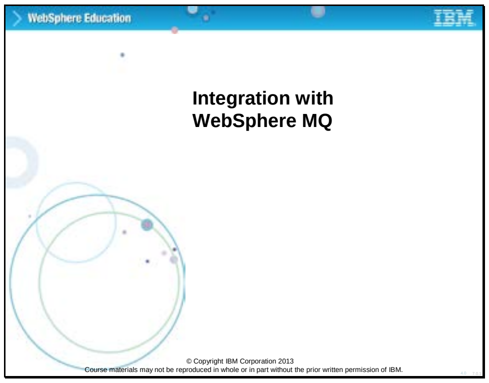
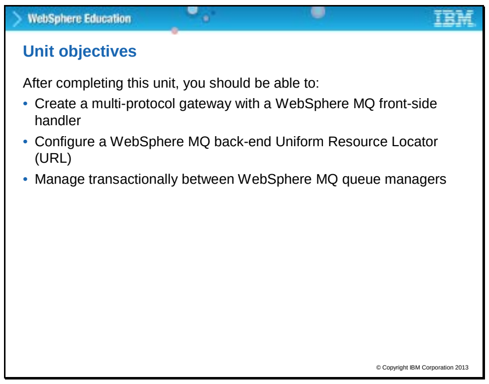
Slide 1



WebSphere Education

Integration with WebSphere MQ

Slide 2

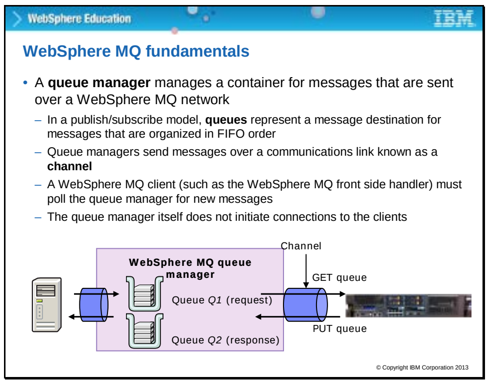


Unit objectives

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 transactionally between WebSphere MQ queue managers

Slide 3


WebSphere Education
IBM

WebSphere MQ fundamentals

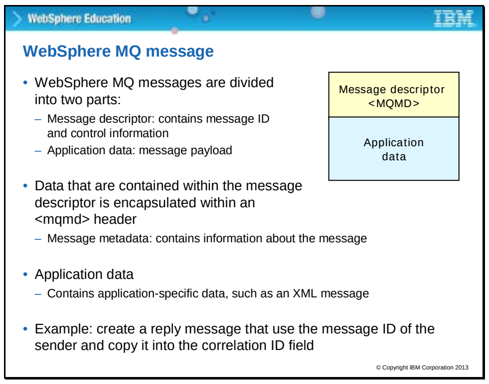
- A **queue manager** manages a container for messages that are sent over a WebSphere MQ network
  - In a publish/subscribe model, **queues** represent a message destination for messages that are organized in FIFO order
  - Queue managers send messages over a communications link known as a **channel**
  - 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

WebSphere MQ fundamentals

Now for a little refresher on WebSphere MQ concepts.
A queue manager manages a container for messages that are sent over a WebSphere MQ network; it is like an electronic delivery service. There are some similarities between a database management system and a queue management system. You might say that queues are like database tables and messages on queues are like rows within tables. But there are some major differences. Databases are designed for long-term storage of data whereas message queuing systems are designed for short-term storage. Databases are designed for random insertion and retrieval of data with indexes whereas message queuing systems are designed for sequential storage and retrieval, with little or no indexing. Databases are generally centralized with the ability for remote application connection whereas message queuing systems are usually distributed as a series of interconnected peer queue managers with locally connected applications.
The most common methodology for message handling within WebSphere MQ is "point-to-point" wherein messages are sent from one application to another. This method is like the way a package can be sent from one person to another by using a courier service. WebSphere MQ also supports a publish/subscribe model, in which queues represent a message destination but instead of being one-to-one, publish/subscribe, as it is often referred to, supports a many-to-many distribution pattern.
When queue managers are interconnected across a network, they send messages to one another over a communications link known as a "message channel". Client applications can also connect to a queue manager over a network by using a different type of communication link that is called an MQI channel. The channel is sometimes called a "client channel", although the correct term is "server connection channel agent" often referred to as SVRCONN.
Unlike "push" technologies like HTTP, an 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.
An important consideration to note here is that there are two ways that an application can connect to a queue manager. If the application is running on the same physical host as the queue manager, it has certain capabilities over and above those capabilities of a "client-attached" application. When an application attaches as

a client, it is running on a physically different system, and connects to the queue manager with a network. DataPower can connect as a client. Now, if you have any exposure to WebSphere MQ, you probably know that there are two modes of client connection. The first mode is standard free client, with limited transactional capability, and the extra-cost “extended transactional client” that implements full two-phase commit support. DataPower implements the former – there is no two-phase commit capability within the DataPower WebSphere MQ interface, although there are some enhancements in that area over and above the normal client, as discovered later in this section.

Slide 4


WebSphere Education

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 that use the message ID of the sender and copy it into the correlation ID field

WebSphere MQ message

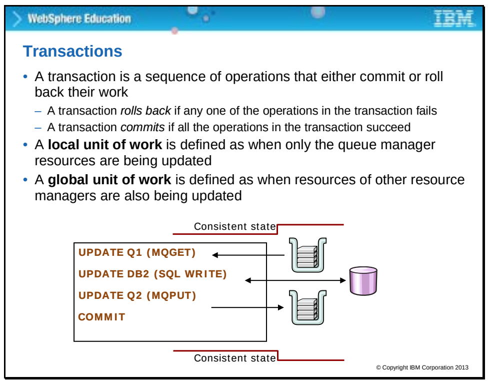
A WebSphere MQ message is divided into two parts – the Message Descriptor, which is a fixed header that contains management information such as the message ID and control information. And the second part is the application data, which is the message payload.

After being parsed by the WebSphere MQ interface, data that is contained within the message descriptor is encapsulated within an <mqmd> header for ease of use. It is message metadata that contains information about the message, such as priority, security, persistence, identification, and timestamp information. You might regard the Message Descriptor as the outside of the envelope, which contains information that the letter carrier wants to know about.

The application data portion of the message contains application-specific data, such as an XML message. But it might also contain non-XML data such as COBOL copybook format fields, or even a binary payload, like a JPEG image. This payload is like the letter inside the envelope – it is intended for the recipient to look at, and the letter carrier never knows anything about it. The maximum size of a WebSphere MQ message application payload is 100 megabytes.

Even though the Message Descriptor is primarily intended for use by the queue manager, applications can manipulate it as well. For example, you might want to create a reply message by using the senders message ID and copy it into the correlation ID field. This usage is a common WebSphere MQ programming convention.

Slide 5


WebSphere Education

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

Transactions

Some definitions that relate to transaction handling.

A transaction is a sequence of operations that either commit or rollback their work.

A transaction rolls back if any one of the operations in the transaction fails.

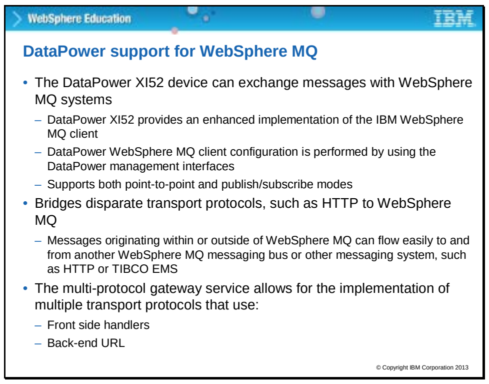
A transaction commits if all the operations in the transaction succeed.

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.

This diagram illustrates a typical host-based application that is directly connected to a queue manager, and also interacting with a database on the same system. The horizontal red line at the top indicates a stable situation in which the queue manager and the database are waiting for some activity. The application begins a transaction. It reads a message from a queue, which causes the queue manager to flag the message for pending delete. It requests an update to the database, which causes the database manager to log those pending changes. It writes a message to a queue, which causes the queue manager to log the pending changes to that queue. Now there is an uncommitted transaction, and the changes that are made to both the queues and the database tables are not yet permanent. Now issue a “commit” command, which causes the queue manager to delete the message that was retrieved from the top queue. And causes the database manager to make the changes to the database permanent, and causes the queue manager to permanently write the message to the bottom queue. Now back to a consistent state. This technique is commonly known as “two-phase commit,” and is only supported by WebSphere MQ applications that are running on the same system as the queue manager. Since DataPower connects as a client, it cannot participate in a two-phase commit scenario between WebSphere MQ and some other resource manager, like a database. But as shown shortly, there are some things DataPower can do with transactionally between queue managers.

Slide 6


WebSphere Education

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 that use:
  - Front side handlers
  - Back-end URL

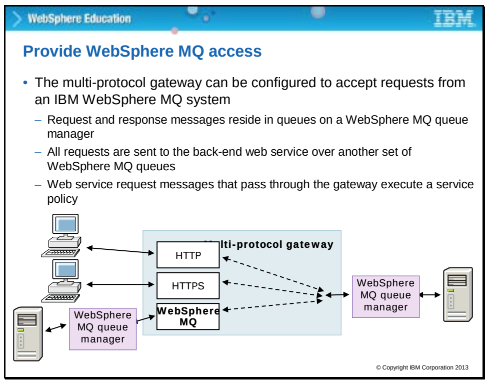
DataPower support for WebSphere MQ

Only the DataPower XI50 can exchange messages with WebSphere MQ systems. It provides an enhanced implementation of the standard WebSphere MQ client, but now stress again that it is NOT the WebSphere MQ Extended Transactional Client. DataPower WebSphere MQ client configuration is performed by using the DataPower management interfaces, and it supports both point-to-point and publish/subscribe (pub-sub) modes of operation.

The whole philosophy behind the XI50 is to provide a bridge between disparate transport protocols such as HTTP and 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 with DataPower.

The multi-protocol gateway service allows for the implementation of multiple transport protocols by using different Front side handlers and different back-end URLs.

Slide 7

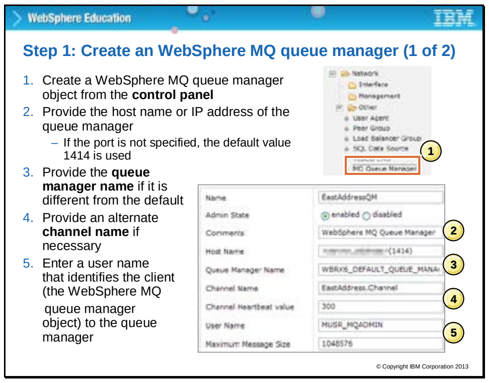


Provide WebSphere MQ access

- 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

Provide WebSphere MQ Access
So here is an illustration of how the XI50 can provide WebSphere MQ access.
The multi-protocol gateway can be configured to accept requests from an IBM WebSphere MQ system with the Front Side Handler. It uses two different queues on the WebSphere MQ system to hold the Request and response messages. This diagram shows how all requests are sent to the back-end web service over another set of WebSphere MQ queues. In other words, assume that there is a web service that communicates with WebSphere MQ, represented by the rightmost server icon in this diagram.
As is the case with HTTP messages, MQ-based web service request messages that pass through the gateway runs a service policy.
This example shows two-way, request-response message processing. Now although DataPower expects all messages to be part of such a “round-trip” conversation, it can pass messages in one direction only. This is as part of a “fire-and-forget” scenario. Although the interface to WebSphere MQ from DataPower assumes a pair of queues, it is not mandatory to use them both.

Slide 8



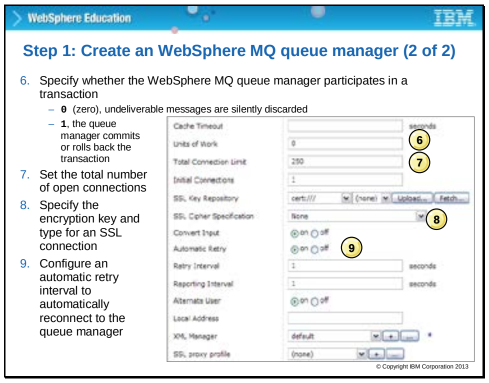
Step 1: Create an WebSphere MQ queue manager (1 of 2)

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	EastAddressQM
Admin State	[✓ enabled] disabled
Comments	WebSphere MQ Queue Manager
Host Name	[redacted](1414)
Queue Manager Name	WBRK6_DEFAULT_QUEUE_MANA
Channel Name	EastAddress.Channel
Channel Heartbeat value	300
User Name	MUSR_MQADMIN
Maximum Message Size	1048576

Step 1: Create an WebSphere MQ queue manager (1 of 2)
Walk through the steps that are needed to set up a DataPower system for WebSphere MQ usage.
First, create a WebSphere MQ queue manager connection object from the control panel, as indicated by number 1. Reiterate the point that DataPower can act as an WebSphere MQ client only – there is NO WebSphere MQ Queue manager that is installed and running on DataPower, even though the menu item implies that one is being created. It should more properly read, “WebSphere MQ Queue manager Connection Object” – it is just an object that defines how to connect remotely to a real queue manager somewhere else in the network.
Number 2 shows where to specify the host name or IP address of the queue manager along with the port number. This field can be entered in traditional format with the IP address followed by a colon followed by the port. Or it can be entered in WebSphere MQ “CONNAME” format, which has the port number in parentheses after the IP address. If the port is not specified, the default value 1414 is used.
Number 3 indicates where to provide the REAL queue manager name on the host system. The value can be left blank if your WebSphere MQ administrator defined a “default” queue manager.
Number 4 indicates where you provide the name of the SVRCONN channel. If your WebSphere MQ administrator allows it, the name can be left blank and DataPower connects to the default SVRCONN on the queue manager. Most WebSphere MQ administrators consider this connection to be a security risk and do not use the default.
Number 5 shows where you can enter a user name that identifies the DataPower client to the queue manager. In this example, the default Windows WebSphere MQ administrative user ID is used – another security risk, according to most WebSphere MQ administrators – so check with yours to find out what they want you to use here.

Slide 9


WebSphere Education

Step 1: Create an WebSphere MQ queue manager (2 of 2)

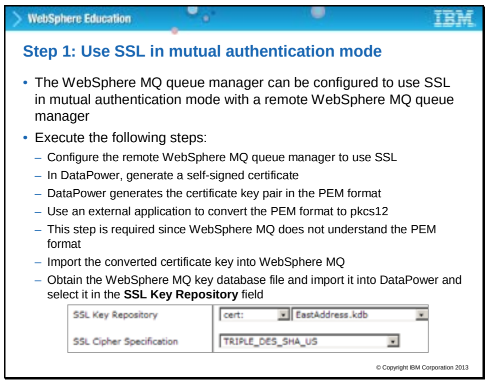
6. Specify whether the WebSphere MQ queue manager participates in a transaction
   - **0** (zero), undeliverable messages are silently discarded
   - **1**, the queue manager commits or rolls back the transaction
7. Set the total number of open connections
8. Specify the encryption key and type for an SSL connection
9. Configure an automatic retry interval to automatically reconnect to the queue manager

Cache Timeout		seconds
Units of Work	0	6
Total Connection Limit	250	7
Initial Connections	1	
SSL Key Repository	cert:/// (none) Upload... Fetch...	
SSL Cipher Specification	None	8
Convert Input	[✓ on] off	
Automatic Retry	[✓ on] off	9
Retry Interval	1	seconds
Reporting Interval	1	seconds
Alternate User	[✓ on] off	
Local Address		
XML Manager	default + ...	*
SSL proxy profile	(none) + ...	

© Copyright IBM Corporation 2013

Step 1: Create an WebSphere MQ queue manager (2 of 2)
Continuing with the definition of the Queue manager connection object...
Number 6 shows where you can specify whether the WebSphere MQ queue manager participates in transaction management. The participation is probably the most important parameter on this page, and it can be set to “0” or “1.” This parameter affects transactional control over the queue manager, depending on other parameters that are set in the Front-Side Handler or the back-end URL (or both). A value of zero causes undeliverable messages are silently discarded. A value of 1 ensures that the Queue manager issues a commit or rollback, which depends on the success or failure of message processing. Rollbacks are covered in more detail later in the presentation.
Number 7 shows where you can set the total number of open connections. Although the Server Connection Channel Agent on the Queue manager can handle many thousands of simultaneous connections, you might want to limit this value for performance reasons.
Number 8 shows where you can specify the encryption key and type for an SSL connection.
Number 9 allows you to configure an automatic retry interval to automatically reconnect to the queue manager in the event of a connection failure.

Slide 10



WebSphere Education

IBM

Step 1: Use SSL in mutual authentication mode

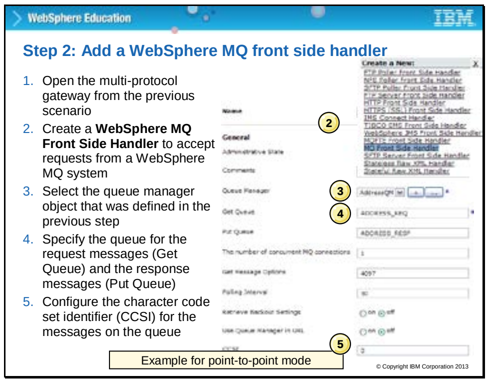
- The WebSphere MQ queue manager can be configured to use SSL in mutual authentication mode with a remote WebSphere MQ queue manager
- Execute the following steps:
  - Configure the remote WebSphere MQ queue manager to use SSL
  - In DataPower, generate a self-signed certificate
  - DataPower generates the certificate key pair in the PEM format
  - Use an external application to convert the PEM format to pkcs12
  - This step is required since WebSphere MQ does not understand the PEM format
  - Import the converted certificate key into WebSphere MQ
  - Obtain the WebSphere MQ key database file and import it into DataPower and select it in the **SSL Key Repository** field

SSL Key Repository	cert: EastAddress.kdb
SSL Cipher Specification	TRIPLE_DES_SHA_US

© Copyright IBM Corporation 2013

Step 1: Use SSL in mutual authentication mode
The WebSphere MQ queue manager connection object within DataPower can be configured to use SSL in mutual authentication mode with a remote WebSphere MQ queue manager. To incorporate SSL in mutual authentication mode, run the following steps:
Configure the remote WebSphere MQ queue manager to use SSL; then, in DataPower, generate a self-signed certificate. Since 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. When that is done, you, or your WebSphere MQ administrator can 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.

Slide 11



WebSphere Education

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

Name	
General	
Administrative State	
Comments	
Queue Manager	AddressQM *
Get Queue	ADDRESS_REQ *
Put Queue	ADDRESS_RESP
The number of concurrent MQ connections	1
Get Message Options	4097
Polling Interval	30
Retrieve Backout Settings	on [✓ off]
Use Queue Manager in URL	on [✓ off]
CCSI	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
Stateful Raw XML Handler
2
3
4
5

Example for point-to-point mode

Step 2: Add a WebSphere MQ front side handler

Next, add a WebSphere MQ front side handler.

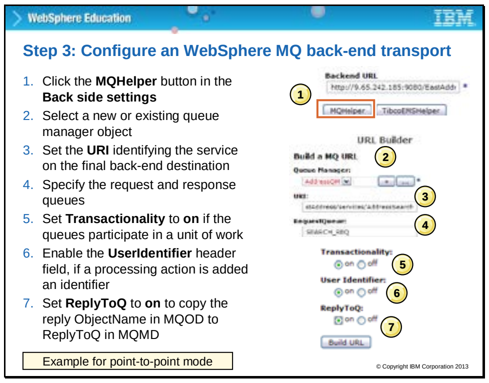
Use the multi-protocol gateway from the previous scenario, and create a WebSphere MQ Front Side Handler to accept requests from a WebSphere MQ system. Number 2 indicates where to select the WebSphere MQ front side handler after which a panel pops up as shown on the lower half of this slide.

Number 3 indicates where to select the queue manager object that was defined in the previous step.

Number 4 is where to specify the Get queue for the request messages and number 5 is the Put queue for response messages. There is no asterisk beside the “Put” queue – the process indicates that it is optional as would be the case in a “fire-and-forget” scenario.

Number 6 allows one to configure the character code set identifier (CCSI) for the messages on the queue. You would normally leave the field blank and allow the queue manager handle it.

Slide 12


WebSphere Education

Step 3: Configure an WebSphere MQ back-end transport

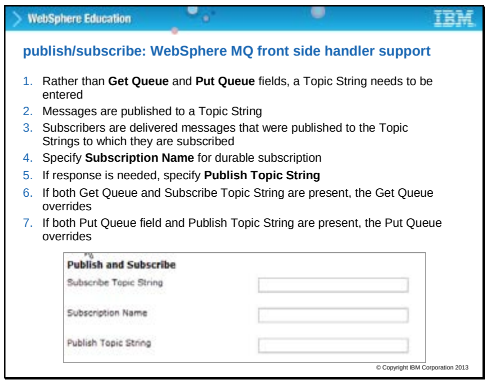
1. Click the **MQHelper** button in the **Back side settings**
2. Select a new or existing queue manager object
3. Set the **URI** identifying the service on the final back-end destination
4. Specify the request and response queues
5. Set **Transactionality** to **on** if the queues participate in a 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

Example for point-to-point mode

Backend URL: http://9.65.242.185:9080/EastAddr
URL Builder
Build a MQ URL
Queue Manager: AddressQM
URI: sAddress/services/AddressSearch
RequestQueue: SEARCH_REQ
Transactionality: [✓ on] off
User Identifier: [✓ on] off
ReplyToQ: [✓ on] off

Step 3: Configure a WebSphere MQ back-end transport
Now look at the back-end transport parameters.
Number 1 shows how to can click the MQHelper button, which opens a panel as shown on the lower half of this slide.
Number 2 is where to select a new or existing queue manager object.
Number 3 allows one to set the URI that identifies the service on the final back-end destination.
Number 4 is where to specify the request and response queues.
Number 5 allows one to set Transactionality to “on” if the queues participate in a unit of work. The process becomes significant in the scenario where to have a queue manager that feeds messages to a Front-Side Handler, and another queue manager at the back-end. More about this topic later.
Number 6 is where to can enable the UserIdentifier header field, which allows a processing action to modify the Message Descriptor User Identifier.
Number 7 is the Set ReplyToQ option. If it is turned “on”, the ReplyQueue parameter you set in number 4 are copied into the ReplyToQ name field in the Message Descriptor. The receiving application can then use this process to send its response to the queue you are expecting. It is like having a return address that is printed on the upper left corner of an envelope.

Slide 13


WebSphere Education

IBM

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

Publish and Subscribe

Subscribe Topic String	
Subscription Name	
Publish Topic String	

Pub-sub: WebSphere MQ front side handler support

As mentioned earlier, WebSphere MQ supports both point-to-point messaging, in addition to publish/subscribe, which is many-to-many. DataPower likewise supports both models.

Rather than Get Queue and Put Queue fields, a Topic String is specified, and messages are published to that topic for later retrieval by applications that subscribe to that same topic.

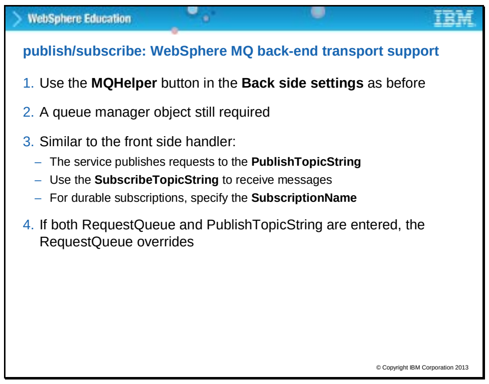
The publish/subscribe model supports a mode of subscription called “durable”, which means that the queue manager collects and holds applicable messages for a specific subscriber even if it is not currently connected. Then, later, when that subscriber reconnects, the queue manager delivers all the saved messages. To enable this mode of operation, each subscriber must supply the queue manager with a unique “subscription name” that identifies it upon reconnection. If a client connects as a “non-durable” subscription, it gets applicable messages sent to it so long as it remains connected. Upon disconnection, the queue manater discards the subscription request.

A DataPower front-side handler can operate in both subscriber mode to get messages, and publisher mode, to send replies. When intending to send responses, one must specify a Publish Topic String.

It is also possible to mix point-to-point and publish/subscribe within the same front-side handler. If both the Get Queue and the Subscribe Topic String are present, the Get Queue takes precedence, and the FSH operates in point-to-point mode for incoming messages. If both the Put Queue field and the Publish Topic String are present, the Put Queue takes precedence, and the FSH operates in point-to-point mode for outgoing messages.

DataPower supports WebSphere MQ Publish-and-Subscribe methodology.

Slide 14


WebSphere Education

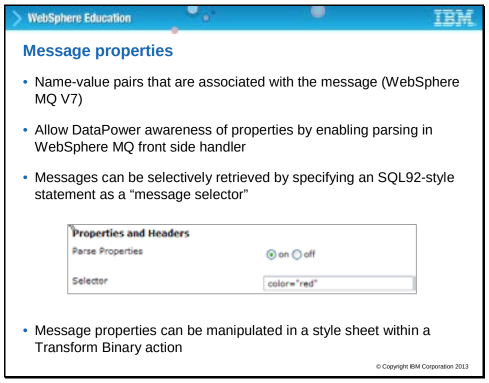
publish/subscribe: WebSphere MQ back-end transport support

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

Pub-sub: WebSphere MQ back-end transport support
Just like the Front Side handler, the back-end transport can also support publish/subscribe.
Like with point-to-point, click the MQHelper button in the Back side settings and it opens a panel as shown here.
A queue manager object that is still required for the connection.
The rest is similar to the front side handler, in that the service publishes requests to the PublishTopicString, and uses the SubscribeTopicString to receive messages.
Durable subscriptions are also supported, and require a SubscriptionName.
The red boxes on this slide are wrongly positioned – one should be surrounding the RequestQueue and PublishTopicString; the other should be surrounding the ReplyQueue and SubscribeTopicString. Similar to the FSH, if both Request Queue and PublishTopicString are entered, the Request Queue overrides, and if both the ReplyQueue and SubscribeTopicString are entered, the Reply Queue overrides.

Slide 15



WebSphere Education

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”

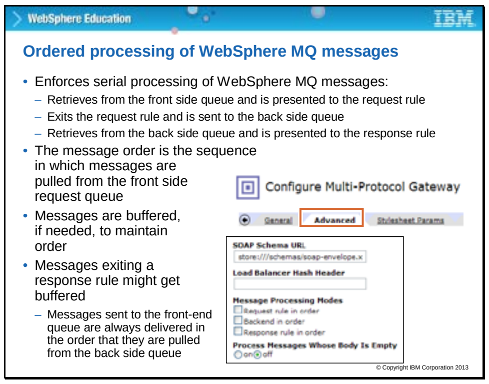
Properties and Headers

Parse Properties	[✓ on] off
Selector	color="red"

- Message properties can be manipulated in a style sheet within a Transform Binary action

Message properties
Since the advent of WebSphere MQ version 7, full support for JMS necessitated the carrying of message properties as name-value pairs within each message in addition to the traditional Message Descriptor. These message properties are parsed out of an incoming message by DataPower, and made available as service variables for use within policy rule actions.
JMS support within WebSphere MQ allows messages to be selectively retrieved by specifying an SQL92-style statement as a “message selector”. However, the process is not recommended as it can cause performance problems within the queue manager. A better technique is to have DataPower read **all** messages from a queue and decide how to handle them within the service policy. Message properties can be manipulated in a style sheet within a Transform Binary action, and DataPower is a lot faster than most servers when it comes to parsing, selecting, and routing messages.
Even though native support for WebSphere MQ JMS is not provided by DataPower, it is possible to participate in such an environment by careful manipulation of the message properties. This process is not an elegant solution, but can provide some useful functionality.
.

Slide 16



WebSphere Education

Ordered processing of WebSphere MQ messages

- Enforces serial processing of WebSphere MQ messages:
  - Retrieves from the front side queue and is presented to the request rule
  - Exits the request rule and is sent to the back side queue
  - Retrieves from the back side queue and is presented to the response rule
- The message order is the sequence in which messages are pulled from the front side request queue
- Messages are buffered, if needed, to maintain order
- Messages exiting a response rule might get buffered
  - Messages sent to the front-end queue are always delivered in the order that they are pulled from the back side queue

Configure Multi-Protocol Gateway

General | Advanced | Stylesheet Params

SOAP Schema URL: store:///schemas/soap-envelope.x

Load Balancer Hash Header:

Message Processing Modes
- ☐ Request rule in order
- ☐ Backend in order
- ☐ Response rule in order

Process Messages Whose Body Is Empty: on [✓ off]

Ordered processing of WebSphere MQ messages

Those of you who are familiar with WebSphere MQ programming would know about the “Logical Order” parameter that is part of the MQGET and MQPUT calls within a WebSphere MQ application. DataPower supports logical order by virtue of the check box under the advanced tab in the MPG. Although WebSphere MQ supposedly handles messages in a sequential fashion, there are various factors that can cause messages to be delivered in a sequence different from how they were initially sent. WebSphere MQ documentation is careful to state that message sequence is NOT assured unless you use the “Logical Order” option.

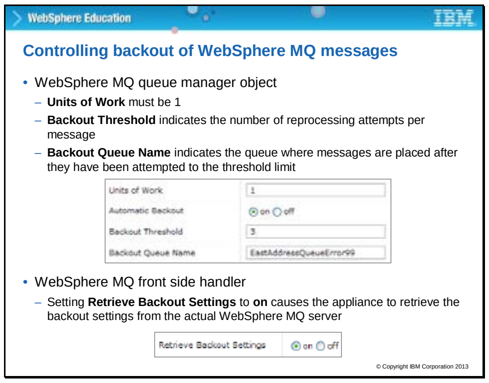
To enforce true serial processing of WebSphere MQ messages, you can check the “Request rule in order” box. This action makes sure that messages are retrieved from the front-side queue and presented to the request rule according to their original sequence. This action assumes that the sending program chose that option when putting the messages to the queue in the first place.

The “Backend in order” check box causes the DataPower back-end interface to tell its destination queue manager to preserve message grouping onto its queues.

The third check box (“Response rule in order”) retrieves messages in logical sequence from the back-side queue and presents them to the response rule in that order.

The message order is the sequence in which messages are pulled from the front-side request queue, and messages are buffered, if needed, to maintain that order. Similarly, messages that exit a response rule might also get buffered, and are sent to the front end queue in the order they were pulled from the back-side queue.

Slide 17



WebSphere Education

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 have been attempted to the threshold limit

Units of Work	1
Automatic Backout	[✓ on] off
Backout Threshold	3
Backout Queue Name	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

Retrieve Backout Settings	[✓ on] off

Controlling backout of WebSphere MQ messages

A few words about backout of WebSphere MQ messages.

Imagine the scenario where you read a message from a queue under syncpoint, encounter an error with that message – which necessitates a rollback – then read a message from that same queue to keep processing. Because of the way WebSphere MQ handles rollbacks, it will deliver that same message the next time you issue a request. If you encounter the same error and roll it back again, the process repeats. This process is commonly referred to as a “poison message.” To prevent this scenario from becoming an endless loop, you can set a back-out threshold. The backout specifies the maximum number of times you are willing to process the same message repeatedly before doing something different. The usual way to break out of this loop is to write the poison message to a different “error” queue; then issue a commit, which removes the message from the input queue permanently.

You can set these parameters in the WebSphere MQ queue manager object.

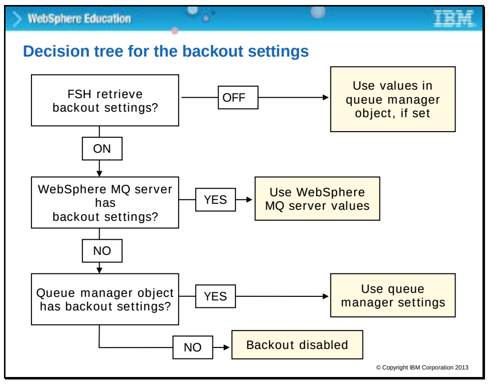
For the process to work, “Units of Work” must be set to 1.

Then, you set “Backout Threshold” to indicate the number of reprocessing attempts per message.

When that threshold limit is reached, the “Backout Queue Name” indicates the queue where messages are placed to break out of the retry loop. Although you can specify a hardcoded Backout Queue Name here within DataPower parameters, it is more common for your WebSphere MQ administrator to set different backout queue names for each local queue. This process gives more granular control over how poison messages are handled.

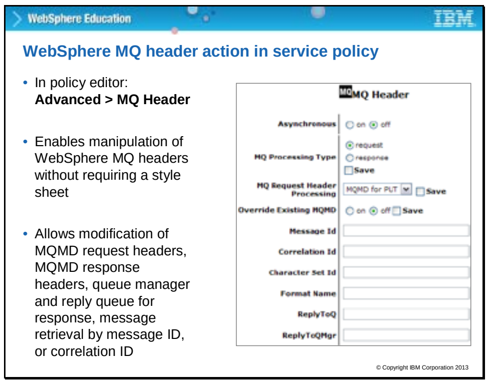
To use the Backout Queue Name that is associated with the local queue rather than the generic one specified in the Queue manager object. There is a parameter that is set on the WebSphere MQ front side handler. Setting Retrieve Backout Settings to “on” causes the appliance to retrieve the backout settings for each local queue from the actual WebSphere MQ server. Sorry about the screen capture that overlays the slide text!

Slide 18


WebSphere Education
IBM
Decision tree for the backout settings
FSH retrieve backout settings?
OFF
Use values in queue manager object, if set
ON
WebSphere MQ server has backout settings?
YES
Use WebSphere MQ server values
NO
Queue manager object has backout settings?
YES
Use queue manager settings
NO
Backout disabled
© Copyright IBM Corporation 2013

Decision tree for the backout settings
This flowchart shows the logical sequence of events that determine where these parameters are found, if at all.
If the Front-Side Handler radio button that requests retrieval of the Queue manager settings is off, use the default values set in the DataPower Queue manager object.
If that button is on, ask the Queue manager to send those values, and use them.
If the Queue manager responds with null values, when again, fall back to the default values set in the DataPower Queue manager object.
If there are no values that are set in the DataPower Queue manager object, transactional control is disabled – it is as though one set the “Units of Work” parameter to zero.

Slide 19


WebSphere Education

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

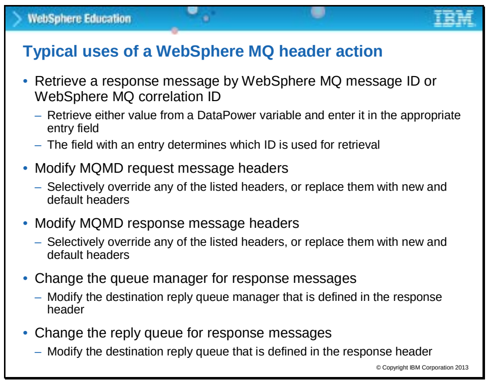
Asynchronous	○ on [✓ ● off]
MQ Processing Type	[✓ ● request] ○ response ☐ Save
MQ Request Header Processing	MQMD for PUT ☐ Save
Override Existing MQMD	○ on [✓ ● off] ☐ Save
Message Id	
Correlation Id	
Character Set Id	
Format Name	
ReplyToQ	
ReplyToQMgr	

WebSphere MQ Header action in service policy

You remember mention about how there is a whole lot of fields in the Message Descriptor that can be manipulated? Here is how to do that. In the policy editor, click and drag the Advanced action to the rule; then choose "MQ Header". The panel that pops up enables manipulation of WebSphere MQ headers without requiring a style sheet, although using a Transform Action with a style sheet would give you more flexibility in controlling these fields.

It allows modification of MQMD request headers, MQMD response headers, queue manager and reply queue for response, message retrieval by message ID or correlation ID. There exists the ability to use variables in many of these fields.

Slide 20



WebSphere Education

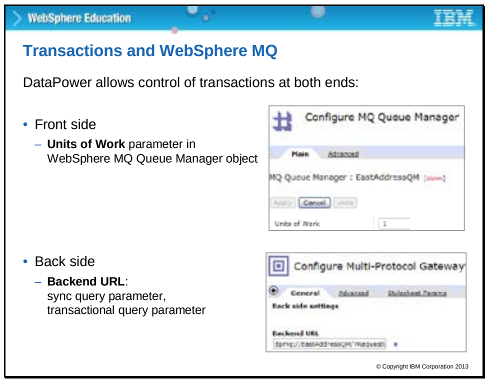
Typical uses of a WebSphere MQ header action

- 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

Typical uses of an WebSphere MQ Header action

Some typical uses of the WebSphere MQ Header action might be:
Retrieve a response message by using the WebSphere MQ message ID or WebSphere MQ correlation ID. This process can be done by retrieving either value from a DataPower variable and entering it in the appropriate entry field. The field with an entry determines which ID is used for retrieval. Blank fields are ignored.
You might modify MQMD request message headers by selectively overriding any of the listed headers, or by replacing them with new and default headers.
You might modify MQMD response message headers in a similar fashion.
You might change the queue manager that is used for response messages by modifying the destination reply queue manager that is defined in the response header. And you might similarly change the reply queue that is used for response messages.

Slide 21



WebSphere Education

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

Configure MQ Queue Manager
Main
Advanced
Apply
Cancel
Undo

MQ Queue Manager : EastAddressQM [down]

Units of Work 1

Configure Multi-Protocol Gateway
General
Advanced
Stylesheet Params
Back side settings

Backend URL dpmq://EastAddressQM/?RequestQ

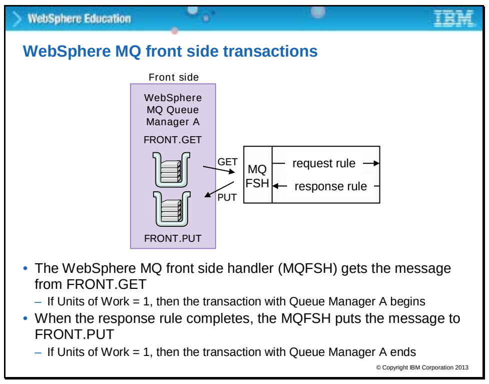
Transactions and WebSphere MQ

DataPower allows control of WebSphere MQ transactions at both ends. On the front-side, the “Units of Work” parameter in WebSphere MQ Queue manager object controls whether messages are pulled from the input queue under syncpoint control, allowing for commit and rollback processing.

On the Back-side, the back-end URL includes two parameters that control how messages are handled on the Request and Reply queues. The Sync parameter and the Transactional parameter are explained over the next few slides.

This situation is where the use of transaction control can become tricky. This complexity can arise especially when there is a queue manager that is accessed by a Front-Side Handler, and a queue manager that is accessed as a back-end server. In some circumstances, the Front-Side and the Back-Side queue manager might be one-and-the-same, which can lead to even more complexity.

Slide 22


WebSphere Education
IBM
Front side
WebSphere MQ Queue Manager A
FRONT.GET
GET
PUT
FRONT.PUT
MQ FSH
request rule
response rule

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

WebSphere MQ front-side transactions

Now walk through a scenario where there is a WebSphere MQ Front-Side handler, and a WebSphere MQ Back-Side URL. Here is the Front-Side.

The WebSphere MQ front side handler (MQFSH) gets the message from FRONT.GET.

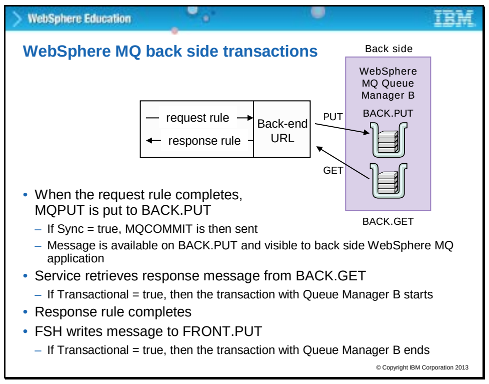
If Units of Work = 1, then the transaction with Queue manager A begins.

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.

Assuming the queue manager object has “units of work” set to “1,” the action of getting a message from the input queue starts a new transaction. The queue manager flags the message for pending delete, and waits for a commit command from the DataPower box. The DataPower service issues the commit only after it successfully writes the reply to the PUT queue, which then causes the deletion of the input message from the input queue. In other words, DataPower regards a complete conversational round trip to be a single unit of work.

Slide 23


WebSphere Education
IBM
Back side
WebSphere MQ Queue Manager B
BACK.PUT
request rule
response rule
Back-end URL
PUT
GET
BACK.GET

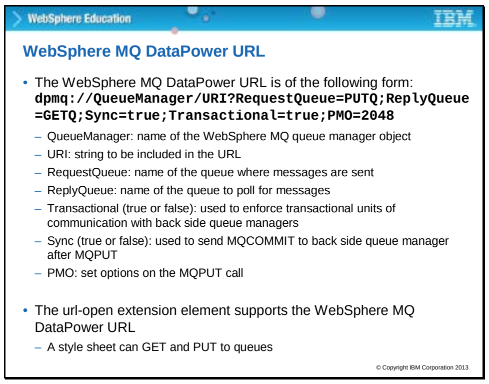
WebSphere MQ back side transactions

- 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

WebSphere MQ back-side transactions

And now, here is the back side.
When the request rule completes, an WebSphere MQ message is put to BACK.PUT.
If Sync = true, a commit command is then sent to the queue manager.
The message is then made available on BACK.PUT and visible to back side WebSphere MQ application.
The DataPower service retrieves the response message from BACK.GET.
If Transactional = true, then the transaction with Queue manager B starts.
When the Response rule completes, the FSH writes the message to FRONT.PUT.
When the message is successfully written to FRONT.PUT, and if Transactional = true, then a commit command is sent to Queue manager B, and the transaction with Queue manager B ends.
There are two parameters that control units of work on the back-side. “Sync” tells the back-end queue manager to assume that a commit command issued at the time when the DataPower service writes a message to the PUT queue. The process might be necessary if the same queue manager is controlling the front-side queues and the back-side queues. It regards the transaction that was started when the message was read by the Front-Side handler as still being open. Therefore it would not allow an application to read from the back-end PUT queue until that transaction is closed. If “Transactional” is true when the message is retrieved from the GET queue, a new unit of work is started and is committed if the message is successfully transmitted back to the Front-Side Handler.

Slide 24


WebSphere Education

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 be included 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

WebSphere MQ DataPower URL

The WebSphere MQ DataPower URL is of the following form:

```
dpmq://QueueManager/URI?RequestQueue=PUTQ;ReplyQueue=GETQ;Sync=true;Tra
nsactional=true;PMO=2048
```

QueueManager is the name of the WebSphere MQ queue manager object.

URI is the string to include in the URL.

RequestQueue is the name of the queue where messages are sent.

ReplyQueue is the name of the queue to poll for messages.

Transactional (true or false) is used to enforce transactional units of communication with back side queue managers, as explained on the previous slide.

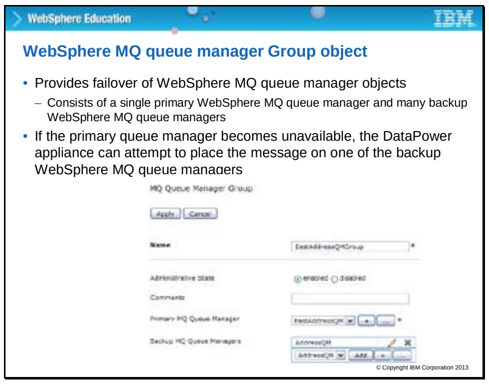
Sync (true or false) is used to send a commit command to the back side queue manager after putting to the queue.

PMO is used to set options on the PUT call. It is an arithmetic addition of a series of options that control such things as generating new message or correlation IDs, and message security context handling. In this example, a value of 2048 tells the queue manager that the DataPower service is in control of setting all context values.

The url-open extension element supports the WebSphere MQ DataPower URL, which means a style sheet can GET and PUT to queues.

Note the protocol prefix “dpmq” has special significance for DataPower in that it refers to a queue manager connection object that is previously defined. DataPower also supports a native WebSphere MQ protocol prefix of “mq” that allows direct connectivity to a queue manager without the necessity of defining a queue manager connection object.

Slide 25


WebSphere Education

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	EastAddressQMGroup *
Administrative State	[✓ enabled] disabled
Comments	
Primary MQ Queue Manager	EastAddressQM *
Backup MQ Queue Managers	AddressQM

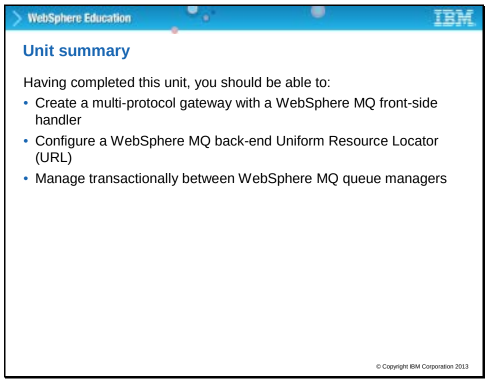
WebSphere MQ queue manager Group object

A word about the WebSphere MQ Queue Manager Group object – an option exists within client-based WebSphere MQ applications to supply a Client Channel Definition Table, which lists candidate queue managers for connection in the event of connection failure. Although DataPower does not support the native CCDT, it simulates it by providing a queue manager group object, as shown here.

When defined, this object provides failover of WebSphere MQ queue manager objects by listing 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 connect to one of the backup WebSphere MQ queue managers.

Slide 26



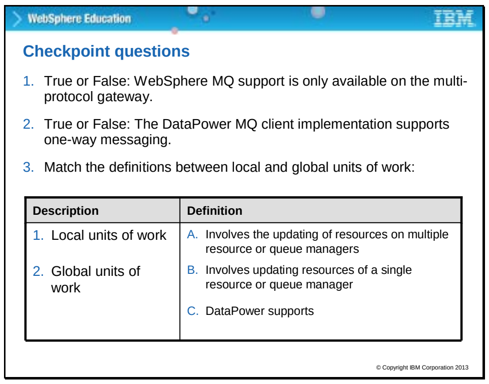
WebSphere Education

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 transactionally between WebSphere MQ queue managers

Slide 27


WebSphere Education

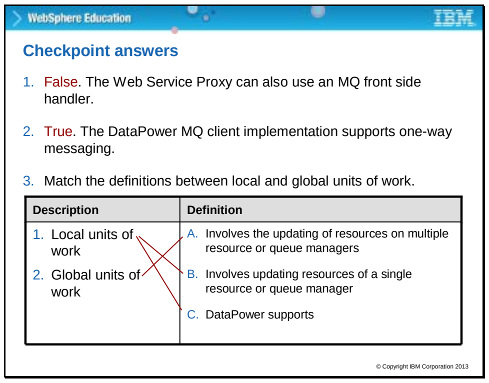
IBM

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

Slide 28



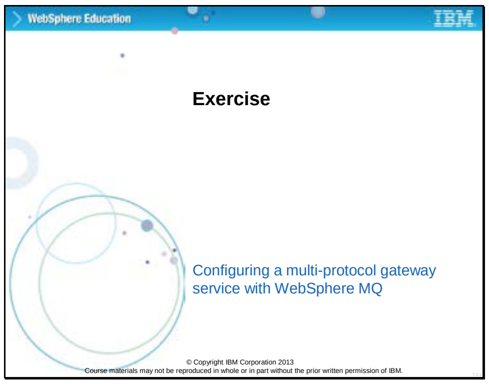
WebSphere Education

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 2. Global units of work	A. Involves the updating of resources on multiple resource or queue managers B. Involves updating resources of a single resource or queue manager C. DataPower supports

Slide 29



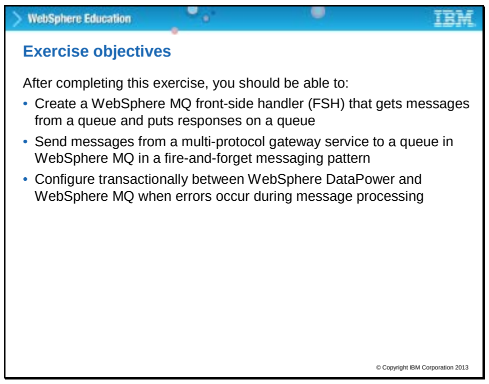
WebSphere Education

IBM

Exercise

Configuring a multi-protocol gateway service with WebSphere MQ

Slide 30



WebSphere Education

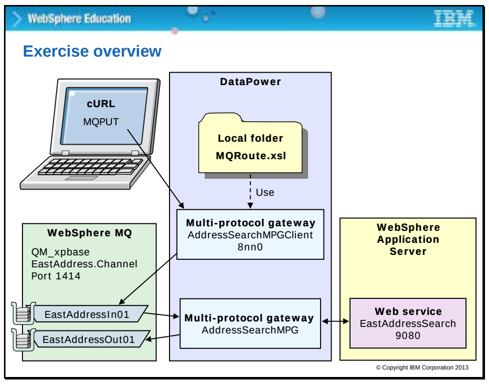
Exercise objectives

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 transactionally between WebSphere DataPower and WebSphere MQ when errors occur during message processing

Notes:

Slide 31



WebSphere Education

Exercise overview