Diagnosing Problems for WebSphere Message Broker (z/OS and Distributed)

Dave Crighton – WebSphere Message Broker L3 Service Delivery Lead
IBM Hursley – davicrig@uk.ibm.com

Tuesday 7th August 2012
Session Number 11508
Agenda

- WMB Recap
- External Components
- Diagnostic Information
- How to diagnose common scenarios
WebSphere Message Broker

- Universal Connectivity FROM anywhere, TO anywhere
  - Simplify application connectivity for a flexible & dynamic infrastructure

- Comprehensive Protocols, Transports, Data Formats & Processing
  - Connect to applications, services, systems and devices
    - MQ, JMS 1.1, HTTP(S), SOAP, REST, File (incl. FTP, FTE, ConnectDirect), Database, TCP/IP, MQTT, CICS, IMS, SAP, SEBL, .NET, PeopleSoft, JDEdwards, SCA, CORBA, email...
  - Understand the broadest range of data formats
    - Binary (C/COBOL), XML, CSV, JSON, Industry (SWIFT, EDI, HL7...), IDOCs, User Defined
  - Built-in suite of request processors
    - Route, Filter, Transform, Enrich, Monitor, Publish, Decompose, Sequence, Correlate, Detect...

- Simple Programming with Patterns & Graphical Data Flows
  - Patterns for top-down, parameterized connectivity of common use cases
    - e.g. Service façades, Message processing, Queue2File...
    - IBM & User defined patterns for development reuse & governance
  - Graphical data flows represent application & service connectivity
    - Custom logic via Graphical mapping, PHP, Java, ESQL, XSL & WTX

- Extensive Management, Performance & Scalability
  - Extensive Administration & Systems Management facilities for developed solutions
  - Wide range of operating system & hardware platforms supported, including virtual & cloud options
  - High performance transactional processing, additional vertical & horizontal scalability
  - Deployment options include Trial, Express, Standard and Advanced

- Connectivity Packs for Industry Specific Content
  - Connectivity Pack for Healthcare includes HL7 Connectors, Patterns & Tooling
Agenda

- WMB Recap
- External Components
- Diagnostic Information
- How to diagnose common scenarios
External Components

- Message Broker Toolkit
- Message Broker Explorer
- Command line
- Third Party Tools
- Web UI

Message Broker Java API

Message Broker REST API

Broker

// Instantiate an object that describes the connection
// characteristics of the broker
BrokerConnectionParameters params =
   new MQBrokerConnectionParameters(brokerHostName, brokerPort, brokerQmgr,
BrokerProxy b = null;

GET /admin/eg/MYEGNAME
From: machine@ibm.com
User-Agent: MyApp/1.0

Complete your sessions evaluation online at SHARE.org/AntelheimEval
Broker View for Application Developers
Message Broker Explorer (MBX)

- Advanced broker management option designed for administrators
- Plug-in to MQ Explorer
- Extra features
  - Create/Manage Configurable Services
  - Performance Views
  - Group brokers using broker sets
  - Offload WS-Security onto Datapower
  - Administration Log
  - Administration Queue
Command line tools

- A wide selection of tools for scripting broker actions
- Requires a configured environment
  - Command console (Windows)
  - mqsisprofile (Linux/UNIX)
  - JCL or ISPF (z/OS)
- Most commands work against local or remote brokers

BIP1121I: Creates an execution group.

Syntax:
`mqsicreateexecutiongroup [brokerSpec] -e egName [-w timeoutSecs] [-v traceFileName]`

Command options:

- `brokerSpec` is one of:
  - (a) `brokerName`: Name of a locally defined broker
  - (b) `-n brokerFileName`: File containing remote broker connection parameters (*.broker)
  - (c) `-i ipAddress -p port -q qMgr`: hostname, port and queue manager of a remote broker

- `-e egName`: name of the new execution group
- `-w timeoutSecs`: maximum number of seconds to wait for the execution group to be created
- `-v traceFileName`: send verbose internal trace to the specified file.
Message Broker API (CMP)

- Java interface that enables the broker administration tools
- Use for custom administration requirements
- Fully documented and samples available

V8 allows you to create and edit message flows too
- Build your entire system programmatically!
Web UI

- Web Administration Console
  - Objective is to provide comprehensive web management interface
  - Focus on non-administrators to understand brokers & resources
  - Supports all major browsers Firefox, IE, Opera, Safari, Chrome
  - Designed as users as a complement MBExplorer
    - MB Administrators can users continue to use MB Explorer

- Easy to configure
  - No extra moving parts - uses internal HTTP listener to serve data
    - Web admin started by default on port 7050
  - Can reconfigure to listen on user port or disable
    - SSL connector configured via mqsichangeproperties
  - Role based access provides custom class user control
    - Default is read-only access to MB resources
    - More authority required to create, change or delete resources

- Using Web Admin
  - Intuitive tree view shows hierarchy of MB resources
  - View resource details with click or button
  - Includes full suite of resources
    - Apps, Libs, Flows, Configurable services etc

- Web Admin & MB Explorer
  - MBX & web admin designed for concurrent use
  - Web admin requires MB8 broker
  - Explorer can manage both MB8 & MB7 brokers

Complete your sessions evaluation online at SHARE.org/AnaheimEval
WMB Runtime Structure

Key components

- `bipservice`
  - “Angel” process, restarts brokers which have terminated unexpectedly

- `bipbroker`
  - Often referred to as the “AdminAgent”
  - The interface to CMP-API applications
  - Responsible for:
    - EG lifecycle
    - Deployment of artifact to EG’s
    - Reporting of EGs

- `biphttplistener`
  - The broker-wide http listener

- `DataFlowEngine`
  - The actual execution group where message flow processing takes place
Agenda

- WMB Recap
- External Components
- Diagnostic Information
- How to diagnose common scenarios
Diagnostic Information in WMB

- Diagnostic Information is available from a multitude of sources
  - System Log
  - Message Flow and Resource Statistics
  - Activity Log
  - Event Monitoring
  - Message Tracking
  - Administration Log
  - Message Flow Debugger
  - Trace
  - Trace Nodes
  - Stdout/Stderr
  - Abends/Dumps

- Problem Diagnosis often requires you to coordinate different pieces of evidence from different places
System and Product Logs

• Diagnostic Information is written to platform specific logging facilities
  • z/OS
    • MVS SYSLOG and JOBLOG
  • Unix and Linux
    • Syslog
  • Windows
    • Event Log
• Messages have a 4 digit “BIP” number
  • For example BIP2153: “About to change an execution group”
• BIP messages only issued to system logs if not handled by flow
stdout/stderr

- Captures everything written to std io
  - Java: System.out.println()
  - C: printf()
- z/OS
  - STDOUT/STDERR DD cards in JOBLOGs
- Distributed
  - Admin Agent
    - $MQSI_WORKPATH/components/<Broker_Name>/stdout and stderr
  - Execution Groups
    - $MQSI_WORKPATH/components/<Broker_Name>/<EG_UUID>/stdout and stderr
Stdout/stderr and Java

- Captures diagnostic statement written by developers
- Often captures exception stacks in both Java Compute Nodes and third party libraries
  - Exception.printStackTrace()
- Destination for JVM based diagnostics
  - Usually enabled by passing the JVM a “-D” parameter on startup
    - Set environment variable IBM_JAVA_OPTIONS or _JAVA_OPTIONS
    - Use mqsichangeproperties
- Examples
  - Classloading trace (-Dibm.cl.verbose=*)
  - JSSE2 “SSL” trace (-Djavax.net.debug=all)
Flow Statistics

- Performance statistics
  - Lightweight non-invasive
  - Reported graphically in MBX
- Subscribe to data
  - Integrate with other products
  - Write your own applications.
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
Activity Log

- New Activity Logging Allows users to understand what a message flow is doing
- End-user oriented
- Focus on easily understood actions and resources
  - “GET message from queue X”, “Update DB table Z”…
- Flexible reporting options
  - All events for a specific flow
  - All events for a specific resource manager
  - Customer filters
  - Available via CMP API, MBX, log files
Event Monitoring

- Generate Monitoring and Audit Events from Message Flows
- Administration and Development Time Configuration
  - Every WMB Node includes a “Monitor” tab to generate events
    - Transaction: Start, End, Rollback (input nodes only)
    - Terminal: Any terminal, any node.
- Operational Control
  - Enable / Disable at runtime (mqsichangeflowmonitoring)
  - Events published on MQ Topics
  - Business Monitor integration
Message Tracking

- Enable Record, Edit and Replay of In-flight Data
  - Comprehensive audit of messages, web, ERP, file & other data
  - Flexible topology: single or multiple brokers for recording, capture & replay

- Data Recording, Capture & Store
  - Graphically configure binary, text, XML payload capture, including whole, partial & multi-field data
  - Source data is currently limited to MB flows, including MB6.1, MB7 & MB8
    - Monitor tab or monitoring profiles identify captured events
  - Capture events on *any broker*, local or remote
    - Any broker EG can be configured as capture agent
    - Configurable service identifies topic, target database
  - Agent stores data in any supported broker database
    - Oracle, DB2, SQL Server, Sybase, Informix…

- Web Tooling to View, Query, Edit data
  - Friendly editors to view, query & edit payloads
    - Key data fields, including application data
  - Independent web admin & capture for scalability
    - Configure multiple EG listeners for web

- Replay for redelivery or flow reprocessing
  - Replay selected data to flows or applications
  - MB admin configures logical destinations
    - Maps to physical protocol, e.g. MQ: {Qmgr, Q}
  - User selects destinations from auto-populated drop-down list
The tools include a lot of information that is useful to the administrator, for example:

- **Administration queue**: What operational changes are currently pending
- **Administration log**: What changes have been recently applied to the broker’s configuration, and by whom?
Trace

- Various trace options are available
  - User Trace – for you
  - Service Trace – for IBM Support
    - Various components: Commands, “Admin Agent”, Execution Group
  - CMP API Trace – for both you and IBM Support
  - CVP Trace – for both you and IBM Support
- Held in binary log files and formatted to text output
- Most detailed debugging option available
- Also highest runtime performance cost
  - User Trace relatively light, Service trace very heavy
Example usertrace

Trace written by version 7002; formatter version 7002 (build S700-FP02)
2011-08-09 21:58:23.159181 6468 UserTrace BIP2632I: Message received and propagated to 'out' terminal of MQ input node 'DebugFlow1.MQ Input'.
2011-08-09 21:58:23.159496 6468 UserTrace BIP6060I: Parser type "Properties" created on behalf of node 'DebugFlow1.MQ Input' to handle portion of incoming message of length 0 bytes beginning at offset '0'.
2011-08-09 21:58:23.159585 6468 UserTrace BIP6061I: Parser type "MQMD" created on behalf of node 'DebugFlow1.MQ Input' to handle portion of incoming message of length '364' bytes beginning at offset '0'. Parser type selected based on value "MQHMD" from previous parser.
2011-08-09 21:58:23.159654 6468 UserTrace BIP6069W: 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.
2011-08-09 21:58:23.160024 6468 UserTrace BIP6061I: Parser type "XMLNSC" created on behalf of node 'DebugFlow1.MQ Input' to handle portion of incoming message of length '143' bytes beginning at offset '364'. Parser type selected based on value "XMLNSC" from previous parser.
2011-08-09 21:58:23.160839 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "InputRoot" at ('.DebugFlow1_Compute.CopyEntireMessage', '2.20'). This resolved to "InputRoot". The result was "ROW... Root Element Type=16777216 NameSpace='' Name='Root' Value=NULL".
2011-08-09 21:58:23.161157 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS INTEGER)" at ('.DebugFlow1_Compute.Main', '4.93'). This resolved to "CAST("3 AS INTEGER"). The result was "3".
2011-08-09 21:58:23.161539 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.Item.Quantity)" at ('.DebugFlow1_Compute.Main', '4.98'). This resolved to "OutputRoot.XMLNSC.Order.Item.Quantity". The result was "3".
2011-08-09 21:58:23.161687 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS INTEGER)" at ('.DebugFlow1_Compute.Main', '4.93'). This resolved to "CAST("3 AS INTEGER"). The result was "3".
2011-08-09 21:58:23.161767 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS INTEGER)" at ('.DebugFlow1_Compute.Main', '4.91'). This resolved to "3 * 7". The result was "21".
2011-08-09 21:58:23.162040 6468 UserTrace BIP4015I: Message propagated to the 'out' terminal of node 'DebugFlow1.MQ Output' with the following message trees: ''. No user action required.
2011-08-09 21:58:23.162866 6468 UserTrace BIP2638I: The MQ output node 'DebugFlow1.MQ Output' attempted to write a message to queue "OUT.DEBUG" connected to queue manager "". The MQCC was '0' and the MQRC was '0'.
2011-08-09 21:58:23.162927 6468 UserTrace BIP2622I: Message successfully output by output node 'DebugFlow1.MQ Output' to queue "OUT.DEBUG" on queue manager "".

Threads encountered in this trace:
6468
Trace Status commands

- **Non-persistent trace option (7.0.0.3/8.0.0.0)**
  - 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 (7.0.0.3/8.0.0.0)**
  - 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

BIP8945I: Service trace settings for execution group 'test1' - mode: 'safe', size: ‘195’ KB
BIP8946I: Service trace is enabled for execution group 'test1' with level 'debug'.
BIP8945I: Service trace settings for execution group 'EG2' - mode: 'safe', size: ‘195’ KB
BIP8947I: Service trace is enabled for message flow 'TestFlow' with level 'debug'.
BIP8948I: User trace settings for execution group 'EG2' - mode: 'safe', size: ‘195’ KB
BIP8949I: User trace is enabled for execution group 'EG2' with level 'debug'.
Trace Nodes

- Configured at design time
- Can be turned on or off at runtime
- Logs parts of the message tree at key points in the flow
- Flexible writing options
  - Local File
  - User Trace
  - Error Log
Example Trace Node Output

```
{0x03000000:NameValue}:X-Server-Name = 'local' (CHARACTER)
{0x03000000:NameValue}:X-Server-Port = '7802' (CHARACTER)
{0x03000000:NameValue}:X-Query-String = ' ' (CHARACTER)
{0x03000000:NameValue}:X-Scheme = 'http' (CHARACTER)

{0x01000000:Folder}:SOAP
    {0x01000000:Folder}:Context = ( [{xmlns}: x1707bc8]
        {0x03000100:Attribute}:operation = 'submitPORequest' (CHARACTER)
        {0x03000100:Attribute}:operationType = 'REQUEST_RESPONSE' (CHARACTER)
        {0x03000100:Attribute}:portType = 'SOAPNodesSampleMessageSetPortType' (CHARACTER)
        {0x03000100:Attribute}:portTypeNamespace = 'http://com.http.orderservice' (CHARACTER)
        {0x03000100:Attribute}:port = 'SOAP_HTTPPort' (CHARACTER)
        {0x03000100:Attribute}:service = 'SOAPNodesSample_HTTP_Service' (CHARACTER)
        {0x03000100:Attribute}:FileName = 'C:\Documents and Settings\All Users\Application Data\IBM\MQSI\components\BRK8\2353'
        {0x03000000:PCDataField}:SOAP_Version = '1.1' (CHARACTER)

    {0x01000000:Folder}:Header = ( [{xmlns}: x1707bc8]
        {0x01000000:Folder}:http://www.acmeorders.com/OrderService:submitPORequest = ( [{xmlns}: x1707bc8]
            {0x03000000:PCDataField}:partNo = '012365' (CHARACTER)
            {0x03000000:PCDataField}:partQuantity = 10 (INTEGER)
            {0x01000000:Folder}:personName = ( [{xmlns}: x1707bc8]
                {0x03000000:PCDataField}:firstName = 'Dave' (CHARACTER)
                {0x03000000:PCDataField}:lastName = 'Crighton' (CHARACTER)

                {0x03000000:PCDataField}:address = ( [{xmlns}: x1707bc8]
                    {0x03000000:PCDataField}:street = 'myStreet' (CHARACTER)
                    {0x03000000:PCDataField}:city = 'myCity' (CHARACTER)
                    {0x03000000:PCDataField}:zipCode = 'myZip' (CHARACTER)

```

Complete your sessions evaluation online at SHARE.org/AnaheimEval
Exception List

- Exception only gets reported to the event log if it is not handled by the flow
- Nested list of exceptions
- Usually innermost exception is the root cause
  - Other exception are a result of the exception passing through other nodes or internal code blocks
- Examine the exception list programmatically for automatic error handling
- Examine exception lists for root cause when post-mortem debugging
Example ExceptionList

### Event Properties

- **Event:** File: C:\build\5800_D\src\MTI\MIforBroker\Mt1ImbParser2\Mt1ImbFIMHandler.cpp
- **Severity:** INTEGER:2
- **Number:** INTEGER:5902
- **Type:** CHARACTER:2
- **Text:** CHARACTER:2

---

**ExceptionList**

- **ExceptionList:**
  - **ParserException**
    - **Severity:** INTEGER:1
    - **Number:** INTEGER:5902
    - **Text:** CHARACTER:1

---

**Exception List:**

- **ExceptionList:**
  - **ParserException**
    - **Severity:** INTEGER:1
    - **Number:** INTEGER:5902
    - **Text:** CHARACTER:2

---

**Event:**

- **Date:** 05/08/2012
- **Source:** WebSphere Broker v8001
- **Time:** 22:06:04
- **Category:** None
- **Type:** Error
- **Event ID:** 5374
- **User:** N/A
- **Computer:** DAVICFIGW500

**Description:**

```
(BRKB.default) Message validation error: An element does not meet the minOcurs constraint. Element: "126"\"SW20" instances: 0 minOcurs: 1 parent: "MTI03" parent index: 1

Element: "126"\"SW20" has 0 instances in the logical tree, but has been defined with a minOccurs constraint of 1 within its parent
```

**Data:**

```
Bytes | Words
------|------
0000: 30 30 42 30 30 30 02 00 .B.R.K.
0010: 30 30 2c 30 30 30 02 00 .B.R.K.
```

---

**SHARE in Anaheim 2012**
Message Flow Debugger

MQ Input → Compute → Java Compute → MQ Output

- Step Over
- Step Into Source
- Resume
- Run To Completion
- Step Return

Variables:
- PartNo: 10001
- Price: 3
- Quantity: 77
- LocalEnvironment
  - Environment
  - ExceptionList

PartNo: CHARACTER: 10001

Complete your sessions evaluation online at SHARE.org/AnaheimEval
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
    - It disables Just In Time (JIT) compilation
Message Flow Debugger

**HOW TO:** Enable the debugger to allow you to debug message flows from the Message Broker toolkit.
**Message Flow Debugger**

**HOW TO:** Configure the Source lookup path to enable you to step through your message flow application.
Abends and DUMPS

- Generated when a process terminates unexpectedly
- z/OS
  - `<component_HFS>/common/errors`
- Distributed
  - `$MQSI_WORKPATH/common/errors`
- Contains useful information
  - Signal code received
  - Stack back trace
  - MVS abend codes
Reading a stack backtrace

- Each line is a native routine which was on the stack at the point of failure
- The top few entries will often be the Broker’s own abend handling routines and should be disregarded
  - ImbAbend::terminateProcessInternal etc
- After the abend handler entries the closer a routine is to the top of the stack the more likely it is to be the culprit
  - But not always, particularly in the case of data corruption the problem may have occurred far earlier during execution
- Message Broker internal classes start with the prefix “Imb”
**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`

<table>
<thead>
<tr>
<th>Traceback:</th>
<th>E Addr</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSA Addr</td>
<td>Program Unit</td>
<td>PU Addr</td>
<td>PU Offset</td>
<td>Entry</td>
<td>E Addr</td>
<td>E Offset</td>
</tr>
<tr>
<td>36F90BD0</td>
<td>CEEVRONU</td>
<td>0707D2B8</td>
<td>+00001004</td>
<td>CEEVRONU</td>
<td>0707D2B8</td>
<td>+00001004</td>
</tr>
<tr>
<td>390253A0</td>
<td>1DF416F8</td>
<td>+00000000</td>
<td>ImAbend::printStackForCurrentThread(int, bool, const void*, vo</td>
<td>1DF416F8</td>
<td>+00000000</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>390257B0</td>
<td>1E221258</td>
<td>+000003C2</td>
<td>ImAbend::terminateProcessInternal(const void*, const bool, vo</td>
<td>1E221258</td>
<td>+000003C2</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>39026080</td>
<td>1DF457F8</td>
<td>+000005BE</td>
<td>IMBCOND</td>
<td>1DF457F8</td>
<td>+000005BE</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>39026120</td>
<td>0707B2E0</td>
<td>+00001252</td>
<td>CEEVRONU</td>
<td>0707B338</td>
<td>+000011FA</td>
<td>CEEPLPKA</td>
</tr>
<tr>
<td>38F9A028</td>
<td>CEEHDSF</td>
<td>06F7C4D0</td>
<td>+00002B4C</td>
<td>CEEHDSF</td>
<td>06F7C4D0</td>
<td>+00002B4C</td>
</tr>
<tr>
<td>36F990A0</td>
<td>CEEHNYUH</td>
<td>06F8B010</td>
<td>+00000992</td>
<td>CEEHNYUH</td>
<td>06F8B010</td>
<td>+00000992</td>
</tr>
<tr>
<td>390261E0</td>
<td>38F39BB0</td>
<td>+000000F2</td>
<td>_NumCompute_evaluate</td>
<td>38F39BB0</td>
<td>+000000F2</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>39027800</td>
<td>33FF07B8</td>
<td>+000004E4</td>
<td>ImbClnode::evaluate(const ImbMessageAssembly*, const ImbData</td>
<td>33FF07B8</td>
<td>+000004E4</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>39028540</td>
<td>201AE2B0</td>
<td>+00000208</td>
<td>ImbDataFlowTerminal::evaluate(const ImbMessageAssembly*)</td>
<td>201AE2B0</td>
<td>+00000208</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>39029220</td>
<td>201ABD70</td>
<td>+00000552</td>
<td>ImbDataFlowTerminal::propagateInner(const ImbMessageAssembly</td>
<td>201ABD70</td>
<td>+00000552</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>39029860</td>
<td>32AC4378</td>
<td>+00000C2E</td>
<td>ImbCommonInputNode::run(ImbOsThread*)</td>
<td>32AC4378</td>
<td>+00000C2E</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>3902B400</td>
<td>32AD3488</td>
<td>+00000046</td>
<td>ImbCommonInputNode::Parameters::run(ImbOsThread*)</td>
<td>32AD3488</td>
<td>+00000046</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>3902B8A0</td>
<td>1DE7FD98</td>
<td>+00000074</td>
<td>ImbThreadPoolThreadFunction::run(ImbOsThread*)</td>
<td>1DE7FD98</td>
<td>+00000074</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>3902C000</td>
<td>1E0A2E80</td>
<td>+00000A8A</td>
<td>ImbOsThread::initializeBootstrapWrapper(void*)</td>
<td>1E0A2E80</td>
<td>+00000A8A</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>3902CD20</td>
<td>1E109E80</td>
<td>+0000025A</td>
<td>ImbOsThread::threadBootstrapWrapper(void*)</td>
<td>1E109E80</td>
<td>+0000025A</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>3902D6A0</td>
<td>1E109E80</td>
<td>+00000000</td>
<td>threadBootstrapWrapper</td>
<td>1E109E80</td>
<td>+00000000</td>
<td>*PATHNAM</td>
</tr>
<tr>
<td>3902D720</td>
<td>0707B2E0</td>
<td>+00001252</td>
<td>CEEVRONU</td>
<td>0707B338</td>
<td>+000011FA</td>
<td>CEEPLPKA</td>
</tr>
<tr>
<td>36F9AE00</td>
<td>CEEOPCM</td>
<td>00035430</td>
<td>+00000908</td>
<td>CEEOPCMM</td>
<td>00035430</td>
<td>+00000908</td>
</tr>
</tbody>
</table>

- 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 lil.

Complete your sessions evaluation online at SHARE.org/AnaheimEval
DFDL Tooling

Simple Types

A simple type defines the allowed values for one or more simple elements.

<table>
<thead>
<tr>
<th>Name</th>
<th>Base Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICX_string</td>
<td>string</td>
</tr>
<tr>
<td>PIC9-Display-Zoned_decimal</td>
<td>decimal</td>
</tr>
</tbody>
</table>
Agenda

- WMB Recap
- External Components
- Diagnostic Information
- How to diagnose common scenarios
Fundamentals of WMB diagnosis

- Problems often occur on busy production systems
- Post mortem information is often incomplete or unavailable
- Good fundamentals are therefore essentials

- Review Symptoms
- Form Hypothesis which explains the problem
- Target collection of diagnostics towards confirming or refuting that hypothesis

- Be aware of impact of various diagnostic tools when using them on a production system
- Where possible perform diagnosis in lower environments
Scenario: WMB won’t start

- First check the JOBLOG, Syslog or Event Viewer

  +BIP8873I MQ91BRK 0 Starting the component verification for component 'MQ91BRK': ImbComponentVerification(78)
  +BIP8875W MQ91BRK 0 The component verification for 'MQ91BRK' has finished, but one or more checks failed: ImbComponentVerification(187)

  - On z/OS check STDOUT/STDERR for MQSICVP
  - On distributed check output of mqsistart and syslog / event viewer

  BIP8873I: Starting the component verification for component 'MQ91BRK'.
  BIP8876I: Starting the environment verification for component 'MQ91BRK'.
  BIP8894I: Verification passed for 'Registry'.
  BIP8894I: Verification passed for 'MQSI_REGISTRY'.
  BIP8894I: Verification passed for 'MQSI_COMPONENT_NAME'.
  BIP8894I: Verification passed for 'MQSI_FILEPATH'.
  BIP8900I: Verification passed for APF Authorization of file '/u/wmqi91/broker/instpath/bin/bipimain'.
  BIP8894I: Verification passed for 'Current Working Directory'.
  BIP8877W: The environment verification for component 'MQ91BRK' has finished, but one or more checks failed.
  One or more of the environment verification checks failed.
  Check the error log for preceding error messages.

  BIP8882I: Starting the WebSphere MQ verification for component 'MQ91BRK'.
  BIP8886I: Verification passed for queue 'SYSTEM.BROKER.ADMIN.QUEUE' on queue manager 'MQ91'.
  BIP8886I: Verification passed for queue 'SYSTEM.BROKER.EXECUTIONGROUP.QUEUE' on queue manager 'MQ91'.
  BIP8886I: Verification passed for queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager 'MQ91'.
  BIP8884I: The WebSphere MQ verification for component 'MQ91BRK' has finished successfully.
  BIP8875W: The component verification for 'MQ91BRK' has finished, but one or more checks failed.
  One or more of the component verification checks failed.
  Check the error log for preceding error messages.
Scenario: Deploy of a Message flow fails

Command line utilities

Message Broker Explorer

Message Broker Toolkit

CMP Applications

Runtime

import com.ibm.broker.config.proxy.*;

public class DeployBAR {
    public static void main(String[] args) {
        BrokerConnectionParameters bcp =
            new MQBrokerConnectionParameters("localhost", 2414,
            "MB7QMGR");

        try {
            BrokerProxy b = BrokerProxy.getInstance(bcp);
            ExecutionGroupProxy eg = b.getExecutionGroupByName("default");

            DeployResult dr = eg.deploy("MyBAR.bar", true, 30000);

            System.out.println("Result = "+dr.getCompletionCode());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
Scenario: 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 of the messages.

- In this scenario the Java compute node cannot find it’s class.
- Has the relevant jar file been deployed or made available in the shared_classes directory?

Details:

BIP4157E: The user-defined node 'Java Compute' could not be deployed.
Details: java.lang.ClassNotFoundException: DebugFlow1_JavaCompute

BIP4395E: Java exception: 'java.lang.ClassNotFoundException'; thrown from class name: 'java.net.URLClassLoader', method name: 'findClass', file: 'URLClassLoader.java', line: '423'

The task was unsuccessful: The deployment was unsuccessful. Check error messages above for explanation.
Scenario: Where’s my 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

  ![Diagram](MQ Input → Compute → Java Compute → MQ Output)

  - A simple MQ In/Out flow with some transformation logic
Scenario: Where’s my message?

- Resource statistics
  - Is there a resource stat available for your output transport that would show if messages are being written?
    - e.g. CICS, CORBA, FTP, File, HTTP Sockets & TCPIP Nodes are available

  ![Graph showing resource statistics]

- No writes are occurring
- So no output messages are being written
Scenario: Where’s my 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.
Scenario: Where’s my message?

- User Trace (also consider Activity Log on WMB V8)
- 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?

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

- Message Flow Debugger
  - Once in the Java source we can step through the code to understand why propagate is never called

```java
MbElement totalElement = orderElement.getFirstElementByPath("Total");

BigDecimal total = (BigDecimal)totalElement.getValue();

if (total.intValue() > 50)
{
    // do nothing, it's not cheap enough
} else {
    // send the message out
    out.propagate[assembly];
}
```

- 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

Complete your sessions evaluation online at SHARE.org
Scenario: Who’s misbehaving

- WMB is often deployed in complex environments
- It’s not always obvious what product is malfunctioning
- Example:
  - Web Services clients are reporting that they are receiving unexpected fault messages
  - WMB is providing a service façade to another Web Service
Scenario: Who’s misbehaving

- Examine the fault content

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenv:Body="soapenv:Body"
    xmlns:soapenv:Fault="soapenv:Fault"
    FaultCode="soapenv:Server">"BIP3113E: Exception detected in message flow webServicesFacade.SOAP Input (broker BRK8)</FaultString>
    <detail>
      <Text>BIP3101E: A Java exception was thrown whilst calling the Java JNI method 'Axis2Requester_processResponseMessageSync'. The Java exception was 'java.lang.NoClassDefFoundError: org.apache.axiom.om.impl.builder.StAXOMBuilder_next(StAXOMBuilder.java:252)' with root cause of 'java.lang.ClassNotFoundException: org.apache.axiom.om.impl.builder.StAXOMBuilder_next'.
      Frame 2: com.ibm.axis.xmp2.api.stax.xml.StAXMessageProvider.throwWrappedXMLStreamException(StAXMessageProvider.java:76)
      Frame 4: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.produceFatalErrorEvent(WebSocketXMLStreamReaderImpl.java:316)
      Frame 5: com.ibm.axis.xmp2.xml.XMLScanner.reportFatalError(DOMXMLScanner.java:4942)
      Frame 7: com.ibm.axis.xmp2.xml.XMLScanner.reportFatalError(DOMXMLScanner.java:2383)
      Frame 8: com.ibm.axis.xmp2.xml.XMLScanner.reportFatalError(DOMXMLScanner.java:2299)
      Frame 10: com.ibm.axis.xmp2.xml.XMLScanner.reportFatalError(DOMXMLScanner.java:1834)
      Frame 11: com.ibm.axis.xmp2.runtime.XMLContext.reportError(XMLContext.java:300)
      Frame 12: com.ibm.axis.xmp2.xml.XMLScanner.nextEvent(DOMXMLScanner.java:1276)
      Frame 14: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.next(XMLStreamReaderImpl.java:183)
      Frame 15: com.ibm.axis.xmp2.api.wsssec.WSSXMLInputFactoryImpl.WSSXMLStreamReaderProxy.next(WSSXMLInputFactoryIMPL.java:55)
      Frame 16: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.nextStAXOMBuilder_next(StAXOMBuilder.java:622)
      Frame 17: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.nextStAXOMBuilder_next(StAXOMBuilder.java:132)
      Frame 18: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.nextStAXOMBuilder_next(StAXOMBuilder.java:664)
      Frame 19: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.nextStAXOMBuilder_next(StAXOMBuilder.java:681)
      Frame 20: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.nextStAXOMBuilder_next(StAXOMBuilder.java:1004)
      Frame 21: com.ibm.axis.xmp2.api.stax.xml.XMLStreamReaderImpl.nextStAXOMBuilder_next(StAXOMBuilder.java:681)
      Correct the error, and if necessary redeploy the flow. File: F:\build\800_D\src\webServicesFacade\Library\ImplSOAPRequestHelper.cpp: 2129: ImplSOAPRequestHelper.
    </detail>
</soapenv:Fault>
</soapenv:Body>
</soapenv:Envelope>
```

- Error parsing the response
Scenario: Who’s misbehaving

- User Trace
  - What did the remote server return?

  See previous error messages for an indication to the cause of the errors.

  RecoverableException: BIP362S: An HTTP error occurred. The HTTP Request-Line was: "POST/acmeOrders/WADDR." The HTTP Request Header bitstream (if any) to be used was: 'X436f6e74656e742d4c656e6774686123434300a43'. The HTTP Request Message Body bitstream (if any) to be used was: 'X3c756f6170656e763456e76656c6f70652070'. The HTTP Reply Header bitstream (if any) received from the server was: 'X485454502f312e312032303020485458c'. The HTTP Reply Message Body bitstream (if any) received from the server was: 'X3c756f6170656e763456e76656c6f70652070'. Ensure that the HTTP data is valid.

  See the following messages for information pertaining to this error.

  RecoverableException: BIP3701E: A Java exception was thrown whilst calling the Java JNI method "'Axis2Reque.
  @: org.apache.axis2.om.impl.builder.StAXOMBuilder.next(StAXOMBuilder.java:252)
  @: org.apache.axis2.om.impl.dom.OMElementImpl.buildNext(OMElementImpl.java:664)
  @: org.apache.axis2.om.impl.dom.OMElementImpl.getFirstOMChild(OMElementImpl.java:681)
  @: org.apache.axis2.om.impl.dom.OMElementImpl.getFirstOMChild(OMElementImpl.java:1006)

  - BIP3633 shows the bitstream sent to and received from the remote server
  - Use a hex to ascii converter to see what was received
Scenario: Who’s misbehaving

- ASCII version of bitstream

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope"
 xmlns:ord="http://www.acmeOrders.com/OrderService">
  <soapenv:Header>
  </soapenv:Header>
  <soapenv:Body>
    <ord:submitPOResponse>
      <orderStatus>AVAILABLE</orderStatus>
      <orderAmt>50</orderAmt>
      <partNo>0123456</partNo>
      <partQuantity>20</partQuantity>
    </ord:submitPOResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

- Remote server has sent invalid data
Summary

- WMB Recap
- External Components
- Diagnostic Information
- How to diagnose common scenarios
<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>Free MQ! - MQ Clients and what you can do with them</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:30</td>
<td>Clustering – the easier way to connect your Queue Managers</td>
<td>MQ on z/OS – vivisection</td>
<td>The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>interpretation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:00</td>
<td>Diagnosing problems for Message Broker</td>
<td>Lock it down - WebSphere MQ Security</td>
<td></td>
<td>Using IBM WebSphere Application Server and IBM WebSphere MQ Together</td>
<td>Spreading the message – MQ pubsub</td>
</tr>
<tr>
<td>12:15</td>
<td>Highly Available Messaging - Rock solid MQ</td>
<td>Putting the web into WebSphere MQ: A look at Web 2.0 technologies</td>
<td>The Doctor is In and Lots of Help with the MQ family - Hands-on Lab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01:30</td>
<td>WebSphere MQ 101: Introduction to the world's leading messaging provider</td>
<td>What's new in the WebSphere MQ Product Family</td>
<td>Extending IBM WebSphere MQ and WebSphere Message Broker to the Cloud</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03:00</td>
<td>First steps with WebSphere Message Broker: Application integration for the messy</td>
<td>What's new in Message Broker V8.0</td>
<td>Under the hood of Message Broker on z/OS - WLM, SMF and more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04:30</td>
<td>The MQ API for Dummies - the Basics</td>
<td>What the **** is going on in my Queue Manager!?</td>
<td>Diagnosing problems for MQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>06:00</td>
<td></td>
<td></td>
<td>For your eyes only - WebSphere MQ Advanced Message Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MQ Q-Box - Open Microphone to ask the experts questions</td>
</tr>
</tbody>
</table>
Copyright and Trademarks

© IBM Corporation 2012. All rights reserved. IBM, the IBM logo, ibm.com and the globe design are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml. Other company, product, or service names may be trademarks or service marks of others.
QR

Complete your sessions evaluation online at SHARE.org/AnaheimEval