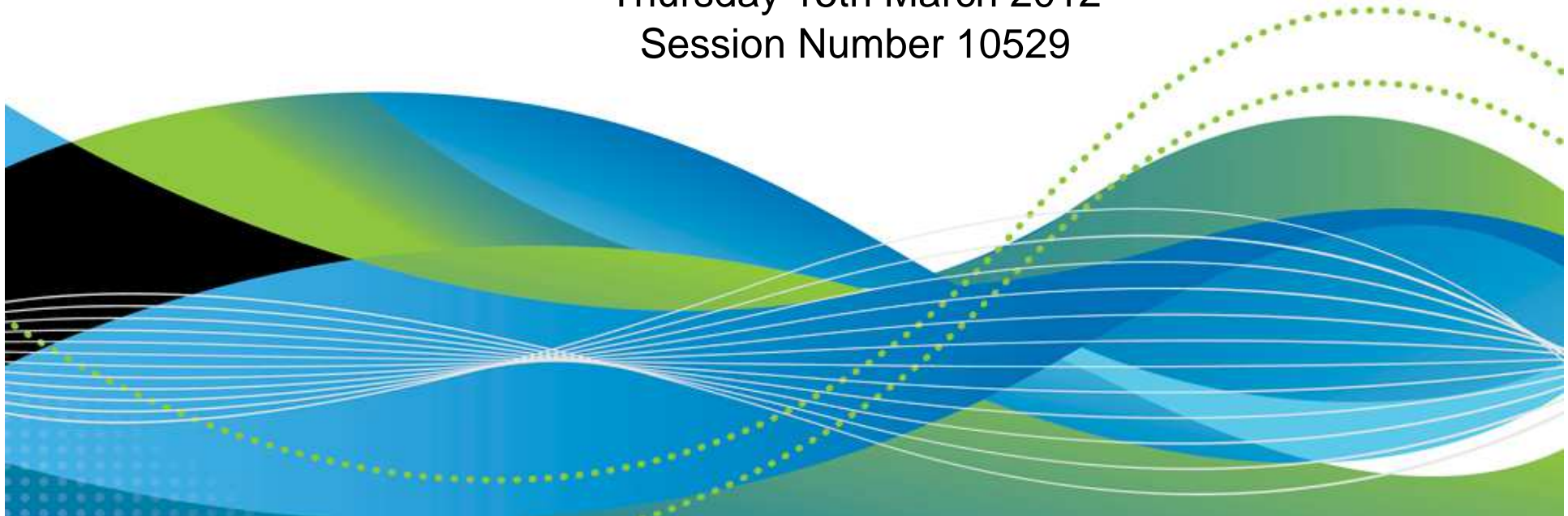


# You think you have problems...well maybe you do. Diagnosing problems for Message Broker

David Coles – WebSphere Message Broker Level 3 Technical Lead,  
IBM Hursley – [dcoles@uk.ibm.com](mailto:dcoles@uk.ibm.com)

Thursday 15th March 2012  
Session Number 10529



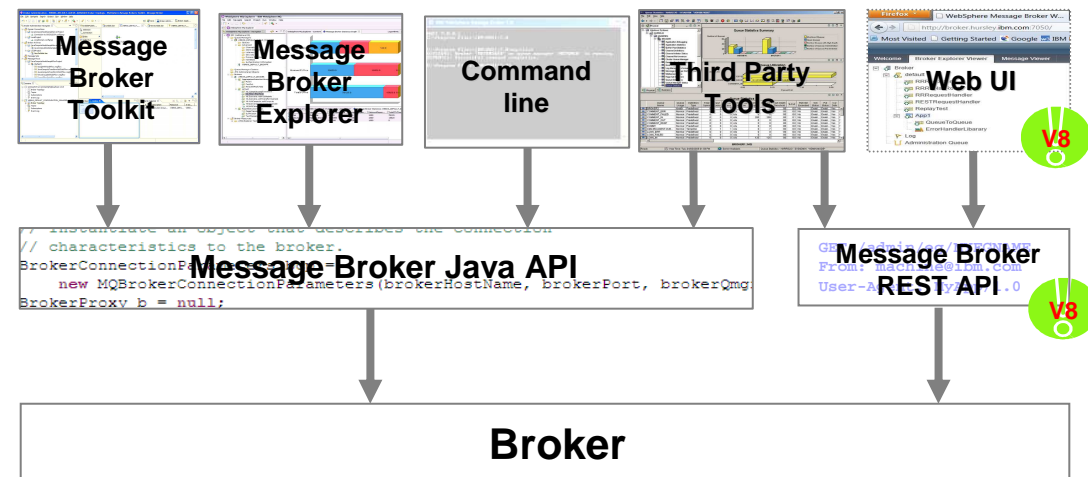
# Agenda

- Diagnostic Information in WMB
- The “moving” parts of a z/OS Broker
- System and Product Logs
- What Trace to use
  - How to understand a User Trace
- What do I do with a Dump / FFDC ?
- What are the common “Status” commands
- “Out the box” Tools available for debugging
  - Debugger
  - Message Flow and Resource Statistics
  - Activity Log
  - Message Tracking
  - Administration Log
- How to Diagnose Common Scenarios
  - My Broker won’t start
  - My Flow won’t deploy and my EG fails
  - Where’s my output message?

# Diagnostic Information in WMB

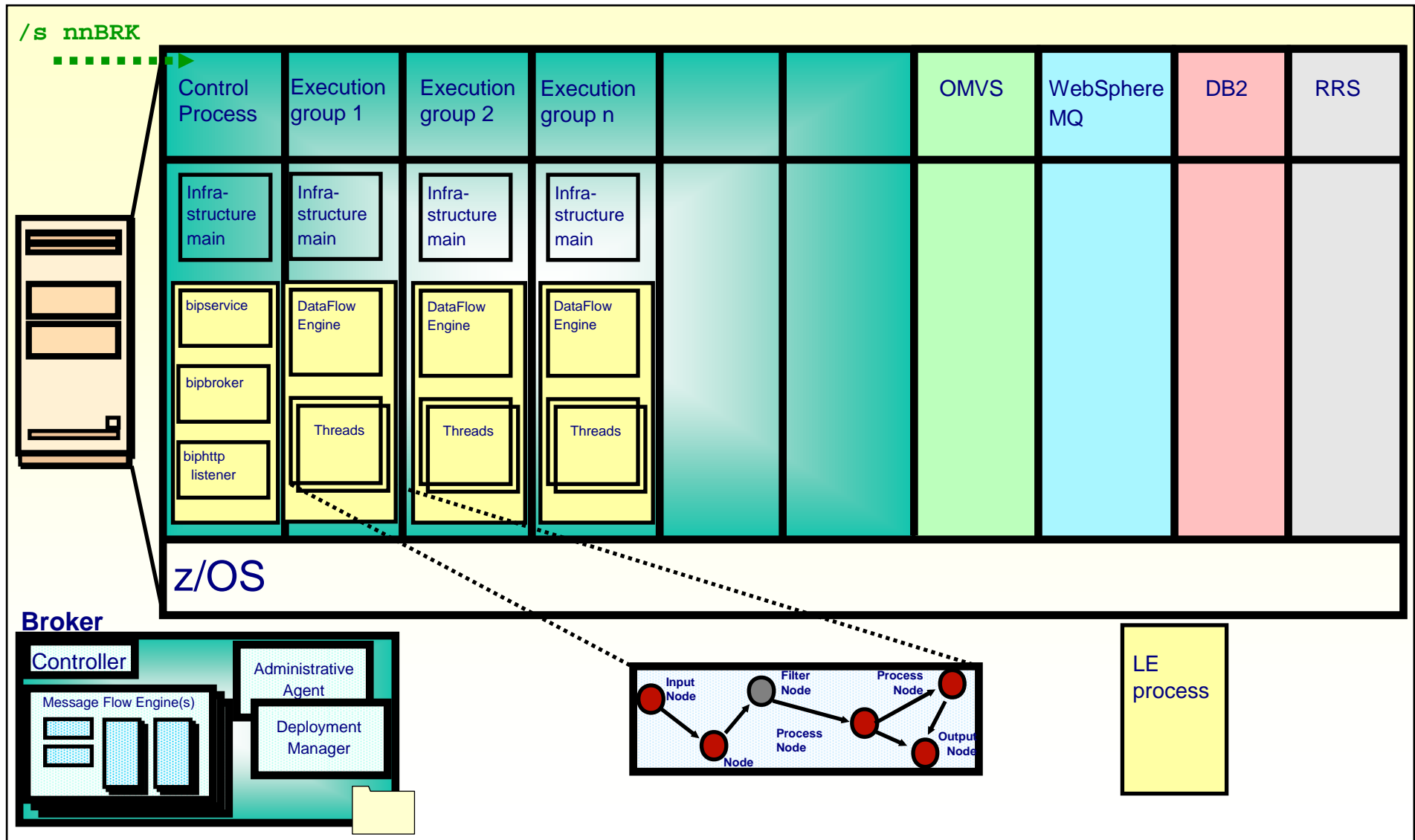
- Diagnostic Information is available from a multitude of sources

- System Log
- Stdout/Stderr
- Trace
- Dumps/FFDC
- Commands
- Debugger
- Message Flow Statistics
- Resource Statistics
- Activity Log
- Message Tracking
- Administration Log



- Problem Diagnosis often requires you to coordinate different pieces of evidence from different places

# The “moving” parts of a z/OS Broker



# Notes: 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 address space 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 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/CMP API and, by extension, WMB Toolkit & Explorer. It manages the deployment of message flows and message flow artifacts, and manages the lifecycle and command reporting of execution groups (EG). When using WebServices nodes with the broker-wde listener then the http listener process runs in this address space.
- Execution Group address spaces. These are where the message flows which have been deployed 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.

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

- 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 WMB uses it for inter-process and inter-platform communication. For example, the AA communicates with the EGs and Toolkit/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.

- **z/OS, Language Environment and Java provide many services.**

- 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).

# System and Product Logs

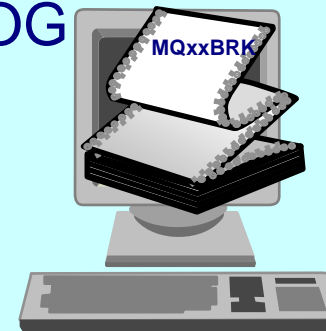


*As you'd expect from a regular z/OS subsystem, the operational output will be available from the MVS SYSLOG and JOBLOGs in the form of BIP product messages.*

*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*

## SYSLOG JOBLOG



```
SDSF JOB DATA SET DISPLAY - JOB MQ91BRK (STC04580)
COMMAND INPUT ==>
PREFIX=MQ*  DEST=(ALL)  OWNER=*  SYSNAME=*
NP  DDNAME      StepName  ProcStep  DSID  Owner      C  Dest
   JESMSG LG  JES2        2  WMQI91    C
   JESJCL    JES2        3  WMQI91    C
   JESYSMSG  JES2        4  WMQI91    C
   ENVFILE   MQ91BRK     101 WMQI91    C
   DSNAOINI  MQ91BRK     102 WMQI91    C
   STDOUT    MQ91BRK     103 WMQI91    C
   STDOUT    MQ91BRK     105 WMQI91    C
   STDERR    MQ91BRK     106 WMQI91    C
```



# What Trace to use and when

Trace is available for separate components which can be formatted to a text output. Held in common/log

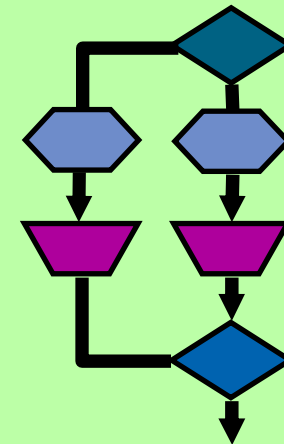
Various trace is available:

- Flow User Trace – for you.
- Internal “Service” Trace – for IBM Support
- Command traces – for all
- CVP trace – for all

## Trace

User

Service



```
Aug 9 15:44 MQ91BRK.c91e9b41-3101-0000-0080-c0fdd5a003a1.trace.bin.0
Aug 9 17:28 MQ91BRK.c91e9b41-3101-0000-0080-c0fdd5a003a1.trace.bin.1
Aug 9 15:44 MQ91BRK.c91e9b41-3101-0000-0080-c0fdd5a003a1.trace.bin.2
Aug 9 15:44 MQ91BRK.c91e9b41-3101-0000-0080-c0fdd5a003a1.trace.bin.3
Aug 9 16:10 MQ91BRK.c91e9b41-3101-0000-0080-c0fdd5a003a1.userTrace.bin.0
Aug 9 17:28 MQ91BRK.c91e9b41-3101-0000-0080-c0fdd5a003a1.userTrace.bin.1
Aug 9 17:28 MQ91BRK.httpListener.trace.bin.0
Aug 9 17:28 MQ91BRK.httpListener.userTrace.bin.0
Aug 9 16:06 MQ91BRK.mqsichangeflowstats.trace.bin.0
Aug 9 16:06 MQ91BRK.mqsichangeflowstats.userTrace.bin.0
Aug 9 15:44 MQ91BRK.mqsichangetrace.trace.bin.0
Aug 9 15:44 MQ91BRK.mqsichangetrace.userTrace.bin.0
Aug 9 15:33 MQ91BRK.mqsicvp.trace.bin.0
Aug 9 17:28 MQ91BRK.mqsicvp.trace.bin.1
Aug 9 17:28 MQ91BRK.mqsicvp.userTrace.bin.0
Aug 9 15:41 MQ91BRK.mqsireadlog.trace.bin.0
Aug 9 15:41 MQ91BRK.mqsireadlog.userTrace.bin.0
Aug 9 16:11 MQ91BRK.mqsireportflowstats.trace.bin.0
Aug 9 16:11 MQ91BRK.mqsireportflowstats.userTrace.bin.0
Aug 9 17:04 MQ91BRK.mqsistop.trace.bin.0
Aug 9 17:04 MQ91BRK.mqsistop.userTrace.bin.0
Aug 9 17:28 MQ91BRK.service.trace.bin.0
Aug 9 17:28 MQ91BRK.service.userTrace.bin.0
```

# 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 21:58:23.160163 6468 UserTrace BIP2537I: Node 'DebugFlow1.Compute': Executing statement "BEGIN ... END;" at ('.DebugFlow1_Compute.Main', '2.2').
2011-08-09 21:58:23.160373 6468 UserTrace BIP2537I: Node 'DebugFlow1.Compute': Executing statement "CopyEntireMessage();" at ('.DebugFlow1_Compute.Main', '3.3').
2011-08-09 21:58:23.160455 6468 UserTrace BIP2538I: Node 'DebugFlow1.Compute': Evaluating expression "CopyEntireMessage();" at ('.DebugFlow1_Compute.Main', '3.8').
2011-08-09 21:58:23.160533 6468 UserTrace BIP2537I: Node 'DebugFlow1.Compute': Executing statement "BEGIN ... END;" at ('.DebugFlow1_Compute.CopyEntireMessage', '1.39').
2011-08-09 21:58:23.160598 6468 UserTrace BIP2537I: Node 'DebugFlow1.Compute': Executing statement "SET OutputRoot = InputRoot;" at
('DebugFlow1_Compute.CopyEntireMessage', '2.3').
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 21:58:23.160905 6468 UserTrace BIP2568I: Node 'DebugFlow1.Compute': Copying sub-tree from "InputRoot" to "OutputRoot".
2011-08-09 21:58:23.161085 6468 UserTrace BIP2537I: Node 'DebugFlow1.Compute': Executing statement "SET OutputRoot.XMLNSC.Order.Total =
CAST(OutputRoot.XMLNSC.Order.Item.Price AS DECIMAL) * CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS INTEGER);" at ('.DebugFlow1_Compute.Main', '4.3').
2011-08-09 21:58:23.161277 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "OutputRoot.XMLNSC.Order.Item.Price" at ('.DebugFlow1_Compute.Main',
'4.44'). This resolved to "OutputRoot.XMLNSC.Order.Item.Price". The result was "3".
2011-08-09 21:58:23.161457 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.Item.Price AS DECIMAL)" at
('DebugFlow1_Compute.Main', '4.39'). This resolved to "CAST('3' AS DECIMAL)". The result was "3".
2011-08-09 21:58:23.161539 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "OutputRoot.XMLNSC.Order.Item.Quantity" at ('.DebugFlow1_Compute.Main',
'4.98'). This resolved to "OutputRoot.XMLNSC.Order.Item.Quantity". The result was "".
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('7' AS INTEGER)". The result was "7".
2011-08-09 21:58:23.161767 6468 UserTrace BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.Item.Price AS DECIMAL) *
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 21:58:23.161844 6468 UserTrace BIP2566I: Node 'DebugFlow1.Compute': Assigning value "21" to field / variable "OutputRoot.XMLNSC.Order.Total".
2011-08-09 21:58:23.161916 6468 UserTrace BIP2537I: Node 'DebugFlow1.Compute': Executing statement "RETURN TRUE;" at ('.DebugFlow1_Compute.Main', '5.3').
2011-08-09 21:58:23.162040 6468 UserTrace BIP4015I: Message propagated to the 'out' terminal of node 'DebugFlow1.Compute' with the following message trees: ".
2011-08-09 21:58:23.162214 6468 UserTrace BIP3904I: Invoking the evaluate() method of node (class='Com.ibm.java.compute.Node', name='DebugFlow1#FCMComposite_1_4').
    About to pass a message to the evaluate() method of the specified node.
    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
```



# Notes: 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 for 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
    - <Broker\_HFS\workpath>/common/errors/CEEDUMP.\*
- Trace files
  - Trace files are always written to the zFS, so make sure you have enough space in the mount.
    - <Broker\_HFS\workpath>/common/log/\*
  - Trace files should be formatted on the customer machine using BIPxxxxx JCL in component PDS.

# Notes: stdout/stderr

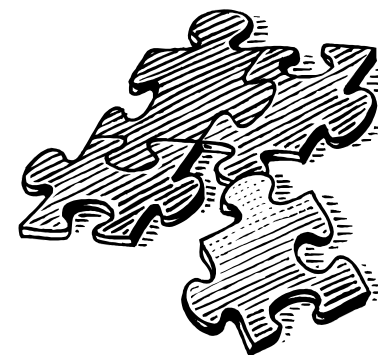
- Useful place to look for errors / debugging
  - Always worth checking for exceptions if problems are occurring
- Each major component redirects its stdout/stderr streams to files
  - Windows
    - *Admin Agent (7.0.0.2)*  
C:\Documents and Settings\All Users\Application Data\IBM\MQSI\components\<brkName>\console.txt
    - *Execution group*  
C:\Documents and Settings\All Users\Application Data\IBM\MQSI\components\<brkName>\<egUUID>\console.txt
  - Linux/Unix
    - *Admin Agent (7.0.0.2)*  
/var/mqsi/components/<brkName>/stdout & stderr
    - *Execution group*  
/var/mqsi/components/<brkName>/<egUUID>/stdout & 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

```

Traceback:
 DSA Addr  Program Unit  PU Addr  PU Offset  Entry      E Addr  E Offset  Statement  Load Mod  Service  Status
38F9DBD0  CEEVRONU    0707D2B8 +00001004 CEEVRONU   0707D2B8 +00001004 CEEPLPKA  HLE7730  Call
390253A0      1DF418F8 +000000DE ImbAbend::printStatsForCurrentThread(int,bool,const void*,vo
                        1DF418F8 +000000DE *PATHNAM  FP2....  Call
39025780      1E221258 +000003C2 ImbAbend::terminateProcessInternal(const void*,const bool,vo
                        1E221258 +000003C2 *PATHNAM  FP2....  Call
39026080      1DF457F8 +000005BE IMBCOND    1DF457F8 +000005BE *PATHNAM  FP2....  Call
39026120      0707B2E0 +00001252 CEEVROND   0707B338 +000011FA CEEPLPKA  HLE7730  Call
38F9A928  CEEHDSP     06F7C4D0 +000024BC CEEHDSP    06F7C4D0 +000024BC CEEPLPKA  HLE7730  Call
38F99DA8  CEEHRNUH    06F8B010 +00000092 CEEHRNUH   06F8B010 +00000092 CEEPLPKA  HLE7730  Call
390261E0      38F39BB0 +000000F2 _NumCompute_evaluate
                        38F39BB0 +000000F2 *PATHNAM                                Exception
39027B00      33EFF078 +000004E4 ImbCniNode::evaluate(const ImbMessageAssembly&,const ImbData
                        33EFF078 +000004E4 *PATHNAM  FP2....  Call
39028840      201AE2B0 +00000208 ImbDataFlowTerminal::evaluate(const ImbMessageAssembly&)
                        201AE2B0 +00000208 *PATHNAM  FP2....  Call
39028920      201AE078 +000000BE ImbDataFlowTerminal::propagateInner(const ImbMessageAssembly
                        201AE078 +000000BE *PATHNAM  FP2....  Call
39029220      201ABD70 +00000552 ImbDataFlowTerminal::propagate(const ImbMessageAssembly&)
                        201ABD70 +00000552 *PATHNAM  FP2....  Call
39029360      32AC4878 +00003C2E ImbCommonInputNode::run(ImbOsThread*)
                        32AC4878 +00003C2E *PATHNAM  FP2....  Call
3902BA00      32AD3488 +00000046 ImbCommonInputNode::Parameters::run(ImbOsThread*)
                        32AD3488 +00000046 *PATHNAM  FP2....  Call
3902BA80      1DE7FD98 +00000074 ImbThreadPoolThreadFunction::run(ImbOsThread*)
                        1DE7FD98 +00000074 *PATHNAM  FP2....  Call
3902C400      1E10A2E8 +000000A8 ImbOsThread::innerThreadBootstrapWrapper(void*)
                        1E10A2E8 +000000A8 *PATHNAM  FP2....  Call
3902CD20      1E109E80 +0000025A ImbOsThread::threadBootstrap(void*)
                        1E109E80 +0000025A *PATHNAM  FP2....  Call
3902D6A0      1E109E38 +00000008 threadBootstrapWrapper
                        1E109E38 +00000008 *PATHNAM  FP2....  Call
3902D720      0707B2E0 +00001252 CEEVROND   0707B338 +000011FA CEEPLPKA  HLE7730  Call
38FAAEE0  CEEOPCMM    00035438 +00000908 CEEOPCMM   00035438 +00000908 CEEBINIT  HLE7730  Call
    
```

- 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.

# Notes: 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 (7.0.0.3/8.0.0.0)**
  - 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'.



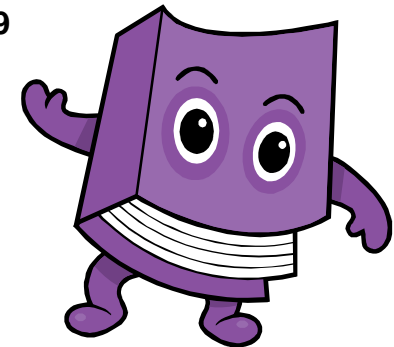
# What level is my broker?



- On z/OS look at the JOBLLOG ☺

BIP9272I 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 '16909047'; THREAD  
'2007049624853938176'; ADDITIONAL INFORMATION: BROKERNAME 'MQ91BRK'  
(OPERATION MODE enterprise); EXECUTIONGROUPUUID '93d22eb0-3101-0000-0080-  
c3ce9c5af203'; EXECUTIONGROUPLABEL 'RALPH'; QUEUEMANAGERNAME '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)

BIP9108I 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

BIPmsgs en\_GB

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: 7003



BIP8997I: Product: WebSphere Message Broker

BIP8998I: CMVC Level: S700-L110525

BIP8999I: Build Type: Production, 32 bit, x86\_nt\_4

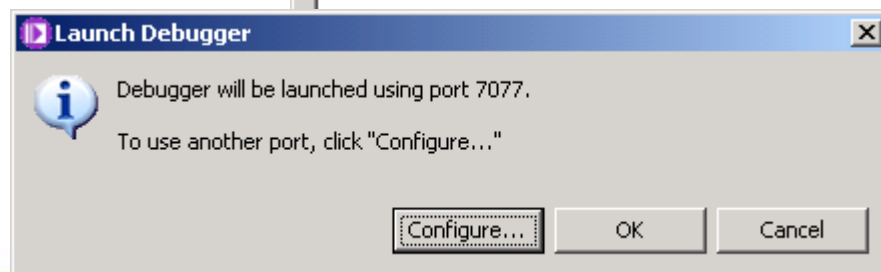
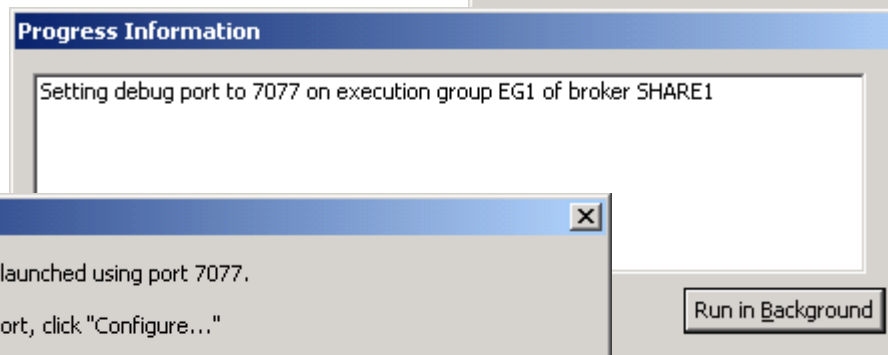
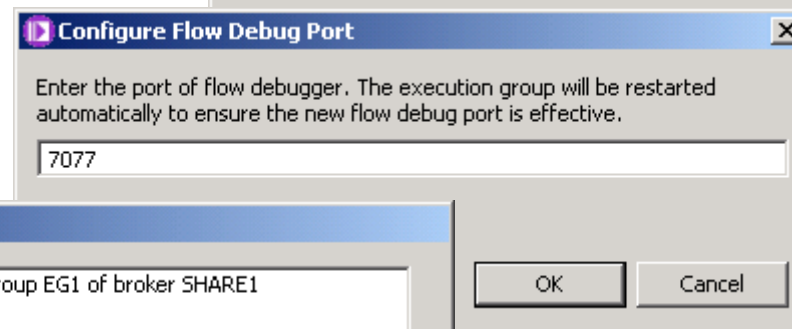
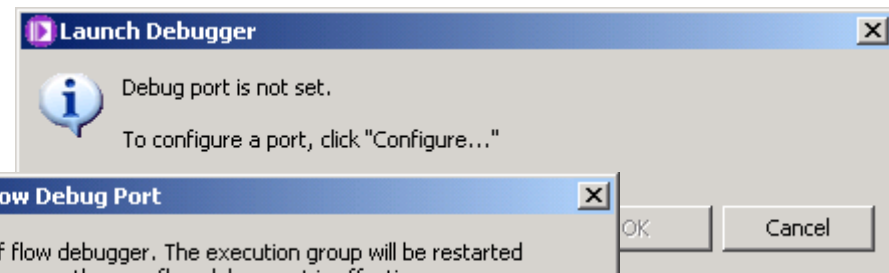
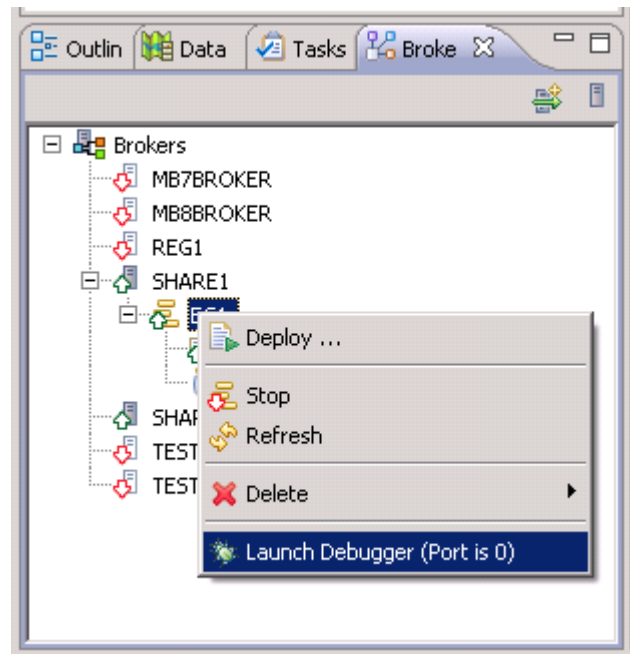
BIP8071I: Successful command completion

# What “out the box” Tools are available?

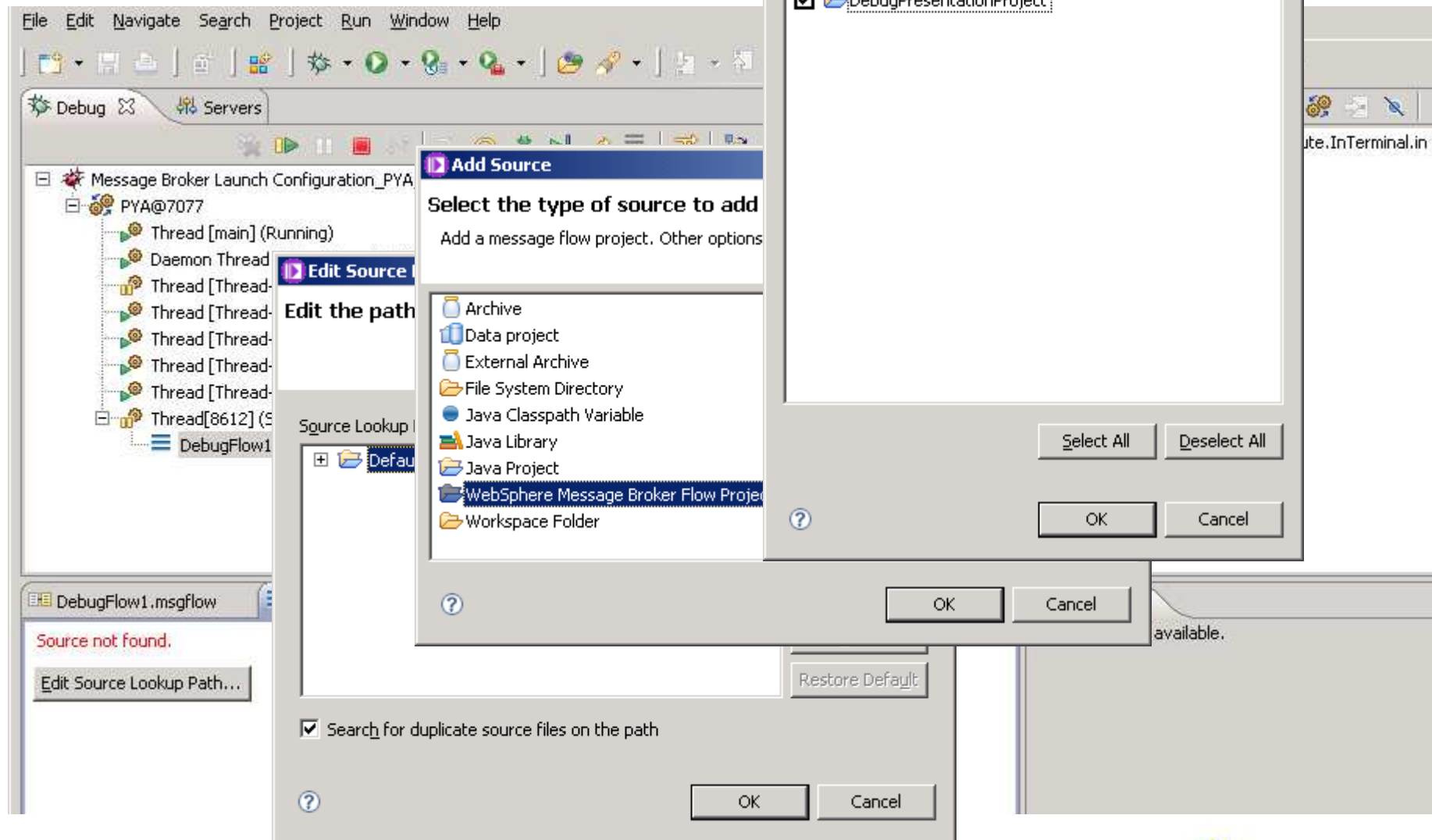
- Flow debugger
  - Useful for debugging message flows during their development
- Flow statistics
  - Useful for highlighting the cause of performance problems
  - Baseline good operation
- Resource stats
  - Useful for highlighting areas of concern in message flows or performance bottlenecks.
- Activity Log 
  - Understand what a message flow is doing at a high level
- Message Tracking 
  - Record, Edit and Replay of In-flight Data
- Administration Log
  - View the results of administration actions & who initiated them

# Message Flow Debugger

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

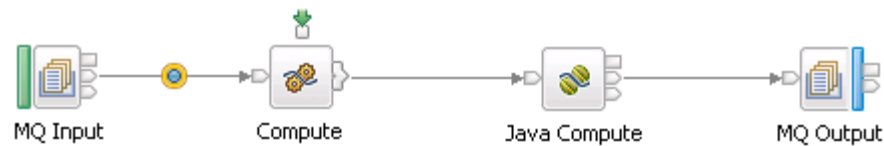


# Message Flow Debugger



**HOW TO:** Configure the Source lookup path to enable you to step through you message flow application

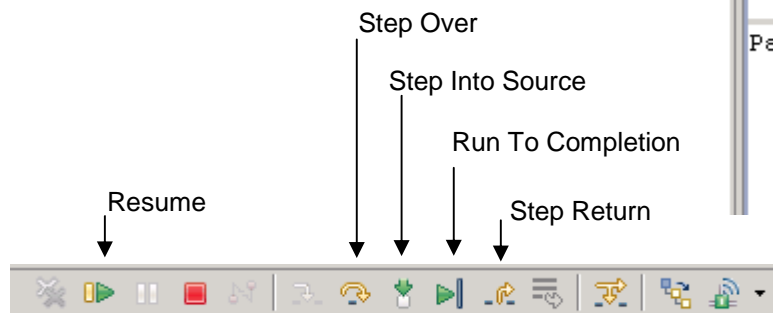
# Message Flow Debugger



Variables Breakpoints

Name	Value
Message	
Properties	
MQMD	
XMLNSC	
Order	
Name	
Item	
PartNo	10001
Price	3
Quantity	77
LocalEnvironment	
Environment	
ExceptionList	

PartNo: CHARACTER: 10001

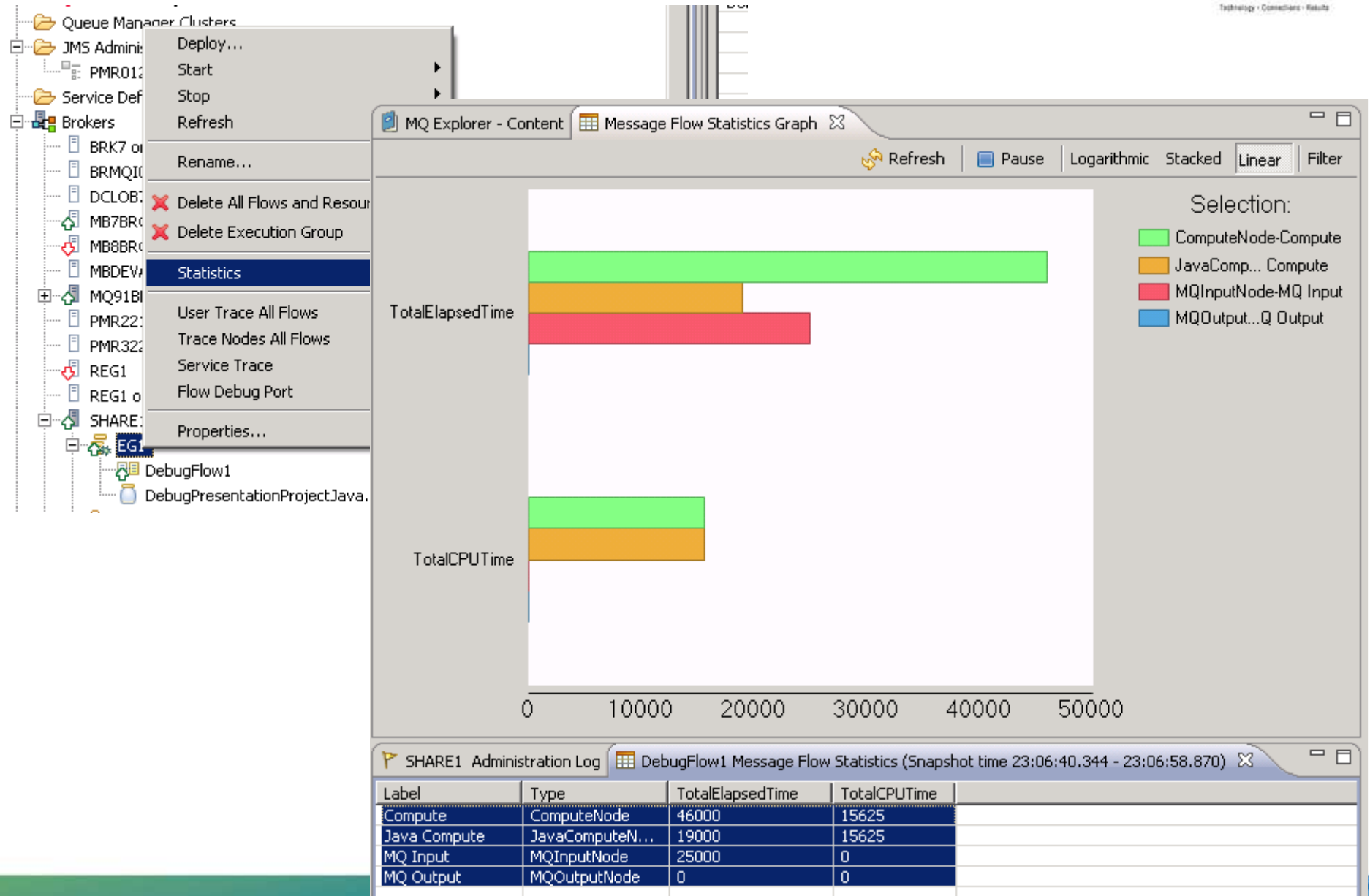


# Notes: Message Flow Debugger

- 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



# Notes: Message Flow Statistics

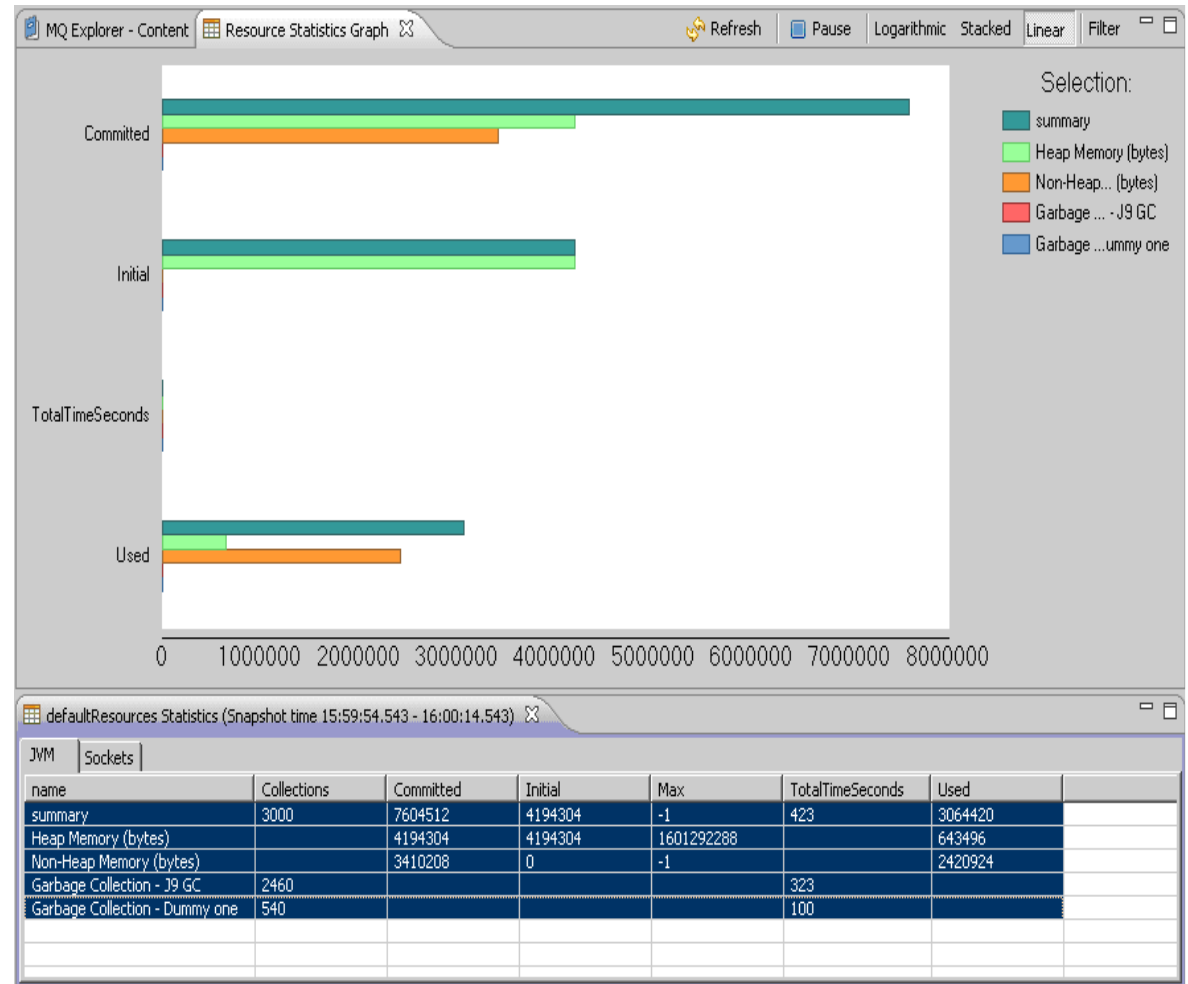


- You can enable and display message flow statistics using MBX.
- You can use the results and graphs to diagnose performance and operational problems.
- During normal operation it can be useful to enable message flow statistics so you can get a feel for what 'normal' looks like as well as to help tuning 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



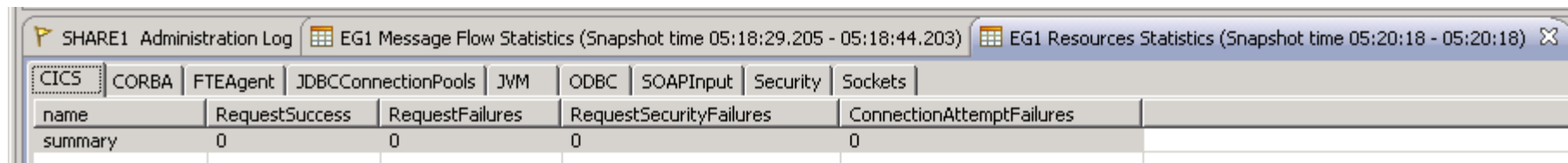
- Find out the current resource usage of a broker or execution group
  - CICS** – successful requests, failures, security failures...
  - CORBA** – Invocations, Success, Failures
  - FTE** – Inbound/Outbound transfers, bytes sent/received...
  - JDBC** – Requests, Cached requests, Providers...
  - JVM** – 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



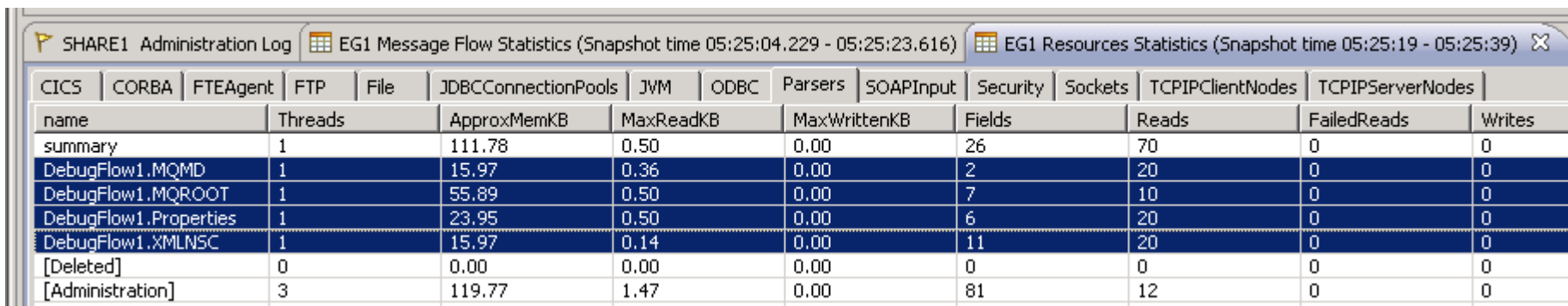
- Each resource reports values specific to the given resource type
- Failure counts are often key values to monitor



SHARE1 Administration Log | EG1 Message Flow Statistics (Snapshot time 05:18:29.205 - 05:18:44.203) | EG1 Resources Statistics (Snapshot time 05:20:18 - 05:20:18) X

CICS	CORBA	FTEAgent	JDBCConnectionPools	JVM	ODBC	SOAPInput	Security	Sockets
name	RequestSuccess	RequestFailures	RequestSecurityFailures	ConnectionAttemptFailures				
summary	0	0	0	0				

- Parser stats provide a great insight to a given flow



SHARE1 Administration Log | EG1 Message Flow Statistics (Snapshot time 05:25:04.229 - 05:25:23.616) | EG1 Resources Statistics (Snapshot time 05:25:19 - 05:25:39) X

CICS	CORBA	FTEAgent	FTP	File	JDBCConnectionPools	JVM	ODBC	Parsers	SOAPInput	Security	Sockets	TCIPClientNodes	TCIPServerNodes
name	Threads	ApproxMemKB	MaxReadKB	MaxWrittenKB	Fields	Reads	FailedReads	Writes					
summary	1	111.78	0.50	0.00	26	70	0	0					
DebugFlow1.MQMD	1	15.97	0.36	0.00	2	20	0	0					
DebugFlow1.MQROOT	1	55.89	0.50	0.00	7	10	0	0					
DebugFlow1.Properties	1	23.95	0.50	0.00	6	20	0	0					
DebugFlow1.XMLNSC	1	15.97	0.14	0.00	11	20	0	0					
[Deleted]	0	0.00	0.00	0.00	0	0	0	0					
[Administration]	3	119.77	1.47	0.00	81	12	0	0					

# Notes: Resource Statistics



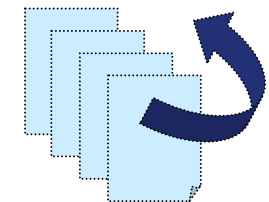
- Every resource is different, so each resource reports different values.
- In the same way as with message flow statistics, it's a good idea to enable resource statistics when things are going well so that you can get a feeling, or some concrete numbers for what to expect.
- The key values to monitor 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.

# Activity Log



- New Activity Logging Allows users to understand what a message flow is doing
  - Complements current extensive product trace by providing end-user oriented trace
  - Can be used by developers, but target is operators and administrators
  - Doesn't require detailed product knowledge to understand behaviour
  - Provides qualitative measure of behaviour
- End-user oriented with external resource lifecycle
  - Focus on easily understood actions & resources
  - "GET message queue X", "Update DB table Z"...
  - Complements quantitative resource statistics
- Flow & resource logging
  - User can observe all events for a given flow
    - e.g. "GET MQ message", "Send IDOC to SAP", "Commit transaction"...
  - Users can focus on individual resource manager if required
    - e.g. SAP connectivity lost, SAP IDOC processed
  - Use event filters to create custom activity log
    - e.g. capture all activity on JMS queue REQ1 and C:D node CDN1
  - Progressive implementation as with resource statistics, starting with JMS and C:D resources
- Comprehensive Reporting Options
  - Reporting via MB Explorer, log files and programmable management (CMP API)
  - Extensive filtering & search options, also includes save data to CSV file for later analysis
- Log Rotation facilities
  - Rotate resource log file when reaches using size or time interval

Message...	Timestamp	Message Summary
i BIP12001I	17-Jun-2011 10:10:50.85...	Connected to JMS provider 'WebSphere_MQ'
i BIP12002I	17-Jun-2011 10:10:50.85...	Created a 'Transaction_None' session for JMS provider 'WebSphere_MQ'
i BIP12004I	17-Jun-2011 10:10:50.93...	Created JMS producer for destination 'ASYNCREQUESTQ'
i BIP12007I	17-Jun-2011 10:10:50.93...	Sent a JMS message to queue 'ASYNCREQUESTQ'
i BIP12004I	17-Jun-2011 10:10:50.52...	Created JMS producer for destination 'ASYNCRECEIVEQ'
x BIP12014E	17-Jun-2011 13:47:51.65...	Failed to send message to 'ASYNCRECEIVEQ'
i BIP12001I	17-Jun-2011 13:47:54.99...	Connected to JMS provider 'WebSphere_MQ'
i BIP12004I	17-Jun-2011 13:47:55.00...	Created JMS producer for destination 'ASYNCRECEIVEQ'





# 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

CD Input Node Properties - CD Input

Configure monitoring events

Events

Enabled	Event Source	Event Source Address
<input checked="" type="checkbox"/>	Transaction start	CD Input.transaction.Start
<input checked="" type="checkbox"/>	Transaction end	CD Input.transaction.End
<input checked="" type="checkbox"/>	Transaction rollback	CD Input.transaction.Rollback

Welcome Broker Explorer Viewer Message Viewer

View Message Replay Message Edit/Replay Message Edit Message View Exception Delete Refresh Filter

Message ID	Correlation ID	Message Exception	Broker	Execution Group	Message Flow
c5743c50-ca64-11e0-ae70-7f0000010000-9		Y N	MB8BROKER	default	QueueToQueue
c5743c50-ca64-11e0-ae70-7f0000010000-9		Y N	MB8BROKER	default	QueueToQueue
c5743c50-ca64-11e0-ae70-7f0000010000-8		Y N	MB8BROKER	default	QueueToQueue

Filter Query

Start Time

End Time

Message ID

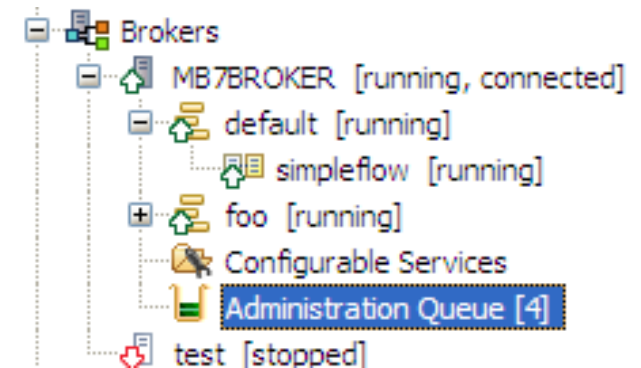
Advanced Filter Query

OK Cancel

# Administration Queue / Log



- 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?



Message	Source	Timestamp	Message Detail
BIP2880I	Change Notification	09-Oct-2009 10:18:30	The property 'ComIbmJVMManager/jvmMaxHeapSize' has changed from '-1' to '102385684' on object 'default' of type 'E
BIP2883I	Change Notification	09-Oct-2009 10:17:13	The resource 'TextMessenger' of type 'MessageFlow' was modified on object 'default' of type 'ExecutionGroup' with pare
BIP2880I	Change Notification	09-Oct-2009 10:17:13	The property 'object.runstate' has changed from 'running' to 'stopped' on object 'TextMessenger' of type 'MessageFlow'
BIP2871I	Administration Result	09-Oct-2009 10:17:12	The request made by user 'Matt' to 'stop' the resource 'TextMessenger' of type 'MessageFlow' on parent 'default' of typ
BIP2271I	Administration Request	09-Oct-2009 10:17:12	The request made by user 'Matt' to 'start' the resource 'TextMessenger' of type 'MessageFlow' on parent 'default' of typ

## Administration Queue

Administration Queue QuickView:

Order	Status	Username	Operation Type	Object Name	Object Type	Creation Time	Elapsed ...	Identifier
1	submitted	Matt	stop	foo	Execution Group	15-Sep-2009 11:55:50	0	DM9715bd43-k
2	pending	Matt	deletechild	default	Execution Group	15-Sep-2009 11:57:46	0	DMe666acfe-c
3	pending	Matt	deploy	foo	Execution Group	15-Sep-2009 11:58:57	3	DM1942c375-k
4	pending	Matt	deploy	foo	Execution Group	15-Sep-2009 11:59:55	1	DM1a81e52f-c



# How to diagnose some common scenarios



# Scenarios

- Message Broker won't start
  - First check the JOBLOG.

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

- Then check the STDOUT/STDERR for MQSICVP

```
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'.
```

```
BIP8907E: Verification failed. Unable to verify Java level.
```

Unable to verify the installed Java level. This error is typically caused by Java not being installed, or a file permissions error.

Ensure Java has been correctly installed, by running the command `java -version`.

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.

```
BIP8894I: Verification passed for 'MQSI_COMPONENT_NAME'.
```

```
BIP8894I: Verification passed for 'MQSI_FILEPATH'.
```

```
BIP8900I: Verification passed for APF Authorization of file '/u/wmqj91/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'.
```

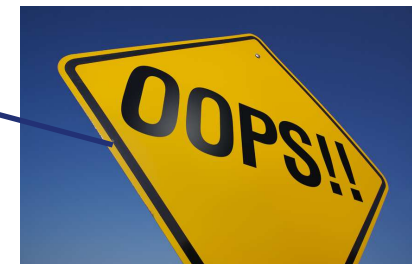
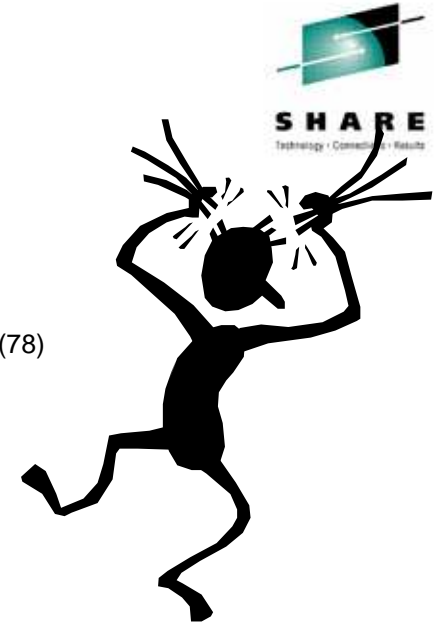
```
BIP8886I: Verification passed for queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager 'MQ91'.
```

```
BIP8884I: The WebSphere MQ verification for component 'MQ91BRK' has finished successfully.
```

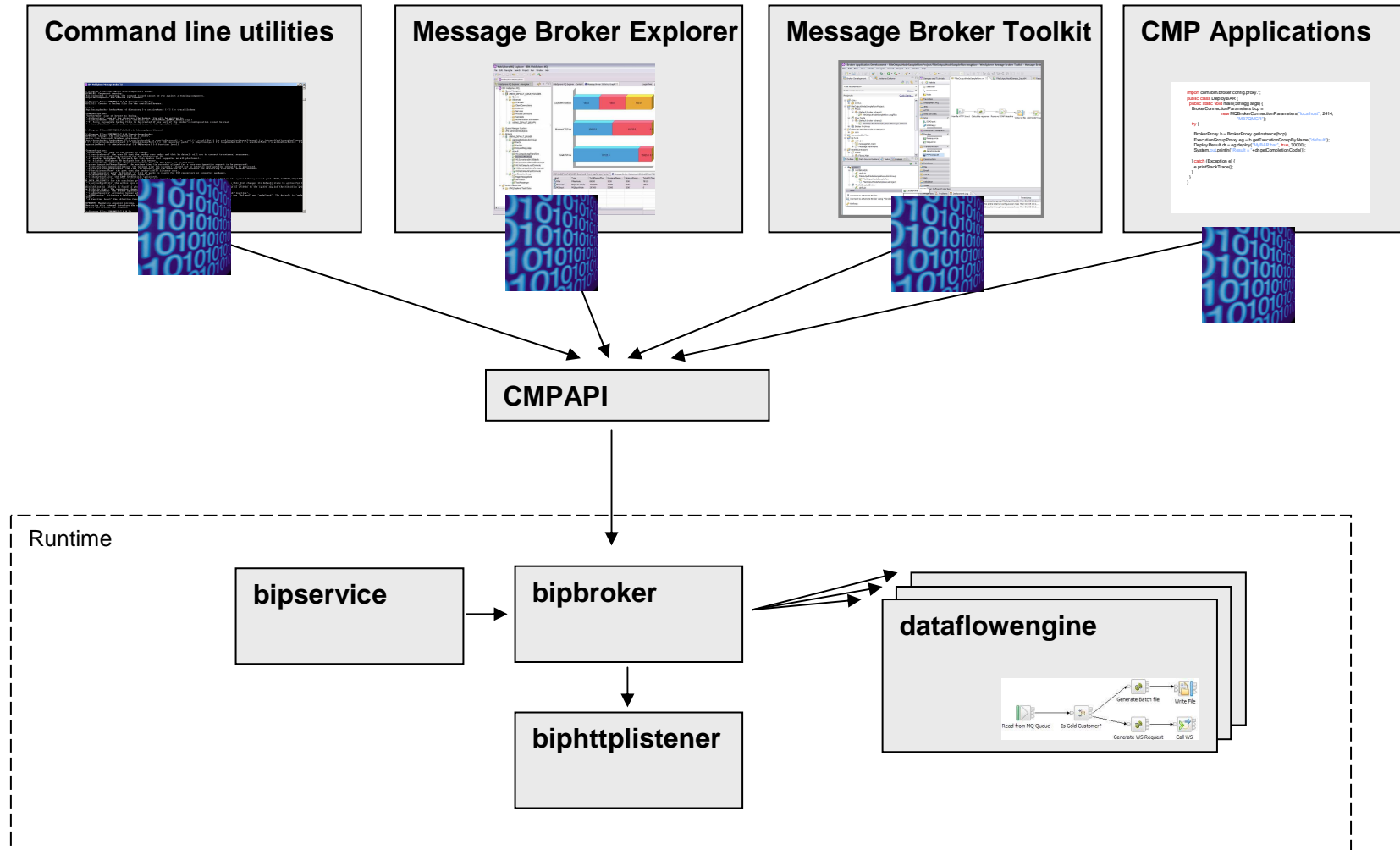
```
BIP8875W: The component verification for 'MQ91BRK' has finished, but one or more checks failed.
```

One or more of the component verification checks failed.

Check the error log for preceding error messages.

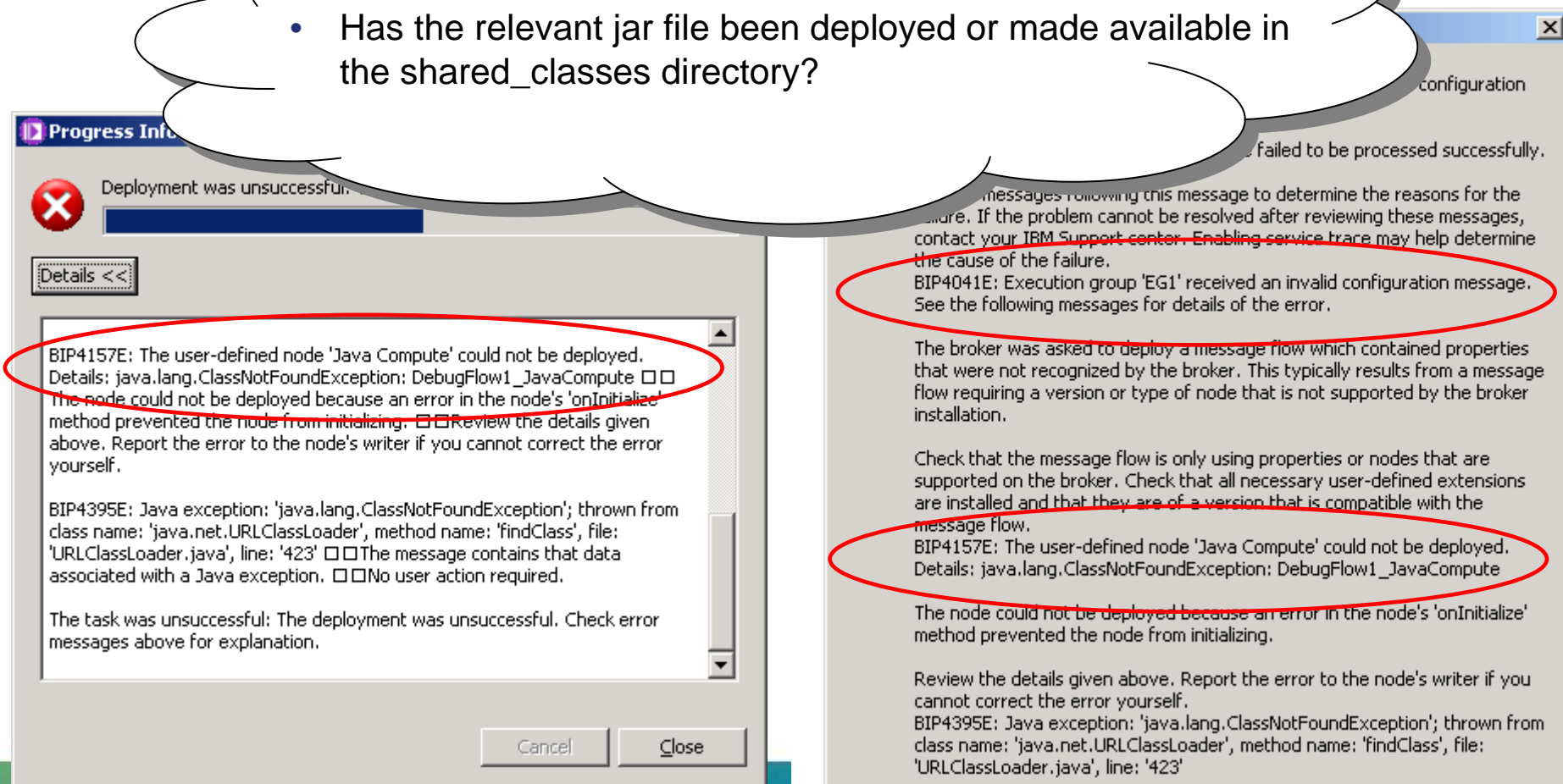


# Deploy of a Message flow fails



# 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 first message in the log is the most important
    - 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?



The screenshot displays the 'Progress Information' dialog box in the IBM Message Broker Toolkit (MBX). The dialog has a red 'X' icon and the title 'Deployment was unsuccessful'. A 'Details <<' button is visible. The main text area contains the following error messages:

**BIP4157E:** The user-defined node 'Java Compute' could not be deployed. Details: java.lang.ClassNotFoundException: DebugFlow1\_JavaCompute. The node could not be deployed because an error in the node's 'onInitialize' method prevented the node from initializing. Review the details given above. Report the error to the node's writer if you cannot correct the error yourself.

**BIP4395E:** Java exception: 'java.lang.ClassNotFoundException'; thrown from class name: 'java.net.URLClassLoader', method name: 'findClass', file: 'URLClassLoader.java', line: '423'. The message contains data associated with a Java exception. No user action required.

The task was unsuccessful: The deployment was unsuccessful. Check error messages above for explanation.

On the right side of the dialog, there is a 'configuration' tab. The text in this tab includes:

Failed to be processed successfully. Review the details given above. Report the error to the node's writer if you cannot correct the error yourself. BIP4041E: Execution group 'EG1' received an invalid configuration message. See the following messages for details of the error.

The broker was asked to deploy a message flow which contained properties that were not recognized by the broker. This typically results from a message flow requiring a version or type of node that is not supported by the broker installation.

Check that the message flow is only using properties or nodes that are supported on the broker. Check that all necessary user-defined extensions are installed and that they are of a version that is compatible with the message flow.

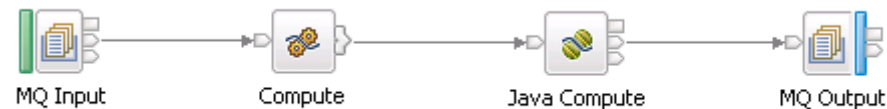
**BIP4157E:** The user-defined node 'Java Compute' could not be deployed. Details: java.lang.ClassNotFoundException: DebugFlow1\_JavaCompute. The node could not be deployed because an error in the node's 'onInitialize' method prevented the node from initializing.

Review the details given above. Report the error to the node's writer if you cannot correct the error yourself. BIP4395E: Java exception: 'java.lang.ClassNotFoundException'; thrown from class name: 'java.net.URLClassLoader', method name: 'findClass', file: 'URLClassLoader.java', line: '423'.



# 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

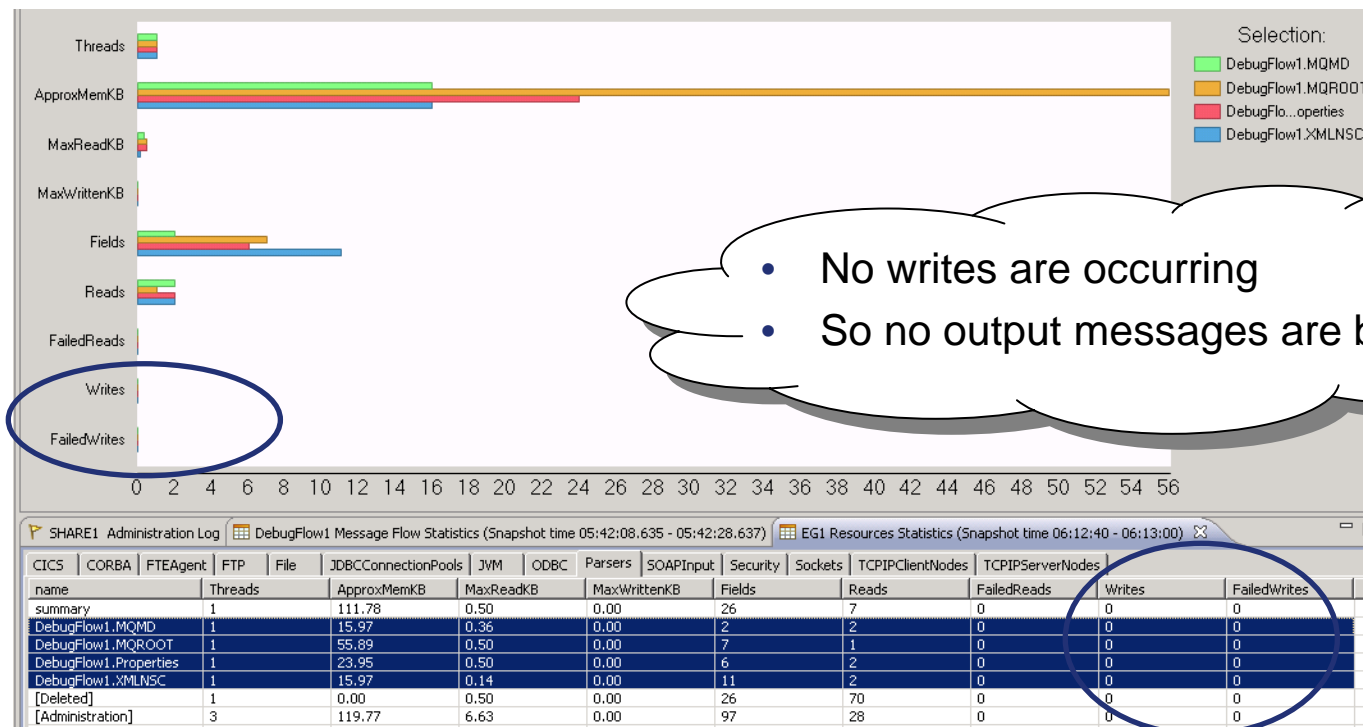


- A simple MQ In/Out flow with some transformation logic

# Where's my message?



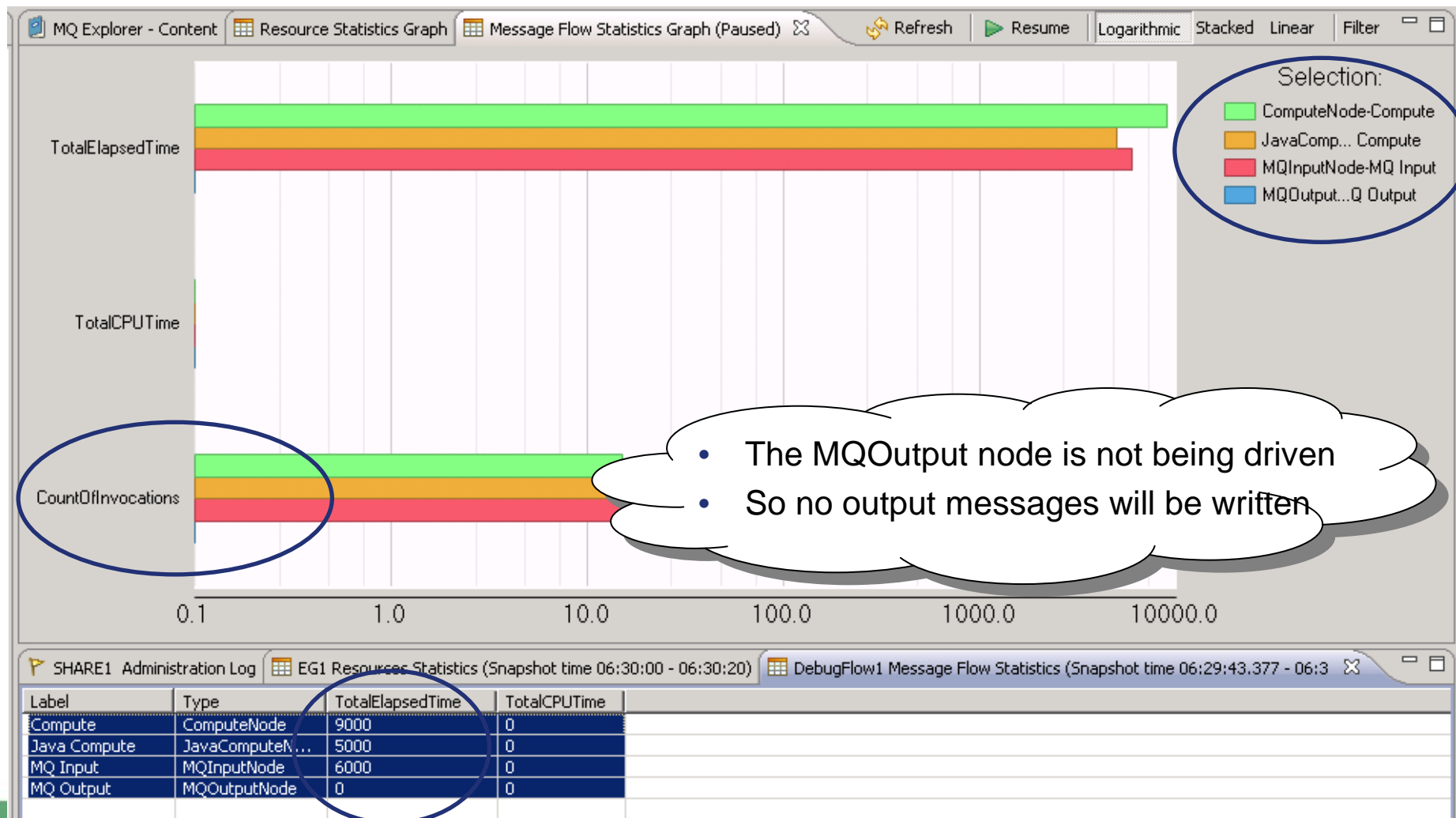
- Resource statistics
  - Is there a resource stat available for your output transport that would show if messages are being written?
    - e.g. CICS, CORBA, FTP, File, HTTP Sockets & TCPIP Nodes are available
    - The parsers output could be useful
    - Are any messages being written?



# Where's my message?



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



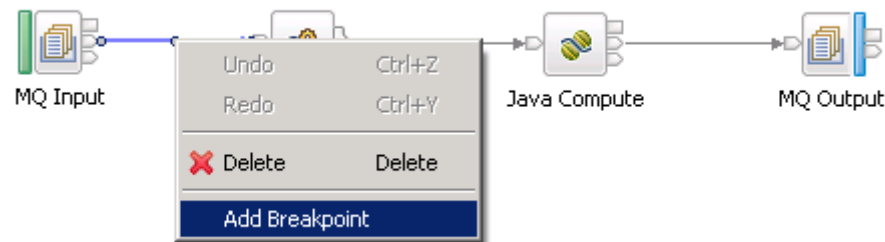


- UserTrace BIP2632l: Message received and propagated to 'out' terminal of MQ input node 'DebugFlow1.MQ Input'.  
UserTrace BIP6060l: Parser type "Properties" created on behalf of node 'DebugFlow1.MQ Input' to handle portion of incoming message of length 0 bytes beginning at offset '0'.  
UserTrace BIP6061l: 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.  
UserTrace BIP6061l: Parser type "XMLNSC" created on behalf of node 'DebugFlow1.MQ Input' to handle portion of incoming message of length '144' bytes beginning at offset '364'. Parser type selected based on value "XMLNSC" from previous parser.  
UserTrace BIP2537l: Node 'DebugFlow1.Compute': Executing statement "BEGIN ... END;" at ('.DebugFlow1\_Compute.Main', '2.2').  
UserTrace BIP2537l: Node 'DebugFlow1.Compute': Executing statement "CopyEntireMessage();" at ('.DebugFlow1\_Compute.Main', '3.3').  
UserTrace BIP2538l: Node 'DebugFlow1.Compute': Evaluating expression "CopyEntireMessage()" at ('.DebugFlow1\_Compute.Main', '3.8').  
UserTrace BIP2537l: Node 'DebugFlow1.Compute': Executing statement "BEGIN ... END;" at ('.DebugFlow1\_Compute.CopyEntireMessage', '1.39').  
UserTrace BIP2537l: Node 'DebugFlow1.Compute': Executing statement "SET OutputRoot = InputRoot;" at ('.DebugFlow1\_Compute.CopyEntireMessage', '2.3').  
UserTrace BIP2539l: Node 'DebugFlow1.Compute': Evaluating expression "InputRoot" at ('.DebugFlow1\_Compute.CopyEntireMessage', '2.3'). This resolved to "InputRoot". The result was "ROW... Root Element Type=16777216 NameSpace=urn:unresolved-schema:2001-09-01:IBM.XM...".  
UserTrace BIP2568l: Node 'DebugFlow1.Compute': Copying sub-tree from "InputRoot" to "OutputRoot".  
UserTrace BIP2537l: Node 'DebugFlow1.Compute': Executing statement "DECIMAL) \* CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS INTEGER)" at ('.DebugFlow1\_Compute.Main', '4.39'). This resolved to "CAST('77' AS INTEGER)".  
UserTrace BIP2539l: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS INTEGER)" at ('.DebugFlow1\_Compute.Main', '4.39'). This resolved to "CAST('77' AS INTEGER)".  
UserTrace BIP2539l: Node 'DebugFlow1.Compute': Evaluating expression "CAST('77' AS INTEGER)" at ('.DebugFlow1\_Compute.Main', '4.91'). This resolved to "3 \* 77". The result was "231".  
UserTrace BIP2566l: Node 'DebugFlow1.Compute': Assigning value "231" to field / variable "OutputRoot.XMLNSC.Order.Total".  
UserTrace BIP2537l: Node 'DebugFlow1.Compute': Executing statement "RETURN TRUE;" at ('.DebugFlow1\_Compute.Main', '5.3').  
UserTrace BIP4015l: Message propagated to the 'out' terminal of node 'DebugFlow1.Compute' with the following message trees: ".  
UserTrace BIP3004l: Invoking the evaluate() method of node (class='ComIbmJavaComputeNode', name='DebugFlow1#FCMComposite\_1\_4'). About to pass a message to the evaluate() method of the specified node.  
No user action required.

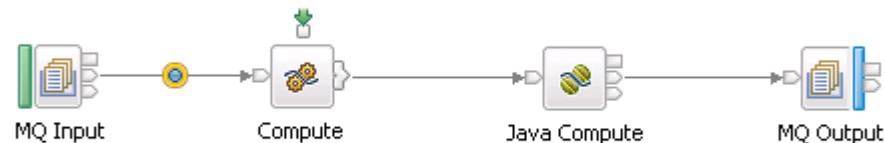


# Where's my message?

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



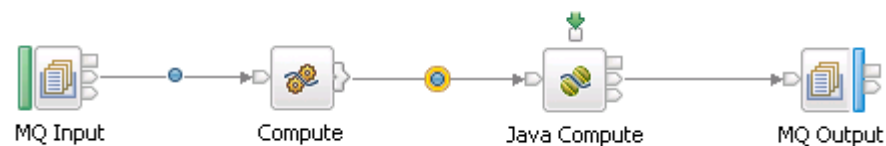
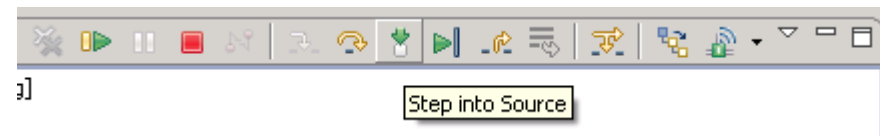
- Fire in a message and the breakpoint triggers



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

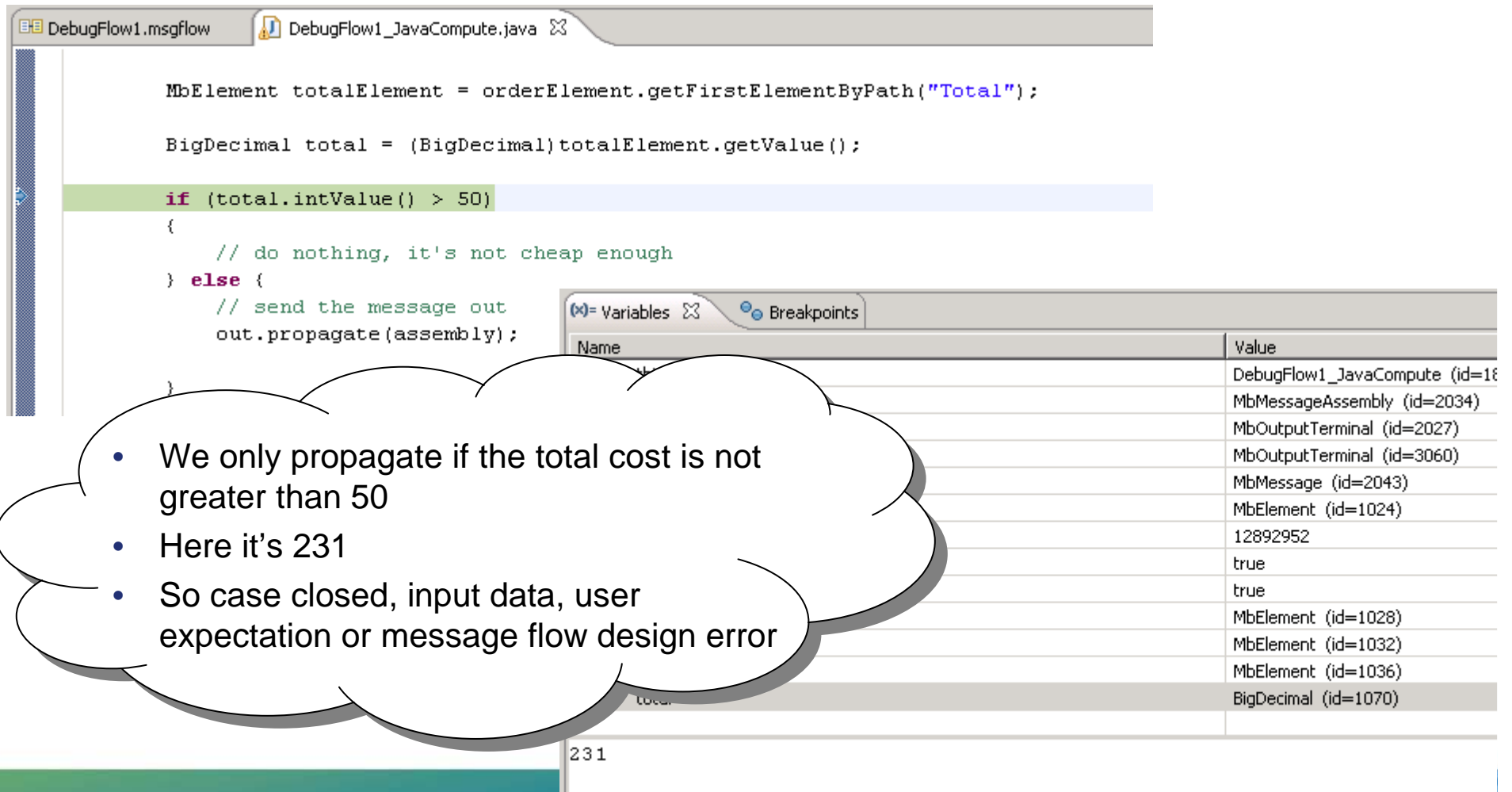
# Where's my message?

- 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



The screenshot shows the Message Flow Debugger interface. The top pane displays the Java source code for `DebugFlow1_JavaCompute.java`. The code is as follows:

```

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);
}
  
```

The bottom pane shows the 'Variables' tab with a table of current variables:

Name	Value
DebugFlow1_JavaCompute (id=18)	
MbMessageAssembly (id=2034)	
MbOutputTerminal (id=2027)	
MbOutputTerminal (id=3060)	
MbMessage (id=2043)	
MbElement (id=1024)	
12892952	
true	
true	
MbElement (id=1028)	
MbElement (id=1032)	
MbElement (id=1036)	
BigDecimal (id=1070)	

A callout bubble contains the following text:

- 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

The value 231 is also visible at the bottom of the code editor window.



# 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



Session 10529

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

# This was session 10529 - The rest of the week .....



	Monday	Tuesday	Wednesday	Thursday	Friday
08:00			Free MQ! - MQ Clients and what you can do with them.	MQ Performance and Tuning on distributed	
09:30		The MQ API for dummies - the basics	The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation	The even darker arts of SMF	CICS Programs Using WMQ V7 Verbs
11:00		Putting the web into WebSphere MQ: A look at Web 2.0 technologies	Message Broker administration	The Do's and Don'ts of z/OS Queue Manager Performance	
		The Doctor is in. Hands-on Lab and Lots of Help with the MQ Family			
12:15		WebSphere MQ: Highly scalable publish subscribe environments		MQ & DB2 – MQ Verbs in DB2 & Q-Replication	
01:30	WebSphere MQ 101: Introduction to the world's leading messaging provider	What's new in WebSphere Message Broker V8.0	The Do's and Don'ts of Message Broker Performance	Diagnosing problems for MQ	
03:00	WebSphere Message Broker 101: The Swiss army knife for application integration	What's new in WebSphere MQ V7.1	WebSphere MQ Security - with V7.1 updates	Diagnosing problems for Message Broker	
04:30	Introduction to the WebSphere MQ Product Family - including what's new in the family products	Under the hood of Message Broker on z/OS - WLM, SMF and more	MQ Java zero to hero	Shared Q including Shared Message Data Sets	
06:00			For your eyes only - WebSphere MQ Advanced Message Security	MQ Q-Box - Open Microphone to ask the experts questions	

# 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](http://www.ibm.com/legal/copytrade.shtml). Other company, product, or service names may be trademarks or service marks of others.