S09439: You think you have problems...well maybe you do. Diagnosing problems for Message Broker

Ralph Bateman - STSM Messaging Serviceability & Support Strategy – ralph@uk.ibm.com
David Coles – WebSphere Message Broker Level 3 Technical Lead, IBM Hursley – dcoles@uk.ibm.com

Thursday 11th August 2011

Agenda

- What are the “moving” parts
- Where is the diagnostic information
  - Syslog
  - Stdout/Stderr
  - Error log
- What Trace to use
  - User Trace
  - Service Trace
- What do I do with a Dump / FFDC ?
- What are the common “Status” commands
- “Out the box” Tools available for debugging
  - Toolkit
  - MBX
- How to Diagnose Common Scenarios
  - My Broker won’t start
  - My Flow wont deploy and my EG fails
  - Where’s my output message?
The “moving” parts of a z/OS Broker

- **Address Space and Process model.**
  - The broker runtime environment is a *collection* of address spaces (AS), which allows natural isolation, recovery and scalability.
  - Each AS contains at least 2 Language Environment (LE) processes. The first, or “infrastructure” process is started authorized so that it can create z/OS components (PCs for SVC dumps etc.), and then returns to problem state. This process only exists on z/OS. After initialization, it creates and monitors a second process, which performs the main brokering function.
  - Other processes in each AS runs platform independent code using C++ and Java (publish/subscribe) to implement brokering function.

- **What are the brokering function address spaces?**
  - Control address space. This is the broker started task address space. The Control process within it is small and monitors for failures of the Administration Agent (AA) process. On z/OS, a console listener thread enables z/OS console interactions with users through the MODIFY interface. The AA process serves as the agent to the configuration manager and, by extension, Workbench. It manages the deployment of message flows and message sets, and manages the lifecycle and command reporting of execution groups (EG). When using WebServices nodes (HTTPInput, HTTPReply) the http listener process runs in this address space.
  - Execution Group address spaces. These are where the message flows deployed from the Configuration Manager execute. The DataFlowEngine process itself contains a number of threads and predefined flows (Configuration) to support the various brokering functions. Multiple EG address spaces remove any concern about (Virtual Storage Constraint Relief) VSCR.

- **z/OS, Language Environment and Java provide many services.**
  - An LE process is a set of threads sharing resources (file handles etc.). An AS serves as a process container. Processes start in the same address space according to the _BPX_SHAREAS (YES,NO) environment variable, and/or authorization requirements. A thread is a unit of execution synonymous with a Task Control Block (TCB).

- **OMVS.** This address space provides several industry standard interfaces (XPG4) that allow the MB processing model and code to be largely platform independent.
  - WebSphere MQ. This is one of the primary transports for dataflows, and WMQ uses it for inter-process and inter-platform communication. For example, the AA communicates with the EGs and CM using XML messages flowed over WMQ.
  - RRS. As the broker runs within regular z/OS address spaces, Resource Recovery Services (RRS) is the transaction manager that enables the coordination of resources (message queues, database tables) accessed by a dataflow.

- **There are a large number of associated address spaces with which the broker interacts.**

- **Notes:**
  - The major processes (not bipimain) written in C++ and Java are supported by the LE and Java runtimes respectively. These use z/OS interfaces for much of their processing.
  - An LE process is a set of threads sharing resources (file handles etc.). An AS serves as a process container. Processes start in the same address space according to the _BPX_SHAREAS (YES,NO) environment variable, and/or authorization requirements. A thread is a unit of execution synonymous with a Task Control Block (TCB).
Where is the diagnostic information

As you’d expect from a regular z/OS subsystem, the operational output will be available from the MVS SYSLOG and JOBLOGs in the form of BIP product messages.

JOBLOGs have the advantage of partitioning messages by address space, typically execution groups. All command output is directed to the JES SPOOL as expected.

Notice the STDOUT/STDERR files to help catch those Java programmers writing to standard output devices.

What Trace to use and when

Trace is available for separate components which can be formatted to a text output. Held in <CO

Various trace is available:
  • Flow User Trace – for you
  • Internal “Service” Trace – for IBM Support
  • Command traces – for all
  • CVP trace – for all
The broker is not capable of handling a message of data type 'MQSTR'. The message broker received a message that requires the handling of data of type 'MQSTR', but the broker does not have the capability to handle data of this type. Check both the message being sent to the message broker and the configuration data for the node. References to the unsupported data type must be removed if the message is to be processed by the broker.

No user action required.

Example usertrace

Where is the diagnostic information

- Useful Output files
  - Now that you've understood the moving parts of a z/OS broker and how to use it, here's a useful file list and a brief content description.

- Job Log and Syslog
  - This is where you'll go to find out most of your local operational information on broker behaviour. Note that the Eclipse tooling or Message Broker Explorer will provide Administration Perspectives applicable for some class of users, here we are considering native operational interaction with the z/OS operating system.

- SYSLOG (SDSF) contains all BIP messages. In the event of a problem, it's also worth looking for messages from other subsystems, e.g. End of Task messages (EOT).

- z/OS Standard Message suppression techniques (MPFLSTxx) can be used to stop any of the commands reaching the MVS log, if you have concern about volume or message importance of messages. This is not usually applicable of Broker users (i.e. the broker doesn’t generate unnecessary messages to the console.)

- FFDC/Abends
  - MiniAbends and FFDCs can be found on zFS to help get first failure information of errors

- Trace files
  - Trace files are always written to the zFS, so make sure you have enough space in the mount.

- No user action required.

Useful file list

Example usertrace

Where is the diagnostic information

- Useful Output files
  - Now that you've understood the moving parts of a z/OS broker and how to use it, here's a useful file list and a brief content description.

- Job Log and Syslog
  - This is where you'll go to find out most of your local operational information on broker behaviour. Note that the Eclipse tooling or Message Broker Explorer will provide Administration Perspectives applicable for some class of users, here we are considering native operational interaction with the z/OS operating system.

- SYSLOG (SDSF) contains all BIP messages. In the event of a problem, it's also worth looking for messages from other subsystems, e.g. End of Task messages (EOT).

- z/OS Standard Message suppression techniques (MPFLSTxx) can be used to stop any of the commands reaching the MVS log, if you have concern about volume or message importance of messages. This is not usually applicable of Broker users (i.e. the broker doesn’t generate unnecessary messages to the console.)

- FFDC/Abends
  - MiniAbends and FFDCs can be found on zFS to help get first failure information of errors

- Trace files
  - Trace files are always written to the zFS, so make sure you have enough space in the mount.
stdout/stderr

• Useful place to look for errors / debugging
  • Always worth checking for exceptions if problems are occurring
• Each major component redirects its stdout/stderr streams to files
  • Windows
    • Admin Agent (7.0.0.2)
      C:\Documents and Settings\All Users\Application Data\BMQMQSI\components\<brkName>\console.txt
    • Execution group
      C:\Documents and Settings\All Users\Application Data\BMQMQSI\components\<brkName>\<egUUID>\console.txt
  • Linux/Unix
    • Admin Agent (7.0.0.2)
      /var/mqsi/components/<brkName>/stderr & stderr
    • Execution group
      /var/mqsi/components/<brkName>\<egUUID>\stderr & stderr
  • z/OS
    • STDOUT / STDERR DD cards in the joblog for both the main broker address space and for any execution groups
• Can be useful for flow developers who use Java and code system.out.println statements for debugging
What do I do with a DUMP / FFDC?

- We bag up as much information as possible and put it in a "DUMP" or FFDC.

- The next steps are just a puzzle waiting to be solved

Call IBM Support or not?

- The traceback is placed into a CEEDUMP file, which resides in the /component_HFS/common/errors directory.
- Each traceback is preceded by the date, time, and unique identifier; for example, CEEDUMP file - CEEDUMP.20100924.171754.84017230
- The abend occurs with an Entry Point name of _NumCompute_evaluate.
- We know that Message Broker always starts Imb so this needs to be looked at by the application team or third party vendor who produced the ill.
What do I do with a DUMP / FFDC?

Dumps are taken by the broker for a number of situations. They do not always mean a code problem.

There are some very simple things that can be done to determine if the issue needs IBM Support help, or just your application teams.

Look at the FFDC files that the Broker produces in /common/errors/CEE* to determine who is at “fault”

If the Entry Point name starts IMB then raise a PMR and contact IBM Support.

If not, then the module may be a third party product or an application node.
Status commands

- Non-persistent trace option (7.0.0.3)
  - How do I find the evidence of what went wrong.
  - New ability to Enable execution group wide trace level that doesn’t survive a restart
  - Helps to capture trace for abend/shutdown situations
  - Stops traces being wrapped during restart

- What traces are running (7.0.0.3)
  - mqsireporttrace is now recursive
  - mqsireporttrace <brkName>
    - Reports all service and user traces which are active
  - mqsireporttrace <brkName> -t
    - Reports all service traces which are active
  - mqsireporttrace <brkName> -u
    - Reports all user traces which are active

What level is my broker?

- On z/OS look at the JOBLOG

  BIP9273 MQ91BRK RALPH 0 THE DATAFLOWENGINE PROCESS HAS REGISTERED SMF 89 SUBTYPE 1 RECORD COLLECTION, RETURN CODE '0', : ImbMain(397) BIP2208I MQ91BRK RALPH 0 EXECUTION GROUP (64) STARTED: PROCESS '16990944'; THREAD '2007049024452383176'; ADDITIONAL INFORMATION: BROKERNAME 'MQ91BRK' (OPERATION MODE='enterprise', EXECUTIONGROUPUUID '93d22eb0-3101-0000-0080-c3c6cb5a2a21'; EXECUTIONGROUPLABEL 'RALPH', QUEUENAME 'MQ91', TRUSTED 'false'; USERID ''; MIGRATIONNEEDED 'false'; BROKERUUID '3e2d440a-b1e4-11e0-a4e8-000000000000'; FILEPATH '/u/wmqi91/broker/instpath'; WORKPATH '/u/wmqi91/broker'; ICU CONVERTER PATH ''. : ImbMain(605)

  BIP9102I MQ91BRK RALPH 0 BROKER SERVICE VALUE IS IMBSERV.V7R0M00.FP02..... : ImbMain(614)

- On distributed run the command: mqsiservice --v

  C:\Program Files\IBM\MQSI\7.0.0.3.L110525_P\mqsiservice --v
  BIPcpu intel_64
  Console OEM CP=437, ICU CCSID=5348
  Default codepage=ibm-5348_P100-1997, in ascii=ibm-5348_P100-1997
  JAVA console codepage name=cp437

  BIP8996I: Version: 7103
  BIP8997I: Product: WebSphere Message Broker
  BIP8998I: CMVC Level: S700-L110525
  BIP8071I: Successful command completion.

  On distributed run the command: mqsibroker --v

  C:\Program Files\IBM\WME\7.0.0.3.L110525_P\mqsibroker --v
  BIPcpu intel_64
  Console OEM CP=437, ICU CCSID=5348
  Default codepage=ibm-5348_P100-1997, in ascii=ibm-5348_P100-1997
  JAVA console codepage name=cp437
  BIP8996I: Version: 7103
  BIP8997I: Product: WebSphere Message Broker
  BIP8998I: CMVC Level: S700-L110525
  BIP8071I: Successful command completion.
What “out the box” Tools are available?

• Message Broker Toolkit
  • Flow debugger
    • Useful for debugging message flows during their development
• Message Broker Explorer
  • Flow statistics
    • Useful for highlighting the cause of performance problems
    • Baseline good operation
  • Resource stats
    • Useful for highlighting areas of concern in message flows or performance bottlenecks.
**Message Flow Debugger**

**HOWTO** Enable the debugger to allow you to debug message flows from the Message Broker toolkit.

**Message Flow Debugger**

**HOWTO** Configure the Source lookup path to enable you to step through your message flow application.
Message Flow Debugger

- Use the Message Flow debugger to debug your message flows
- Set breakpoints on the connections between nodes
- At each stage you can view (and edit) the Message Trees
- Step into ESQL or Java compute nodes
- Requires the enablement of the JVMDebug port on the execution group you wish to debug
  - Don’t do this on production machines as it hits performance
Message Flow Statistics

Message Flow Statistics - Notes

- You can enable and display message flow statistics using MBX.
- You can use the results and graphs to diagnose performance and operational problems.
- During normal operation it can be useful to enable message flow statistics so you can get a feel for what ‘normal’ looks like as well as to help tuning flows and machines for performance.
- During operational issues you may notice that certain nodes are using more/less cpu than normal, or maybe a certain node is showing no usage data, so perhaps is no longer being driven when it should. All quantifiable differences could be indicative of an issue worth investigating, or provide clues to help solve other problems.
- With the statistics view open in MBX click on the execution group or message flow to change the scope of the data displayed. Clicking on the execution group will restrict the information to the flow level, whereas clicking on a given flow will show the statistics at a node level.
Resource Statistics

- Find out the current resource usage of a broker or execution group:
  - CICS – successful requests, failures, security failures...
  - CORBA – Invocations, Success, Failures
  - FTE – Inbound/Outbound transfers, bytes sent/received...
  - JDBC – Requests, Cached requests, Providers...
  - JVM – Memory used, thread count, heap statistics...
  - ODBC – Connections, Closures, Errors, Successes
  - SOAPInput – Inbound messages, Replies, Failures, Policy Sets
  - Security – Operations, Success, Failures, Cache usage...
  - Sockets – Total sockets, message sizes, Kb sent/received
  - Parsers – Memory usage, message elements created/deleted, parser count

- More resource types being added in the future

Resource Statistics - Examples

- Each resource reports values specific to the given resource type
- Failure counts are often key values to monitor

Parser stats provide a great insight to a given flow
Resource Statistics - Notes

- Every resource is different, so each resource reports different values.
- In the same way as with message flow statistics, it's a good idea to enable resource statistics when things are going well so that you can get a feeling, or some concrete numbers for what to expect.
- The key values to monitor are the failure counts.
- When you’re having trouble and you know that your flows make use of certain resources take a look at the failure counts or connection errors to see if they are highlighting any issues.
- Other key values are the bytes sent/received. If these are not increasing when they should be then they might point at a problem.
- Message Broker does not attempt to provide analytics or averages on the values. The raw numbers are presented and published to allow customers and external tools to provide the analytics.
- As with message flow statistics the resource statistics data format is documented and the data is published as XML messages to a given topic allowing customers and other applications to subscribe to and process the data.
- As well as looking for errors through the failed reads and failed writes counts, parser resource statistics can be used to size your message flows as they given an approximation of the memory used by the message flow for parsing and the logical tree structure. You can also see the maximum size input and output buffers used, so you can tell the max message sizes being processed, and you can see the number of fields being created, which can all point at problems with flow design, or if these numbers change over baseline figures, of a change in message size/type being processed.
How to diagnose some common scenarios

Scenarios

• Message Broker won’t start
  • First check the JOBLOG.
  • Then check the STDOUT/STDERR for MQSICVP

If Java has been correctly installed, see the preceding messages for further information about the cause of this failure, and the actions that you can take to resolve it.
Deploy of a Message flow fails

- Whether you deploy using the Message Broker Toolkit, MBX or via the command line you will see any deploy errors being reported
- Always make sure to read all the messages.
  - The first or last message might not always contain the most relevant information

In this scenario the Java compute node cannot find its class
- Has the relevant jar file been deployed or made available in the shared_classes directory?
Where’s my output message?

• A user reports that they’re not receiving any messages
• So where are the messages going and what can you look at?
  • Resource statistics
  • Message Flow statistics
  • User trace
  • Message Flow Debugger
• We’ll see how all of the above can be used to piece together the pieces of the puzzle

• The message flow

  MQ Input  Compute  Doc Control  MQ Output

• A simple MQ In/Out flow with some transformation logic

Where’s my output message?

• Resource statistics
  • Is there a resource stat available for your output transport that would show if messages are being written?
    • MQ is not yet available, so not helpful here
    • But CICS, CORBA, FTP, File, HTTP Sockets & TCPIP Nodes are available
    • The parsers output could be useful
    • Are any messages being written?

• No writes are occurring
• So no output messages are being written
Where’s my output message?

- Message Flow statistics
  - Are all nodes in the flow being driven as expected?

The MQOutput node is not being driven
- So no output messages will be written.

Where’s my output message?

- User Trace
  - Can we see why the MQOutput node is not being driven?
  - We saw earlier what to look for in a User Trace, can we see that here?

UserTrace BIP2632I: Message received and propagated to 'out' terminal of MQ input node 'DebugFlow1.MQ Input'.

... then the trace ends ....
- The last thing the trace shows is the JavaCompute node being invoked
- This never propagates to its output terminal
- Why?
Where’s my output message?

- Message Flow Debugger
  - Enable and connect to the debug port
  - Add a breakpoint to the message flow

- Fire in a message and the breakpoint triggers

- We can then step through the message flow and into the ESQL and Java code

Where’s my output message?

- Message Flow Debugger
  - As the message is never propagated from the JavaCompute node we need to see why
  - When the flow is paused on the connection between the compute and JavaCompute nodes we can step into the source
Where’s my output message?

• Message Flow Debugger
  • Once in the Java source we can step through the code to understand why
    propagate is never called
  
  ```java
  EnElement totalElement = orderElement.xmlElementByPath("Total");
  BigDecimal total = (BigDecimal)totalElement.getXmlNode().getNumericValue();
  if (total.getDouble() > 50)
  {
    // do nothing, it’s not cheap enough
    // call the message out
    if (propagate) {messageOut();}
  }
  
  60 Variables 53 Breakpoints
  Variable       Value
  totalElement   EnElement
  orderElement   EnElement
  
  • We only propagate if the total cost is not
    greater than 50
  • Here it’s 231
  • So case closed, input data, user
    expectation or message flow design error

Summary

• What are the “moving” parts
• Where is the diagnostic information
• What Trace to use
• What do I do with a Dump / FFDC ?
• What are the common “Status” commands
• “Out the box” Tools available for debugging
• How to Diagnose Common Scenarios
### This was session 09431 - The rest of the week ......

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>08:00</td>
<td>More than a buzzword: Extending the reach of your MQ messaging with Web 2.0</td>
<td>Batch, local, remote, and traditional MVS - file processing in Message Broker</td>
<td>The Do’s and Don’ts of Queue Manager Performance</td>
<td>Lyn’s Story Time - Avoiding the MQ Problems Others have Hit</td>
<td>MQ Project Planning Session</td>
</tr>
<tr>
<td>09:30</td>
<td>WebSphere MQ 101: Introduction to the world’s leading messaging provider</td>
<td>So, what else can I do? - MQ API beyond the basics</td>
<td>The Do’s and Don’ts of Message Broker Performance</td>
<td>All About WebSphere MQ File Transfer Edition</td>
<td>The Do’s and Don’ts of Message Broker Performance</td>
</tr>
<tr>
<td>11:00</td>
<td>MQ Publish/Subscribe</td>
<td>Diagnosing problems for Message Broker</td>
<td>Using the WMQ V7 Verbs in CICS Programs</td>
<td>Diagnosing problems for Message Broker</td>
<td>What’s new for the MQ Family and Message Broker</td>
</tr>
<tr>
<td>12:15</td>
<td>MQ Freestie! Top 5 SupportPacs</td>
<td>The doctor is in. Hands-on lab and lots of help with the MQ family</td>
<td>The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation</td>
<td>MQ Project Planning Session</td>
<td></td>
</tr>
<tr>
<td>01:30</td>
<td>Diagnosing problems for MQ</td>
<td>WebSphere Message Broker 101: The Swiss army knife for application integration</td>
<td>Keeping your eye on it all - Queue Manager Monitoring &amp; Auditing</td>
<td>Getting your MQ JMS applications running, with or without WAS</td>
<td>Using the WMQ V7 Verbs in CICS Programs</td>
</tr>
<tr>
<td>03:00</td>
<td>The MQ API for dummies - the basics</td>
<td>Under the hood of Message Broker on z/OS - WLM, SMF and more</td>
<td>Message Broker Patterns - Generate applications in an instant</td>
<td>Message Broker Patterns - Generate applications in an instant</td>
<td>Using the WMQ V7 Verbs in CICS Programs</td>
</tr>
<tr>
<td>04:30</td>
<td>Message Broker administration for dummies</td>
<td>For your eyes only - WebSphere MQ Advanced Message Security</td>
<td>Keeping your MQ service up and running - Queue Manager clustering</td>
<td>Message Broker Patterns - Generate applications in an instant</td>
<td>Using the WMQ V7 Verbs in CICS Programs</td>
</tr>
<tr>
<td>06:00</td>
<td>Free MQ - MQ Clients and what you can do with them</td>
<td>MQ-Q-Box - Open Microphone to ask the experts questions</td>
<td>Diagnosing problems for Message Broker</td>
<td>MQ Project Planning Session</td>
<td></td>
</tr>
</tbody>
</table>

### Session 09439

You think you have problems...well maybe you do. Diagnosing problems for Message Broker