Enabling WebSphere MQ Traffic with WebSphere DataPower - Use Case Scenarios

Chin Sahoo (chintam3@us.ibm.com)
DataPower SOA Appliances Support
May 18, 2010
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
    
    ```xml
    <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
- MQCOMIT 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

- MQCOMIT 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 qmqr 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 qmqr is configured to handle message routing to remote qmqr 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 qmgrp
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.

**MQ Queue Manager: LINUX-MQ**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin State</td>
<td>enabled</td>
</tr>
<tr>
<td>Comments</td>
<td></td>
</tr>
<tr>
<td>Host Name</td>
<td>MQ-LNX(1520)</td>
</tr>
<tr>
<td>Queue Manager Name</td>
<td>DPT</td>
</tr>
<tr>
<td>Channel Name</td>
<td>SYSTEM.DEF.SVRCONN</td>
</tr>
<tr>
<td>Channel Heartbeat</td>
<td>300</td>
</tr>
<tr>
<td>User Name</td>
<td>mom</td>
</tr>
<tr>
<td>Maximum Message Size</td>
<td>1048576 bytes</td>
</tr>
<tr>
<td>Cache Timeout</td>
<td>30 seconds</td>
</tr>
<tr>
<td>Units of Work</td>
<td>1</td>
</tr>
<tr>
<td>Automatic Backout</td>
<td>on</td>
</tr>
<tr>
<td>Backout Threshold</td>
<td>2</td>
</tr>
<tr>
<td>Backout Queue Name</td>
<td>ERROR.Q</td>
</tr>
<tr>
<td>Total Connection Limit</td>
<td>25</td>
</tr>
<tr>
<td>Initial Connections</td>
<td>1</td>
</tr>
<tr>
<td>SSL Key Repository</td>
<td>cert:///</td>
</tr>
</tbody>
</table>

**SSL Cipher Specification**

- None

**SSL Proxy Profile**

- [none]

**Convert Input**

- on off

**Automatic Retry**

- on off

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retry Interval</td>
<td>10 seconds</td>
</tr>
<tr>
<td>Retry Attempts</td>
<td>6 attempts</td>
</tr>
<tr>
<td>Long Retry Interval</td>
<td>1200 seconds</td>
</tr>
<tr>
<td>Reporting Interval</td>
<td>20 seconds</td>
</tr>
<tr>
<td>Alternate User</td>
<td>on off</td>
</tr>
<tr>
<td>Local Address</td>
<td></td>
</tr>
<tr>
<td>XML Manager</td>
<td>default</td>
</tr>
</tbody>
</table>

**WebSphere® Support Technical Exchange**

18 of 44
Use Case-1: MQ Front Side Handler

<table>
<thead>
<tr>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin State</td>
</tr>
<tr>
<td>enabled</td>
</tr>
<tr>
<td>disabled</td>
</tr>
<tr>
<td>Comments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Queue Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINUX-MQ</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Get Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE1</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

| Put Queue             |
|                       |

<table>
<thead>
<tr>
<th>The number of concurrent MQ connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Get Message Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>4097</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Polling Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Retrieve Backout Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>on(✓)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Queue Manager in URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>on(✓)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publish and Subscribe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe Topic String</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Subscription Name</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Publish Topic String</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Properties and Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse Properties</td>
</tr>
<tr>
<td>on(✓)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selector</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Exclude Message Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS Bridge Header (MQCIM)</td>
</tr>
<tr>
<td>Dead Letter Header (MQDTH)</td>
</tr>
<tr>
<td>IMS Information Header (MQIIIH)</td>
</tr>
<tr>
<td>Rules and Formatting Header (MQRFH1)</td>
</tr>
<tr>
<td>Rules and Formatting Header (MQRFH2)</td>
</tr>
<tr>
<td>Work Information Header (MQWIN)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header to extract Content-Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Async Put</td>
</tr>
<tr>
<td>on(✓)</td>
</tr>
</tbody>
</table>
Use case-1: Example MQMD.Format

```xml
<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

```xml
<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>
      <name>
        <Nm>jms_text</Nm>
      </name>
      <name>
        <Nm>queue:///ALERTSX_INPUT_QUEUE</Nm>
      </name>
      <Rto>/queue:///ALERTSX_REPLY_QUEUE</Rto>
      <Tms>1197997507781</Tms>
      <Pri>0</Pri>
      <Dlv2</Dlv>
    </NameValueData>
  </MQRFH2>
</variable>
```

```
<variable name="rfh2Str">
  <dp:serialize select="SRFH2" omit-xml-decl="yes"/>
</variable>
```
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

Header Injection Parameters

<table>
<thead>
<tr>
<th>Direction</th>
<th>Header Name</th>
<th>Header Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back</td>
<td>MQMD</td>
<td><code>&lt;MQMD&gt;&lt;Format&gt;MQSTR&lt;/Format&gt;&lt;/MQMD&gt;</code></td>
</tr>
</tbody>
</table>

Header Suppression Parameters

<table>
<thead>
<tr>
<th>Direction</th>
<th>Header Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back</td>
<td>MQRPHZ</td>
</tr>
</tbody>
</table>

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

Crypto Profile: MQ-Crypto-Profile [up]

- Admin State: enabled, disabled
- Identification Credentials: MQ-Crypto-IDCred
- Validation Credentials: MQ-Crypto-ValCred
- Ciphers: RC4
- Options: Enable default settings, Disable SSL version 2, Disable SSL version 3, Disable TLS version 1
- Send Client CA List: on, off
Use Case 2: SSL Identity and Validation Credentials

Configure Crypto Identification Credentials

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

Configure Crypto Validation Credentials

Crypto Validation Credentials: MQ-Crypto-ValCred [up]
MQ Routing: Complete MQMD Header Example

```xml
<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>MOSTR</Format>
    <Priority>0</Priority>
    <Persistence>0</Persistence>
    <MsgId>414d5120454942544853303120202049cd019922fb7f07</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/>
</MQMD>
```
MQ Routing: MQOD header Injection Example

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

```xml
<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)

```xml
<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 Queue Manager 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

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
- 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
- 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
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

```bash
> dis q(Queue.Steve.X150) TYPE(HANDLE)
  4 : dis q(Queue.Steve.X150) TYPE(HANDLE)
AMQ8450: Display queue status details.
  QUEUE(Queue.Steve.X150) TYPE(HANDLE)
  APPLTAG(WebSphere Datapower MQClient) APPLTYPE(USER)
  BROWSE(NO) CHANNEL(MYCHANNEL.SVRCONN)
  CONNAME(www.xxx.xxx.xxx.xxx) HSTATE(VOICE)
  INPUT(SHARED) INQUIRE(NO)
  OUTPUT(NO) PID(7136)
  QMURID(0.1470754) SET(NO)
  TID(3258) ...
  URID(XA_FORMATID[00000000] XA_GTRID[] XA_BQUAL[])
  URRTYPE(QMGR) USERID(myuser)
... output like this for each connection

> dis q(Queue.Steve.X150) TYPE(HANDLE) CONNAME INPUT OUTPUT
  5 : dis q(Queue.Steve.X150) TYPE(HANDLE) CONNAME INPUT OUTPUT
AMQ8450: Display queue status details.
  QUEUE(Queue.Steve.X150) TYPE(HANDLE)
  CONNAME(www.xxx.xxx.xxx.xxx) INPUT(SHARED)
  OUTPUT(NO)
... output like this for each connection
```