



IBM Software Group

# Enabling WebSphere MQ Traffic with WebSphere DataPower - Use Case Scenarios

Chin Sahoo ([chintam3@us.ibm.com](mailto:chintam3@us.ibm.com))

DataPower SOA Appliances Support

May 18, 2010



WebSphere® Support Technical Exchange



# Agenda

- MQ Traffic Patterns in DataPower
- MQ Message Processing Mode
- Use Case Scenarios
- MQ Routing in DataPower
- Trouble Shooting
- Useful Links
- Questions and Answers

# MQ Traffic Patterns in DataPower

- DataPower's Multi-Protocol Gateway Service (MPGW) is used to handle these traffic patterns
- MQ <=> MQ (MQ in front & MQ in back)
  - ▶ MQ with Units-of-Work (UoW)
  - ▶ MQ with SSL Channel
- MQ <=> HTTP(S) (MQ in front & HTTP(s) in back)
  - ▶ HTTP(S) => MQ
  - ▶ MQ => HTTP(S)
- MQ <=> JMS (MQ in front & JMS in back)
- MQ <=> TIBCO EMS (MQ in front & TIBCO in back)
- MQ <=> FTP(S) (MQ in front & FTP(S) in back)

# MQ Message Processing Mode

- Datagram traffic using MPGW
- Datagram with custom error handling
- Datagram with distribution list
- Datagram with transactionality
- Request/Reply traffic using MPGW
- Request/Reply with Dynamic Routing
- Request/Reply with Model Queue
- Request/Reply with transactionality

# Datagram Traffic using MPGW Service

- ▶ Request Rule only, no Response Rule
- ▶ May have Error Rule
- ▶ Process Backend Errors is “off” under the advanced tab of the Multi-Protocol Gateway (MPGW) Service
- ▶ Request Type as “XML”, “SOAP”, “non-XML” or “pass-thru”
- ▶ Response Type as “pass-thru”
- ▶ Request MQMD is not altered
- ▶ Backside MQ URL only specifies the request queue

# Datagram with custom error processing

- ▶ Request Rule and Response Rule
- ▶ Must capture response code in response rule using “dp:response-header('x-dp-response-code’)” and use “dp:reject” to invoke error rule if “response code” is “2xxx”
- ▶ “Process Backend Errors” is “on” under the advanced tab of the MPGW
- ▶ Must use “var://service/error-ignore = 1” in error rule to handle ROLLBACK when units-of-work is enabled



# Datagram with distribution list

- Typical scenario when the same message is distributed to multiple queues
- MQ Distribution Lists is an optimal way of fanning out messages
  - Inject MQOD request header with multiple destinations

```
<MQOD>
  <MQOR><ObjectName>Queue1</ObjectName></MQOR>
  <MQOR><ObjectName>Queue2</ObjectName></MQOR>
  <MQOR><ObjectName>Queue3</ObjectName></MQOR>
  <MQOR><ObjectName>Queue4</ObjectName></MQOR>
</MQOD>
```
  - And four separate calls become a single one. Performance will increase significantly
  - Inject MQOD headers for the backend MQ qmgr using DataPower's extension function `<dp:set-request-header name="MQOD" value="$mqodStr"/>` in custom stylesheet or MPGW's Header injection Tab

# Datagram with Transactionality (UoW)

On the front side of the MPGW service:

- Transactions are enabled when units-of-work (UoW) is set to “1” on MQ QM object
- Both GET and PUT operations use the same connection
- MQCOMMIT performed only after successful PUT
- Front and backside use same Queue Manager (qmgr) configured in the MQ QM object



# Datagram with UoW Continued..

On the backend MQ URL for the MPGW service:

- Transactions are enabled when MQ URL contains “Transactional=true” parameter for Datagram traffic
- Transactions are enabled when MQ URL contains “Sync=true;Transactional=true” parameters for Request/Reply traffic
- MQCOMMIT is performed immediately after PUT when MQ URL contains “Sync=true” parameter
- Use Automatic Backout as “on” with Backout Threshold and Backout Queue name in QM Object
- Set service variable “var://service/error-ignore” to “1” to handle ROLLBACK in the error rule

# Request/Reply Traffic using MPGW

- ▶ Request, Response and Error Rules
- ▶ Must capture response code in response rule using “dp:response-header('x-dp-response-code')” and use “dp:reject” to invoke error rule if “response code” is “2xxx”
- ▶ “Process Backend Errors” should be “on” under the advanced tab of the MPGW
- ▶ Must use “var://service/error-ignore = 1” in error rule to handle ROLLBACK if UoW is enabled

# Request/Reply Traffic with Dynamic Routing

Scenario 1: If MQMD.ReplyToQ value exists and MQMD.ReplyToQMgr's value is same as the qmgr name configured in the MQ QM object

- Set DataPower's internal header "ReplyToQ" to an empty string using extension function `<dp:set-response-header name="ReplyToQ" value="" />` using stylesheet in both request and response rules
- Save MQMD.ReplyToQ to a context variable in request rule
- Save MQMD.ReplyToQMgr to a context variable in request rule
- Inject MQOD headers with these values in the response rule for the front side client
- Make sure the local qmgr is configured to handle message routing to remote qmgr based on MQOD

## Request/Reply Traffic with Dynamic Routing Continued

### **Scenario 2: If MQMD.ReplyToQ value exists and MQMD.ReplyToQMgr's value is different than the qmgr name configured in the MQ QM object**

- Set DataPower's internal header "ReplyToQ" to an empty string using extension function `<dp:set-response-header name="ReplyToQ" value=""/>` using stylesheet in both request and response rules
- Set DataPower's internal header "ReplyToQM" to an empty string using extension function `<dp:set-response-header name="ReplyToQM" value=""/>` using stylesheet in both request and response rules
- Save MQMD.ReplyToQ to a context variable in request rule
- Save MQMD.ReplyToQMgr to a context variable in request rule
- Inject MQOD headers with these values in the response rule for the front side client
- Make sure the local qmgr is configured to handle message routing to remote qmgr based on MQOD

## Request/Reply Traffic with Model Queue

- A model queue defines a set of queue attributes that are used as a template for creating a dynamic queue.
- Dynamic queues are created by the queue manager when an application issues an MQOPEN request specifying a queue name that is the name of a model queue.
- The dynamic queue that is created in this way is a local queue whose attributes are taken from the model queue definition.



## Request/Reply Traffic with Model Queue Continued

- The Backend MQ URL contains “Model=true” parameter with model queue name as part of the “ReplyQueue” tag
- Example of MQ URL with “Model=true”
  - `dpmq://MQ-AIX/?RequestQueue=Q1;ReplyQueue=MQ1;Model=true`
  - Note: MQ1 is the name of the model queue defined in qmgr

# Message delivery modes

## DataPower supports 1-phase COMMIT

- The same MQ qmgr must be used in MQ front side handlers and MQ URL openers
- All processing actions must be synchronous
- The same connection is shared across all MQ operations within a transaction
- Guaranteed “once-and-only-once” message delivery

In all other cases it is “at-least-once” message delivery, i.e. no message will ever be lost

# Message delivery modes Continued

Scenario 1: When same MQ qmgr is used in both front side handler and back side MQ URL opener

- Message from Input Queue (GETQ) will be resent if input connection fails. No duplicates in Output Queue (PUTQ)

Scenario 2: When two separate MQ qmgrs are used in front side handler and back side MQ URL opener

- Message from Input Queue (GETQ) will be resent if input connection fails. Duplicate message may appear in Output Queue (PUTQ). No message loss.





# Use Case Scenarios

- Use Case 1:
- Traffic pattern is MQ-to-MQRFH2
- MPGW service with MQ as the front side handler, creates MQRFH2 message and delivers the message as datagram to the backend MQ qmgr
- Units-of-work (UoW)
- Same qmgr for both front side and back side



# Use Case-1 Configuration – QM Object

## Configure MQ Queue Manager

Configuration successfully saved.

Main **Advanced**

MQ Queue Manager: LINUX-MQ [up]

Apply Cancel Delete Undo

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Host Name	<input type="text" value="MQ-LNX(1520)"/> *
Queue Manager Name	<input type="text" value="DP1"/>
Channel Name	<input type="text" value="SYSTEM.DEF.SVRCONN"/>
Channel Heartbeat	<input type="text" value="300"/> seconds
User Name	<input type="text" value="mqm"/>
Maximum Message Size	<input type="text" value="1048576"/> bytes
Cache Timeout	<input type="text" value="30"/> seconds
Units of Work	<input type="text" value="1"/>
Automatic Backout	<input checked="" type="radio"/> on <input type="radio"/> off
Backout Threshold	<input type="text" value="2"/>
Backout Queue Name	<input type="text" value="ERROR.Q"/>
Total Connection Limit	<input type="text" value="25"/>
Initial Connections	<input type="text" value="1"/>
SSL Key Repository	<input type="text" value="cert:///"/> <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/>

SSL Cipher Specification	<input type="text" value="None"/>
SSL Proxy Profile	<input type="text" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>
Convert Input	<input checked="" type="radio"/> on <input type="radio"/> off
Automatic Retry	<input checked="" type="radio"/> on <input type="radio"/> off
Retry Interval	<input type="text" value="10"/> seconds
Retry Attempts	<input type="text" value="6"/> attempts
Long Retry Interval	<input type="text" value="1200"/> seconds
Reporting Interval	<input type="text" value="20"/> seconds
Alternate User	<input checked="" type="radio"/> on <input type="radio"/> off
Local Address	<input type="text"/>
XML Manager	<input type="text" value="default"/> <input type="button" value="+"/> <input type="button" value="..."/> *

# Use Case-1: MQ Front Side Handler

MQ Front Side Handler: mq-rfh2-fsh [up]

Apply Cancel Undo

**General**

Admin State  enabled  disabled

Comments

Queue Manager    \*

Get Queue  \*

Put Queue

The number of concurrent MQ connections:

Get Message Options:

Polling Interval:  seconds

Retrieve Backout Settings:  on  off

Use Queue Manager in URL:  on  off

CCSI:

---

**Publish and Subscribe**

Subscribe Topic String:

Subscription Name:

Publish Topic String:

---

**Properties and Headers**

Parse Properties:  on  off

Selector:

Exclude Message Headers:

- CICS Bridge Header (MQCIH)
- Dead Letter Header (MQDLH)
- IMS Information Header (MQIIH)
- Rules and Formatting Header (MQRFH)
- Rules and Formatting Header (MQRFH2)
- Work Information Header (MQWIH)

Header to extract Content-Type:

---

**Advanced**

Async Put:  on  off

# Use case-1: Example MQMD.Format

```
<xsl:variable name="MQMDStr">
  <MQMD>
    <Format>MQHRF2</Format>
  </MQMD>
</xsl:variable>
<xsl:variable name="MQMDStr2">
  <dp:serialize select="$MQMDStr" omit-xml-decl="yes"/>
</xsl:variable>
<dp:set-request-header name=""MQMD""
  value="$MQMDStr2"/>
```

# Use Case-1: Example MQRFH2 Header

```

<xsl:variable name="RFH2">
  <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:///ALERTSX.INPUT.QUEUE</Dst>
        <Rto>queue:///ALERTSX.REPLY.QUEUE</Rto>
        <Tms>1197997507781</Tms>
        <Pri>0</Pri>
        <Dlv>2</Dlv>
      </jms>
    </NameValue>
    <NameValue>
      <usr>
        <FinalDestinationNode>YES</FinalDestinationNode>
        <DestinationNodes>PERMS</DestinationNodes>
        <version>VERSION_1</version>
        <ReplyTo>TRUE</ReplyTo>
      </usr>
    </NameValue>
    </NameValueData>
  </MQRFH2>
</xsl:variable>
<xsl:variable name="rfh2Str">
  <dp:serialize select="$RFH2" omit-xml-decl="yes"/>
</xsl:variable>
<xsl:message dp:priority="info">
  <xsl:copy-of select="concat('New RFH2 header : ', $rfh2Str)"/>
</xsl:message>
<dp:set-request-header name="'MQRFH2'" value="$rfh2Str"/>

```

## Use Case-2: MQRFH2-to-MQ


- MPGW service to process message
- Front side handler receives MQRFH2 message
- DataPower removes the MQRFH2 headers from the front side
- Suppress MQRFH2 header for the backend
- Injects MQMD.Format header for the backend
- Message is created for the backend MQ
- UoW in the MQ QM Object
- Same qmgr for both front and back sides
- SSL Channel for the QM Object

## Use Case-2: Excludes headers from MQ FSH

Exclude Message Headers

- CICS Bridge Header (MQCIH)
- Dead Letter Header (MQDLH)
- IMS Information Header (MQIIH)
- Rules and Formatting Header (MQRFH)
- Rules and Formatting Header (MQRFH2)
- Work Information Header (MQWIH)

## Use Case-2: Example Header Injection and Suppression

 **Configure Multi-Protocol Gateway**

General | **Advanced** | Stylesheet Params | **Headers** | Monitors | WS-Addressing | WS-ReliableMessaging

Apply | Cancel | Delete | [Export](#) | [View Log](#) | [View Status](#) | [Show Probe](#) | [Validate Conformance](#) | [Help](#)

Multi-Protocol Gateway status: [up]

**Header Injection Parameters**

Direction	Header Name	Header Value
Back	MQMD	<MQMD> <Format>MQSTR</Format> </MQMD>

Add

---

**Header Suppression Parameters**

Direction	Header Tag
Back	MQRFH2

Add



## Use Case 2: Configuring QM Object with SSL

- Use of SSL Proxy Profile
  - ▶ Forward Crypto Profile instead of SSL Key Repository
  - ▶ Consistent approach with HTTPS configuration
  - ▶ No additional tooling to create SSL Key Repository
  - ▶ Ciphers MUST match MQ Server Queue Manager
  - ▶ Careful which Cipher is used if you have a choice



## Use Case 2: Configuring QM with SSL Continued..



### Configure Crypto Profile

#### Main

Crypto Profile : MQ-Crypto-Profile [up]

Apply

Cancel

Undo

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Identification Credentials	MQ-Crypto-IDCred [v] [ + ] [ ... ]
Validation Credentials	MQ-Crypto-ValCred [v] [ + ] [ ... ]
Ciphers	RC4
Options	<input checked="" type="checkbox"/> Enable default settings <input checked="" type="checkbox"/> Disable SSL version 2 <input type="checkbox"/> Disable SSL version 3 <input checked="" type="checkbox"/> Disable TLS version 1 *
Send Client CA List	<input type="radio"/> on <input checked="" type="radio"/> off

# Use Case 2: SSL Identity and Validation Credentials

## Configure Crypto Identification Credentials

**Main**

Crypto Identification Credentials : MQ-Crypto-IDCred [up]

Apply Cancel Delete Undo [Export](#) | v

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Crypto Key	MQ-Crypto-Key + ... *
Certificate	MQ-Crypto-Cert + ... *
Intermediate CA Certificate	(empty) Add + ...

## Configure Crypto Validation Credentials

**Main**

Crypto Validation Credentials : MQ-Crypto-ValCred [up]

Apply Cancel Delete Undo

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Certificates	VerisignClass3SecureServerCA Add + ...
Certificate Validation Mode	Match exact certificate or immediate issuer
Use CRL	<input checked="" type="radio"/> on <input type="radio"/> off
Require CRL	<input type="radio"/> on <input checked="" type="radio"/> off
CRL Distribution Points Handling	Ignore

# MQ Routing: Complete MQMD Header Example

```
<MQMD>
  <StrucId>MD</StrucId>
  <Version>1</Version>
  <Report>0</Report>
  <MsgType>1</MsgType>
  <Expiry>-1</Expiry>
  <Feedback>0</Feedback>
  <Encoding>546</Encoding>
  <CodedCharSetId>819</CodedCharSetId>
  <Format>MQSTR</Format>
  <Priority>0</Priority>
  <Persistence>0</Persistence>
  <MsgId>414d512045494254485330312020202049cd019922fb7f07</MsgId>
  <CorrelId>0000000000000000000000000000000000000000000000000000000000000000</CorrelId>
  <BackoutCount>0</BackoutCount>
  <ReplyToQ>CLIENT.REPLY.QUEUE</ReplyToQ>
  <ReplyToQMgr>CLIENTQM</ReplyToQMgr>
  <UserIdentifier>userid</UserIdentifier>
  <AccountingToken>0000000000000000000000000000000000000000000000000000000000000000</AccountingToken>
  <ApplIdentityData>
  </ApplIdentityData>
  <PutApplType>6</PutApplType>
  <PutApplName>WebSphere Datapower MQClient</PutApplName>
  <PutDate>20090403</PutDate>
  <PutTime>21595756</PutTime>
  <ApplOriginData>
  </ApplOriginData>
</MQMD>
```

## MQ Routing: MQOD header Injection Example

- ▶ MQOD headers used for Distributed MQ Queue Manager (qmgr)

```
<xsl:variable name="newMQODStr">
  <MQOD>
    <Version>2</Version>
    <ObjectName>CLIENT.REPLY.QUEUE</ObjectName>
    <ObjectQMgrName>CLIENTQM</ObjectQMgrName >
  </MQOD>
</xsl:variable>

<xsl:variable name="mqodStr">
  <dp:serialize select="$newMQODStr"
    omit-xml-decl="yes"/>
</xsl:variable>

<dp:set-request-header name="'MQOD'"
  value="$mqodStr"/>
```

## MQ Routing: MQOD header Injection Example Continued

- ▶ MQOD headers used for Cluster MQ Queue Manager (qmgr)

```
<xsl:variable name="clusMQODStr">
  <MQOD>
    <Version>2</Version>
    <ObjectName>CLIENT.REPLY.QUEUE</ObjectName>
  </MQOD>
</xsl:variable>
<xsl:variable name="clus-mqodStr2">
  <dp:serialize select="$clusMQODStr"
    omit-xml-decl="yes"/>
</xsl:variable>
<dp:set-request-header name="'MQOD'"
  value="$clus-mqodStr2"/>
```

**Note: The qmgr is not included in MQOD for cluster MQ environment**

# MQ Routing: Use of Static and Dynamic URL

- Static MQ URL opener
  - Using MQ Queue Manager configuration object
  - URL uses dpmq:// prefix, like `dpmq://QM?RequestQueue=...`, where QM is the name of MQ QM configuration object
- Dynamic MQ URL opener
  - Doesn't require statically defined MQ Queue Manager object
  - URL is using mq:// prefix and has the following format `mq://host:port?QueueManager=<QM_NAME>.....`, where QM\_NAME is the name of MQ Queue Manager running on a specific host and listening on a specific port

# Troubleshooting – DataPower Side

- Enable log level to “debug” using trouble shooting icon on the control panel
- Enable probe for the particular MPGW service
- Run few transactions and observe the system log
- Look for MQ Reason Code(s) and errors in the system log
- Understand the MQ Reason Code(s) using MQ supportpac “[ma0k](#)” available at <http://www-01.ibm.com/support/docview.wss?uid=swg24000652>

## **MQ supportpac Link:**

<http://www-01.ibm.com/support/docview.wss?rs=977&uid=swg27007205>





# Troubleshooting – MQ Server Side

- MQ function
- Access via: `su – mqm`
- Display queue status
  - ▶ Processes – how many connections on get/put
  - ▶ Queue depth
  - ▶ Uncommitted messages present?
  - ▶ Queue Handles
    - Who has open connections?
    - Are those connections input or output?

# Summary

- Traffic Patterns such as MQ-to-MQ, MQ-to-HTTP, MQ-to-JMS, MQ-to-TIBCO and MQ-to-FTP were discussed
- MQ Message Processing Modes (Datagram, Request/Reply and its variation with transactionality were presented
- Configurations involving use case scenarios for MQ-to-MQ, MQ-to-MQRFH2 and MQRFH2-to-MQ were discussed



# Summary Continued..

- MQ Routing based on MQMD and MQOD
- MQ Routing based on static and dynamic URLs
- MQ SSL Channel configuration using SSL proxy profile
- Error Handling – Capturing “response code” in response rule with “x-dp-response-code”



# Summary Continued..

- Trouble Shooting Techniques
  - ▶ DataPower side trouble shooting
  - ▶ MQ server side “runmqsc” to check the Queue Status for “IPPROCS”
  - ▶ Check Queue Status with Handles
    - dis qs(qname) type(handles)
    - dis qs(qname) type(handle) conname input output



# References

- IBM® WebSphere DataPower SOA Appliances webGUI Guide
- IBM WebSphere DataPower SOA Appliances Reference Guide
- MQ Series Application Programming Reference
- IBM WebSphere DataPower XSLT extension elements, extension functions, and variables Guide

# Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at:  
[http://www.ibm.com/software/websphere/support/supp\\_tech.html](http://www.ibm.com/software/websphere/support/supp_tech.html)
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at:  
<http://www.ibm.com/developerworks/websphere/community/>
- Join the Global WebSphere User Group Community:  
<http://www.websphere.org>
- Access key product show-me demos and tutorials by visiting IBM Education Assistant:  
<http://www.ibm.com/software/info/education/assistant>
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically:  
<http://www.ibm.com/software/websphere/support/d2w.html>
- Sign up to receive weekly technical My Notifications emails:  
<http://www.ibm.com/software/support/einfo.html>

# We Want to Hear From You!

## Tell us about what you want to learn

Suggestions for future topics  
Improvements and comments about our webcasts  
We want to hear everything you have to say!

**Please send your suggestions and comments to:**  
[wsehelp@us.ibm.com](mailto:wsehelp@us.ibm.com)

# Questions and Answers



# BACKUP CHARTS



# Troubleshooting – “runmqsc” examples

```
dis qs(Queue.STEVE.XI50)
```

```
1 : dis qs(Queue.STEVE.XI50)
```

```
AMQ8450: Display queue status details.
```

Queue(Queue.STEVE.XI50)	Type(Queue)
CurDepth(0)	IPProcs(10)
LGetDate( )	LGetTime( )
LPutDate( )	LPutTime( )
MonQ(Off)	MsgAge( )
OPProcs(0)	QTime( , )
Uncom(No)	

# Troubleshooting – “runmqsc” examples

```
dis qs(*)
  2 : dis qs(*) IPPROCS
AMQ8450: Display queue status details.
  QUEUE(Queue.STEVE.XI50)          TYPE(Queue)
  CURDEPTH(0)                     IPPROCS(10)
AMQ8450: Display queue status details.
  QUEUE(Queue.STEVE.XI50)          TYPE(Queue)
  CURDEPTH(0)                     IPPROCS(10)
... more queues to follow, including system queues ...
```

# Troubleshooting – “runmqsc” examples

```

dis qs(Queue.STEVE.XI50) TYPE(HANDLE)
  4 : dis qs(Queue.STEVE.XI50) TYPE(HANDLE)
AMQ8450: Display queue status details.
  QUEUE(Queue.STEVE.XI50)                TYPE(HANDLE)
  APPLTAG(WebSphere Datapower MQClient)  APPLTYPE(USER)
  BROWSE(NO)                             CHANNEL(MYCHANNEL.SVRCONN)
  CONNAME(XXX.XXX.XXX.XXX)               HSTATE(ACTIVE)
  INPUT(SHARED)                          INQUIRE(NO)
  OUTPUT(NO)                              PID(7136)
  QMURID(0.14707544)                     SET(NO)
  TID(3258)
  URID(XA_FORMATID[00000000] XA_GTRID[] XA_BQUAL[])
  URTYPE(QMGR)                            USERID(myuser)
  ... output like this for each connection

dis qs(Queue.STEVE.XI50) TYPE(HANDLE) CONNAME INPUT OUTPUT
  5 : dis qs(Queue.STEVE.XI50) TYPE(HANDLE) CONNAME INPUT OUTPUT
AMQ8450: Display queue status details.
  QUEUE(Queue.STEVE.XI50)                TYPE(HANDLE)
  CONNAME(XXX.XXX.XXX.XXX)               INPUT(SHARED)
  OUTPUT(NO)
  ... output like this for each connection

```