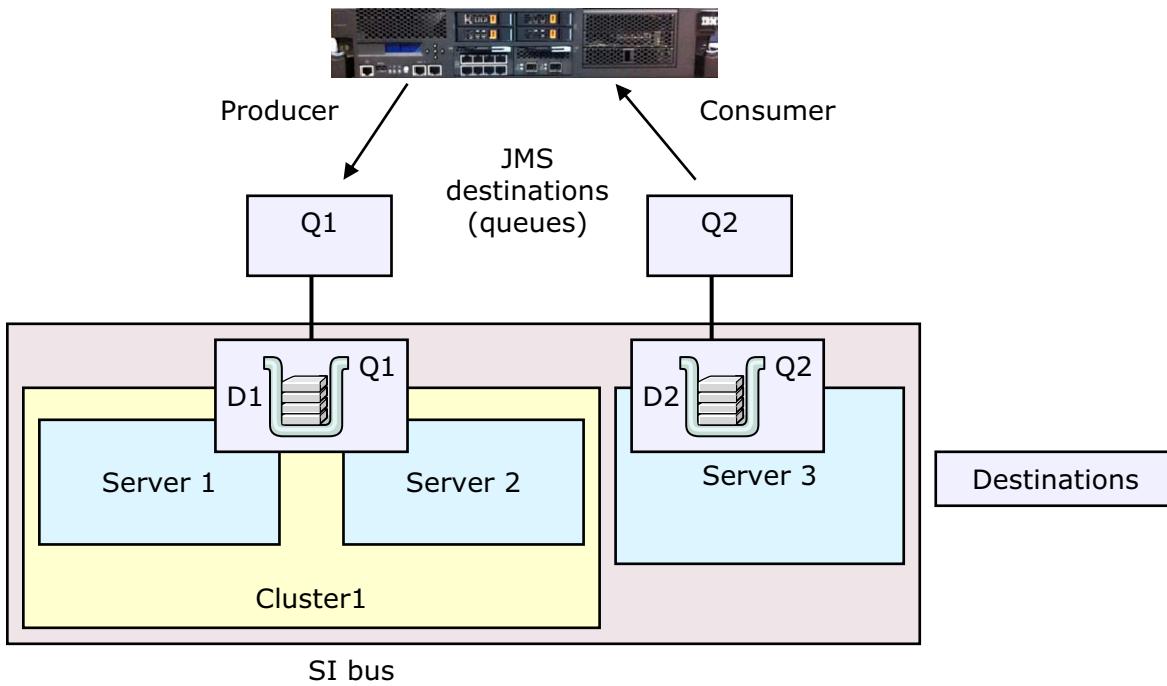
Any application can exchange messages with any other application by using a **destination**.

A message-producing application, that is, a **producer**, can produce messages for a destination regardless of which messaging engine the producer uses to connect to the bus.

A message-consuming application, that is, a **consumer**, can consume messages from a destination (whenever that destination is available) regardless of which messaging engine the consumer uses to connect to the bus.

JMS queue resources on SIBus

- JMS applications use JMS **queues** for point-to-point messaging



© Copyright IBM Corporation 2013

Figure 19-7. JMS queue resources on SIBus

WE4013.0

Notes:

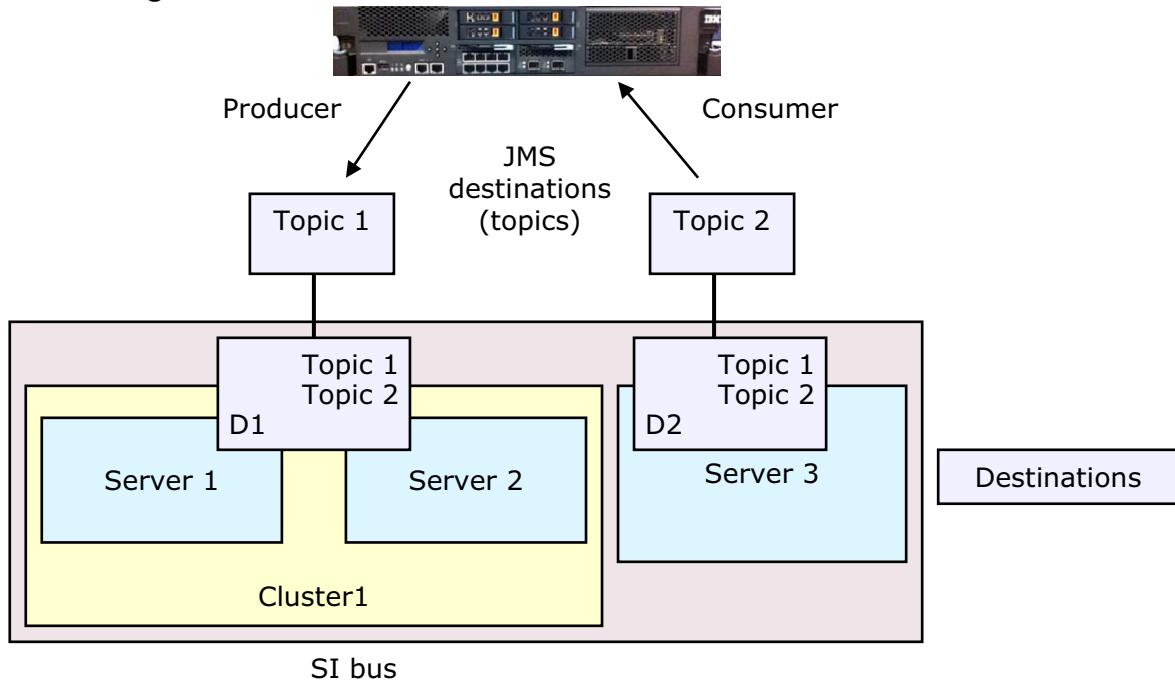
JMS destinations are configured on the application server. They encapsulate the name of the actual queue destination on the service integration bus.

D1 and D2 are destinations.

Q1 and Q2 are JMS queues.

JMS topic resources on SIBus

- JMS applications use JMS **topics** to publish and subscribe to messages



© Copyright IBM Corporation 2013

Figure 19-8. JMS topic resources on SIBus

WE4013.0

Notes:

An application such as DataPower interacts with a JMS topic, which is a JMS resource that is configured on the default messaging provider.

JMS applications receive messages only from a topic when it is connected to a server.

WebSphere JMS support

- DataPower supports JMS messaging to the default messaging provider in WebSphere Application Server V7 by using the **service integration bus**
 - Firmware uses the IBM JFAP (JetStream Formats and Protocols) to communicate with the service integration bus
- Scenario: Send SOAP messages over JMS by using the DataPower WebSphere JMS front side handler
 - SOAP/JMS web service implementation can read messages from a JMS queue and put response onto another queue

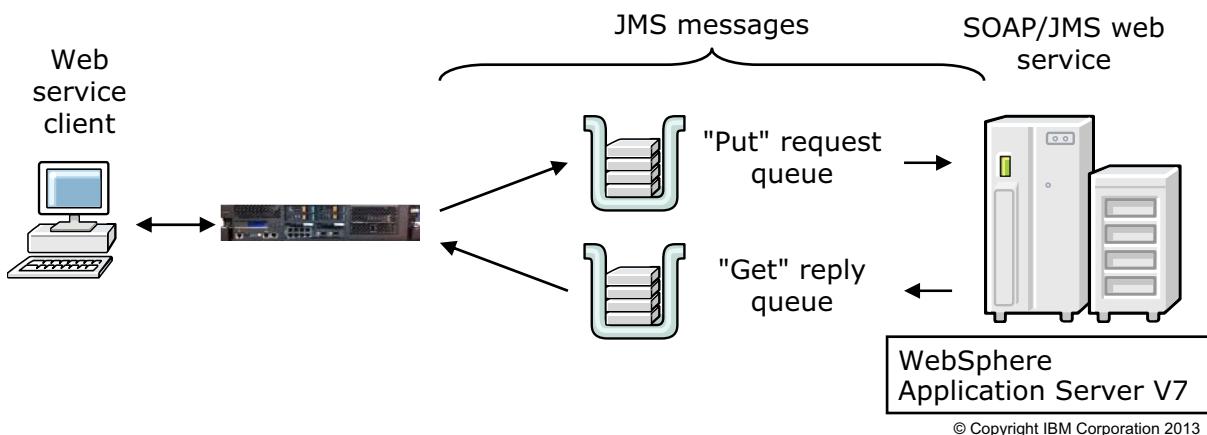


Figure 19-9. WebSphere JMS support

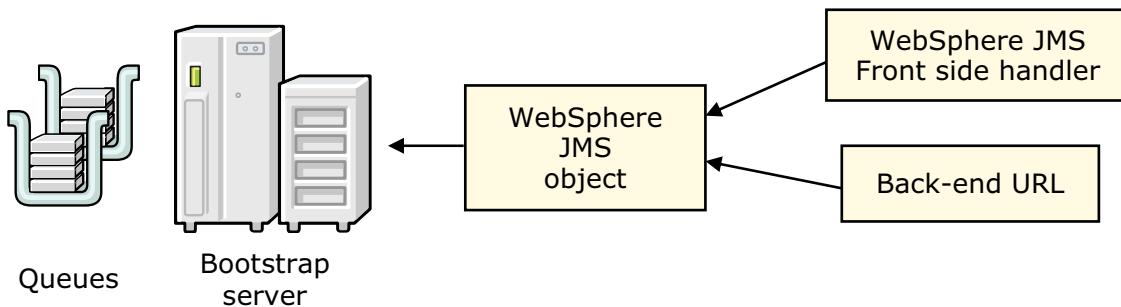
WE4013.0

Notes:

You must create and configure both a service integration bus and the JMS resources for a JMS client to invoke a SOAP/JMS web service that is running on WebSphere Application Server V7.

DataPower and WebSphere JMS interaction

- The **WebSphere JMS** object represents the connection to the bootstrap server on WebSphere Application Server
 - Remote WebSphere JMS applications are unable to communicate directly with a messaging engine on a bus – the bootstrap server enables this communication
- The **WebSphere JMS Front Side Handler** uses the WebSphere JMS object to communicate with specific queues
- The **Backend URL** also uses the WebSphere JMS object to manage communication to a back end JMS service



© Copyright IBM Corporation 2013

Figure 19-10. DataPower and WebSphere JMS interaction

WE4013.0

Notes:

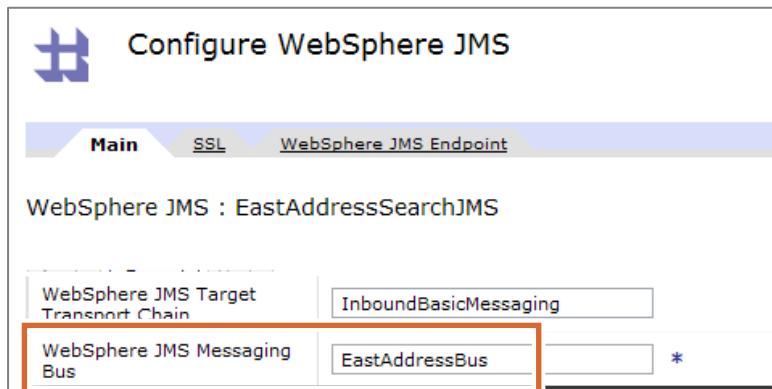
The default WebSphere JMS port is 7276.

The WebSphere JMS FSH and the back-end URL refer to the WebSphere JMS object, and they also define which queues on that object are being accessed.



WebSphere JMS object: Main (1 of 2) - messaging bus

- The WebSphere JMS server object requires the name of the **WebSphere JMS Messaging Bus**
- The **Transport Chain** is predefined on the application server



© Copyright IBM Corporation 2013

Figure 19-11. WebSphere JMS object: Main (1 of 2): Messaging bus

WE4013.0

Notes:

The WebSphere JMS messaging bus name is specified in WebSphere Application Server.

The options for the **WebSphere JMS Target Transport Chain** are:

- InboundBasicMessaging** (the default): specifies the predefined InboundBasicMessaging transport chain (JFAP-TCP/IP)
- InboundHTTPMessaging**: specifies the predefined InboundHTTPMessaging transport chain (tunnels JFAP by using HTTP wrappers)
- InboundHTTPSMessaging**: specifies the predefined InboundHTTPSMessaging transport chain (tunnels JFAP by using HTTPS wrappers)
- InboundSecureMessaging**: specifies the predefined InboundSecureMessaging transport chain (JFAP-SSL-TCP/IP)

There are more specifications on the tab; they are covered on the next slide.

WebSphere JMS object: Main (2 of 2) - optional settings

- User Name:**
Account name that is used to access the server
- Default Message Type:** Byte or Text, is useful when the message type cannot be determined from the headers
- Automatic Retry:** Attempts to reestablish a lost connection

| | |
|---|---|
| Admin State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments | <input type="text"/> |
| User Name | <input type="text"/> |
| Password | <input type="password"/> |
| Confirm Password | <input type="password"/> |
| Transactional | <input type="radio"/> on <input checked="" type="radio"/> off |
| Memory Threshold | 268435456 bytes |
| Maximum Message Size | 1048576 bytes |
| Default Message Type | Byte <input type="button" value="▼"/> |
| Total Connection Limit | 64 |
| Maximum number of Sessions per Connection | 100 |
| Automatic Retry | <input checked="" type="radio"/> on <input type="radio"/> off |
| Retry Interval | 1 seconds |
| Enable JMS-Specific Logging | <input type="radio"/> on <input checked="" type="radio"/> off |

© Copyright IBM Corporation 2013

Figure 19-12. WebSphere JMS object: Main (2 of 2): Optional settings

WE4013.0

Notes:

Transactional: Enables (**on**) or disables (**off**) transaction-based processing, in which messages are acknowledged only after the transaction succeeds. Transaction-based processing is disabled by default.

Memory Threshold: Specifies the maximum memory allocation for pending messages. Enter an integer (within the range 1048576 through 1073741824) that specifies the maximum memory (in bytes) that is allocated for pending messages. By default, the maximum memory allocation is set at 268,435,456 bytes.

Maximum Message Size: Specifies the maximum message size that the WebSphere Application Server JMS object can support. Enter an integer (with the range 0 through 1073741824) that specifies the maximum message size in bytes. By default, the maximum message size is set at 1048576 bytes. You can use the special value **0** to disable the enforcement of maximum message sizes.

Default Message Type:

- Byte:** The message payload is accessed as a Java byte array.

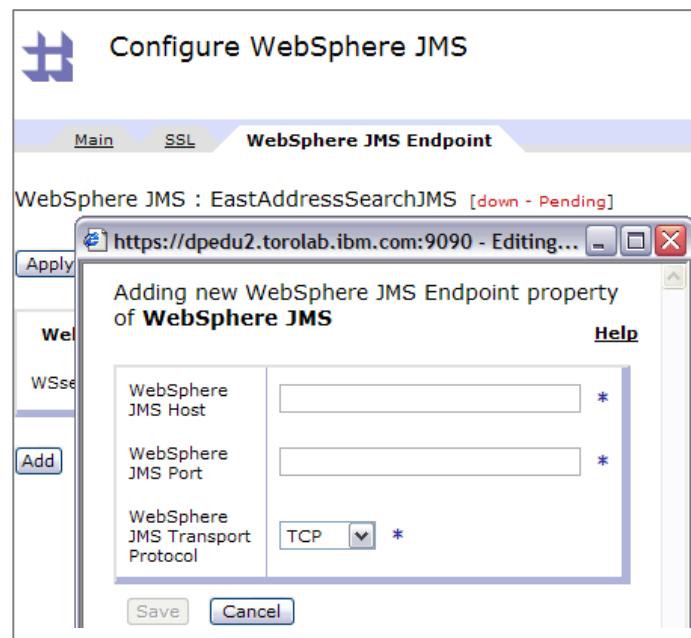
- **Text:** The message payload is accessed as a Java string value. This message type is suitable for SOAP and XML input.

Enable JMS-Specific Logging: Enables or disables an expanded JMS logging facility. Disabled is the default state.



WebSphere JMS object: WebSphere JMS Endpoint

- The **WebSphere JMS Endpoint** in the WebSphere JMS object is used to connect to a bootstrap server on WebSphere Application Server



© Copyright IBM Corporation 2013

Figure 19-13. WebSphere JMS object: WebSphere JMS Endpoint

WE4013.0

Notes:

The default WebSphere JMS port is 7276.

WebSphere JMS Transport Protocol: Selects the predefined transport chain that the WebSphere bootstrap server supports and is used for information exchange between the WebSphere JMS object and the bootstrap server. The choices are: TCP, SSL, HTTP, and HTTPS.

The screenshot shows the 'WebSphere Education' header with a blue navigation bar. Below it, the main title 'Communicating to WebSphere JMS' is displayed in blue. To the right is the 'IBM' logo.

Create a New:

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

Back side settings

Backend URL: http://9.65.242.185:9080/EastAddr *

Helper Buttons: MQHelper, TibcoEMSHelper, WebSphereJMSHelper (highlighted with a red box), IMSConnectHelper

© Copyright IBM Corporation 2013

Figure 19-14. Communicating to WebSphere JMS

WE4013.0

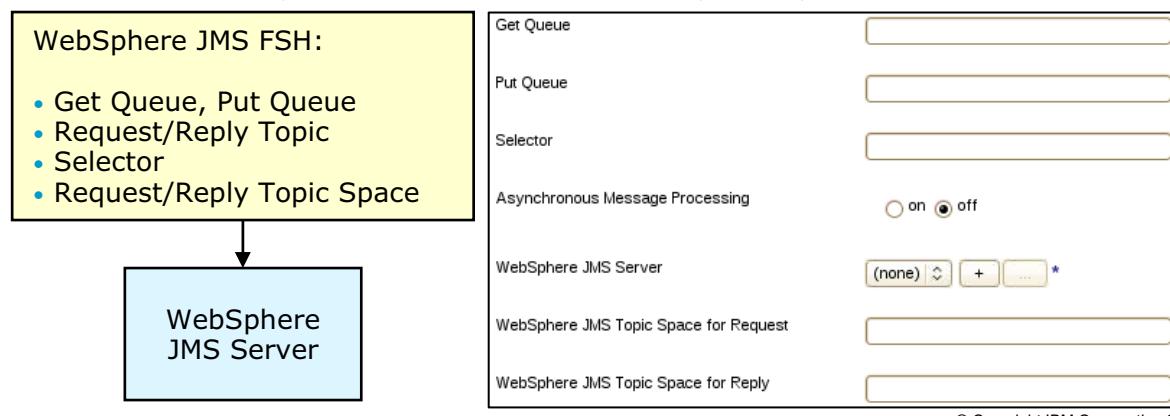
Notes:

A response queue is optional. The WebSphere JMS front side handler also supports one-way messaging.

WebSphere JMS Front Side Handler

The multi-protocol gateway and web service proxy service use the WebSphere JMS Front Side Handler (FSH) object to support JMS messaging

- Two messaging models use the Get and Put Queues:
 - Point-to-point messaging: Queues
 - Publish/subscribe messaging: Topics
- Select an existing WebSphere JMS server object – it defines how to connect to default messaging provider in WebSphere Application Server V7
- The **Selector** field allows you to filter the Get queue for messages to process
- The Request/Reply Topic space is used to uniquely identify topics in multiple destinations



© Copyright IBM Corporation 2013

Figure 19-15. WebSphere JMS Front Side Handler

WE4013.0

Notes:

The actual queues and topics that are defined in the WebSphere JMS FSH are mapped to destinations on the SIBus messaging engine.

The **Selector** field uses an SQL-like syntax for specifying the filter conditions.



WebSphere JMS back-end URL

- Multi-protocol gateway

| | |
|--|---------------------------------------|
| Backend URL: | dpwasjms://EastAddressSearchJMS/?Re * |
| <input type="checkbox"/> MQHelper <input type="checkbox"/> TibcoEMSHelper <input checked="" type="checkbox"/> WebSphereJMSHelper <input type="checkbox"/> IMSConnectHelper | |

Build a WebSphere JMS URL

| | |
|------------------------------|----------------------|
| Server: | EastAddressSearchJMS |
| Request Queue: | EastAddressQueueReq |
| Reply Queue: | EastAddressQueueResp |
| Request Topic Space: | |
| Response Topic Space: | |
| Selector: | |
| Timeout: | |

- Web service proxy

| | |
|---|----------------------|
| Remote | |
| Protocol | WebSphere JMS |
| DPWASJMS | EastAddressSearchJMS |
| <input type="button"/> + <input type="button"/> ... Remote URI : ?RequestQueue=EastAddressQueueReq | |

© Copyright IBM Corporation 2013

Figure 19-16. WebSphere JMS back-end URL

WE4013.0

Notes:

The generated JMS URL in this example is: `dpwasjms://EastAddressSearchJMS/?RequestQueue=EastAddressQueueReq&ReplyQueue=EastAddressQueueResp`

The generated URL is specific to DataPower.

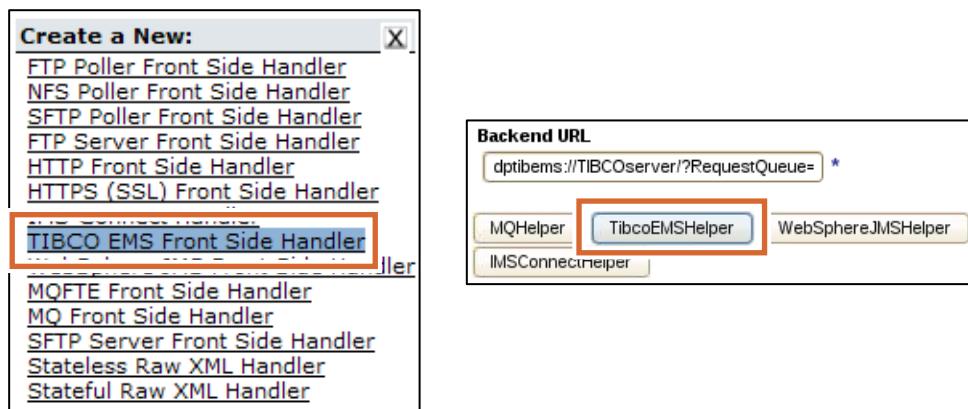
The actual queues and topics that are defined in the WebSphere JMS FSH are mapped to destinations on the SIBus messaging engine.

The **Selector** field uses an SQL-like syntax for specifying the filter conditions.



TIBCO EMS support

- DataPower supports JMS messaging with TIBCO EMS
- The TIBCO EMS object represents the connection to the JMS support in TIBCO EMS
 - Similar to the WebSphere JMS object
- Use a TIBCO EMS front side handler and a TIBCO EMS back-end URL
 - Similar to the equivalent WebSphere objects



© Copyright IBM Corporation 2013

Figure 19-17. TIBCO EMS support

WE4013.0

Notes:

JMS support with TIBCO EMS is similar to the support with WebSphere Application Server.

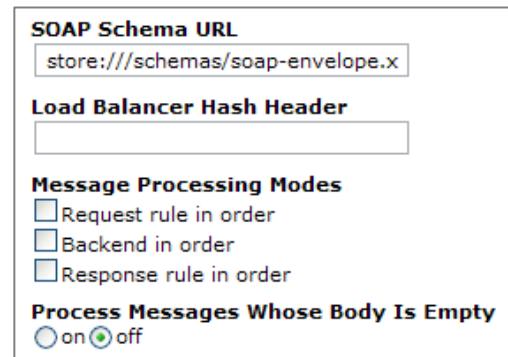
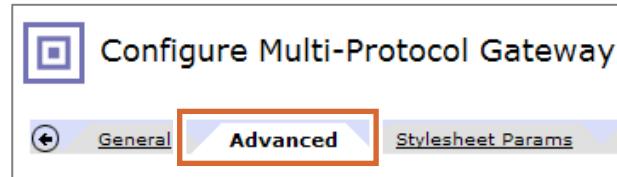


Ordered processing of JMS messages

- Enforces serial processing of JMS messages when the message:
 - Is retrieved from the front side queue and is presented to the request rule
 - Exits the request rule and is sent to the back side queue
 - Is retrieved from the back side queue and presented to the response rule

- The message order is the sequence in which messages are pulled from the front side request queue

- Messages are buffered, if needed, to maintain order
- Messages exiting a response rule might get buffered:
 - Messages sent to the front end queue are always delivered in the order in which they are pulled from the back side queue



© Copyright IBM Corporation 2013

Figure 19-18. Ordered processing of JMS messages

WE4013.0

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 processing the first message. The back-end request queue accepts whatever message arrives first, except in the case where you enforce back-end in order serial processing. In that case, the appliance buffers messages so that it sends messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Backend in order: Enforces the serial processing of messages that are delivered to the back-end request queue. If necessary, the appliance buffers messages that complete request rule processing out of order. The appliance delivers messages to the back-end request queue in the same order in which they were pulled from the front-end request queue.

Response rule in order: Enforces serial processing of messages for actions in the response rule. The appliance initiates and completes response rule processing for messages in the order in which they were pulled from the back-end reply queue. The appliance starts the response rule for the second message in the reply queue only after it completes processing the first message. The appliance always buffers messages so that it sends messages to the front-end reply queue in the same order in which they were pulled from the back-end reply queue.

Ordered messaging applies to WebSphere JMS and TIBCO EMS.

The **web service proxy** has the message processing modes under the **Advanced Proxy Settings** tab.

Unit summary

Having completed this unit, you should be able to:

- Configure a WebSphere JMS front-side handler to send JMS messages to the default messaging provider in WebSphere Application Server V7
- Configure a back-end WebSphere JMS URL to communicate with the default messaging provider in WebSphere Application Server V7
- Describe the components of the service integration bus on WebSphere Application Server V7

© Copyright IBM Corporation 2013

Figure 19-19. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. True or False: A point-to-point model can have only one consumer receive a particular message. In the publish/subscribe model, many consumers can register and receive messages.
2. True or False: The DataPower WebSphere JMS support allows you to send messages to non- WebSphere Application Server JMS providers.
3. Select the three mandatory steps to configure a DataPower WebSphere JMS front side handler:
 - A. Define the queues and destinations
 - B. Define the WebSphere JMS endpoint
 - C. Enter the name of the service integration bus
 - D. Turn on the JMS CloudBurst

© Copyright IBM Corporation 2013

Figure 19-20. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. **True.** A point-to-point model can have only one consumer receive a particular message. In the publish/subscribe model, many consumers can register and receive messages.
2. **False.** The DataPower WebSphere JMS works only with the default messaging provider in WebSphere Application Server V7.
3. **A, B, and C.** Select the three mandatory steps to configure a DataPower WebSphere JMS FSH:
 - ✓ A. Define the queues and destinations
 - ✓ B. Define the WebSphere JMS endpoint
 - ✓ C. Enter the name of the service integration bus
 - D. Turn on the JMS CloudBurst

© Copyright IBM Corporation 2013

Figure 19-21. Checkpoint answers

WE4013.0

Notes:

Unit 20. Course summary

What this unit is about

This unit summarizes the course, explains the class evaluation process, 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
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- 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
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study

© Copyright IBM Corporation 2013

Figure 20-1. Unit objectives

WE4013.0

Notes:

Course learning objectives

- Describe how WebSphere DataPower SOA Appliances are configured, including the role of XSL Transformations (XSLT)
- Configure a DataPower service to protect against a new class of XML-based threats
- Create a web service proxy to virtualize web service applications
- Implement web services security
- Create and configure cryptographic objects
- Configure Secure Sockets Layer (SSL) to and from WebSphere DataPower SOA Appliances
- Configure a multi-protocol gateway (MPGW) to handle multiple protocols for a single service
- Configure a service level monitoring (SLM) policy to handle service processing violations
- Enforce service level policies to manage traffic to and from WebSphere DataPower SOA Appliances
- Configure support for IBM WebSphere MQ and WebSphere Java Message Service (JMS)
- Use logs and probes to troubleshoot services
- Handle errors in service policies

© Copyright IBM Corporation 2013

Figure 20-2. Course learning objectives

WE4013.0

Notes:

Course review (1 of 3)

- The XG45/XI52/XI50 blade IBM WebSphere DataPower SOA Appliances secure XML and web service applications through **three main services**:
 - XML firewall
 - Web service proxy
 - Multi-protocol gateway
- The **web application firewall** is used to secure and offload processing from web-based applications
- The **XML firewall service** secures XML-based applications against XML threats
 - In addition to the features inherited from the XSL proxy, the XML firewall supports message-level security processing actions and content based routing
- **XML threats** use weaknesses in regular firewalls, XML processors, and applications that work with XML data
 - The XG45/XI52/XI50 blade appliances provide high-performance services that protect against a wide range of XML-based attacks

© Copyright IBM Corporation 2013

Figure 20-3. Course review (1 of 3)

WE4013.0

Notes:

Course review (2 of 3)

- Clients can connect to the back-end service through the **multi-protocol gateway**, over a number of different transport and application protocols
 - Protocol handlers are available for HTTP and HTTPS protocols, FTP, raw XML messages, TIBCO EMS, and IBM WebSphere MQ systems
 - Includes all capabilities of XML firewall type
- The **Web Service Proxy** provides features tailored to web service applications
 - Service-level monitoring and processing policies can be applied at the service definition, interface (portType), and operation level
 - Web service virtualization acts as a single endpoint for a group of operations that are implemented by different back-end services
- The **Encrypt**, **Decrypt**, **Verify**, and **Sign** processing actions provide XML encryption and digital signatures down to the message field level
- Services use the **cryptographic** tools to configure SSL communication, XML encryption, and digital signatures

© Copyright IBM Corporation 2013

Figure 20-4. Course review (2 of 3)

WE4013.0

Notes:

Course review (3 of 3)

- The **Authentication, Authorization, and Auditing** processing action restricts access to resources and monitors XML application traffic
 - A wide range of message-level security specifications are supported, including WS-Security, SAML, XACML, SPNEGO, LTPA, OAuth, Kerberos, and others
- **Message monitors** and **service-level monitors** can filter, shape, and throttle XML traffic through a DataPower SOA appliance
- **WebSphere MQ** is a protocol that is commonly used for DataPower front end or back end transport
 - WebSphere MQ header manipulation and transactionality is supported
- DataPower **JMS** support works with WebSphere JMS and TIBCO EMS

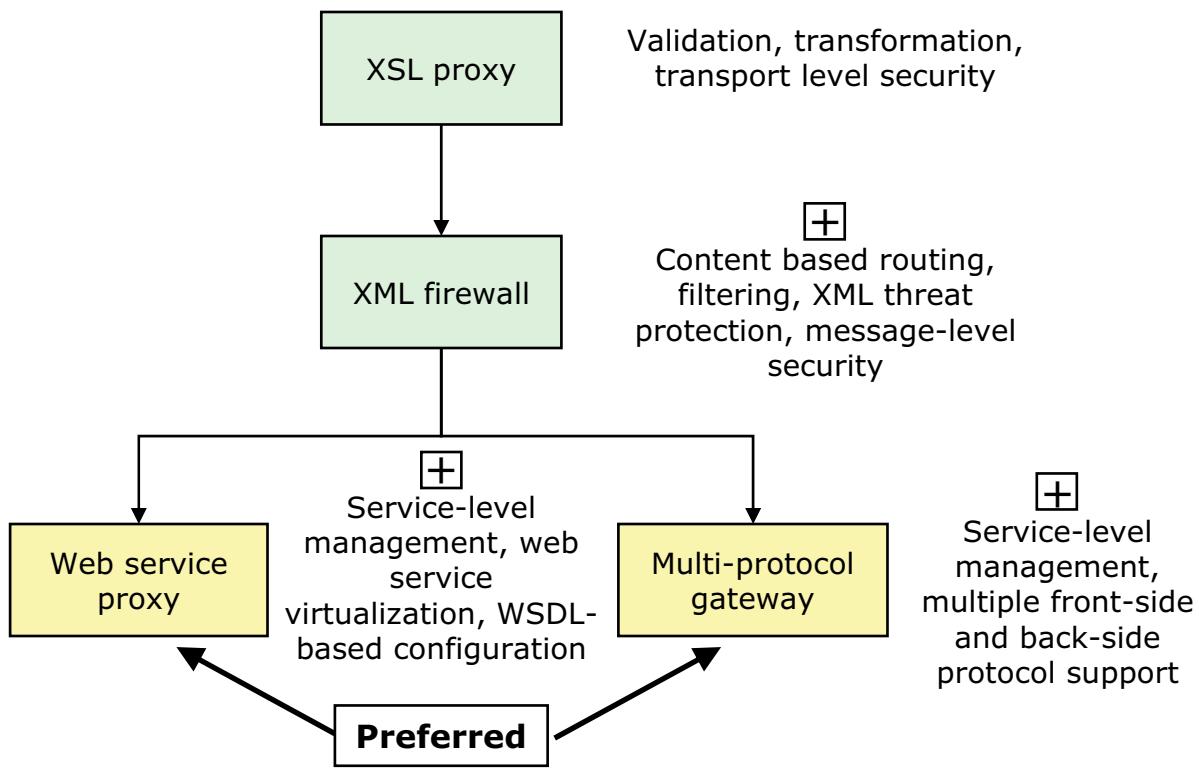
© Copyright IBM Corporation 2013

Figure 20-5. Course review (3 of 3)

WE4013.0

Notes:

DataPower services feature hierarchy



© Copyright IBM Corporation 2013

Figure 20-6. DataPower services feature hierarchy

WE4013.0

Notes:

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

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

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

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

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

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

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

Class evaluation

- Your comments about this class are useful to WebSphere Education
- Feedback on the site, curriculum, and instructor tell what was good about the class and what can be improved
- Take the time to complete the course evaluation on the IBM Training website: <http://www.ibm.com/training/osart>

© Copyright IBM Corporation 2013

Figure 20-7. Class evaluation

WE4013.0

Notes:



Lab exercise solutions

- Solutions are available in the `dplabs` subdirectory:

`<labfiles>/dplabs/Solution`

- Remember to change
 - Port numbers
 - Back-end server (**Network > Interface > DNS Settings > Static Hosts**)
 - Front IP addresses (**Network > Interface > Host Alias**)

© Copyright IBM Corporation 2013

Figure 20-8. Lab exercise solutions

WE4013.0

Notes:



To learn more on this subject

- WebSphere Education website:
 - www.ibm.com/websphere/education
- Training paths:
 - www.ibm.com/software/websphere/education/paths/
 - Select **WebSphere Application Integration > WebSphere DataPower**
 - Identify the next courses in this sequence
- Resource Guide:
 - Contains information on many useful sources of information
 - Many of these sources are free
 - See the handout in your class materials, or download a copy
 - <http://www.ibm.com/developerworks/wikis/display/WEinstructors/WebSphere+Resource+Guide>
- Email address for more information:
 - websphere_skills@us.ibm.com

© Copyright IBM Corporation 2013

Figure 20-9. To learn more on this subject

WE4013.0

Notes:

Additional DataPower education

Developer: Advanced

- Available as individual SPVCs or consolidated standard course
- Covers topics such as:
 - DataPower variables, extensions, and flow control
 - Access control and custom security policies
 - FTP and DataPower
 - SQL and DataPower
 - RESTful services on 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 2013

Figure 20-10. Additional DataPower education

WE4013.0

Notes:

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.



References

- DataPower Information Center
 - <http://pic.dhe.ibm.com/infocenter/wsdatap/v5r0m0/index.jsp>
- IBM Redbooks:
 - <http://www.redbooks.ibm.com>
 - Search on “DataPower”
- DataPower SOA Appliance Customer Forum:
 - <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1198>
- developerWorks articles:
 - <http://www.ibm.com/developerworks/>
 - Search on “DataPower”

© Copyright IBM Corporation 2013

Figure 20-11. References

WE4013.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Explain how the course met its learning objectives
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study

© Copyright IBM Corporation 2013

Figure 20-12. Unit summary

WE4013.0

Notes:

Appendix A. Web application firewall service

What this unit is about

This course provides an appendix unit and an appendix exercise on the web application firewall (WAF). Since the WAF is not commonly used, this material is not covered in class. It is provided for your self-study.

In this unit, you learn how to create a web application firewall to offload tasks and protect access to their web applications.

What you should be able to do

After completing this unit, you should be able to:

- Configure a web application firewall to protect a back-end web application
- Use a AAA policy to protect access through the web application firewall
- Validate parameters from an HTTP request by using name-value profiles
- Protect the web application from phishing attacks by using built-in threat protection

How you will check your progress

- Checkpoint
- Exercise A: Creating a firewall and HTTP proxy for a web application (optional)



Unit objectives

After completing this unit, you should be able to:

- Configure a web application firewall to protect a back-end web application
- Use a AAA policy to protect access through the web application firewall
- Validate parameters from an HTTP request by using name-value profiles
- Protect the web application from phishing attacks by using built-in threat protection

© Copyright IBM Corporation 2013

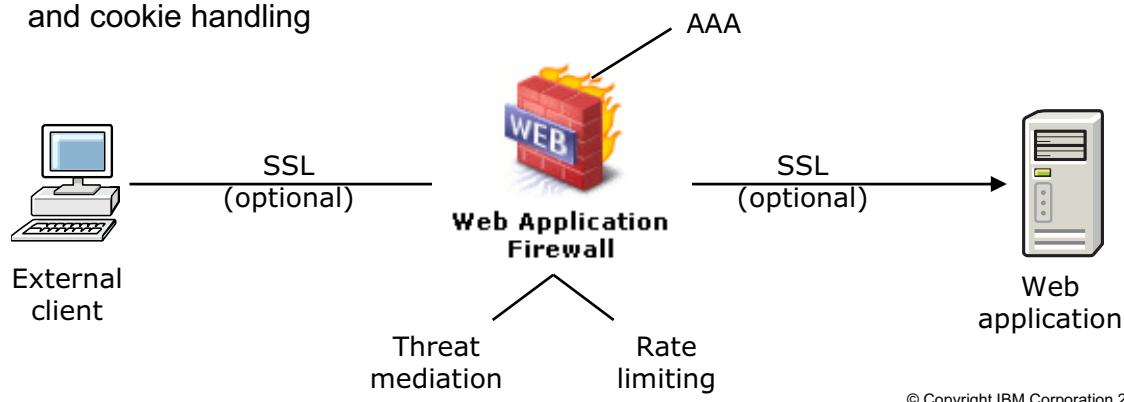
Figure A-1. Unit objectives

WE4013.0

Notes:

What is a web application firewall?

- A web application firewall is used to secure and offload processing from web-based applications
 - Proxies back-end web application
 - Listens for requests on multiple Ethernet interfaces and TCP ports
 - Provides threat mediation, AAA, and SSL
 - Can limit the number of requests or simultaneous connections to web application
 - Session management
 - Web-based validation and cookie handling



© Copyright IBM Corporation 2013

Figure A-2. What is a web application firewall?

WE4013.0

Notes:

The DataPower appliance does not host the web application. It runs the web application firewall service.

Proxying web-based applications prevents the client from connecting to the back-end application directly.

Threat mediation, AAA, and SSL are provided by creating their respective DataPower objects. These objects can be reused from other configurations.

A web application firewall can validate name-value pairs that are passed in the HTTP request. It can also check for malicious content in these parameters.

When to use the web application firewall

- The web application firewall is primarily used for offloading functions from web-based service applications that execute on application servers
 - Performs AAA functions, such as authentication, by HTTP basic authentication
 - Protects against malicious attacks by URL-encoded strings
 - Secures communication to the DataPower appliance by SSL
 - DataPower appliance can also communicate securely with the back-end web application
- Designed to support both HTTP and SOAP/HTTP traffic
 - Better support for HTTP-based traffic
 - Little support for processing rules, which are typically used for XML-based traffic
- For XML-only traffic, use other DataPower services such as XML firewall, web service proxy, or multi-protocol gateway

© Copyright IBM Corporation 2013

Figure A-3. When to use the web application firewall

WE4013.0

Notes:

You can URL-encode HTTP post requests. URL encoding ensures that ASCII characters that are not supported in URL strings are converted to a URL character.

Comparison with other DataPower services

- DataPower services are designed to offload functions performed by back-end applications
- The following table describes the purpose and function of each service

| Service | Purpose | Features |
|--------------------------|--|--|
| Web application firewall | <ul style="list-style-type: none"> • Send and receive HTTP traffic to and from web applications • Limited support for XML traffic | <ul style="list-style-type: none"> • Functions: AAA, HTTP threat protection, rate limit (limit requests), SSL • Front-side and back-end transport: HTTP |
| XML firewall | <ul style="list-style-type: none"> • Send and receive XML traffic over HTTP to and from XML-based applications | <ul style="list-style-type: none"> • Functions: AAA, XML encryption, XML signatures, XML threat protection, content based routing, SSL, and monitors • Front-side and back-end transport: HTTP |
| Web service proxy | <ul style="list-style-type: none"> • Receives web services (SOAP-based) traffic over multiple transports and forwards to web service applications over HTTP or IBM WebSphere MQ | <ul style="list-style-type: none"> • Functions: Same as XML firewall, plus service-level monitoring • Front-side transport: HTTP, WebSphere MQ, FTP, JMS, Tibco EMS, Raw XML • Back-end transport: HTTP, WebSphere MQ |
| Multi-protocol gateway | <ul style="list-style-type: none"> • Send and receive XML traffic over multiple transports | <ul style="list-style-type: none"> • Functions: Same as XML firewall • Front-side and back-end transport: HTTP, WebSphere MQ, FTP, JMS, TIBCO EMS, raw XML |

© Copyright IBM Corporation 2013

Figure A-4. Comparison with other DataPower services

WE4013.0

Notes:

The list of the functions of each service is not exhaustive. Only the main features are emphasized.

The web application firewall can reuse the AAA policy and SSL proxy profile objects that other services create.

HTTP threat protection is different from the XML threat protection.

Example scenario that uses web application firewall

- An online business allows customers to purchase tickets for concerts and sporting events
- Users are required to supply a user name and password to access the site
- Due to the demand of certain tickets, the requests must be rate-limited to ensure that the system is not overloaded with too many requests
- Each request is also checked for HTTP-based attacks that use a URL string and SQL injections
- Due to the sensitivity of the data, all transactions are encrypted by using SSL, and session timeouts are also enforced

© Copyright IBM Corporation 2013

Figure A-5. Example scenario that uses web application firewall

WE4013.0

Notes:

DataPower implementation for scenario

- Create a web application firewall to accept HTTP requests over SSL and send them to the back-end web application if they are processed correctly
 - Configure SSL from the client to the DataPower appliance
 - Authenticate users against an LDAP server by using the credentials in the HTTP basic authentication header
 - Set up HTTP threat protection to check for malicious requests
 - Configure rate limit for the service to ensure no more than 1000 requests per second from any user
 - Any request that exceeds this threshold is dropped

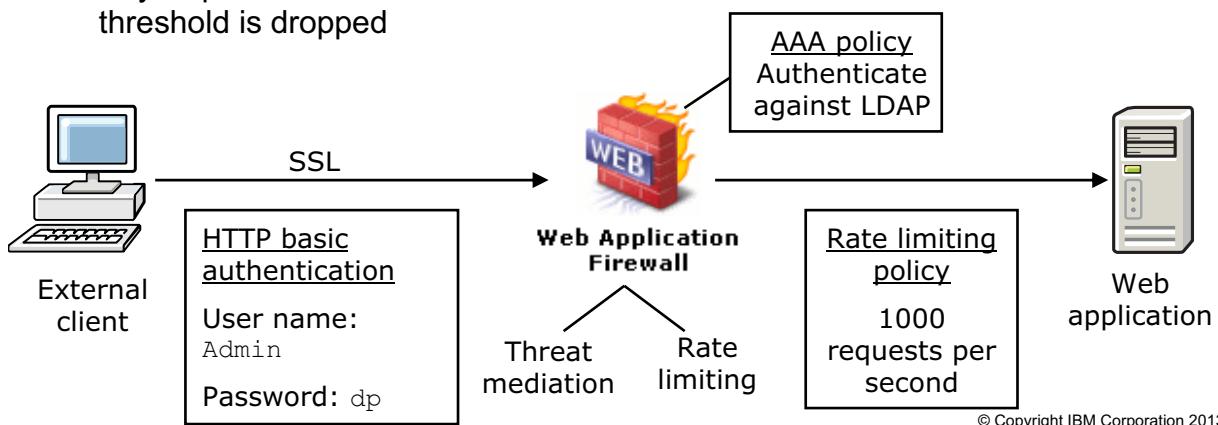


Figure A-6. DataPower implementation for scenario

WE4013.0

Notes:

You can configure SSL either by creating or by using an existing SSL proxy profile object.

You can configure authentication by creating a AAA policy that uses the HTTP basic authentication credentials and validates them against an LDAP server.

HTTP threat protection can be set by checking for attacks on the URL query strings.

You can configure rate limiting to set the policy on the number of requests that are allowed.



Configuration steps that use the wizard

Perform the following steps to create and configure a web application firewall:

1. From the control panel, click the **Web Application Firewall** icon
2. Click **Add Wizard**
 - Each wizard page aids in creating the required objects for the web application firewall
3. Enter the destination host and TCP port number
 - Optional: support SSL
4. Enter the Ethernet interface and TCP port number to which the web application firewall listens for requests
 - Optional: support SSL
5. Create or select an existing AAA policy
6. Add or use an existing header name-value pair profile for either requests or responses
7. Specify HTTP threat protection and how to process cookies



© Copyright IBM Corporation 2013

Figure A-7. Configuration steps that use the wizard

WE4013.0

Notes:

The wizard approach to creating a web application firewall configures the most commonly used functions. When you used the wizard to create the web application firewall, you were able to add or modify the entire configuration.

The following objects can be reused from other services:

- AAA policy
- SSL proxy profile
- Matching rule
- Error rule

The screenshot shows the 'WebGUI Configuration page (1 of 2)' for a service named 'AddressWAF'. The page is divided into several sections:

- General Configuration:** Includes fields for 'Name' (AddressWAF), 'Summary', and a note about configuration settings.
- Back side settings (2):** Includes 'Remote Host' (1.2.3.4) and 'Remote Port' (80).
- Front side settings (3):** Includes a table for 'Source Addresses' with one entry: IP 0.0.0.0, Port 9080, SSL (off), Action Remove.
- Top-level objects (4-7):** A box labeled 'Top-level objects that the web application firewall configures' contains the following objects:
 - XML Manager:** Default object.
 - SSL Proxy Profile:** (none) object.
 - Default Error Policy:** (none) object.
 - Application Security Policy:** AddressWAF_security_policy object.

Figure A-8. WebGUI Configuration page (1 of 2)

WE4013.0

Notes:

1. The **Web Application Firewall Name** is given to the service.
2. The **Back Side Settings** specify the host address and TCP port number where the requests are forwarded.
3. The **Front Side Settings** contain a list of IP (Ethernet interface) and TCP port numbers to which the service listens for requests. If **Use SSL** is selected, then the SSL proxy profile contains the server-side SSL proxy configuration.
4. The **XML Manager** object handles style sheet and document processing options. All DataPower services use this **default** XML Manager object.
5. The **SSL Proxy Profile** specifies the client and server profiles (if SSL is required) that are used for communication. Both the back and front side SSLs are supported through this object.
6. The **Default Error Handling Policy** specifies the actions that take place when errors occur in the back-end web application.
7. The **Security Policy** specifies the policy for the web application firewall.




WebGUI Configuration page (2 of 2)

| Advanced settings | |
|---|--|
| <p>Settings that you can modify depending on your environment. For most environments, the default values are appropriate.</p> | |
| Timeout Settings | Streaming Settings |
| Basic timeouts apply during a transaction. Persistent timeouts apply between transactions. | |
| Front Side Timeout <input type="text" value="120"/> seconds * | Streaming Settings Whether to stream or buffer requests (from client) and responses (from server). |
| Back Side Timeout <input type="text" value="120"/> seconds * | Stream Output to Front <input type="button" value="Buffer Messages"/> |
| Front Persistent Timeout <input type="text" value="180"/> seconds * | Stream Output to Back <input type="button" value="Buffer Messages"/> |
| Back Persistent Timeout <input type="text" value="180"/> seconds * | |
| Protocol Settings | Security Settings |
| HTTP versions set the version the client and server expect. Priority controls scheduling. Follow redirects controls whether to resolve redirects. | |
| HTTP Response Version <input type="button" value="HTTP 1.1"/> | Normalize URI <input checked="" type="radio"/> on <input type="radio"/> off |
| HTTP Version to Server <input type="button" value="HTTP 1.1"/> | Rewrite Errors <input checked="" type="radio"/> on <input type="radio"/> off |
| Service Priority <input type="button" value="Normal"/> | Delay Error Messages <input checked="" type="radio"/> on <input type="radio"/> off |
| Follow Redirects <input checked="" type="radio"/> on <input type="radio"/> off | Delay Error Messages for the Duration of <input type="text" value="1000"/> msec * |

Configure the timeout period, supported HTTP protocols, streaming settings, and request/response security enforcement

© Copyright IBM Corporation 2013

Figure A-9. WebGUI Configuration page (2 of 2)

WE4013.0

Notes:

If **Request** or **Response security** is turned **off**, then requests are passed through without any enforcement of security policies.

Normalizing the URI ensures that all unsupported URI characters are escaped.

The HTTP protocol version can be overridden in the web application security policy.

Front and back side settings

- The web application firewall rewrites the client host address and port number URL with the remote host address and port number
 - Under “Front side settings,” select the **use SSL** check box to require clients to connect by using SSL
 - Need to configure a server-side SSL proxy profile

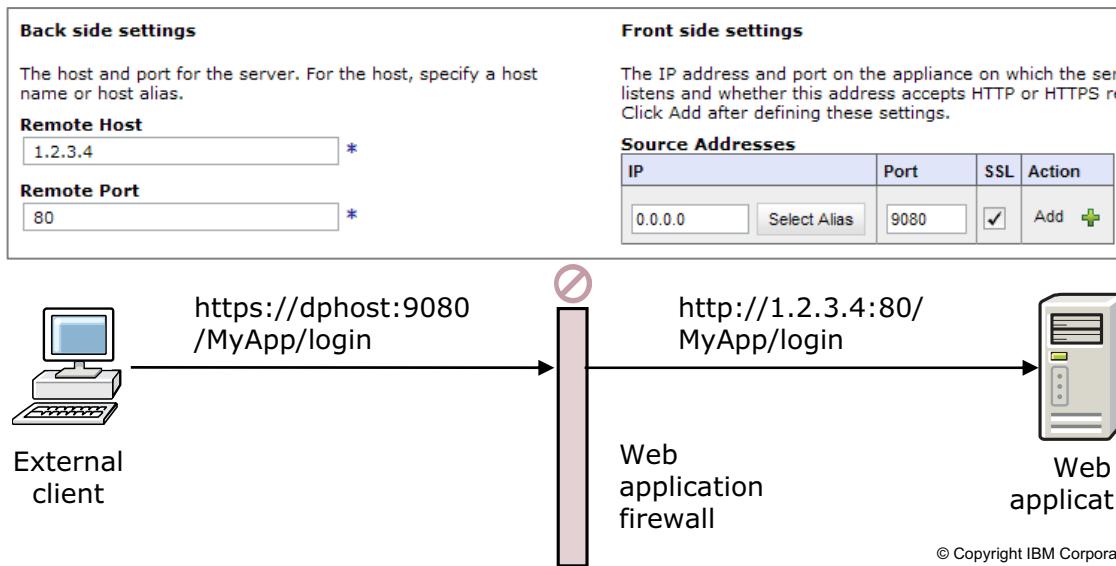


Figure A-10. Front and back side settings

WE4013.0

Notes:

The front side settings have SSL **on**, so the external client must connect by using **https**.

The part of the URI after the port number is the same URI that is sent to the back-end web application.

In this scenario, the web application firewall service must be configured as an SSL server.

WebSphere Education

Configure SSL proxy profile

Configure an SSL proxy profile when SSL communication is required

- SSL proxy profile contains information about client and server-side SSL communication
- Directionality: reverse (server), forward (client), or two-way (client and server)
- Crypto profile: contains references to crypto certificates and keys that are used during SSL communication

SSL Proxy Profile: AddressSearchSSL [up]

[Apply](#) [Cancel](#) [Undo](#)

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

| | |
|--------------------------------------|---|
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| SSL Direction | <input type="button" value="Reverse"/> * |
| Reverse (Server) Crypto Profile | <input type="button" value="AliceCryptoProfile"/> + ... * |
| 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 Optional | <input type="radio"/> on <input checked="" type="radio"/> off |
| Always Request Client Authentication | <input type="radio"/> on <input checked="" type="radio"/> off |

© Copyright IBM Corporation 2013

Figure A-11. Configure SSL proxy profile

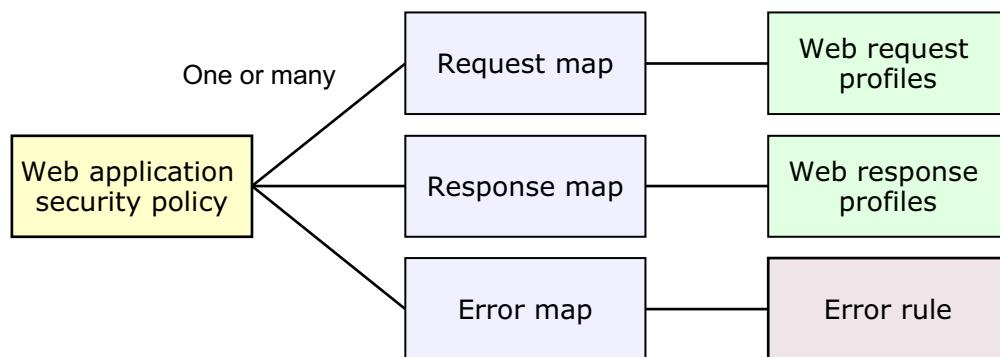
WE4013.0

Notes:

If clients are required to communicate with the web application firewall securely (**Client > DataPower**), then select **reverse** for the direction and select a reverse (server) cryptographic profile. If the web application firewall acts as both client and server in SSL communication, select **both-ways** for directionality.

Web application security policy

- Web application security policy is defined by using three maps:
 - Request: references a web request profile
 - Response: references a web response profile
 - Error: references an error rule
- Multiple maps can be defined per request and response
 - Maps execute based on matching rule definition
- Each profile implements the security policy configuration
 - AAA, HTTP threat protection, rate limit, session management, and more



© Copyright IBM Corporation 2013

Figure A-12. Web application security policy

WE4013.0

Notes:

The web application security policy consists of a set of maps. There are three types of map: **Request**, **Response**, and **Error**. More than one type of map can be defined. A map is chosen to run based on the matching rule. More than one map can run for a request.



Web application security map

- A web application firewall security policy must define a request and response map
 - Error map is optional
- Each map references a matching rule and map rule
 - The matching rule object specifies the conditions to execute the map rule
 - Can reuse object from previously defined rules, for example, match URLs with */services
 - The map rule defines the security policy
- Multiple maps can be configured with order in the profile

| Match | Rule | Reorder | Delete |
|--------------------------|--------------------|---------|--------|
| AddressWAF_match_request | AddressWAF_request | up down | Remove |

Matching Rule

Request Profile

© Copyright IBM Corporation 2013

Figure A-13. Web application security map

WE4013.0

Notes:

Select a matching rule and map rule, and then click **Add Request Map** to add the map to the web application security policy.

The screen capture in the slide shows a configured request map. Open the web application security policy, and select the **Request maps** tab to view this page.

Request maps

After completing this topic, you should be able to:

- Configure the request map objects

© Copyright IBM Corporation 2013

Figure A-14. Request maps

WE4013.0

Notes:



Configure request map profile

- Using the map rule, you can configure a profile that contains:
 - AAA policy
 - Rate limiting
 - Threat protection
 - Supported HTTP methods and protocols
 - HTTP parameter validation
 - Cookie handling
 - Session management
 - Accepted content types



| Main | | Profile | Methods & Versions | Processing | Name-Value |
|--|---------------------------------------|-------------------------------------|--|--------------------------------|-----------------------------|
| Web Request Profile: AddressWAF_request [up] | | | | | |
| <input type="button" value="Apply"/> | <input type="button" value="Cancel"/> | <input type="button" value="Undo"/> | Export | View Log | View Status |
| Administrative State | | | <input checked="" type="radio"/> enabled | <input type="radio"/> disabled | |
| Comments | | | <input type="text"/> | | |
| Style | | | <input type="button" value="Admission"/> | | |

© Copyright IBM Corporation 2013

Figure A-15. Configure request map profile

WE4013.0

Notes:

The **Rule** field is used to open the map rule profile.

Select the various tabs in the map rule to configure the objects. Most of the objects are available in the **Profile** tab, specifically the items that are listed in the slide.

In the **Main** tab, the **Style** field enables you to specify the rules for multiple profiles. The value **Admission** specifies that if one map profile runs correctly, then the web application security policy passes. The value **Prerequisite** is used when a map profile that is already passed requires another map profile to succeed for the entire web application security policy to pass.



Map rule objects: Request

- **Allow SSL:**
{Allow | Require | Deny}
values for client-side SSL
- **AAA Policy:** select an existing AAA policy from the context menu or create a AAA policy
- **Shared Secret Key:** used for both signing and encrypting
- **Rate Limiting:** used to enforce limits on the number of requests and connections
- **Access Control List:** list of valid IP addresses that are allowed to access the service
- **Error Policy:** actions to take if error occurs in profile
- **Session Policy:** specify the start page and session timeout
- **Content-Type List:** the acceptable MIME headers for each message

The screenshot shows the 'Profile' tab selected in the 'Web Request Profile: AddressWAF_request' dialog. The configuration includes:

| Object | Action |
|---------------------|----------------|
| Allow SSL | Allow |
| AAA Policy | AddressWAF_AAA |
| Shared Secret Key | (none) |
| Rate Limiting | (none) |
| Access Control List | (none) |
| Error Policy | (none) |
| Session Policy | AddressWAF_SP |
| Content-Type List | (empty) |

© Copyright IBM Corporation 2013

Figure A-16. Map rule objects: Request

WE4013.0

Notes:

The three values for **Allow SSL** are specific to client-side SSL only:

- **Allow:** neutral to the presence of SSL
- **Require:** SSL is required
- **Deny:** SSL is not allowed

Use the **Content-Type List** to specify the list of acceptable MIME headers for each message that passes through the web application firewall.

Rate limiting

- The rate limiter object is used to restrict the number of request messages and connections within a time interval
 - Similar to a message monitor, but uses different objects
 - Requests over the specified rate can be rejected, shaped, or logged
 - Limit the number of users that are connected and number of connections per user

| | | | |
|------------------------|---|-------------------------|---|
| Name | <input type="text"/> | * | Request limit and action for violation of limit |
| Admin State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled | | |
| Comments | <input type="text"/> | | |
| Rate | <input type="text"/> 500 | transactions-per-second | |
| Enforcement Style | <input type="button" value="Reject"/> | | Specify maximum simultaneous users and connections per user |
| Distinct Users | <input type="text"/> 10000 | * | |
| Concurrent Connections | <input type="text"/> 0 | * | |

© Copyright IBM Corporation 2013

Figure A-17. Rate limiting

WE4013.0

Notes:

A value of 0 in the **Concurrent Connections** field implies that there is no limit on the number of user connections.

The rate value is applied for each user.

Error policy

The error policy object contains actions to execute when errors occur during execution of the web application security policy

- The **Mode** specifies the **Error** action:
 - **Error rule:** run a specified error rule
 - **Proxy:** enter a URL to fetch its contents to return to the client
 - **Redirect:** redirect the client to the specified URL
 - **Standard:** do nothing
- **Monitor:** create or use an existing monitor to count the number of error messages
 - Independent of the mode selected

Error Policy

Apply Cancel

Name: *

Administrative State: enabled disabled

Comments:

Mode: Standard

Monitor: (none)

© Copyright IBM Corporation 2013

Figure A-18. Error policy

WE4013.0

Notes:

Changing the value of the **Mode** field enables new fields in the web page. For example, selecting the mode **Error Rule** shows a new field that is called **Error Rule**, where you can select an existing error rule or create a new one.

The web application firewall also defines a default error policy that is applicable for the entire service. The error policy that is defined in the request map overrides the service-wide error policy, if it was defined.

The difference between the proxy and redirect occurs in the client web browser. Using redirect, the client is forwarded to the URL that is returned. The proxy mode keeps the client at the same URL, but the contents of the URL that was entered in the web page are returned.



Session policy

- The session policy object is used to specify a list of allowable start pages and session timeout
 - Start Pages:** Select a matching rule for specifying the set of allowable start pages

Session Management Policy

| | | |
|------------------------------|---|---|
| Name | <input type="text"/> | * |
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled | |
| Comments | <input type="text"/> | |
| Auto Renew | <input checked="" type="radio"/> on <input type="radio"/> off | |
| Session Lifetime | <input type="text" value="3600"/> | seconds |
| Address Independent Sessions | <input type="radio"/> on <input checked="" type="radio"/> off | |
| Start Pages | <input type="text" value="(none)"/> | <input type="button" value="+"/> <input type="button" value="..."/> |

© Copyright IBM Corporation 2013

Figure A-19. Session policy

WE4013.0

Notes:

The **Start Pages** field allows you to restrict the URI that can be sent to the web application firewall. Use a matching rule to specify valid URIs that can be sent.



Name-value profiles (1 of 2)

- A name-value profile is used to validate name-value pairs that are formatted as `operation=1&input1=CA` and received from:
 - HTTP headers
 - Cookie values
 - URL-encoded query strings
 - URL-encoded request messages
- These values are typically submitted as part of an HTTP request or response
- Name-value profiles allow you to validate the name and ensure that it contains allowable values
 - Names and values are validated by using a regular expression

© Copyright IBM Corporation 2013

Figure A-20. Name-value profiles (1 of 2)

WE4013.0

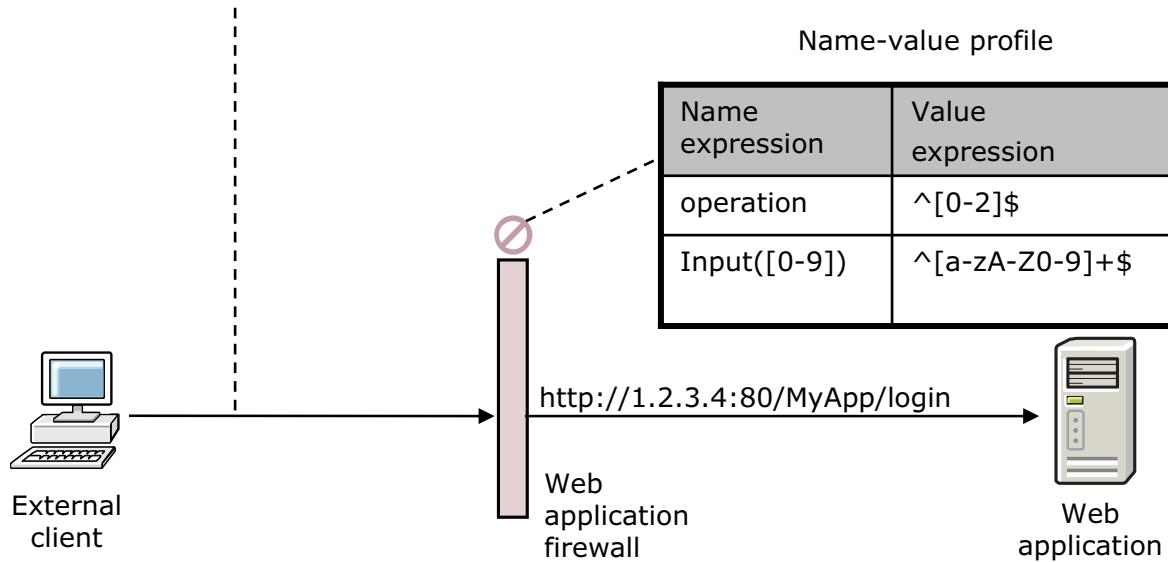
Notes:

The name-value pairs typically are entered into an HTML form. In the slide, the name `operation` is the field name and the value `1` is the value that is entered or selected in the field. The name `input1` is the name of the second field, and the value `CA` is entered. All HTML form controls have name-value pairs.

Name-value profiles (2 of 2)

- The name-value profile uses regular expressions for ensuring correct name-value pairs

URL query string
`https://dphost:9080/MyApp/login?operation=1&input1=CA`



© Copyright IBM Corporation 2013

Figure A-21. Name-value profiles (2 of 2)

WE4013.0

Notes:

The name and value expressions are specified by using PCRE regular expressions.

This type of validation might also occur inside the request HTML page by using client-side JavaScript.

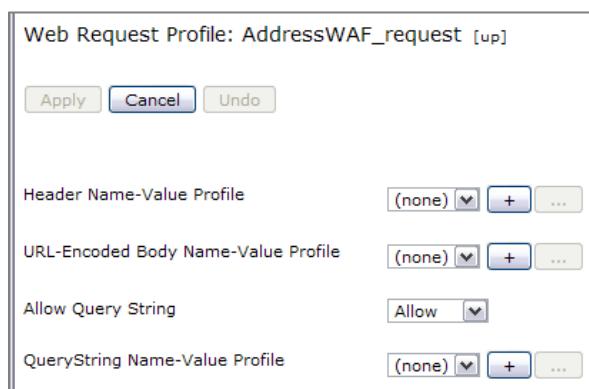
The URL query string contains the name-value pairs after the question mark (?). Other types of requests pass these parameters differently.

The `operation` name regular expression must have the name **operation**. The value expression of `^[0-2]$` means that the correct values can be either 0, 1, or 2.

The `input([0-9])` name regular expression must start with **input** and end with a number 0–9. The possible values for the input name are any alphabetic character or numeral.

Using name-value profiles

- Name-value profiles can be specified for:
 - HTTP headers that use the **Header Name-Value Profile** field
 - HTML form values that use the **URL-Encoded Body Name-Value Profile** field
 - Query strings that contain name-value pairs that use the **QueryString Name-Value Profile** field
 - A cookie file stores name-value pairs: defined on the **Cookies** tab
- The **Allow Query String {allow | require | deny}** determines the permissibility of query strings



© Copyright IBM Corporation 2013

Figure A-22. Using name-value profiles

WE4013.0

Notes:

HTTP header name-value pairs are passed in the HTTP header; for example, the **content-type** HTTP header.

URL-encoded body name-value pairs are name-value pairs that are passed from HTML forms by using the HTTP POST method.

Query string name-value pairs are passed from HTML forms by using the HTTP GET method.

Cookie name-value pair validation is specified on a separate page.

Creating name-value profile object

- The name-value profile object is used to:
 - Specify size constraints on the name and values
 - Build a validation list that consists of name and value expressions

The screenshot shows two side-by-side panels within a web-based configuration tool.

Main Panel:

- Name:** Name-Value Profile
- Buttons:** Apply, Cancel
- Administrative State:** enabled (radio button selected)
- Comments:** (empty text field)
- Maximum Count:** 256
- Total Size:** 128000
- Maximum Name Length:** 512
- Maximum Value Length:** 1024
- No Match Policy:** Strip (dropdown menu)
- No Match XSS Policy:** on (radio button selected)

Validation List Panel:

- Name:** Name-Value Profile
- Buttons:** Apply, Cancel
- Validation List:**

| Name Expression | Value Constraint | Failure Policy | Map Value | Check XSS | XSS (Cross Site Scripting Protection Patterns File) |
|-----------------|------------------|----------------|-----------|-----------|---|
| (empty) | | | | | |

 - Name Expression:** (empty text field)
 - Value Constraint:** (empty text field) Value Helper
 - Failure Policy:** Error (dropdown menu)
 - Check XSS:** on (radio button selected)

© Copyright IBM Corporation 2013

Figure A-23. Creating name-value profile object

WE4013.0

Notes:

You can create a name-value profile object to set constraints on the size and possible values for the name-value pairs that are passed from an HTTP request.

Maximum Count refers to the maximum number of name-value pairs that are allowed.

The **No Match XSS** policy looks in the value string for malicious content. This policy provides some protection from cross-site scripting. It defines the actions to take when a name-value pair is not listed in the validation list. The possible actions are:

- Error:** generates an error and rejects the request
- Pass-through:** leaves the name-value pair untouched
- Strip:** removes any invalid name-value pairs
- Set:** sets the name-value pair to a constant

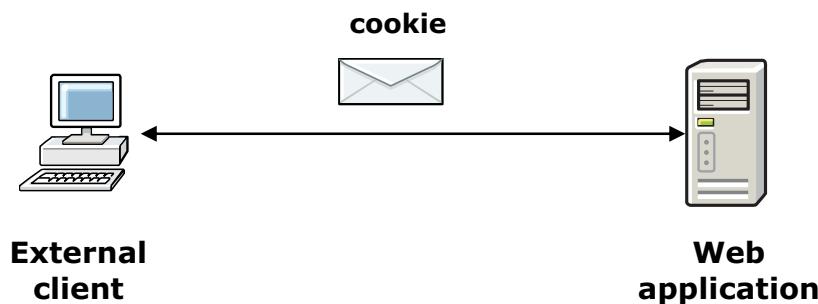
The validation list contains a list of name-value expressions. Within this list, a failure policy defines the action that is taken when a name-value pair fails validation. If the **Failure Policy** is **set**, then the **Map Value** contains the constant that is assigned. The **Failure**

Policy values are similar to the **No Match policy** values that are defined in the **General** tab.

By setting the failure policy, for example to **strip**, you can prevent phishing attacks from intermediaries who might insert malicious parameters.

Cookies

- A cookie is a message that sent initially by the web server to a client web browser
 - Client sends cookie to server on subsequent requests
- Cookies store name-value pairs
 - To maintain sessions between a client and web application, the cookie stores a unique session ID value
 - The web application uses the session ID value inside the cookie to retrieve session data



© Copyright IBM Corporation 2013

Figure A-24. Cookies

WE4013.0

Notes:

Instead of the web application that sends a cookie to the client as depicted in the slide, the web application firewall sends the client a cookie.



Configure cookie support

- Configure cookies on the web application firewall to filter name-value pairs and sign-encrypt cookies
 - Allow Cookies:** {allow | require | deny} determines the permissibility of cookies
 - Sign/Encrypt Cookies:** selects whether to sign or encrypt cookies
 - Enter a pass phrase for signing or encrypting the cookie
 - Cookie Content Name-Value Profile:** specifies a name-value profile object to validate name-value pairs in cookies

Web Request Profile: AddressWAF_request [up]

Allow Cookies: Allow

Sign or Encrypt Cookies: None

Cookie Content Name-Value Profile: (none)

Sign or Encrypt All Cookies:

© Copyright IBM Corporation 2013

Figure A-25. Configure cookie support

WE4013.0

Notes:

When the **Allow Cookies** value is set to **allow**, it does not matter whether the client sends a cookie on each request.



HTTP threat protection

- Built-in threat protection for HTTP request and responses
 - The **Filter Unicode**, **Filter Dot Dot**, and **Filter .exe** options check for the presence of the respective values after the URI is normalized
 - Messages are rejected if they are found
 - Fragmented URI policy determines how to process URIs that contain the number sign (#) character

Web Request Profile: AddressWAF_request [up]

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

| | |
|-----------------------|--|
| Minimum Size | <input type="text" value="0"/> bytes |
| Maximum Size | <input type="text" value="128000000"/> bytes |
| SQL Injection Filter | <input type="checkbox"/> |
| Maximum URI Length | <input type="text" value="1024"/> |
| Filter Unicode | <input checked="" type="checkbox"/> |
| Filter Dot Dot | <input checked="" type="checkbox"/> |
| Filter .exe | <input checked="" type="checkbox"/> |
| Fragmented URI Policy | Truncate ▼ |

© Copyright IBM Corporation 2013

Figure A-26. HTTP threat protection

WE4013.0

Notes:

The web application firewall has built-in protection from phishing or malicious attacks. The SQL injection attack has the same protection as that offered by XML threat protection.

URIs containing the number sign (#) character are used for moving within the same page.



Applying processing rules to messages

- Determine how to process requests that contain XML (text or XML) or non-XML in the HTTP MIME header
 - For an XML request, specify **text/xml** as a valid **content-type** header in the **Profile** tab
- In the **XML Processing** field, select either SOAP or XML
 - The profile can validate the message for conformance to XML or SOAP
 - After validation, a processing rule can execute
 - Processing rule contains actions, such as **Transform**, **AAA**, and many more
- In the **Non-XML Processing** field, you can select:
 - **No processing** (default): do nothing to the message
 - **Side effect rule**: specify a rule that contains actions that do not modify the message
 - **Binary rule**: specify processing rule on message, which is not treated as XML



Figure A-27. Applying processing rules to messages

WE4013.0

Notes:

To use the side effect **Non-XML Processing** option, you can specify a rule that contains actions that do not modify the message, such as AAA and Log actions. This type of rule cannot access the input or output context.

The profile tab that contains the content-type header is available by modifying the request map profile.

HTTP request/response configuration

• Use the **Multipart Form** tab to select the acceptable HTTP methods and versions

• Use the **Multipart Form** tab to configure how to handle binary data that are received from a web browser

- Single requests are broken up into parts
- Uses content type: multipart/form-data

© Copyright IBM Corporation 2013

Figure A-28. HTTP request/response configuration

WE4013.0

Notes:

Multipart forms are used when you send binary data or text that contains non-ASCII characters to the server. Multipart data are sent from web browsers when files are uploaded to a server.

When the **Restrict Sub-Content Types** option is **on**, the content-type of each part is checked against the content-types in the **Profile** tab.

Each control within the HTML form is contained within a separate part in the message.

For an example of a multipart data request, see:

<http://www.w3.org/TR/html4/interact/forms.html#h-17.13.4.2>

Response maps

After completing this topic, you should be able to:

- Configure the request map objects

© Copyright IBM Corporation 2013

Figure A-29. Response maps

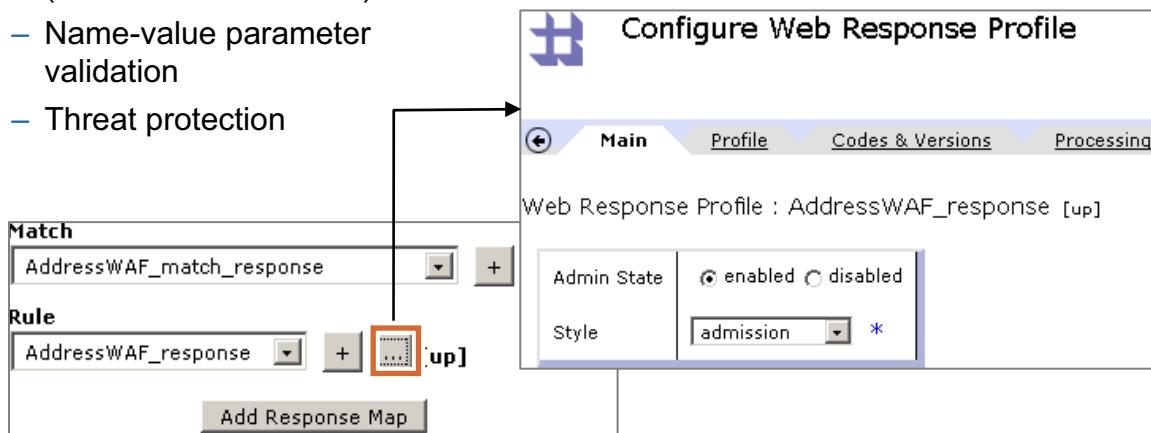
WE4013.0

Notes:



Configure response map profile

- Using the response map rule, you can configure a *profile* that contains (subset of the request map profile):
 - Error policy
 - Accepted content types
 - Supported HTTP methods and protocols
 - Message processing (XML versus non-XML)
 - Name-value parameter validation
 - Threat protection



© Copyright IBM Corporation 2013

Figure A-30. Configure response map profile

WE4013.0

Notes:

The response map profile object is generated when you use the wizard to create a web application firewall.

Error handling

After completing this topic, you should be able to:

- Describe the differences between the three types of error handling in the web application firewall

© Copyright IBM Corporation 2013

Figure A-31. Error handling

WE4013.0

Notes:

Error handling types

Three types of error handling:

- Default error handling policy: handles errors that are received from the back-end service
- Error map: selects a processing rule to execute when an error occurs and matching criteria are satisfied
 - Applies to all requests and responses in the web application firewall
 - Executes based on matching rule
- Error policy: handles errors for requests or responses
 - Defined in the request or response map profile

Either the error map or error policy can handle request or response errors

- These objects are not shared

© Copyright IBM Corporation 2013

Figure A-32. Error handling types

WE4013.0

Notes:



WAF forms-based login

After completing this topic, you should be able to:

- Describe the differences between the three types of error handling in the web application firewall

© Copyright IBM Corporation 2013

Figure A-33. WAF forms-based login

WE4013.0

Notes:



What is forms-based login?

- Forms-based authentication allows one to use an HTML form to obtain credentials from users who are attempting to access secured (private) web pages on an application server
- Such as:
 - Account name
 - Password
 - Or both

© Copyright IBM Corporation 2013

Figure A-34. What is forms-based login?

WE4013.0

Notes:

Forms-based authentication allows you to use an HTML form to obtain credentials from users who are attempting to access secured (private) web pages on an application server. Typically, this information is an account name, a password, or both.

When a user attempts to access a secured web page, the web application firewall redirects the request to a forms-based login page. This page provides fields where the user can provide credentials.

If the request passes authentication and authorization, the web application firewall sets a cookie in the client browser and forwards the request to the application server. The application server starts a session. The session uses this cookie to preserve authentication for all subsequent requests.

If the request fails authentication, the web application firewall redirects the user to an error page.

DataPower provides a login page that is located in the store directory (store:///loginPage.htm).



WAF user authentication

- Basic authentication (send user name and password with each transaction)
- Customer demand for forms-based login (J2EE specification, allowing custom login pages)
- J2EE assumes heavyweight server-side “container” to maintain state for the lifetime of the session

© Copyright IBM Corporation 2013

Figure A-35. WAF user authentication

WE4013.0

Notes:



Forms-based login overview (J2EE)

- User attempts to access protected page within application
- User's browser is redirected to a custom login page
- Login page contains HTML form that collects user name and password
- Submitting form sends user name and password to application server
- Application server validates login, attaches authenticated identity to container (server-side representation of the session), and redirects browser back to original page
- Next, browser transaction within that container can use authenticated identity

© Copyright IBM Corporation 2013

Figure A-36. Forms-based login overview (J2EE)

WE4013.0

Notes:



Implementing in WAF

- Offloading authentication from the application
- Mediate access to pages within the application that is based on request maps within the WAF security policy
- Need at least two request maps:
 - Pages that require authentication
 - Pages that do not require authentication (including the login form)
- Each map specifies a AAA policy, which name the same **FormsLoginPolicy** object in their EI sections
- **FormsLoginPolicy** is new configuration object that configures forms-based login

© Copyright IBM Corporation 2013

Figure A-37. Implementing in WAF

WE4013.0

Notes:

The process to enable form-based authentication and authorization for a web application firewall service is as follows:

1. Define an HTML forms-based authentication policy.
2. Create a AAA policy for secured (private) web pages and a web request profile that references this AAA policy.
3. Create a AAA policy for unsecured (public) web pages and a web request profile that references this AAA policy.

WebSphere DataPower XI52

IBM

Configure an Application Security Policy

[Help](#)

[General](#) [Request Maps](#) [Response Maps](#) [Error Maps](#)

Create / Edit a list of Request Maps

| Match | Role | Reorder | Delete |
|----------------------|-------------------------|---------|--------|
| Unauthenticated_Rule | Unauthenticated_Profile | up down | Remove |
| Authenticated_Rule | Authenticated_Profile | up down | Remove |

Matching Rule
Unauthenticated_Rule [+](#) [...](#)

Request Profile
Unauthenticated_Profile [+](#) [...](#)

[Add Request Map](#)

[Commit](#) [Cancel](#)

- Two request maps are configured on the WAF object
 - One map is for an authenticated request
 - The second map is for an unauthenticated request

© Copyright IBM Corporation 2013

Figure A-38. WAF security policy request map

WE4013.0

Notes:

WebSphere Education

WAF forms login request rule

Configure Web Request Profile

This configuration has been added and not yet saved.

Main **Profile** Methods & Versions Processing Name Value Cookie Multipart

Web Request Profile: Authenticated_Profile [up]

Apply Cancel Undo Export | View Log | View Status | Help

Allow SSL: Allow

AAA Policy: **Test-WAF-AAA-Policy** (highlighted with a red box)

Shared Secret Key: (none) + ...

Rate Limiting: (none) + ...

Access Control List: (none) + ...

Error Policy: (none) + ...

- The request map profile specifies a AAA policy

© Copyright IBM Corporation 2013

Figure A-39. WAF forms login request rule

WE4013.0

Notes:

Configure AAA Policy

This configuration has been added and not yet saved.

AAA Policy: Test-WAF-AAA-Policy [up]

Methods

- HTTP Authentication Header
- Password-carrying UsernameToken Element from WS-Security Header
- Derived-key UsernameToken Element from WS-Security Header
- SAML Artifact
- Client IP Address
- Subject DN from Certificate in the Message's signature
- Token Extracted from the Message
- Token Extracted as Cookie Value
- LTPA Token
- Processing Metadata
- Custom Template
- HTML Forms-based Authentication *

HTML Forms-based Login Policy Test-WAF-Login-Policy + ... *

Export | View Log | View Status | Help
Flush Cache

Both AAA policies (authentication and unauthentication) specify forms-based login as their EI method, and refer to the same forms login policy

© Copyright IBM Corporation 2013

Figure A-40. WAF-authenticated AAA policy

WE4013.0

Notes:



WAF Forms Login Policy (1 of 3)

 Configure HTML Forms Login Policy
This configuration has been added and not yet saved.

Main

HTML Forms Login Policy: Test-WAF-Login-Policy [up]

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

General

Admin State enabled disabled

Comments

Security

Use SSL for Login on off *

SSL Port

Enable Session Migration on off *

© Copyright IBM Corporation 2013

Figure A-41. WAF Forms Login Policy (1 of 3)

WE4013.0

Notes:

WebSphere Education

IBM

WAF Forms Login Policy (2 of 3)

| | |
|----------------------------------|--|
| Client-side URL fragments | |
| Login | /LoginPage.htm * |
| Error | /ErrorPage.htm * |
| Logout | /LogoutPage.htm * |
| Default URL | / * |
| Location of HTML pages | |
| Forms Storage Location | Appliance * |
| On-appliance Login Form | store:/// <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <input type="button" value="Edit..."/> <input type="button" value="View..."/> |
| On-appliance Error Page | store:/// <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <input type="button" value="Edit..."/> <input type="button" value="View..."/> |
| On-appliance Logout Page | store:/// <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <input type="button" value="Edit..."/> <input type="button" value="View..."/> |

© Copyright IBM Corporation 2013

Figure A-42. WAF Forms Login Policy (2 of 3)

WE4013.0

Notes:

WAF Forms Login Policy (3 of 3)

| | |
|------------------------------|--|
| Login form properties | |
| User Name Field Name | <input type="text" value="j_username"/> * |
| Password Field Name | <input type="text" value="j_password"/> * |
| Target URL Field Name | <input type="text" value="originalUrl"/> * |
| Form POST Action URL | <input type="text" value="/j_security_check"/> * |
| | |
| Timeouts | |
| Inactivity Timeout | <input type="text" value="600"/> Seconds * |
| Session Lifetime | <input type="text" value="10800"/> Seconds * |

© Copyright IBM Corporation 2013

Figure A-43. WAF Forms Login Policy (3 of 3)

WE4013.0

Notes:

The significant content of the login form is as follows:

- The user name and password fields (`j_username` and `j_password`) are in accordance with the J2EE specification.
- A hidden *redirection URL* field (`originalUrl`) records the URL that the user was attempting to contact before being redirected to the login form. After authentication, the user is redirected to this URL.
- Any form action (post) tells the browser what to do when a user clicks **Login**. In accordance with the J2EE specification, `j_security_check` is the security check.

Note: To work with form-based authentication, this value must be a path fragment within the URL (`j_security_check`).



Unit summary

Having completed this unit, you should be able to:

- Configure a web application firewall to protect a back-end web application
- Use a AAA policy to protect access through the web application firewall
- Validate parameters from an HTTP request by using name-value profiles
- Protect the web application from phishing attacks by using built-in threat protection

© Copyright IBM Corporation 2013

Figure A-44. Unit summary

WE4013.0

Notes:

Checkpoint questions

1. True or False: Session management is provided by issuing a cookie to the client. The list of valid URIs can also be configured by using this object.
2. Select the four items that can be configured with a name-value profile object:
 - A. HTTP headers
 - B. XML documents
 - C. Cookie values
 - D. XSL style sheet
 - E. URL-encoded query strings
 - F. URL-encoded request messages
3. True or False: DataPower does provide a working HTML login page, which is in the `store:///subdirectory`.
4. Match the firewall with its correct definition:

| Description | Definition |
|--|--|
| <ol style="list-style-type: none"> 1. Web application firewall 2. XML firewall | <ol style="list-style-type: none"> A. Limited support of XML or binary requests B. Handles mainly HTTP requests C. Designed to handle XML traffic D. Process messages against a set of actions |

© Copyright IBM Corporation 2013

Figure A-45. Checkpoint questions

WE4013.0

Notes:

Write your answers here:

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

Checkpoint answers

1. **True.** Session management is provided by issuing a cookie to the client. The list of valid URLs can also be configured by using this object.
2. **A, C, E, and F.** Select the four items that can be configured with a name-value profile object:
 - A. HTTP headers
 - B. XML documents
 - C. Cookie values
 - D. XSL style sheet
 - E. URL-encoded query strings
 - F. URL-encoded request messages
3. **True.** DataPower does provide a working HTML login page, which is in the `store:///subdirectory`.
4. Match the firewall with its correct definition:

| Description | Definition |
|-----------------------------|--|
| 1. Web application firewall | A. Limited support of XML or binary requests B. Handles mainly HTTP requests C. Designed to handle XML traffic D. Process messages against a set of actions |
| 2. XML firewall | A. Limited support of XML or binary requests B. Handles mainly HTTP requests C. Designed to handle XML traffic D. Process messages against a set of actions |

© Copyright IBM Corporation 2013

Figure A-46. Checkpoint answers

WE4013.0

Notes:

Appendix exercises



Creating a firewall and HTTP proxy
for a web application

Configuring WebSphere JMS

© Copyright IBM Corporation 2013
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

T01

Figure A-47. Appendix exercises

WE4013.0

Notes:

Exercise objectives

After completing the first exercise, you should be able to:

- Use the web application firewall wizard to create a web application firewall
- Implement a security policy on a web application firewall
- Create a reverse-proxy to virtualize requests to web applications

After completing the second exercise, you should be able to:

- Identify the fields in the service integration bus configuration on WebSphere Application Server V7.0 that are needed to configure the WebSphere DataPower JMS object
- Create a multi-protocol gateway service that invokes the East Address Search web service over the JMS transport

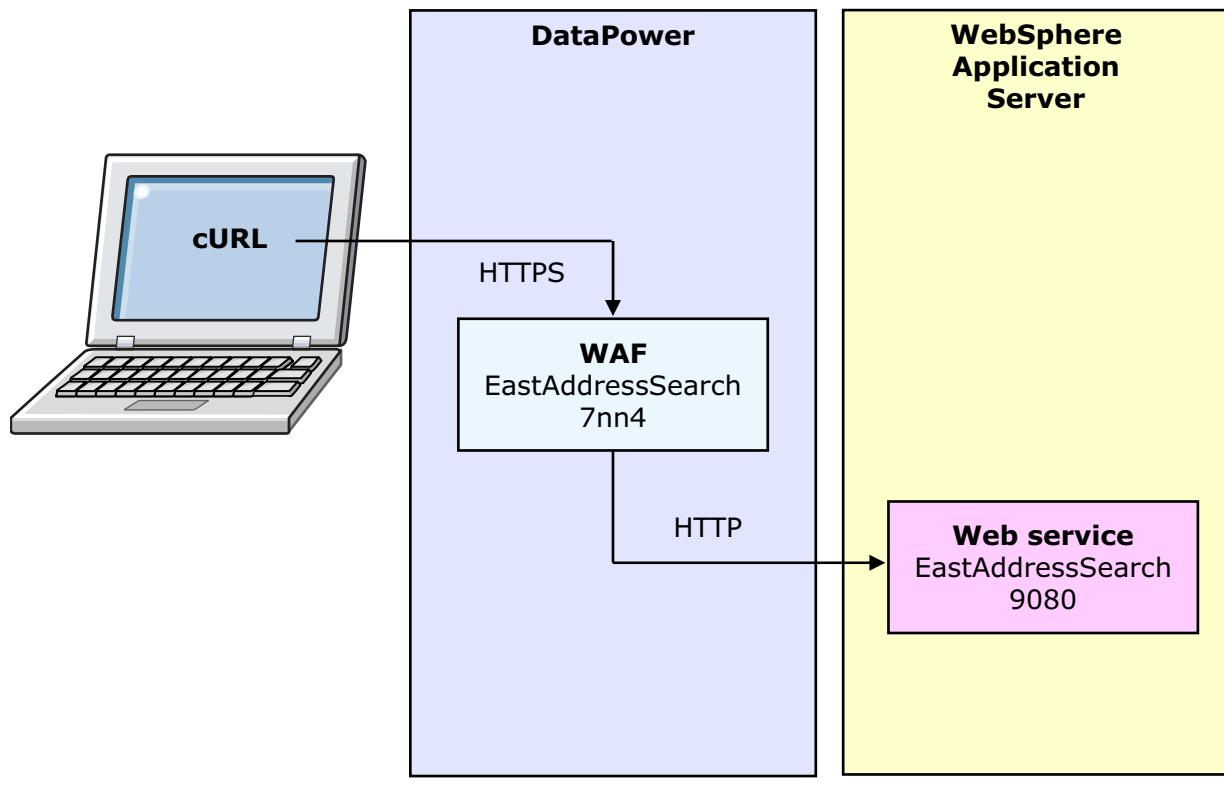
© Copyright IBM Corporation 2013

Figure A-48. Exercise objectives

WE4013.0

Notes:

Exercise overview



© Copyright IBM Corporation 2013

Figure A-49. Exercise overview

WE4013.0

Notes:

Glossary of abbreviations and acronyms

A

AAA authentication, authorization, and auditing
AC alternating current
ACL access control list
AES Advanced Encryption Standard
AMP Appliance Management Protocol
AO Application Optimization
API application programming interface
ARP Address Resolution Protocol
ASCII American Standard Code for Information Interchange

B

B2B business-to-business
B2C business-to-consumer
BEA BEA Systems, Inc.

C

CA certificate authority
CBR content based routing
CCSID coded character set identifier
CD ROM Compact Disk Read Only Memory
CGI Common Gateway Interface
CHML Compact HTML
CIDR Classless Inter-Domain Routing
cKVM concurrent keyboard-video-mouse
CLI command-line interface
CPU central processing unit
CR carriage return
CRL certificate revocation list
CSR certificate signing request
CSR customer service representative
CSS cascading style sheet

D

DAP Directory Access Protocol
DB database
DES Data Encryption Standard
DH Diffie-Hellman
DIME Direct Internet Message Encapsulation
DIT directory information tree

DMZ demilitarized zone
DN distinguished name
DNS Dynamic Name Server
DOM Document Object Model
DOP data-oriented programming
DoS denial-of-service
DSS Digital Signature Standard
DTD document type definition

E

EAL Evaluation Assurance Level
EDIINT Electronic Data Interchange-Internet Integration
EMS Enterprise Message Service
EMS Enterprise Messaging System
EoN Edge of Network
ESB enterprise service bus
EULA end-user license agreement
EXSLT Extensions to Extensible Stylesheet Language Transformation

F

FIFO first-in first-out
FIPS Federal Information Processing Standard
FIX Financial Information Exchange
FO formatting object
FSH front-side handler
FTP File Transfer Protocol
FTPS File Transfer Protocol over SSL

G

GB gigabyte
GSS Generic Security Services
GUI graphical user interface

H

HMAC hash-based message authentication code
HR human resources
HREF hypertext reference
HSM Hardware Security Module
HSRP Hot Standby Router Protocol
HTML Hypertext Markup Language
HTTP Hypertext Transfer Protocol
HTTPS HTTP over SSL

I

ICAP Internet Content Adaptation Protocol

ICMP Internet Control Message Protocol
ICRX extended identity context reference
IDE integrated development environment
IDEA International Data Encryption Algorithm
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
ILD Intelligent load distribution
IMS Information Management System
IP Internet Protocol
IPSec IP Security
iSCSI Internet Small Computer Systems Interface
ITS Interoperability Test Service

J

JAXP Java API for XML Parsing
JAXP Java API for XML Processing
JDBC Java Database Connectivity
JDK Java Development Kit
JFAP JetStream Formats and Protocols
JKS Java Key Store
JMS Java Message Service
JRE Java runtime environment
JSON JavaScript Object Notification

K

KB kilobyte

L

LAN local access network
LDAP Lightweight Directory Access Protocol
LDIF LDAP Data Interchange Format
LED light-emitting diode
LLM Low Latency Messaging

M

MAC message authentication code
MB megabyte
MFA message filter action
MIB Management Information Base
MIME Multipurpose Internet Mail Extensions
MM message monitor
MMXDoS Multiple message XML denial-of-service
MPG multi-protocol gateway
MPGW multi-protocol gateway
MQ Message Queue
MQFTE MQ File Transfer Edition
MT message type

MTOM Message Transmission Optimization Mechanism

N

NAT network address translation
NFS Network File System
NG New Generation
NIC network interface card
NSS network security services
NSTISSC National Security Telecommunications and Information Systems Security Committee
NTP Network Time Protocol

O

OASIS Organization for the Advancement of Structured Information Standards
OCSP Online Certificate Status Protocol
OID object ID
OSI Open Systems Interconnection
OTMA Open Transaction Management Access

P

PCF Processing Control File
PCRE Perl-compatible regular expressions
PDF Portable Document Format
PDP policy decision point
PED PIN Entry Device
PEM privacy enhanced mail
PEP policy enforcement point
PI processing instruction
PIN personal identification number
PKCS Public Key Cryptography Standard
PKI public key infrastructure
PKIX Public Key Infrastructure for X.509 Certificates (IETF)
PMR program maintenance request

Q

QoS quality of service

R

RACF Resource Access Control Facility
RAID Redundant Array of Independent Disks
RAM random access memory
RBM role-based management
RDBMS relational database management system
RDN relative distinguished name
REL Rights Expression Language

RFC Request for Comments**RSA** Rational Software Architect**RSA** Rivest, Shamire, and Adleman**S**

SAML Security Assertion Markup Language
SAS Serial Attached SCSI
SAX Simple API for XML
SCP Secure Copy Protocol
SCSI Small Computer System Interface
SFTP Secured File Transfer Protocol
SHA1 Secure Hash Algorithm, Version 1
SLA service level agreement
SLES SUSE Linux Enterprise Server
SLM service level management
SLM service level monitoring
SMS session management server
SMTP Simple Mail Transfer Protocol
SNMP Simple Network Management Protocol
SOA service-oriented architecture
SOAP usage note: SOAP is not an acronym; it is a word in itself (formerly an acronym for Simple Object Access Protocol)
SOL serial-over-LAN
SOMA SOAP Configuration Management
SPNEGO Simple and Protected GSS-API Negotiation Mechanism
SPVC self-paced virtual class
SQL Structured Query Language
SSH Secure Shell
SSL Secure Sockets Layer
SSO single sign-on
SwA SOAP with Attachments

T

Tcl Tool Control Language (often pronounced as “tickle”)
TCO total cost of ownership
TCP Transmission Control Protocol
TCP/IP Transmission Control Protocol/Internet Protocol
TDES Triple Data Encryption Standard
TEP Tivoli Enterprise Portal
TFIM Tivoli Federated Identity Manager
TIA Telecommunications Industry Association
TIBCO The Information Bus Company
TIM Tivoli Identity Manager
TLS Transport Layer Security
TM Transaction Manager

TTL Time to Live**U**

UDDI Universal Description, Discovery, and Integration
UDP User Datagram Protocol
UNIX Uniplexed Information and Computing System
URI Uniform Resource Identifier
URL Uniform Resource Locator
USB Universal Serial Bus
UTC Coordinated Universal Time

V

VIP virtual IP address
VM virtual machine
VLAN virtual local area network
VRRP Virtual Router Redundancy Protocol

W

W3C World Wide Web Consortium
WAF web application firewall
WAFW web application firewall
WML Wireless Markup Language
WS web services
WSDL Web Services Description Language
WSDM Web Services Distributed Management
WSP web service proxy
WTS Web Token Service
WWW World Wide Web

X

XA eXtended Architecture
XCF cross-system coupling facility
XDoS XML denial of service
XHTML Extensible Hypertext Markup Language
XML Extensible Markup Language
XMLDS XML digital signature
XML-PI XML processing instructions
XPath XML Path Language
XSD XML Schema Definition
XSL Extensible Stylesheet Language
XSLT Extensible Stylesheet Language Transformation

Y

Z

z/OS zSeries operating system

IBM
®