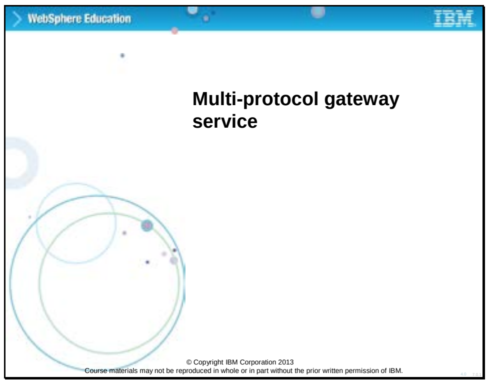
Slide 1



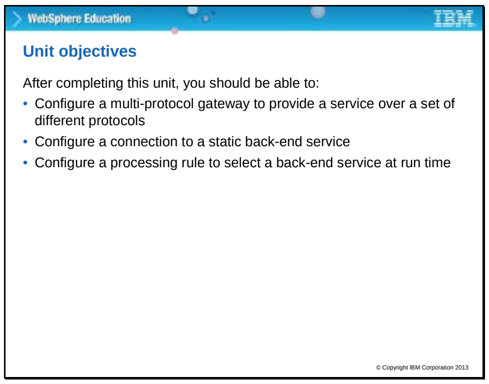
WebSphere Education

IBM

Multi-protocol gateway service

The multi-protocol gateway service.

Slide 2



WebSphere Education

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

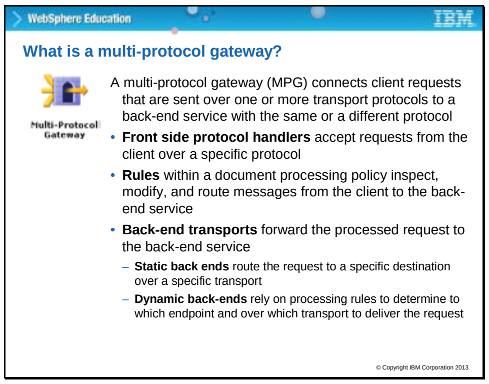
This unit assumes that students are familiar with the Web Service Proxy and XML firewall services on the IBM WebSphere DataPower SOA Appliance. Topics that are covered include processing actions and objects that relate to the multi-protocol gateway.

This unit takes a similar approach to the AAA unit in that it does not cover all of the protocol handlers. It covers the scenarios in which you can use the multi-protocol gateway service. The two scenarios that are covered are:

Providing multiple back-end transports by HTTP and HTTPS

Determining the back-end transport dynamically at execution by a style sheet

Slide 3


WebSphere Education IBM

What is a multi-protocol gateway?

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

What is a multi-protocol gateway?

So, what is a Multi-Protocol Gateway (MPGW)?

It connects client requests that are sent over one or more transport protocols to a back-end service by using the same or a different protocol.

It has one or more front side protocol handlers that accept requests from the client, each one tailored for a specific protocol.

Rules within a document processing policy inspect, modify, and route messages from the client to the back-end service, just like in an XML Firewall.

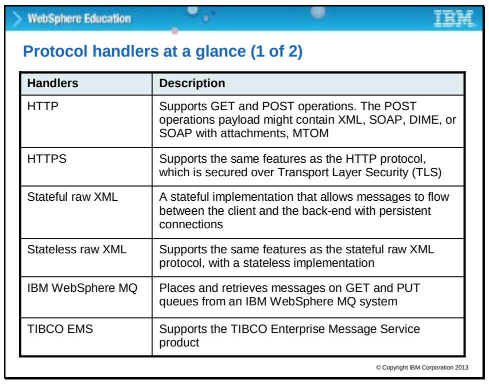
Back-end transports forward the processed request to the back-end service by using various protocols.

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.

It is important to note that a single MPGW service can have several different protocols by being supplied by several different front side handlers all feeding to the same set of rules. The MPGW and the dynamic back-ends might also be connected by using various protocols, all being fed from the same single MPG. It might be regarded as a protocol “fan-in, fan-out” situation.

Slide 4


WebSphere Education

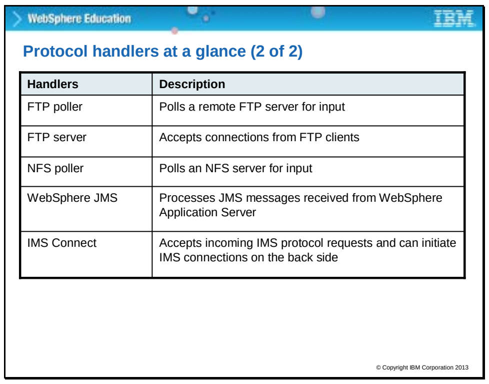
IBM

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

Protocol handlers at a glance (1 of 2)
Now look at the protocols that are handled by the MPG.
HTTP is a familiar protocol that supports GET and POST operations. The payload in POST operations might contain XML, SOAP, DIME, SOAP with attachments, or MTOM. More about POST operation payloads later.
HTTPS is the secure version of HTTP, and it supports the same features. HTTPS is secured over Transport Layer Security (TLS).
Stateful raw XML is a stateful implementation that allows messages to flow between the client and the back-end server by using persistent connections.
Stateless raw XML supports the same features as the stateful raw XML protocol, by using a stateless implementation.
The WebSphere MQ handler places and retrieves messages on GET and PUT queues from an IBM WebSphere MQ system. DataPower acts as an WebSphere MQ client only, not an WebSphere MQ server, and it does not support the WebSphere MQ Extended Transactional Client.
The TIBCO EMS handler supports the TIBCO Enterprise Message Service product.

Slide 5


WebSphere Education IBM

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

Protocol handlers at a glance (2 of 2)

The FTP poller handler polls a remote FTP server for input, and can rename or delete the file that is used for input when completed. It can be configured to poll periodically, looking for new files at the remote location.

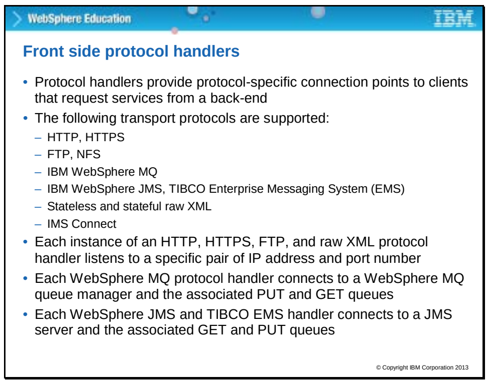
The FTP server handler accepts connections from FTP clients and allows them to upload files, from which records are extracted and processed by the policy rules within the MPG.

The NFS poller handler polls an NFS server for input in much the same way as the FPT poller operates.

The WebSphere JMS handler processes JMS messages received from WebSphere Application Server similar to the WebSphere MQ handler. This processing is not a generic JMS implementation in that it does not use a JNDI and connection factories. It specifically links only to the WebSphere Application Server proprietary interfaces.

The IMS Connect handler accepts incoming IMS protocol requests and can initiate IMS connections on the back-side.

Slide 6


WebSphere Education

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

Front-side protocol handlers

As covered on the previous slides, protocol handlers provide protocol-specific connection points to clients that request services from a back-end server, and they support all these protocols:

HTTP, HTTPS

FTP, NFS

IBM WebSphere MQ

IBM WebSphere JMS, TIBCO Enterprise Messaging System (EMS)

Stateless and stateful raw XML

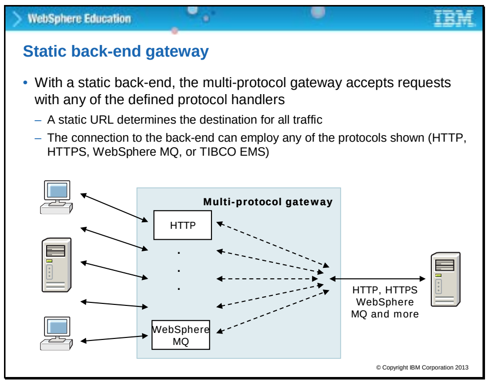
IMS Connect

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 refers to a WebSphere MQ queue manager and the associated PUT and GET queues that are used for communication.

Each WebSphere JMS and TIBCO EMS handler refers to a JMS server and the associated GET and PUT queues.

Slide 7


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

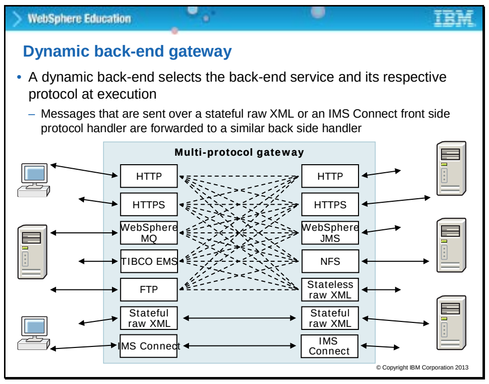
Static back-end gateway

With a static back-end, the multi-protocol gateway accepts requests through any of the defined protocol handlers.

A static URL determines the destination for all traffic.

The connection to the back-end server can employ any of the protocols shown (HTTP, HTTPS, WebSphere MQ, or TIBCO EMS).

Slide 8



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

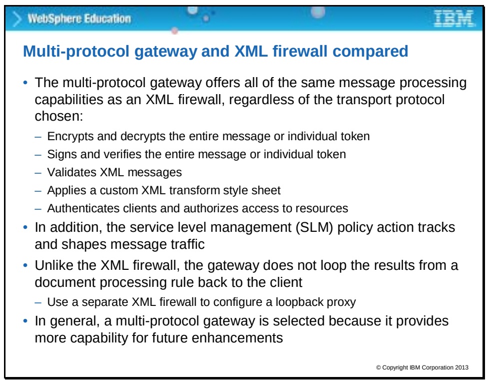
Dynamic back-end gateway

A dynamic back-end selects the back-end service and its respective protocol at run time. So, technically, you might have a single MPG acting as a message-switching service as shown in this diagram: taking messages from anywhere, and sending them anywhere.

Messages that are sent over a stateful raw XML front side protocol handler must be forwarded to a stateful raw XML back side handler.

This example is extreme; it is unlikely that anyone would want to have a single MPG acting as a universal router.

Slide 9


WebSphere Education

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

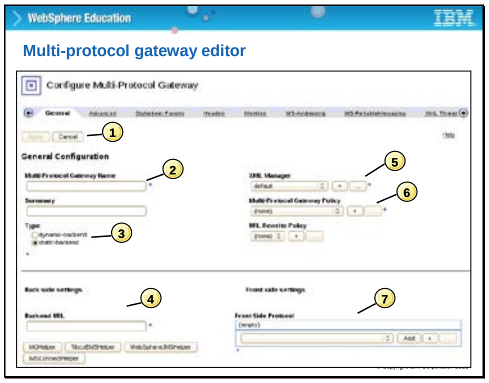
Multi-protocol gateway and XML firewall compared

So, why use the MPG service rather than the XML Firewall? Well, the multi-protocol gateway offers all of the same message processing capabilities as an XML firewall, regardless of the transport protocol chosen. It can encrypt and decrypt the entire message or individual fields. It can sign and verify the entire message or individual fields. It can validate XML messages by using a schema document; it can apply a custom XML transform style sheet; it can authenticate clients and authorize access to resources.

In addition, the service level management (SLM) policy action monitors and shapes message traffic.

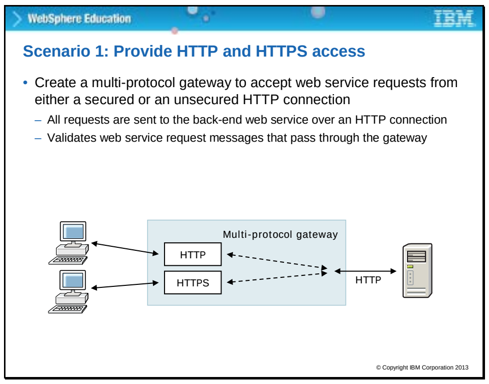
Unlike the XML firewall, the gateway cannot loop the results from a document processing rule back to the client. If you really want loopback capability for testing an MPG, you can use a separate XML firewall to configure a loopback proxy, and “back-end” it onto the MPG. In general, whenever you have a choice between using an XML Firewall, or a multi-protocol gateway, the MPG should be selected because it provides more capability for future enhancements.

Slide 10



Multi-protocol gateway editor
As you can see, the MPG configuration editor looks remarkably similar to the XML Firewall configuration editor.
Number 1 shows the usual Apply and Cancel buttons.
Number 2 allows you to name your MPG.
Number 3 allows you to select whether it uses a dynamic or a static back-end. You do not have a loop-back option, like the XML Firewall.
Number 4 configures the back-end. Notice there Is some extra buttons that allow you to generate a URL to handle connections other than a normal web service.
Number 5 is where you choose your XML manager.
Number 6 allows you to select, create, or edit your Policy.
Number 7 is where you can populate a list of front-side handlers from a pick list of choices. As is usual with such lists, you can create a front-side handler dynamically.

Slide 11


WebSphere Education
IBM

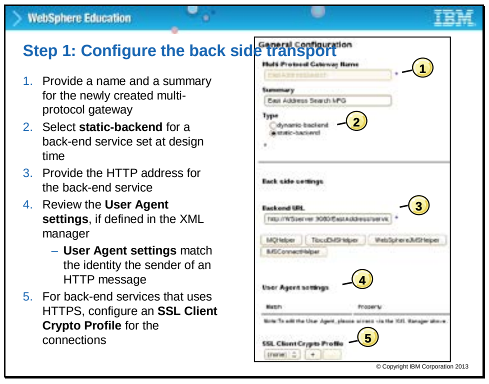
Scenario 1: Provide HTTP and HTTPS access

- 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

Multi-protocol gateway
HTTP
HTTPS
HTTP

Scenario 1: Provide HTTP and HTTPS access
Look at some typical uses of the MPG. The first scenario shows how to create a multi-protocol gateway to accept 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, and the Service Policy validates web service request messages that pass through the gateway.

Slide 12



WebSphere Education

IBM

Step 1: Configure the back side transport

1. Provide a name and a summary for the newly created multi-protocol gateway
2. Select **static-backend** for a back-end service 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 uses HTTPS, configure an **SSL Client Crypto Profile** for the connections

General Configuration

Multi Protocol Gateway Name: EastAddressSearch

Summary: East Address Search MPG

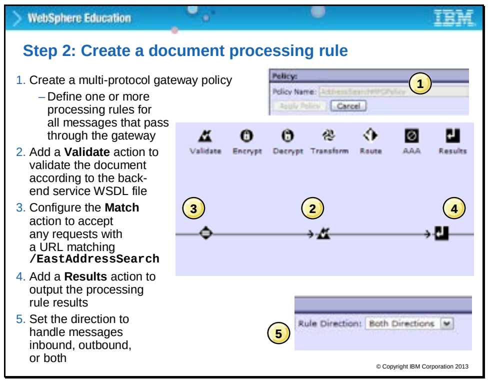
Type: ( ) dynamic-backend [✓ static-backend]

Back side settings

Backend URL: http://WSserver:9080/EastAddress/servic

Step 1: Configure the back-end transport
Here is how to would configure the MPG to handle this scenario.
First, provide a name and a summary for the newly created multi-protocol gateway.
Then, select a static-backend and provide the HTTP address for the back-end service.
Check the User Agent settings, if they were defined in the XML manager. The user Agent settings match the identity of the sender of an HTTP message.
And finally, for back-end services that use HTTPS, configure an SSL Client crypto Profile for the connections.

Slide 13

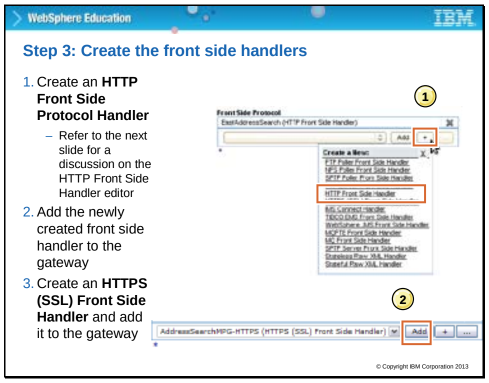


Step 2: Create a document processing rule

1. Create a multi-protocol gateway policy
   - Define one or more processing rules for all messages that pass through the gateway
2. Add a **Validate** action to validate the document according to the back-end service WSDL file
3. Configure the **Match** action to accept any requests with a URL matching **`/EastAddressSearch`**
4. Add a **Results** action to output the processing rule results
5. Set the direction to handle messages inbound, outbound, or both

Step 2: Create a document processing rule
Next, define how to process the messages, so create a multi-protocol gateway policy, which includes one or more processing rules for all messages that passes through the gateway.
In this example, add a Validate action, highlighted as number 2 to validate the document according to the back-end service WSDL file.
Also, configure the Match Action, indicated as number 3 to accept any requests with a URL matching /EastAddressSearch.
Then, add a Results action (number 4) to output the processing rule results to the server.
Finally, set the direction to handle messages inbound, outbound, or both, before clicking “Apply Policy” to save the changes.

Slide 14



Step 3: Create the front side handlers

1. Create an **HTTP Front Side Protocol Handler**
   – Refer to 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

1
Front Side Protocol
EastAddressSearch (HTTP Front Side Handler)
Add
+
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
2
AddressSearchMPG-HTTPS (HTTPS (SSL) Front Side Handler)
Add
+
...

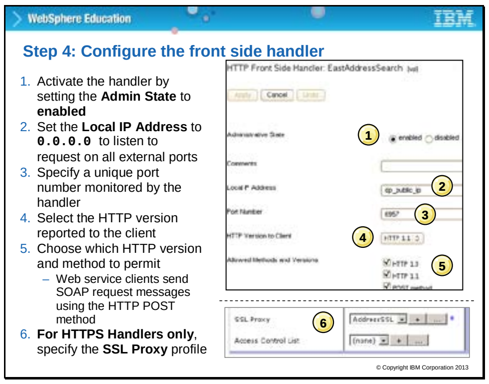
Step 3: Create the front side handlers

Next, create an HTTP Front Side Protocol Handler, which is covered in more detail on the next slide.

Create and add a front side handler to the gateway. You begin this action by clicking the plus sign beside the Front Side Handle pick list, which causes a menu to be displayed. Choose the protocol that you want, and a configuration screen appear. (See that on the next slide.) There is a minor error on this slide in that the button indicated as #1 should actually be the plus sign, not “Add”.

When you create a Front Side Handler, you can click “Add” to add it to the list, which is correctly labeled as number 2.

Slide 15


WebSphere Education

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 monitored by the handler
4. Select the HTTP version reported to the client
5. Choose which HTTP version and method to permit
   – Web service clients send SOAP request messages using the HTTP POST method
6. **For HTTPS Handlers only**, specify the **SSL Proxy** profile

HTTP Front Side Handler: EastAddressSearch [xd]

Apply | Cancel | Undo

Administrative State	(1) [✓ enabled] disabled
Comments	
Local IP Address	dp_public_ip (2)
Port Number	6957 (3)
HTTP Version to Client	(4) HTTP 1.1
Allowed Methods and Versions	✓ HTTP 1.0 ✓ HTTP 1.1 ✓ POST method (5)

SSL Proxy (6)	AddressSSL
Access Control List	(none)

© Copyright IBM Corporation 2013

Step 4: Configure the front side handler

For example, configure the HTTP front-side handler. It would look like the diagram.

Number 1 shows how to activate or deactivate the handler by setting the Admin State radio button.

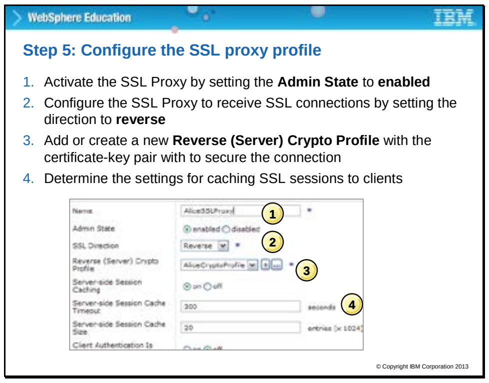
Number 2 shows how to set the Local IP address to 0.0.0.0 to listen to requests on all external Ethernet interfaces.

Number 3 shows how to specify a unique port number that is monitored by the handler.

Numbers 4 is where to select the HTTP version that is reported to the client, and also choose which HTTP version and method to allow, indicated by number 5. Web service clients send SOAP request messages by using the HTTP POST method.

When you configure the HTTPS Handlers, the lower part of the form has some extra fields for configuring the SSL Proxy Profile, as indicated by number 6.

Slide 16



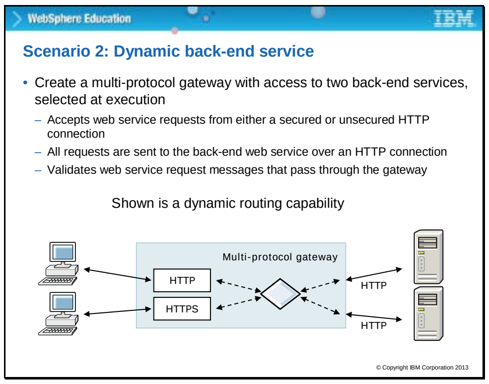
Step 5: Configure the SSL proxy profile

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	AliceSSLProxy *
Admin State	[✓ enabled] disabled
SSL Direction	Reverse *
Reverse (Server) Crypto Profile	AliceCryptoProfile *
Server-side Session Caching	[✓ on] off
Server-side Session Cache Timeout	300 seconds
Server-side Session Cache Size	20 entries (x 1024)
Client Authentication Is	on [✓ off]

Step 5: Configure the SSL Proxy profile
Here is how to configure the SSL Proxy Profile.
Number 1 shows how to activate the SSL Proxy by setting the Admin State to enabled.
Number 2 is where to configure the SSL Proxy to receive SSL connections by setting the direction to reverse.
Number 3 allows one to add or create a new Reverse (Server) crypto Profile with the certificate-key pair used to secure the connection.
Number 4 shows where to determine the settings for caching SSL sessions to clients.

Slide 17


WebSphere Education

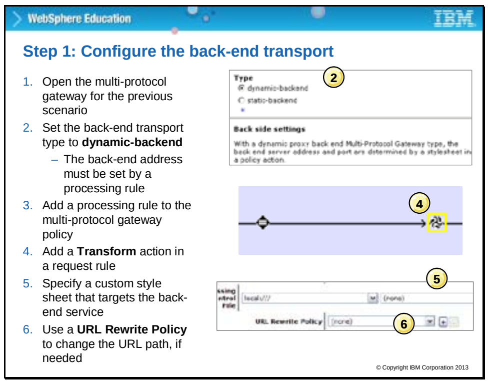
Scenario 2: Dynamic back-end service

- Create a multi-protocol gateway with access to two back-end services, 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

Scenario 2: Dynamic back-end service
Now for a second scenario:
In this example, create a multi-protocol gateway with access to two back-end services, selected dynamically at run time.
This MPG 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.
It validates web service request messages that pass through the gateway.
The slide illustrates a dynamic routing capability.

Slide 18


WebSphere Education

Step 1: Configure the back-end transport

1. Open the multi-protocol gateway for the previous scenario
2. Set the back-end transport type to **dynamic-backend**
   - The back-end address must be set by a processing rule
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

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.

URL Rewrite Policy: (none)

Step 1: Configure the back-end transport

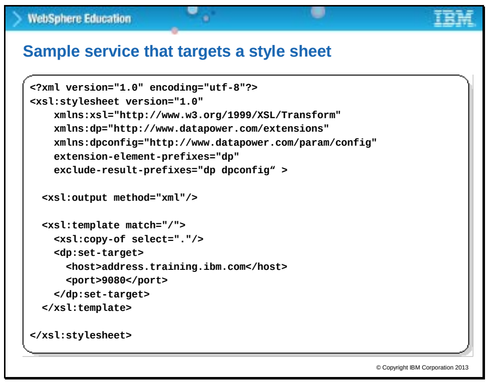
Now go through the steps that are needed to set up this MPG. First, configure the back-end transport.

Open the multi-protocol gateway for the previous scenario, then set the back-end transport type to dynamic-backend, as indicated by number 2. The back-end address is set by a processing rule, so add a processing rule to the multi-protocol gateway policy. Then, add a Transform action to it, as indicated by number 4.

Edit the Transform Action to specify a custom style sheet that targets the back-end service, as indicated by number 5.

And finally, number 6 shows how to use a URL Rewrite Policy to change the URL path, if needed.

Slide 19

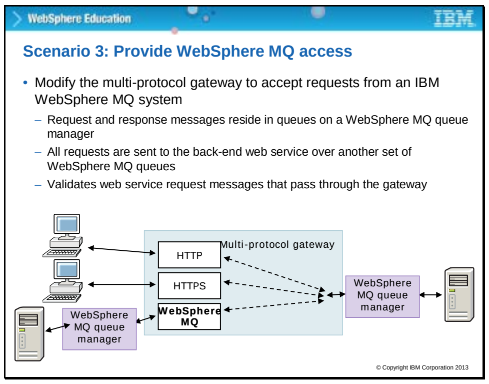


Sample service that targets a style sheet

```
<?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>
```

Sample service that targets style sheet
Here is a snippet of XSL code that shows how certain DataPower extensions can be used to set parameters such as “host” and “port.”

Slide 20

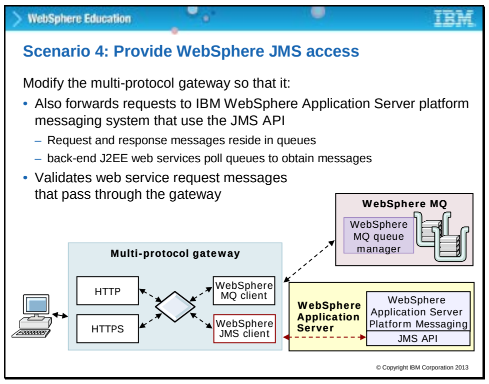

WebSphere Education

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

Scenario 3: Provide WebSphere MQ access
And now, for the third scenario, a modifying the previous example to add another front-side handler that gets messages from an WebSphere MQ queue in parallel with receiving messages with an HTTP and HTTPS. In this example, is also using WebSphere MQ as the back-end server. There can be issues that relate to transactional control between queue managers in such a scenario. WebSphere MQ in covered in greater detail a later section.
To implement this scenario, one must modify the multi-protocol gateway to accept requests from an IBM WebSphere MQ system.
Request and response messages exist 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.
The MPG includes Policy Rules that validate web service request messages that pass through the service.

Slide 21


WebSphere Education

Scenario 4: Provide WebSphere JMS access

Modify the multi-protocol gateway so that it:

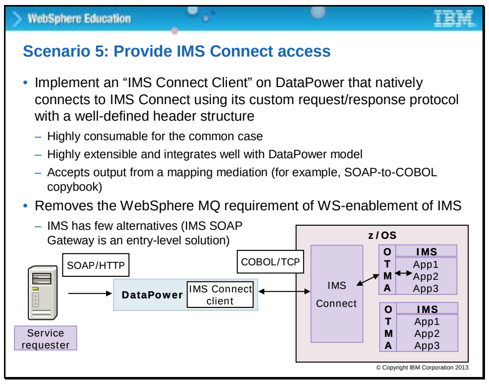
- Also forwards requests to IBM WebSphere Application Server platform messaging system that use the JMS API
  - Request and response messages reside in queues
  - back-end J2EE web services poll queues to obtain messages
- Validates web service request messages that pass through the gateway

Multi-protocol gateway
HTTP
HTTPS
WebSphere MQ client
WebSphere JMS client
WebSphere MQ
WebSphere MQ queue manager
WebSphere Application Server
WebSphere Application Server Platform Messaging
JMS API

© Copyright IBM Corporation 2013

Scenario 4: Provide WebSphere JMS access
For the fourth scenario, WebSphere Application Server’s JMS is added as an extra back-end server, and are dynamically choosing which messaging system to use based on message content.
Modify the multi-protocol gateway so that it also forwards requests to IBM WebSphere Application Server messaging system by using the JMS API. Request and response messages exist in queues, and the back-end Java Platform, Enterprise Edition web services poll those queues to obtain messages.
When again, MPG validates web service request messages that pass through the service.

Slide 22


WebSphere Education
IBM

Scenario 5: Provide IMS Connect access

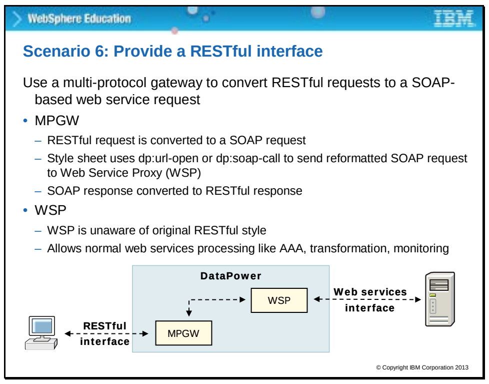
- Implement an “IMS Connect Client” on DataPower that natively connects to IMS Connect using its custom request/response protocol with a well-defined header structure
  - Highly consumable for the common case
  - Highly extensible and integrates well with DataPower model
  - Accepts output from a mapping 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)

Scenario 5: Provide IMS Connect access

The fifth scenario is aimed at existing systems still by using IMS. In this example, DataPower provides an IMS Connect client that effectively allows access to the IMS transactions as though they were web services.

To do processing, one implements 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. It is highly consumable for the common case, is highly extensible, and integrates well with the DataPower model. It also accepts output from a mapping mediation, such as a SOAP-to-COBOL copybook.

In the past, the only way to integrate with IMS was to connect with WebSphere MQ. The DataPower IMS Connect Client removes that requirement by effectively making IMS available as a web service. Besides, WebSphere MQ and now DataPower, IMS has few alternatives to such integration, although there is an entry-level solution in the form of the IMS SOAP Gateway.



WebSphere Education

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 converted to RESTful response
- WSP
  - WSP is unaware of original RESTful style
  - Allows normal web services processing like AAA, transformation, monitoring

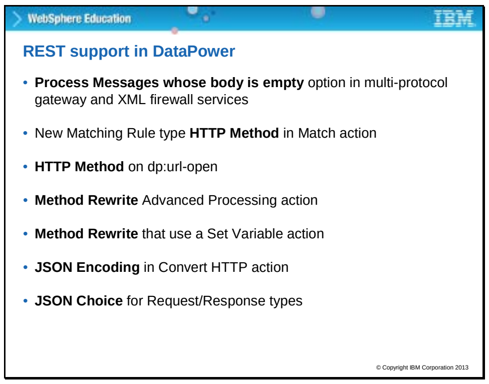
Scenario 6: Provide a RESTful interface

And finally, for the sixth scenario, the topic is how to implement a RESTful interface. Representational State Transfer is supported as a front-side handler, allowing such transactions to interface with other systems.

In this example, a multi-protocol gateway is used to convert RESTful requests to a SOAP-based web service request that is then handled by a Web Services Proxy before being sent to a back-end web server.

In the MPG, the RESTful request is converted to a SOAP request, and a style sheet then uses dp:url-open or dp:soap-call to send reformatted SOAP request to the WSP. SOAP responses from the WSP are then converted back into to RESTful responses. The WSP is unaware of the original RESTful style; it just receives the SOAP request from the MPG like any other SOAP request. The WSP does its normal web services that process like AAA, transformation, and monitoring as the message passes through the service policy.

Slide 24



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 use a Set Variable action
- **JSON Encoding** in Convert HTTP action
- **JSON Choice** for Request/Response types

REST support in DataPower

Here are some of the parameters that support REST in DataPower.

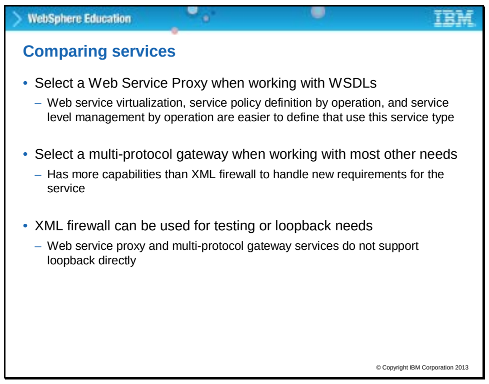
There is a "Process Message whose body is empty" option in the multi-protocol gateway and XML firewall services, and a new Matching Rule type called "HTTP Method" in the Match Action.

One can specify "HTTP Method" in the dp:url-open XSL extension.

In the Advanced Processing Action, there is a "Method Rewrite" option, which can also be configured by using the Set Variable action.

And the Convert HTTP action supports JSON encoding.

Slide 25


WebSphere Education

IBM

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 that use 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

Comparing services

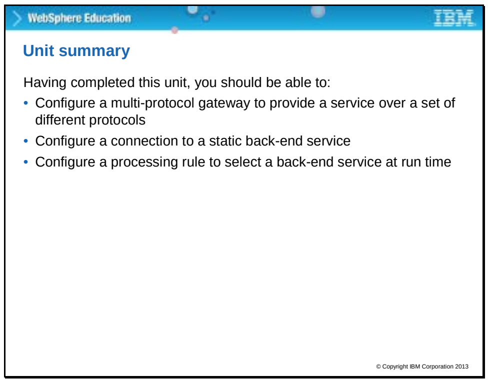
Given that there are several services that seem to offer similar capabilities, when is one chosen over the other?

Generally speaking, select a web service proxy when working with WSDLs since it offers web service virtualization, service policy definition by operation, and service level management by operation. All of which are easier to define by using this service type.

Select a multi-protocol gateway when working with most other needs. It has more capabilities than XML firewall to handle new requirements for the service.

The XML firewall can be used for testing or loopback needs. Remember, the web service proxy and multi-protocol gateway services do not support loopback directly.

Slide 26

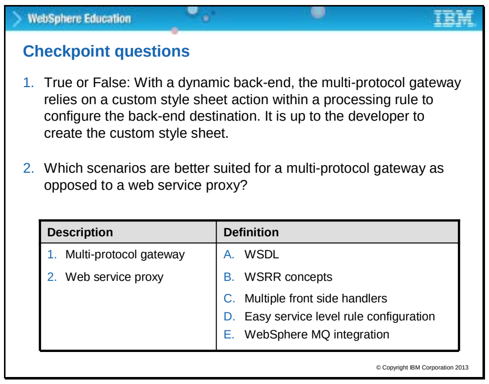

WebSphere Education

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

Slide 27


WebSphere Education IBM

Checkpoint questions

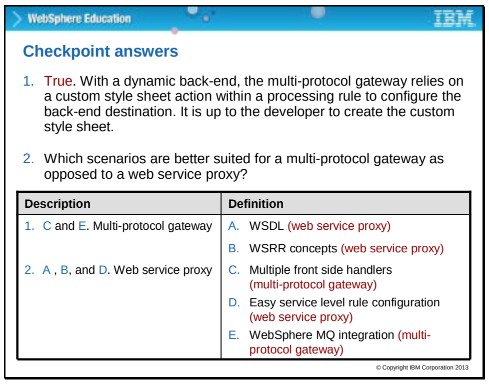
1. True or False: With a dynamic back-end, the multi-protocol gateway relies on a custom style sheet action within a processing rule to configure the back-end destination. It is up to the developer to create the custom style sheet.

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

Description	Definition
1. Multi-protocol gateway	A. WSDL
2. Web service proxy	B. WSRR concepts
	C. Multiple front side handlers
	D. Easy service level rule configuration
	E. WebSphere MQ integration

Take a few moments to read over the questions and write down their answers in the blank spaces in the student notes.

Slide 28



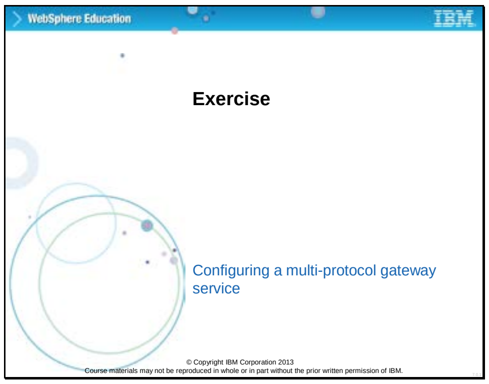
Checkpoint answers

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

The correct answers are highlighted in red.

Slide 29



WebSphere Education

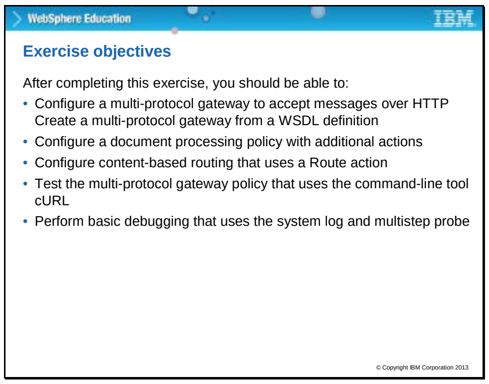
IBM

Exercise

Configuring a multi-protocol gateway service

This slide provides an introduction to the demonstration.

Slide 30



WebSphere Education

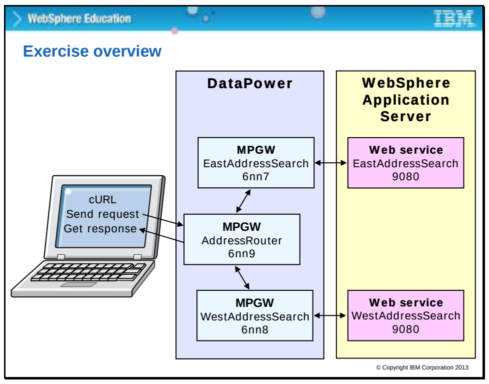
Exercise objectives

After completing this exercise, you should be able to:

- Configure a multi-protocol gateway to accept messages over HTTP Create a multi-protocol gateway from a WSDL definition
- Configure a document processing policy with additional actions
- Configure content-based routing that uses a Route action
- Test the multi-protocol gateway policy that uses the command-line tool cURL
- Perform basic debugging that uses the system log and multistep probe

Here is a list of the exercise objectives.

Slide 31



Exercise overview

DataPower
WebSphere Application Server
MPGW EastAddressSearch 6nn7
Web service EastAddressSearch 9080
cURL
Send request
Get response
MPGW AddressRouter 6nn9
MPGW WestAddressSearch 6nn8
Web service WestAddressSearch 9080