DataPower-MQ Integration Deep Dive

Robin Wiley
(Robin Wiley Training)

Your Presenter: Robin Wiley

- **Senior Instructor, IBM Messaging Products**
  - MQ Administration & Application Development
  - DataPower Administration & Service Development
  - Integration Bus Administration & Message Flow Development
- **IBM Certified:**
  - MQ Administrator
  - MQ Solution Designer
  - DataPower Solution Implementer

- **Over 40 years IT Industry Experience**
  - Network Integration
  - Managing Software Development
- **Experience: 20 years with MQ; 10 with DataPower**
  - Technical Architecture, Project Management, Installation, Training
- **Effective Instructor**
  - Over 35 years experience in corporate training and adult education
  - Brings magic to the classroom (Member, Academy of Magical Arts)

Agenda

Focus:

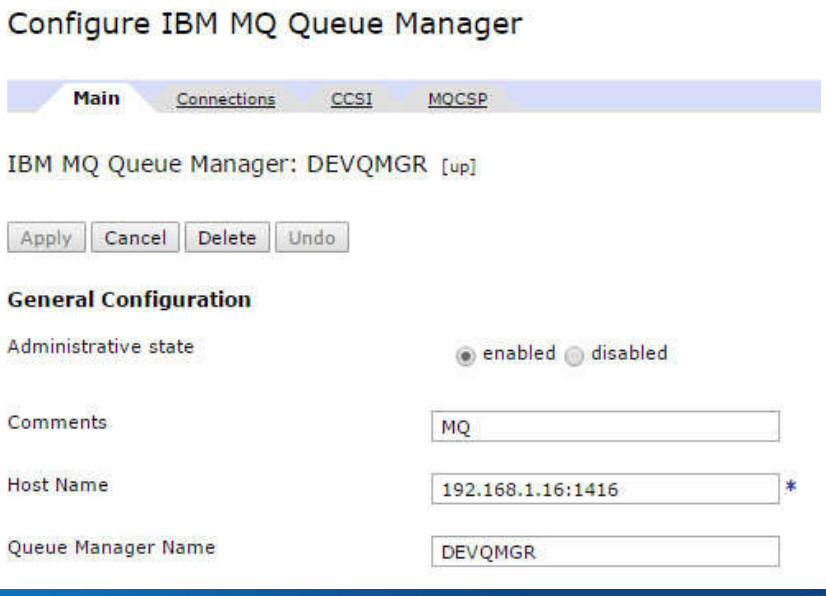
- MQ 9 & DP 7.6

Topics:

- Queue Manager Object
- MQ Front Side Handler
- MQ URL
- Message Headers
- Error Handling
- Transaction Management

Queue Manager Object

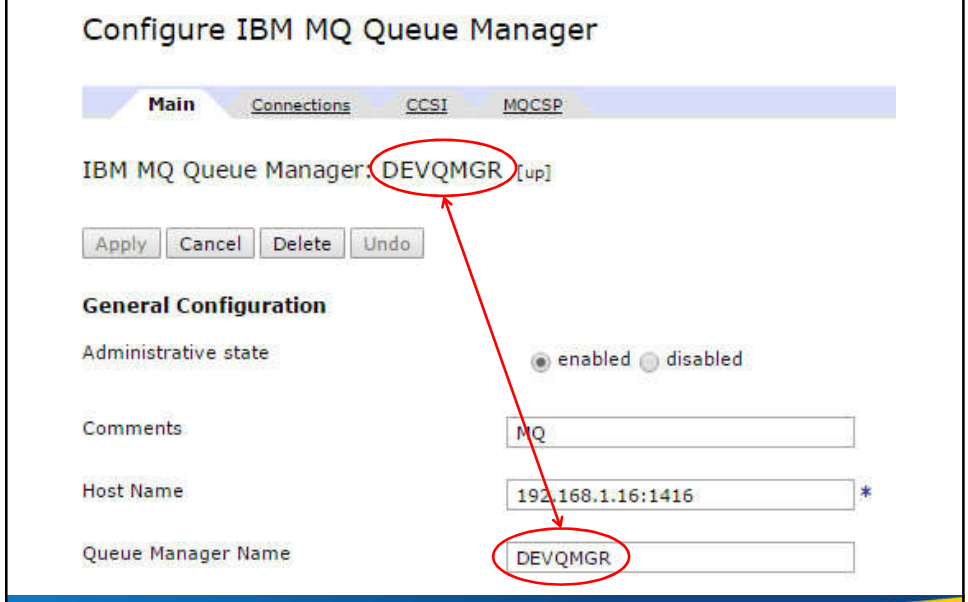
Queue Manager Object



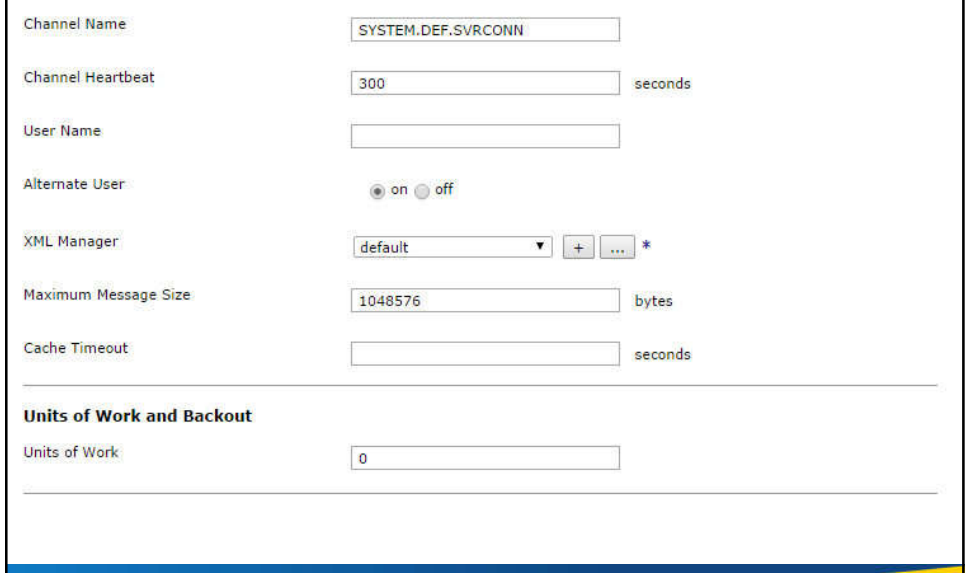
Queue Manager Object

- **Host Name (IPv4)**
  - address:port — 192.168.57.1:1414
  - address(port) — 192.168.57.1(1414)
  - address — 192.168.57.1 *default port 1414*
- **Host Name (IPv6)**
  - [address]:port — [2202::148:248]:1414
  - address(port) — 2202::148:248(1414)
  - address — 2202::148:248 *default port 1414*
- **Host Name (DNS)**
  - hostname:port — myserver:1414
  - hostname(port) — myserver(1414)
  - hostname — myserver *default port 1414*

Queue Manager Object



Queue Manager Object



Queue Manager Object

- **Channel Name**
  - SVRCONN name as defined on the Queue Manager
- **Channel Heartbeat (seconds)**
  - Approximate time between heartbeat flows on the channel
  - 0 = no heartbeat flow exchanged
  - Does not set the heartbeat on the channel
  - Negotiates heartbeat value with channel definition -- greater is used
- **Cache Timeout (seconds)**
  - How long the appliance keeps alive a dynamic connection in the connection cache
  - Must be greater than the negotiated heartbeat interval but less than the Queue Manager keep alive interval (defined on the host)

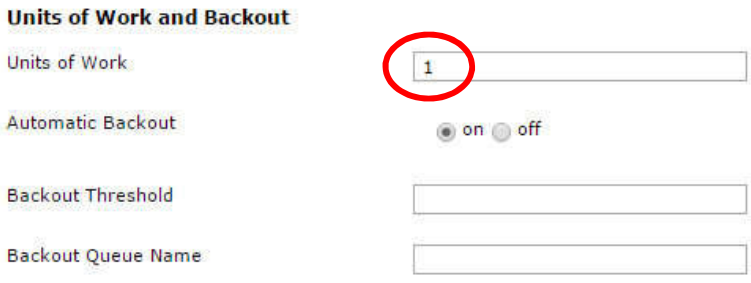
Queue Manager Object

- **User Name**
  - Supplied to Queue Manager at connection
  - Maximum 12 characters
- **Alternate User**
  - Enables or disables MQOD.AlternateUserId
  - Off = use Message Descriptor User Identifier for queue authorization
  - On (default) = use Object Descriptor Alternate User Identifier for queue authorization (need to create Object Descriptor)
- **XML Manager**
  - Recommend using a dedicated XML Manager per QM object

Queue Manager Object

- **Maximum Message Size**
  - Limit the size of the MQ message payload
- **Units of Work**
  - Controls syncpoint processing (transaction management)
  - Affects MQ Front Side Handlers only
  - Two values: 0 or 1
- **Units of Work = 0**
  - No syncpoint control
  - Front Side Handler MQGET is immediate and irrevocable
  - If error, message integrity is responsibility of DataPower
- **Units of Work = 1**
  - Enables syncpoint control
  - Front Side Handler MQGET has an implied MQGMO_SYNCPOINT
  - If error, message is rolled back via implied MQBACK

Queue Manager Object



- **Automatic Backout**
  - Enable the use of BOTHRESH and BOQUEUE
- **Backout Threshold**
  - Used if BOTHRESH not set on input queue
- **Backout Queue Name**
  - Used if BOQUEUE not set on input queue

Queue Manager – Connections Tab

Queue Manager – Connections Tab

- **Total Connection Limit:**
  - Connection pool size of the QM object
  - Default value is 250
  - Can tune the total connection limit for performance
- **Initial Connections:**
  - Maximum simultaneous initial connection requests
  - Default value is 1
  - If too high, may flood the Queue Manager
- **Local Address**
  - Local address for outbound connections
  - Specific local interface and port
  - For a range of ports, use (1414,1420) or x.x.x.x(1414,1420)

Queue Manager – Connections Tab

- **Automatic Retry**
  - On: Attempt to reconnect to the Queue Manager if connection dropped
  - Off: Disable and re-enable the Queue Manager object to reconnect
- **Retry Interval**
  - Time interval between attempts to retry failed connections
  - Recommend 10 to 15 seconds (default is 1 second)
  - Low value can spike CPU and memory usage
- **Retry Attempts**
  - Number of attempts to retry the failed connections
  - After threshold reached, the Long Retry Interval is used instead
  - Default value of 0 (zero) disables Long Retry Interval
  - Recommend non-zero value

Queue Manager – Connections Tab

- **Long Retry Interval**
  - Interval in seconds to retry connection after Retry Attempts
  - Recommend 600 seconds (default is 1800)
  - Must be greater than the Retry Interval (if less, ignored)
- **Reporting Interval**
  - How often to log retries (seconds)
  - Suppresses duplicate log entries
  - Recommend setting this the same as Retry Interval

Queue Manager – Connections Tab

- **Sharing Conversations**
  - Maximum conversations sharing single TCP/IP connection
  - Value is negotiated between SVRCONN SHARECNV setting and DataPower (lower value takes effect)
  - Value of 0 means **NO** Shared Conversations
    - Suppresses MQ V7+ features (Administrator stop-quiesce; Heartbeating; Read ahead; Client asynchronous consume)
  - Value of 1 means **NO** Shared Conversations
    - Allows MQ V7+ features (Administrator stop-quiesce; Heartbeating; Read ahead; Client asynchronous consume)
  - Value > 1 means Shared Conversations permitted
    - Allows MQ V7+ features (Administrator stop-quiesce; Heartbeating; Read ahead; Client asynchronous consume)
  - Can impact performance of clients (unless V7+ features used)

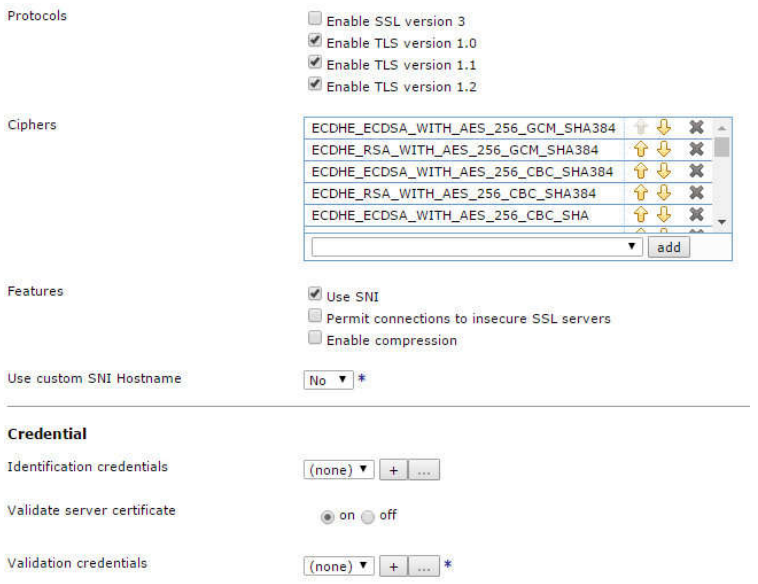
Queue Manager – Connections Tab

Queue Manager – Connections Tab

- **SSL Client Type: Client Profile**
  - Select the SSL Client Profile object to use from the pick list
  - Must be used for connection to a z/OS host
- **SSL Client Type: Proxy Profile**
  - Deprecated – recommend using Client Profile instead
  - Select the SSL Proxy Profile object to use from the pick list
- **SSL Key Repository**
  - Select the location of the key database file
- **SSL Version 3 Support**
  - Permit SSL v3 or not
- **SSL Cipher Specification**
  - Choose the Cipher Spec to use

Client Profile Object

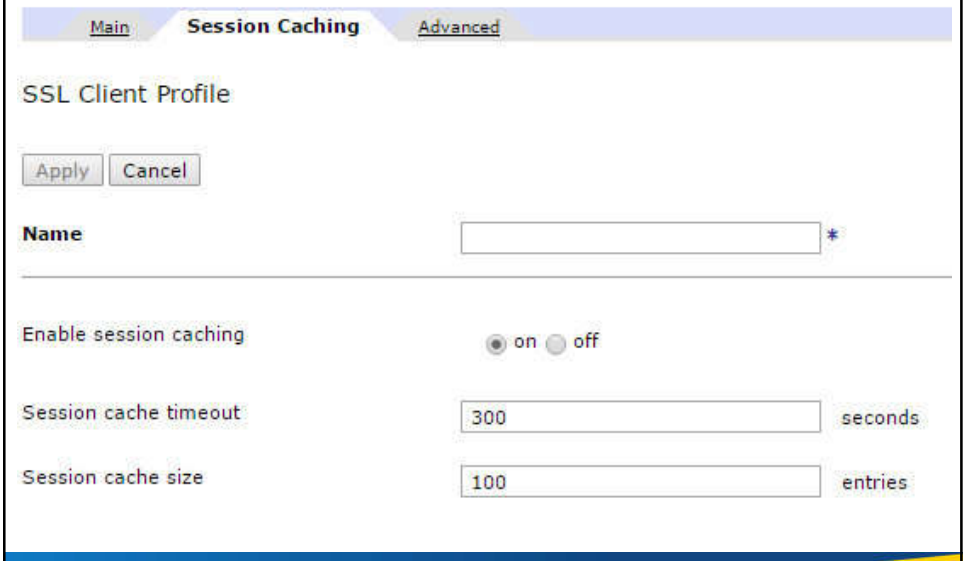
Client Profile – Main Tab



Client Profile – Main Tab

- **Protocols:**
  - Choose the protocols to be supported
- **Ciphers:**
  - Choose the ciphers to be supported
- **Use SNI:**
  - Send the Server Name Indication (SNI) TLS extension in the client hello message
- **Permit connections to insecure SSL servers:**
  - Allow connection to potentially vulnerable servers
- **Enable compression:**
  - Allow SSL compression
  - Not recommended – can allow CRIME or BREACH attacks
- **Identification credentials:**
  - If mutual authentication requested by server
- **Validate server certificate:**
  - Check the credentials presented by the server (Default: On)
- **Validation credentials:**
  - Crypto Validation Credential object used for server certificate validation

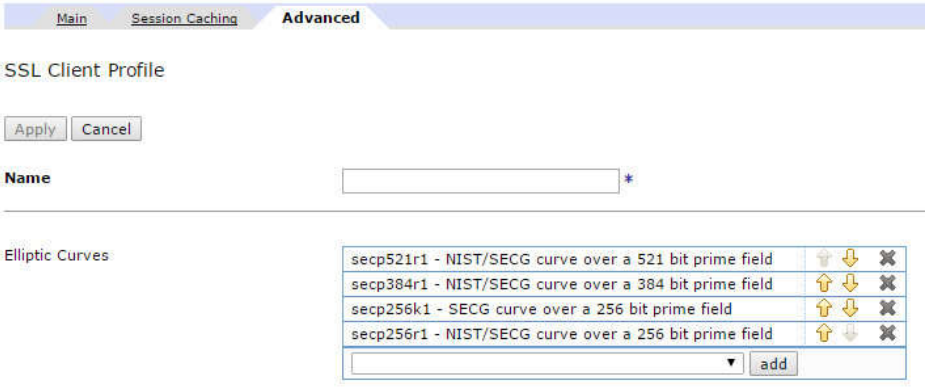
Client Profile – Session Caching Tab



Client Profile – Session Caching Tab

- **Enable session caching:**
  - Allow SSL session caching
- **Session Cache Timeout:**
  - How long before cache is flushed
  - Maximum: 86,400 seconds (24 hours)
- **Session Cache Size:**
  - How many entries to be cached
  - Maximum: 500,000

Client Profile – Advanced Tab



- **Elliptical Curves:**
  - Build a list of acceptable Elliptical Curve algorithms (RFC 4492)
  - Allows equivalent security to current cryptosystems like RSA but smaller key size
  - Favored for mobile technology

Back to the Queue Manager Object

Queue Manager – CCSI Tab

- **Coded Character Set ID**
  - Presented to the SVRCONN channel during connection
  - Same as setting MQCCSID Environment Variable
- **Convert Input**
  - On: Ask the Queue Manager to convert messages using the CCSID (default)
  - Off: No conversion

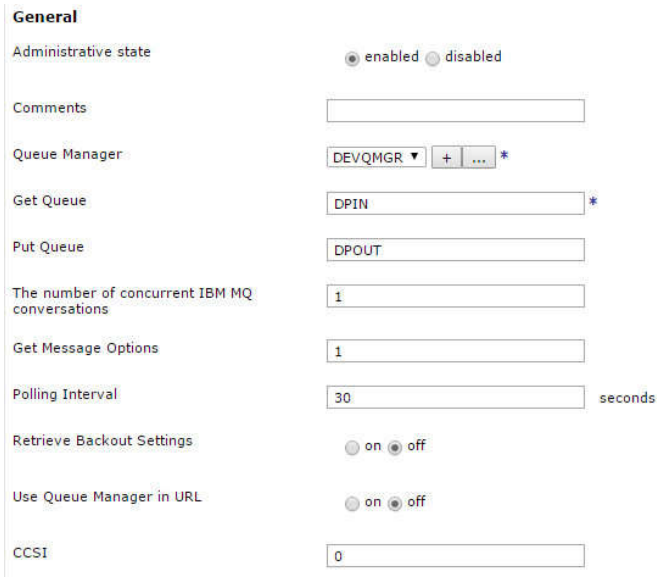
Queue Manager – MQCSP Tab

Queue Manager – MQCSP Tab

- **Defines the MQCSP Data Structure**
  - Simulates passing MQCSP using MQCONNX
- **MQCSP User ID**
  - Sent via MQCSP if present
  - If blank (and Password Alias set to “none”), no MQCSP is sent
- **MQCSP Password Alias**
  - Points to the encrypted password stored within DataPower
  - Password is sent in clear text in MQCSP after retrieval from the Alias

MQ Front Side Handler Object

MQ Front Side Handler



MQ Front Side Handler

- **Get Queue**
  - Name of queue to get messages from
  - Mandatory, unless Pub/Sub being used
- **Put Queue**
  - Optional, because:
    - May be "one-way" messaging (fire and forget)
    - May be using Reply-To Queue
    - May be dynamically allocated by Response Rule code
- **The number of concurrent MQ conversations:**
  - Number of parallel active and pending MQGETs for the Get Queue
  - Recommend value of 1 (in high throughput situations, may use up to 5)
  - Regardless of this setting, multiple FSH threads will still use multiple connections
  - If greater than 1, monitor Queue Manager for workload
  - If greater than 1, Backout Threshold must be this value plus 1
  - If using message ordering (MQGMO_LOGICAL_ORDER), set it to 1

MQ Front Side Handler

- **Get Message Options**
  - Allows the use of any MQGMO_Options parameters
  - Overrides any specific parameters set elsewhere
  - Default “1” (MQGMO_WAIT)
- **Polling Interval**
  - How long to wait on an empty queue (seconds)
  - Equivalent to Wait Interval with conventional MQ applications
  - Low value increases network traffic
  - Recommend default of 30
- **Retrieve Backout Settings**
  - Get BOTHRESH and BOQUEUE from the Get Queue
  - Issues MQINQ before ***every*** MQGET – potential performance hit
  - Only relevant if queue parameters were set by MQ administrator
  - Recommend set "off" and use Queue Manager Object settings

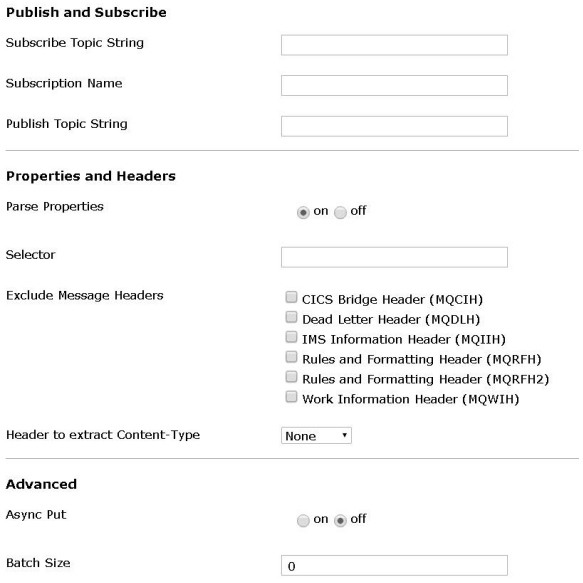
Some MQGMO Options

0	MQGMO_NONE
1	MQGMO_WAIT
2	MQGMO_SYNCPOINT
4	MQGMO_NO_SYNCPOINT
8	MQGMO_SET_SIGNAL
16	MQGMO_BROWSE_FIRST
32	MQGMO_BROWSE_NEXT
64	MQGMO_ACCEPT_TRUNCATED_MSG
128	MQGMO_MARK_SKIP_BACKOUT
256	MQGMO_MSG_UNDER_CURSOR
512	MQGMO_LOCK
1024	MQGMO_UNLOCK
2048	MQGMO_BROWSE_MSG_UNDER_CURSOR
4096	MQGMO_SYNCPOINT_IF_PERSISTENT
8192	MQGMO_FAIL_IF_QUIESCING
16384	MQGMO_CONVERT
32768	MQGMO_LOGICAL_ORDER
65536	MQGMO_COMPLETE_MSG
131072	MQGMO_ALL_MSGS_AVAILABLE
262144	MQGMO_ALL_SEGMENTS_AVAILABLE

MQ Front Side Handler

- **Use Queue Manager in URL**
  - Defines the behavior of var://service/URL-in when a QM Group is specified
  - If on, the variable returns the name of the chosen Queue Manager
  - If off, the variable returns the name of the Queue Manager Group
  - Default off
- **CCSI**
  - Sets the CCSID in the MQ Message Descriptor
  - If blank or zero, default is ISO-8859-1 (latin-1)
  - For MQCCSI_EMBEDDED enter 4294967295
  - For MQCCSI_INHERIT enter 4294967294

MQ Front Side Handler



MQ Front Side Handler

- **Subscribe Topic String**
  - Pub/Sub topic string for subscription
  - If Get Queue also defined, this is ignored
- **Subscription Name**
  - Used to establish or resume a Durable Subscription
- **Publish Topic String**
  - Pub/Sub topic string for response publication
  - If Put Queue also defined, this is ignored
- **Parse Properties**
  - Extracts MQ V7 (and above) Message Properties into Node Set
  - Minor overhead, so leave off unless needed

MQ Front Side Handler

- **Selector**
  - Allows selective retrieval of messages based on properties
  - Forces sequential search of queue so may be inefficient
- **Exclude Message Headers**
  - Strip off selected MQ header types
- **Header to extract Content-Type**
  - Can obtain Content-Type from
    - MQMD
    - RFH
    - RFH2

MQ Front Side Handler

- **Async Put**
  - Put message to queue without waiting for a response
  - Do not use when Queue Manager units-of-work is 1
  - Recommend only use where performance is an issue
- **Batch Size**
  - Number of messages to handled as a singe commit or rollback operation
  - Recommend leave this as zero – each message is a separate transaction

MQ Back-End URL

MQ Back-End URL

- **General Syntax:**
  - dpmq://mqQueueManagerObject/URI?<parameters>
- **RequestQueue=*requestQueueName***
  - Name of the backend MQ request queue
- **ReplyQueue=*replyQueueName***
  - Name of the backend MQ reply queue
- **Sync=true**
  - Issues a Commit call when a message is put on Request Queue
- **Transactional=true**
  - Begins a new transaction when getting a message from the ReplyQueue
- **GMO=*optionsValue***
  - MQGMO_Options parameter value when getting from Reply Queue
- **PMO=*optionsValue***
  - MQPMO_Options parameter value when putting to Request Queue

MQ Back-End URL

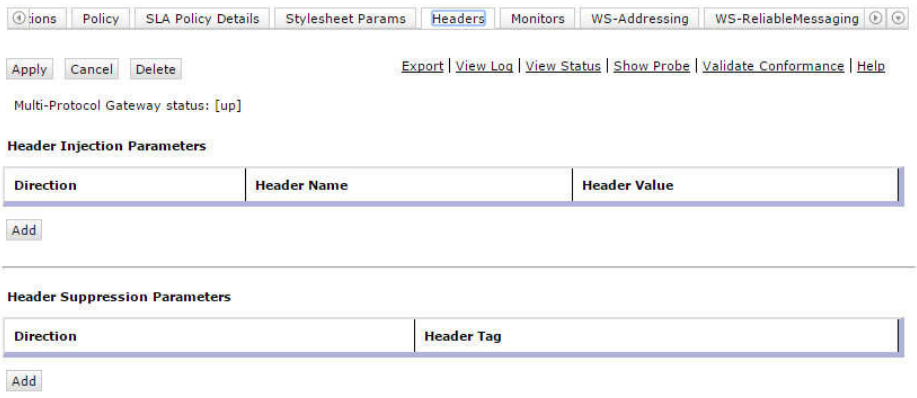
- **ParseHeaders={true|false}**
  - Specifies whether to parse and strip headers from message
- **SetReplyTo={true|false}**
  - Specifies whether to set MD ReplyToQ during Put
- **AsyncPut={true|false}**
  - Specifies whether to use Asynchronous Put
  - Only valid when using MQ V7 (and above)
- **Browse={first|next|current}**
  - Controls non-destructive retrieval of messages

MQ Back-End URL

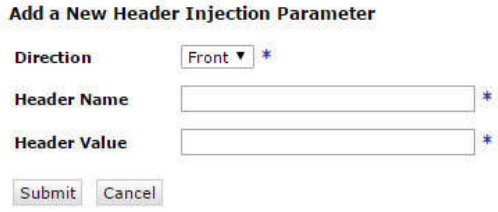
- **ContentTypeHeader=*header***
  - Which MQ header identifies the content type of the message
- **ContentTypeXPath=expression**
  - XPath expression to extract the content type of message
- **ParseProperties={on|off}**
  - Parse message properties
- **PublishTopicString=*string* and SubscribeTopicString=*string***
  - Specifies topic to use with Pub/Sub (MQ V7 and above)
- **SubscriptionName=*string***
  - Specifies name for a durable subscription (MQ V7 and above)
- **Selector=*expression***
  - SQL92 style query filtering on message properties
  - Performance hit

Multi-Protocol Gateway Parameter Settings

MPGW Headers Tab – MQ Headers



MPGW Headers Tab – MQ Headers



- **Using Header Injection (Header Tab)**
- **Example setting Format and Persistence:**
  - Direction: Front (for FSH MQPUT)
  - Direction: Back (for Backend MQPUT)
  - Header Name: MQMD
  - Header Value:
    <MQMD><Format>MQSTR</Format><Persistence>1</Persistence></MQMD>

MQ Programmatic Control

MQ Headers – Programmatic Manipulation

- **Using the Stylesheet method (page 1 of 2)**

```
<?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"
    extension-element-prefixes="dp"
    exclude-result-prefixes="dp">

  <xsl:output method="xml"/>
  <xsl:template match="/">
     <xsl:variable name="newMQMDStr">
          <MQMD>
                    <Format>MQSTR</Format><Persistence>1</Persistence>
          </MQMD>
     </xsl:variable>
     <xsl:variable name="mqmdStr">
    <dp:serialize select="$newMQMDStr" omit-xml-decl="yes"/>
     </xsl:variable>
     <xsl:message dp:priority="debug">
          <xsl:value-of select="concat('The New MQMD : ', $mqmdStr)"/>
```

MQ Headers – Programmatic Manipulation

- **Using the Stylesheet method (page 2 of 2)**

```
    </xsl:message>
    <!-- for request rule -->
    <dp:set-request-header name="'MQMD'" value="$mqmdStr"/>
    <!-- for response rule -->
    <!-- <dp:set-response-header name="'MQMD'" value="$mqmdStr"/> -->
     <!-- adding MQ header when MQ URL open call is usedfor MQPUT-->
     <!--
     <xsl:variable name="mqHeaders">
         <header name="MQMD"><xsl:value-of select="$mqmdStr"/></header>
     </xsl:variable>
     <xsl:variable name="sendmessage">
    <dp:url-open
       target="dpmq://DP4/?RequestQueue=QUEUE6"
            http-headers="$mqHeaders"
  response="responsecode-ignore">
            <xsl:copy-of select="." />
        </dp:url-open>
      </xsl:variable>
  -->
  </xsl:template>
</xsl:stylesheet>
```


MQ Headers – Programmatic Manipulation

- **Context variable method to inject the MQMD header**
  - For the following code to work:
  - Set Transform Action's OUTPUT context to "EVENTS"
  - Set Result Action's INPUT context to "EVENTS"

```
<xsl:variable name="MQMDStr">
    <MQMD>
       <Expiry>8000</Expiry>
       <Priority>0</Priority>
       <Format>MQSTR</Format>
    </MQMD>
</xsl:variable>
<xsl:variable name="MQMDStr2">
    <dp:serialize select="$MQMDStr" omit-xml-decl="yes"/>
</xsl:variable>
<dp:set-variable name="'var://context/EVENTS/_extension/header/MQMD'"
 value="$MQMDStr2"/>
```

MQ Headers – Programmatic Manipulation

- **JMS Headers as Message Properties**
  - Must set FSH “Parse Properties” to be “on”
  - Must set “Exclude RFH2” to be “off”
  - Message Properties appear as “MQMP” header

Protocol Headers:

MQMD	<MQMD><StrucId>MD </StrucId><Version>1</Version><Report>0</Report><MsgType>8</MsgType> <Expiry>-1</Expiry><Feedback>0</Feedback><Encoding>546</Encoding> <CodedCharSetId>819</CodedCharSetId><Format>MQSTR </Format><Priority>4</Priority> <Persistence>1</Persistence><MsgId>414d5120444556514d4752202020209637e75722087c02</MsgId> <CorrelId>000000000000000000000000000000000000000000000000</CorrelId><BackoutCount>0</BackoutCount> <ReplyToQ> </ReplyToQ><ReplyToQMgr>DEVQMGR </ReplyToQMgr><UserIdentifier>virtuser </UserIdentifier> <AccountingToken>16010515000000dd7a5d8bea372ba8a0058899e80300000000000000000000000b</AccountingToken> <ApplIdentityData> </ApplIdentityData><PutApplType>11</PutApplType> <PutApplName>C:\Windows\System32\java.exe</PutApplName><PutDate>20160926</PutDate> <PutTime>20093683</PutTime><ApplOriginData> </ApplOriginData></MQMD> [View Parsed]
MQMP	<MQMP><Property name="mcd.Msd" type="string">jms_text</Property><Property name="jms.Dst" type="string">queue://DEVQMGR/DPIN</Property><Property name="jms.Tms" type="string">1474920576830</Property><Property name="jms.Dlv" type="string">2</Property></MQMP>

MQ Headers – Programmatic Manipulation

- **JMS Headers as RFH2**
  - Must set FSH “Parse Properties” to be “off”
  - Must set “Exclude RFH2” to be “on”
  - Message Properties appear as “MQRFH2” headers

X-MQRFH2-Data-0	<mcd><Msd>jms_text</Msd></mcd>
X-MQRFH2-Data-1	<jms><Dst>queue://DEVQMGR/DPIN</Dst><Tms>1474922529242</Tms><Dlv>2</Dlv></jms>
MQRFH2	<MQRFH2><StrucId>RFH </StrucId><Version>2</Version><Encoding>546</Encoding> <CodedCharSetId>819</CodedCharSetId><Format>MQSTR </Format><Flags>0</Flags> <NameValueCCSID>1208</NameValueCCSID><NameValueData><NameValue><mcd><Msd>jms_text</Msd> </mcd> </NameValue><NameValue><jms><Dst>queue://DEVQMGR/DPIN</Dst><Tms>1474922529242</Tms> <Dlv>2</Dlv></jms> </NameValue></NameValueData></MQRFH2>

MQ Error Handling

- **MQ error handling example:**

```
   <xsl:template match="/">
       <xsl:variable name="mqrc" select="dp:response-header('x-dp-response-code')"/>
       <xsl:variable name="ecode" select="dp:variable('var://service/error-code')"/>
       <xsl:variable name="errMsg" select="concat('** The Response Code ** : ', $mqrc, ' and ** Error
Code ** :', $ecode )"/>

       <xsl:choose>
           <xsl:when test="(starts-with($mqrc, '2') and (string-length(normalize-space($mqrc))= 4)) or
($ecode != '0x00000000')">
               <xsl:message dp:priority="debug">
                   <xsl:value-of select="$errMsg"/>
               </xsl:message>
               <dp:set-variable name="'var://context/ERROR/err-msg'" value="$errMsg"/>
               <dp:reject override="true"><xsl:value-of select="$errMsg"/></dp:reject>
           </xsl:when>
           <xsl:otherwise>
               <xsl:message dp:priority="debug">
                   <xsl:value-of select="$errMsg"/>
               </xsl:message>
               <dp:accept/>
           </xsl:otherwise>
       </xsl:choose>
 </xsl:template>
```


MQ Conversational Processing

- **Backend application must copy MsgId to CorrelId**
  - DataPower Back-End retrieves reply using CorrelId
- **MQPUT1 Simulation**
  - Create MQ Object Descriptor header with Queue Manager name in it
  - Request Rule issues MQOPEN/MQPUT/MQCLOSE to back end Queue Manager
- **ReplyToQ Usage**
  - If set, Response Rule sends message there
- **ReplyToQmgr Usage**
  - Can be set to send to a different Queue Manager
  - If destination is a Cluster, no need to supply ReplyToQmgr

MQ Conversational Processing

- **XSL code snippet to set ReplyToQ and ReplyToQmgr in a Request Rule:**

```
<xsl:variable name="rule-type" select="dp:variable('var://service/transaction-rule-type')"/>
<xsl:choose>
   <!-- Request Rule only -->
   <xsl:when test="$rule-type = 'request'">
       <xsl:variable name="entries" select="dp:request-header('MQMD')"/>
       <xsl:variable name="header"  select="dp:parse($entries)"/>
       <!-- save ReplyToQ and ReplyToQMgr values -->
       <dp:set-variable name="'var://context/MYMQMD/ReplyToQ'"    value=""$header//ReplyToQ"/>
       <dp:set-variable name="'var://context/MYMQMD/ReplyToQMgr'" value=""$header//ReplyToQMgr"/>
       <xsl:message dp:priority="debug">
          <xsl:value-of select="concat ('Request MQMD : ', dp:request-header('MQMD'))"/>
       </xsl:message>
   </xsl:when>
```


MQ Conversational Processing

- **XSL code snippet to set ReplyToQ and ReplyToQmgr in a Response Rule:**

```
    <!-- Response rule only -->
    <xsl:when test="$rule-type = 'response'">
        <xsl:variable name="custMQODStr">
            <MQOD>
               <Version>2</Version>
               <ObjectName>
                   <xsl:value-of select="dp:variable('var://context/MYMQMD/ReplyToQ')"/>
               </ObjectName>
               <ObjectQMgrName>
                   <xsl:value-of select="dp:variable('var://context/MYMQMD/ReplyToQMgr')"/>
               </ObjectQMgrName>
            </MQOD>
        </xsl:variable>
        <xsl:variable name="mqodStr">
            <dp:serialize select="$custMQODStr" omit-xml-decl="yes"/>
        </xsl:variable>
        <xsl:message dp:priority="debug">
            <xsl:value-of select="concat('Response MQOD : ', $mqodStr)"/>
        </xsl:message>
        <dp:set-response-header name="'MQOD'" value="$mqodStr"/>
    </xsl:when>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

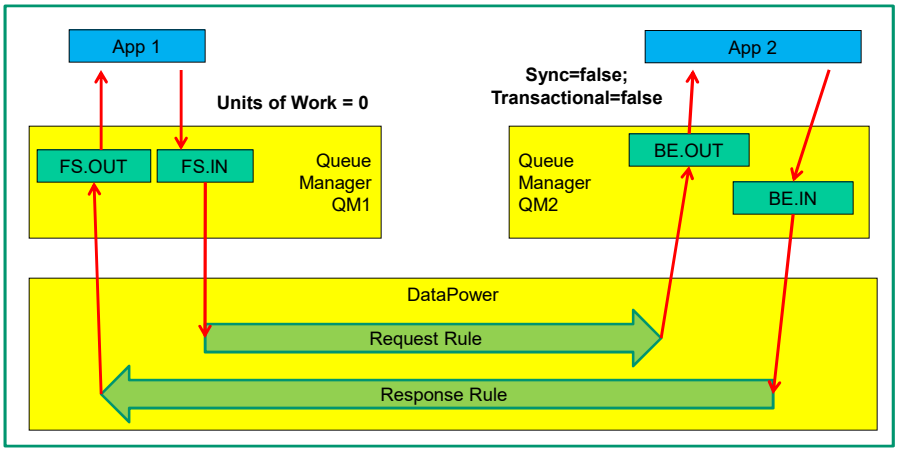
Transactional Processing

MQ Transactional Processing

- **DataPower is a standard MQ Client**
  - It does **NOT** offer Extended Transactional Client functionality
  - **NO** XA two-phase commit
- **DataPower is considered an application by MQ**
  - Therefore, no inherent message integrity
- **If the same Queue Manager at front and back:**
  - True message integrity
  - Once and once-only delivery
- **If different Queue Managers at front and back**
  - No possibility of two-phase commit
  - Message integrity assured if DataPower configured properly
  - Possibility of messages sent more than once

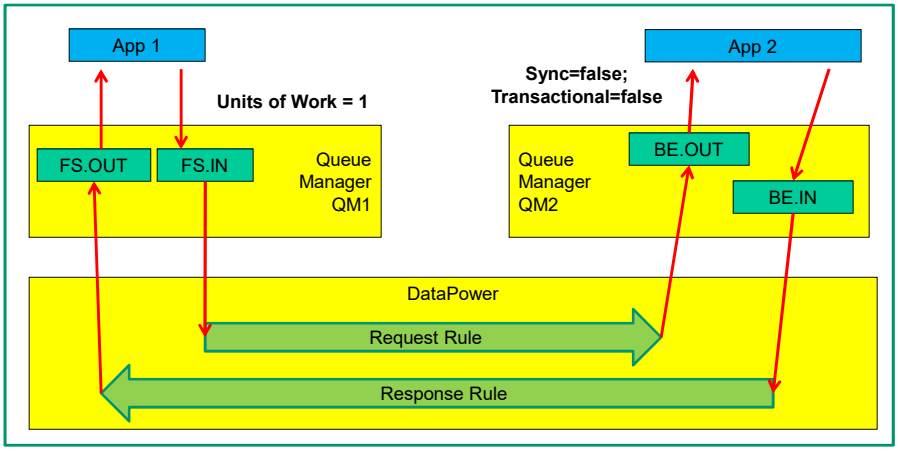
Scenario: Two Different Queue Managers

No Transactional Control



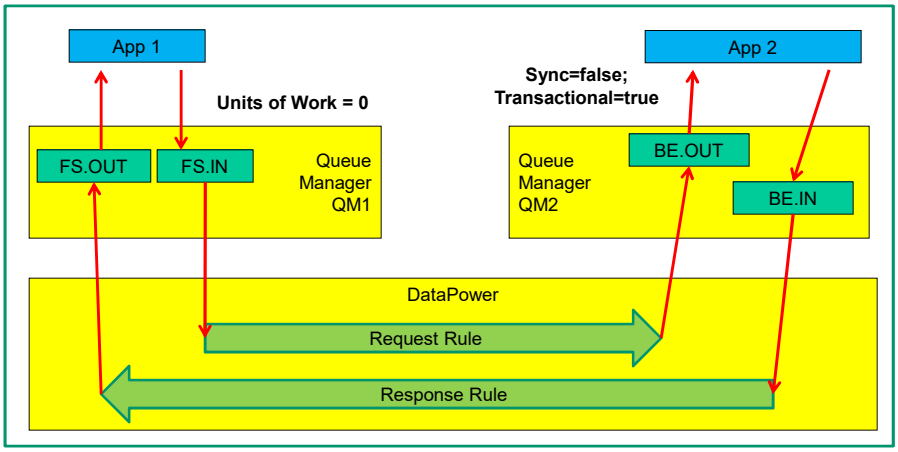
Scenario: Two Different Queue Managers

Front Side Transactional Control only



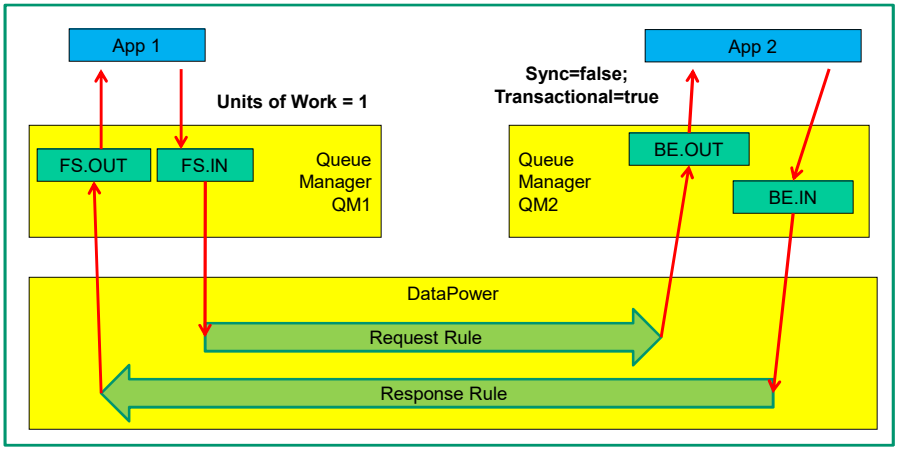
Scenario: Two Different Queue Managers

Back End Transactional Control only



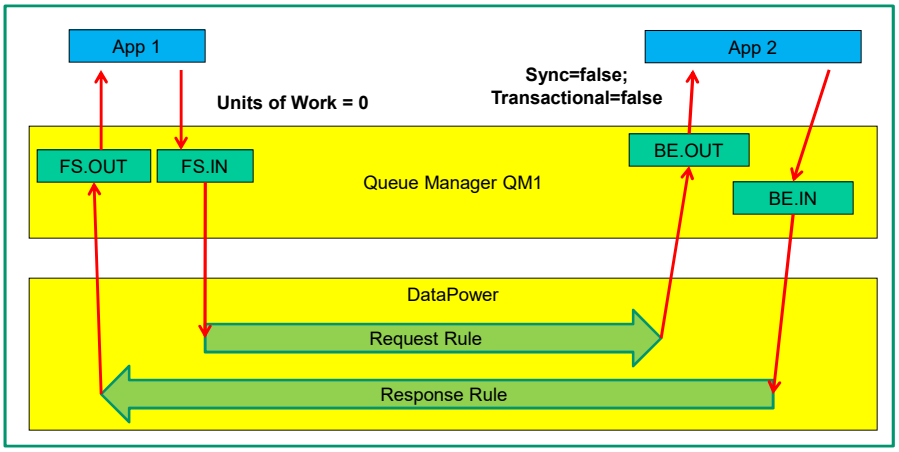
Scenario: Two Different Queue Managers

Front Side and Back End Transactional Control



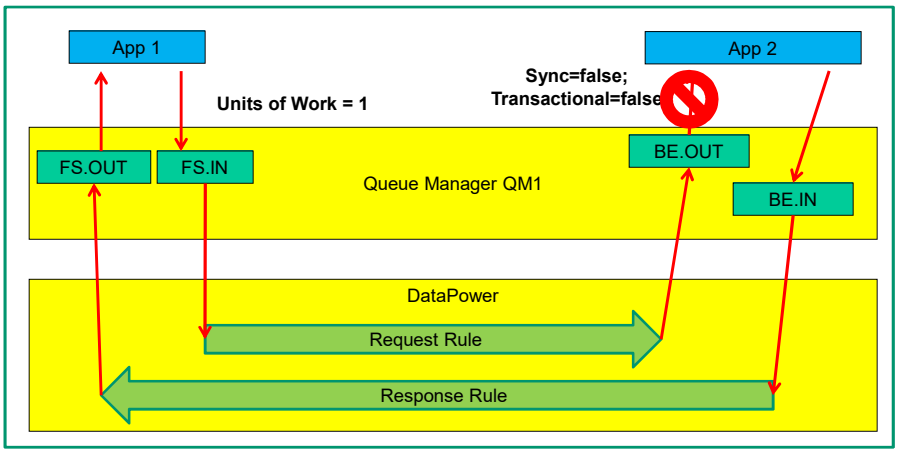
Scenario: One Queue Manager (Front & Back)

No Transactional Control



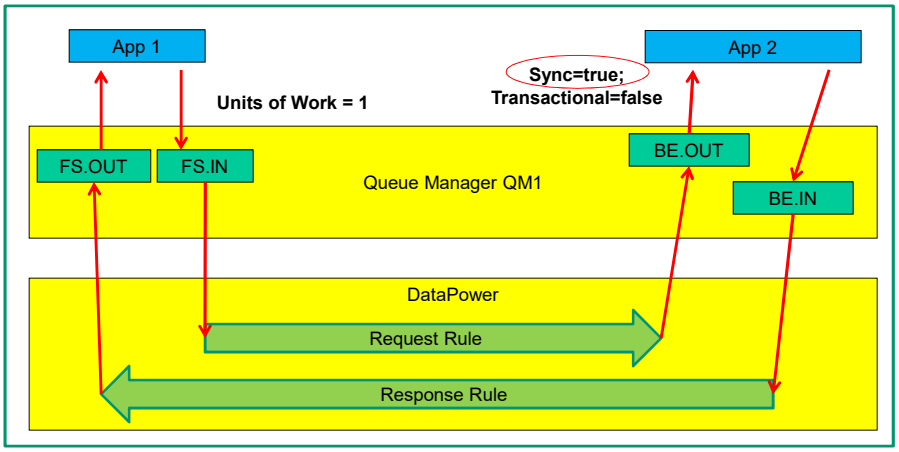
Scenario: One Queue Manager (Front & Back)

Front Side Transactional Control only



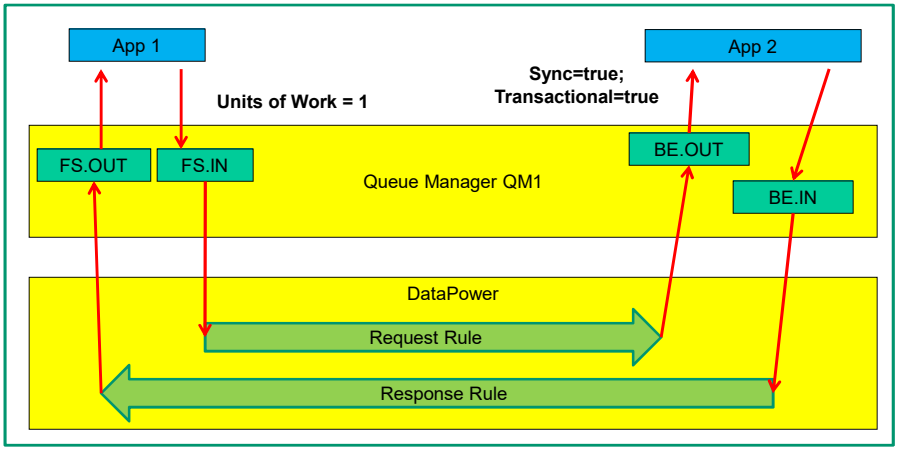
Scenario: One Queue Manager (Front & Back)

Front Side Transactional Control only

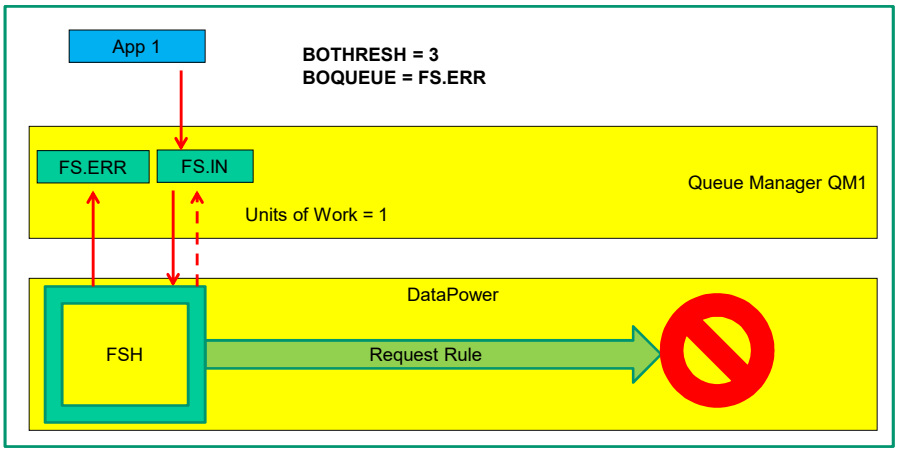


Scenario: One Queue Manager (Front & Back)

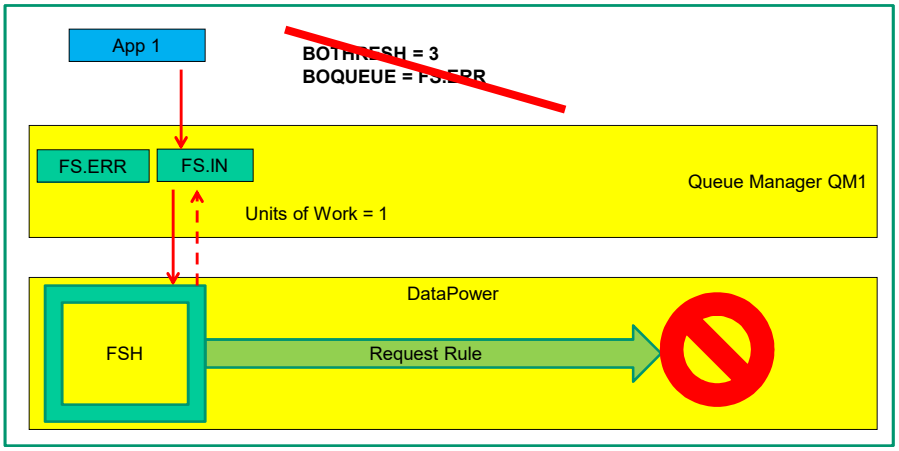
Front Side *and* Back End Transactional Control



Poison Message



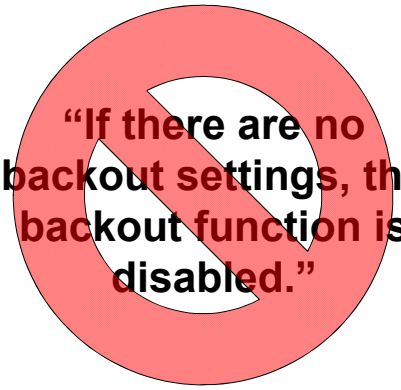
Poison Message



Poison Message

DataPower Knowledge Center Documentation

https://www.ibm.com/support/knowledgecenter/en/SS9H2Y_7.6.0/com.ibm.dp.doc/retrieve-backout-setting_mqfsh.html



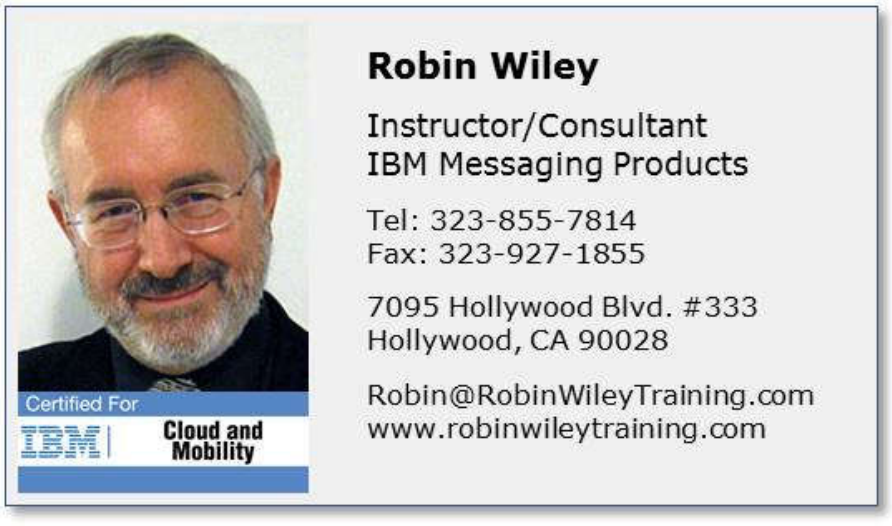
Questions & Answers

End of Session

Thank You!

Contact: **Robin@RobinWileyTraining.com**

Handouts: **RobinWileyTraining.com/MQTC2017**



MQ Technical Conference v2.0.1.7