#SHAREorg

SHARE
Technology · Connections · Results

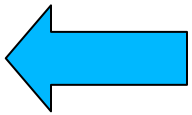
Diagnosing Problems for WebSphere Message Broker (z/OS and Distributed)

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

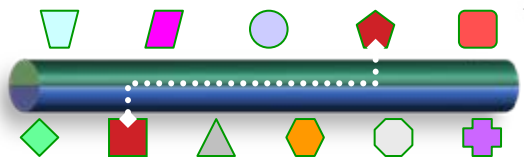
Tuesday 7th August 2012
Session Number 11508

SHARE in Anaheim 2012

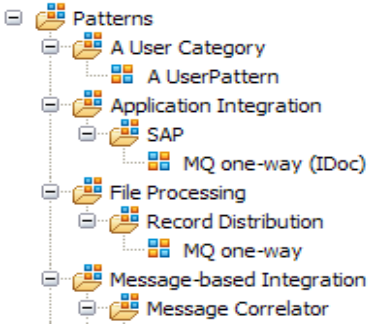
Agenda

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

WebSphere Message Broker

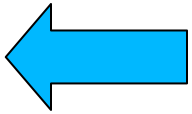


- Universal Connectivity FROM anywhere, TO anywhere
  - Simplify application connectivity for a flexible & dynamic infrastructure
- Comprehensive Protocols, Transports, Data Formats & Processing
  - Connect to applications, services, systems and devices
    - MQ, JMS 1.1, HTTP(S), SOAP, REST, File (incl. 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

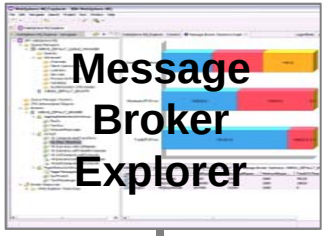


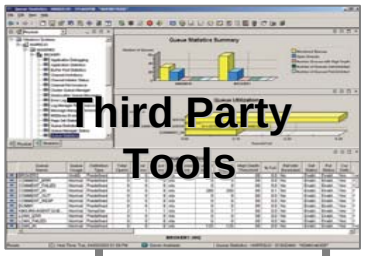
- Extensive Management, Performance & Scalability
  - Extensive Administration & Systems Management facilities for developed solutions
  - Wide range of operating system &hardware platforms supported, including virtual & cloud options
  - High performance transactional processing, additional vertical & 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

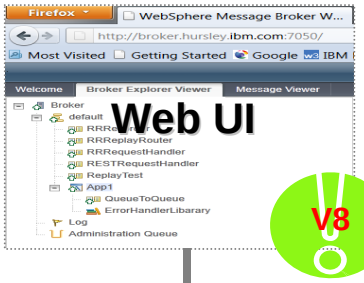
Agenda

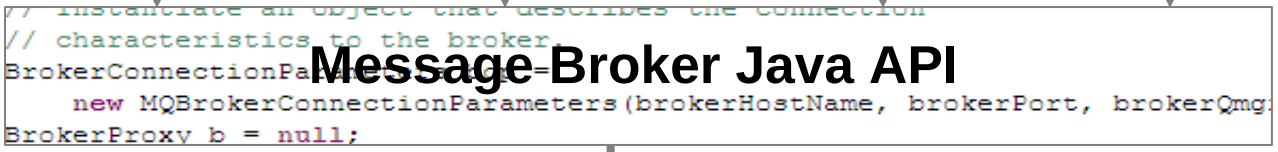
- WMB Recap
- External Components 
- Diagnostic Information
- How to diagnose common scenarios

External Components



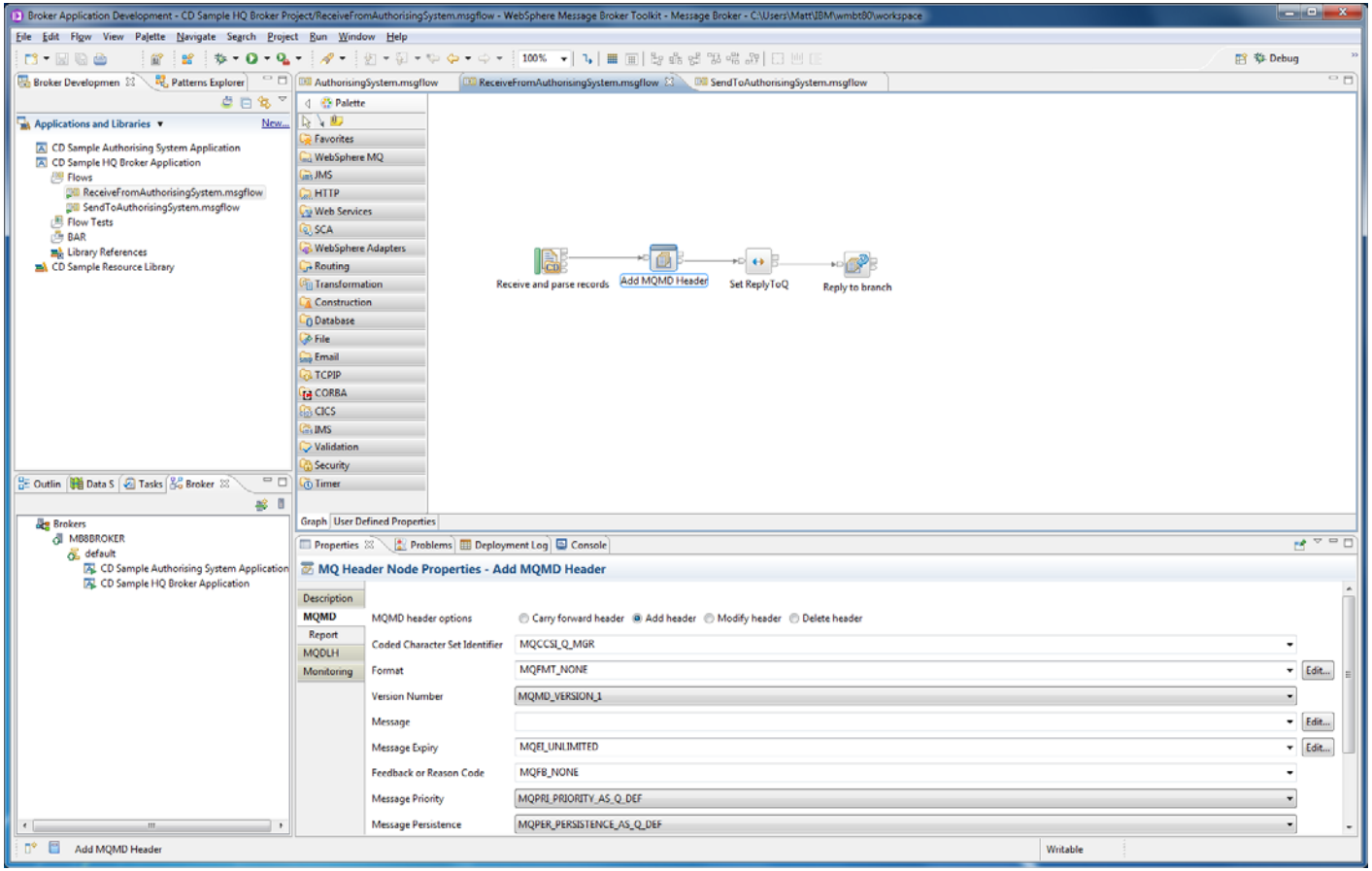






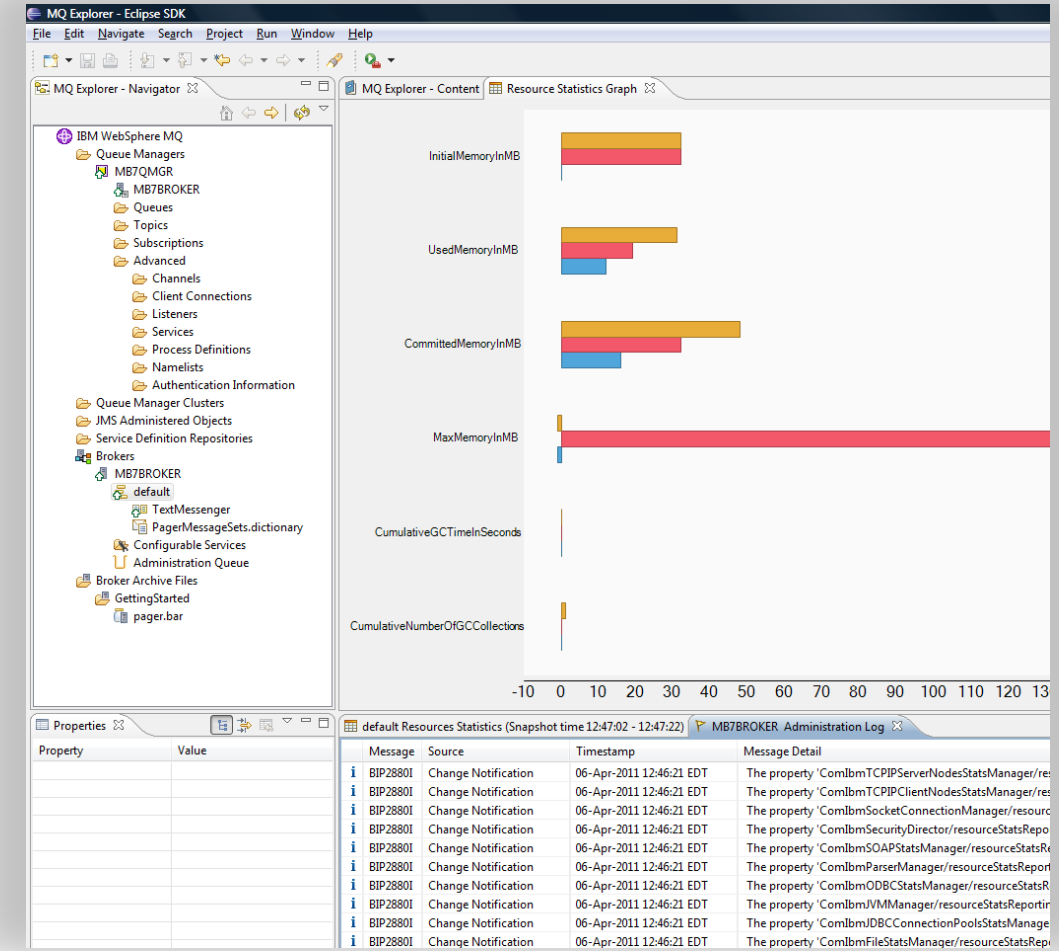
Broker

Broker View for Application Developers



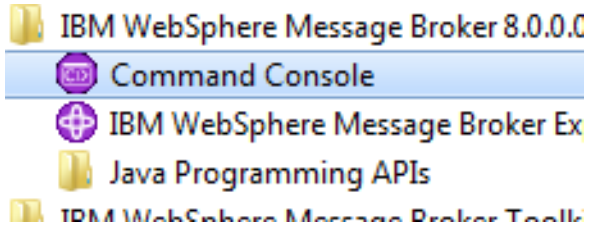
Message Broker Explorer (MBX)

- 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

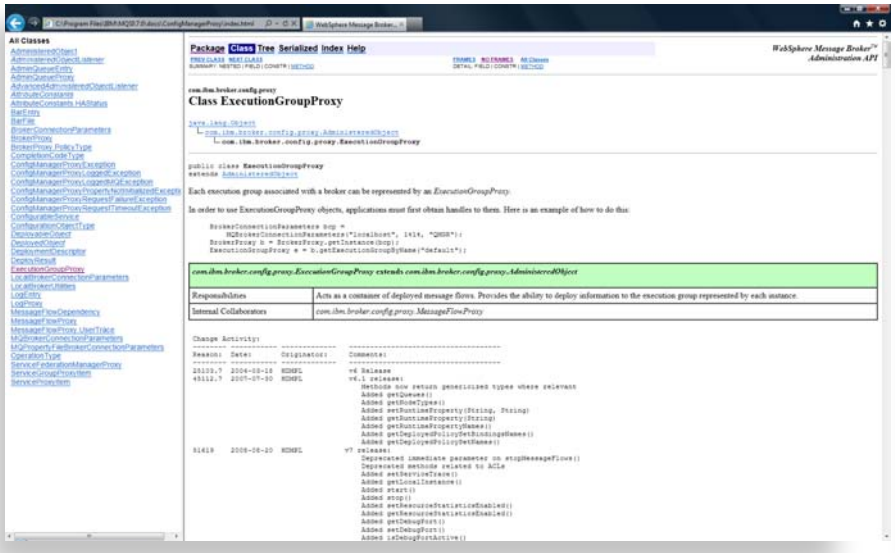
- A wide selection of tools for scripting broker actions
- Requires a configured environment
  - Command console (Windows)
    mqsiprofile (Linux/UNIX)
    JCL or ISPF (z/OS)
- Most commands work against local or remote brokers

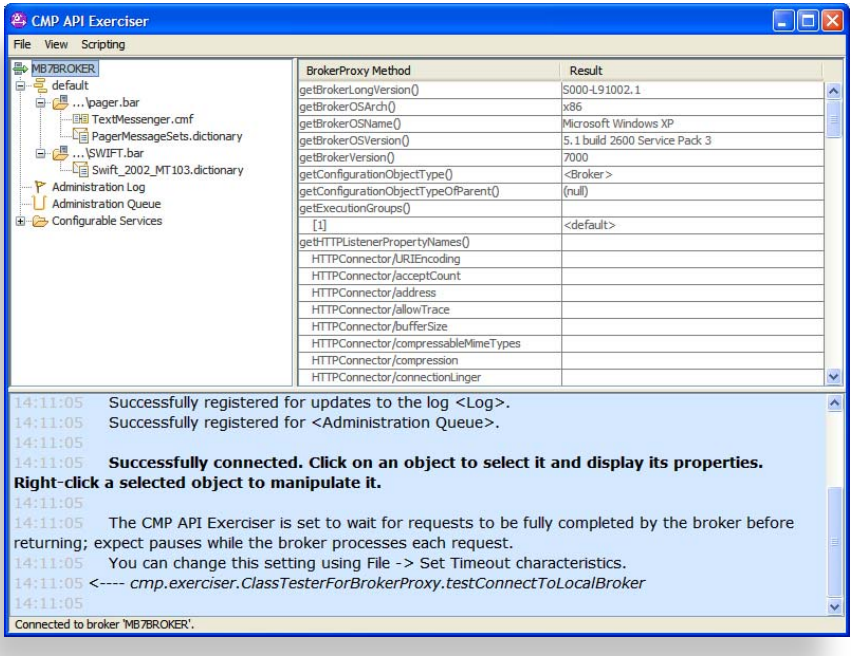



```
BIP1121I: Creates an execution group.

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





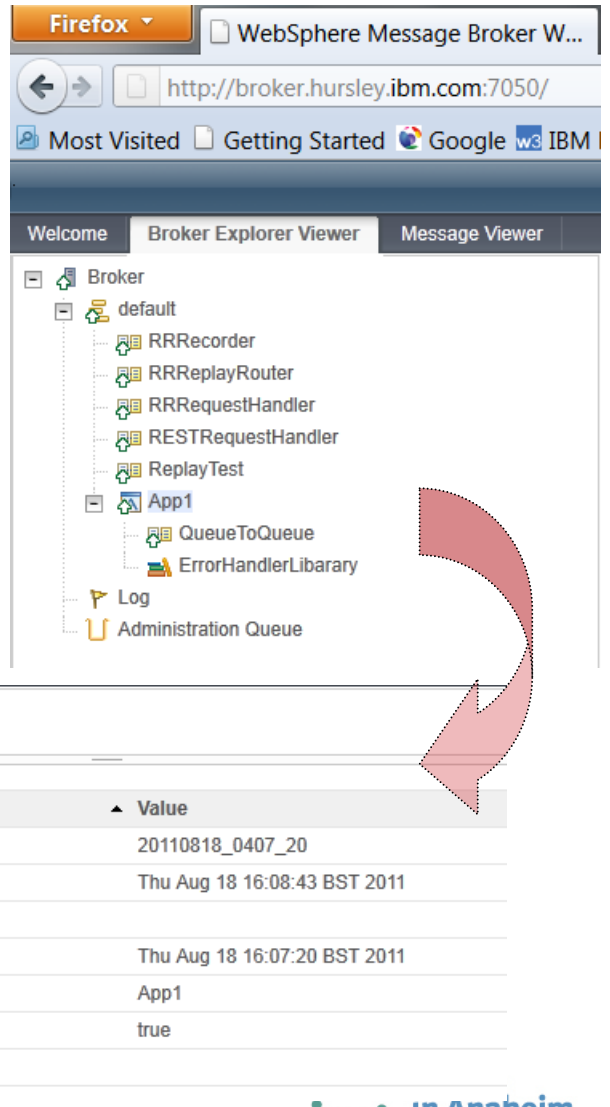
New in V8

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

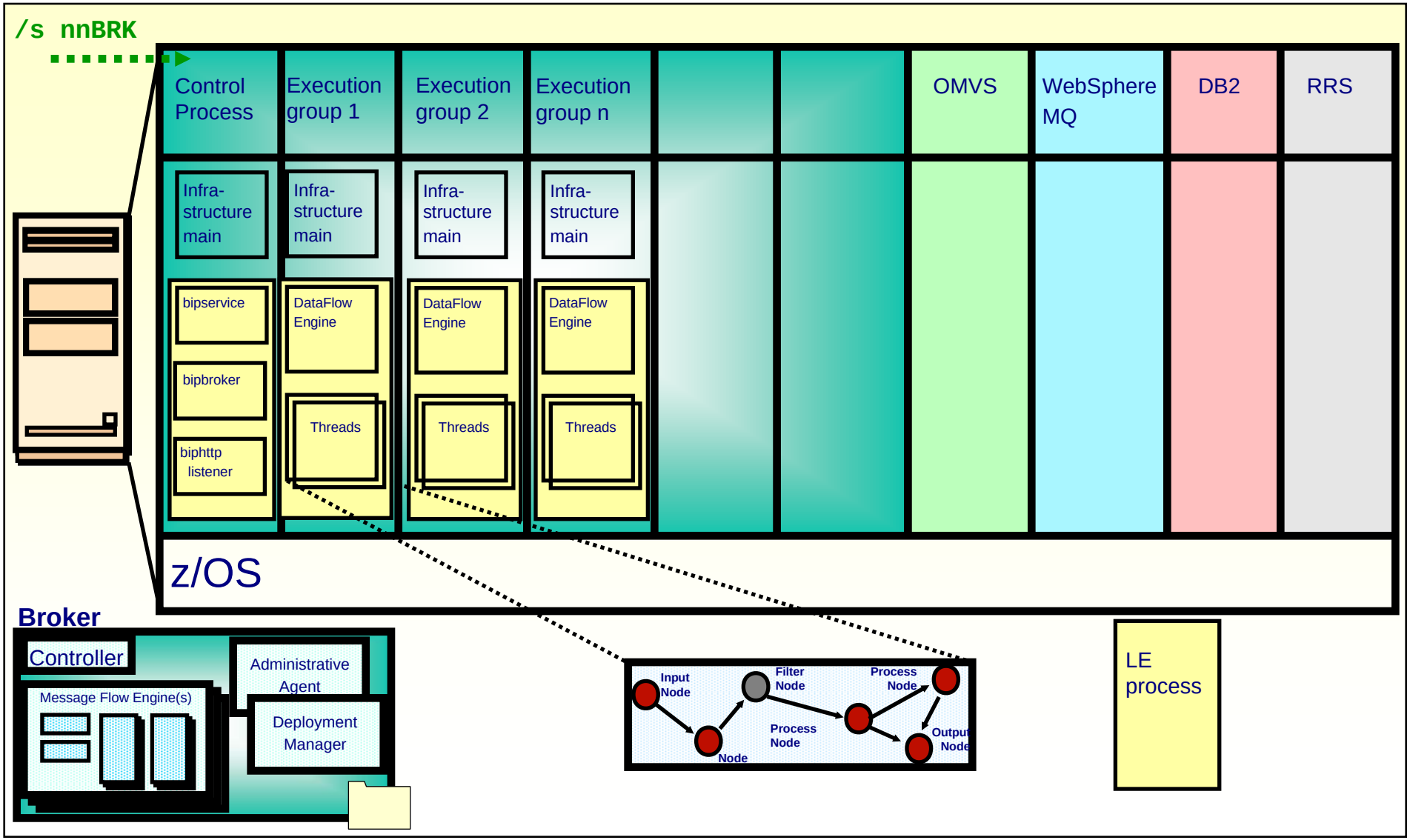
Web UI

New in V8

- 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



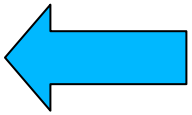
WMB Runtime Structure z/OS



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

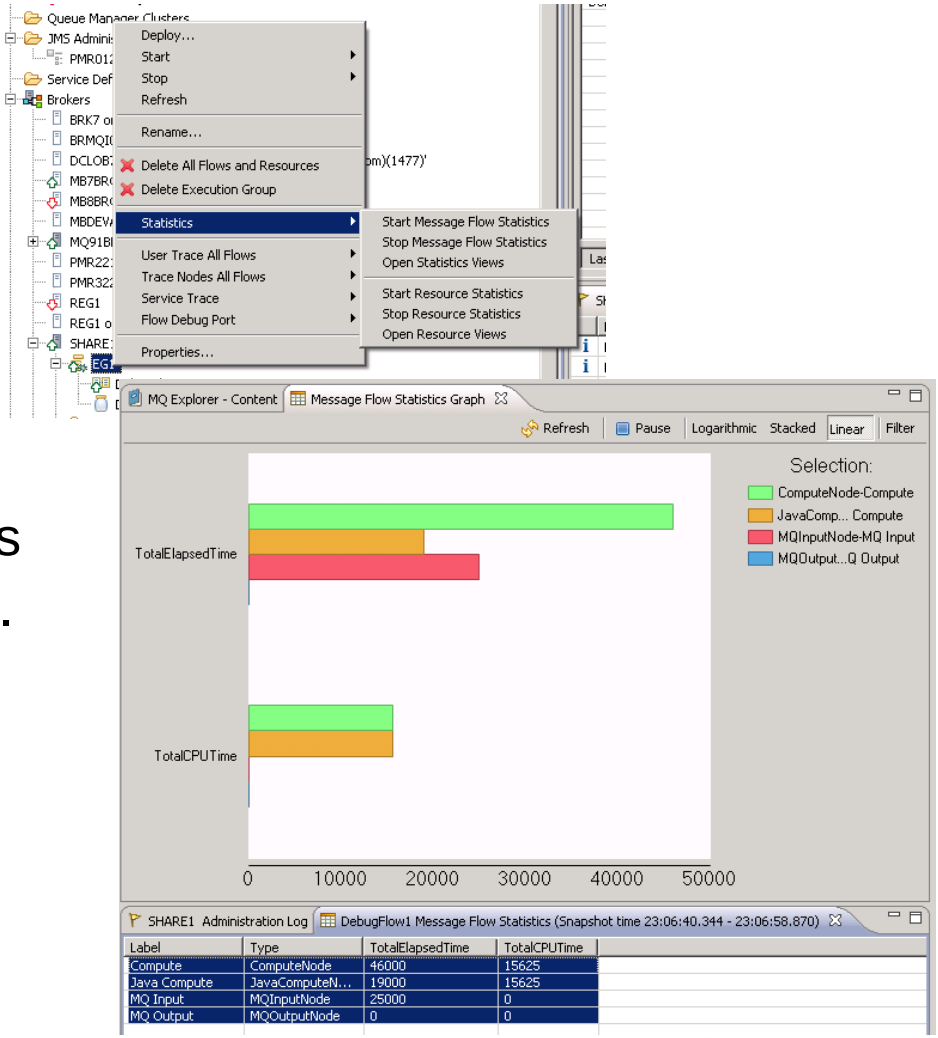
- Captures everything written to std io
  - Java: System.out.println()
  - C: printf()
- z/OS
  - STDOUT/STDERR DD cards in JOBLOGs
- Distributed
  - Admin Agent
    - $MQSI_WORKPATH/components/<Broker_Name>/stdout and sttderr
  - Execution Groups
    - $MQSI_WORKPATH/components/<Broker_Name>/<EG_UUID>/stdout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

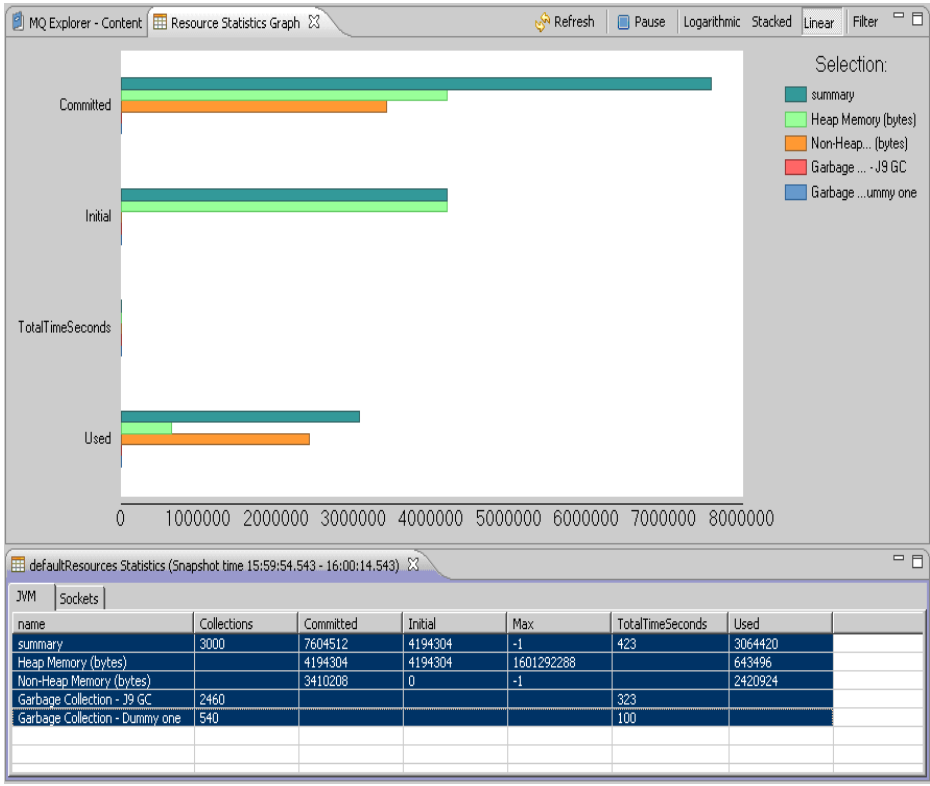
Flow Statistics

- 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

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)

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)

CICS | CORBA | FTEAgent | FTP | File | JDBCConnectionPools | JVM | ODBC | Parsers | SOAPInput | Security | Sockets | TCPIPClientNodes | TCPIPServerNodes

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

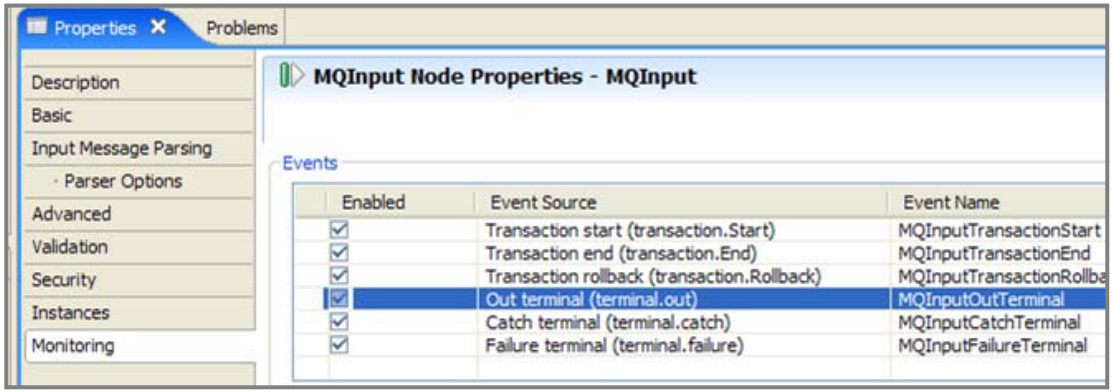
Activity Log

- 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

	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
⊗	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

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



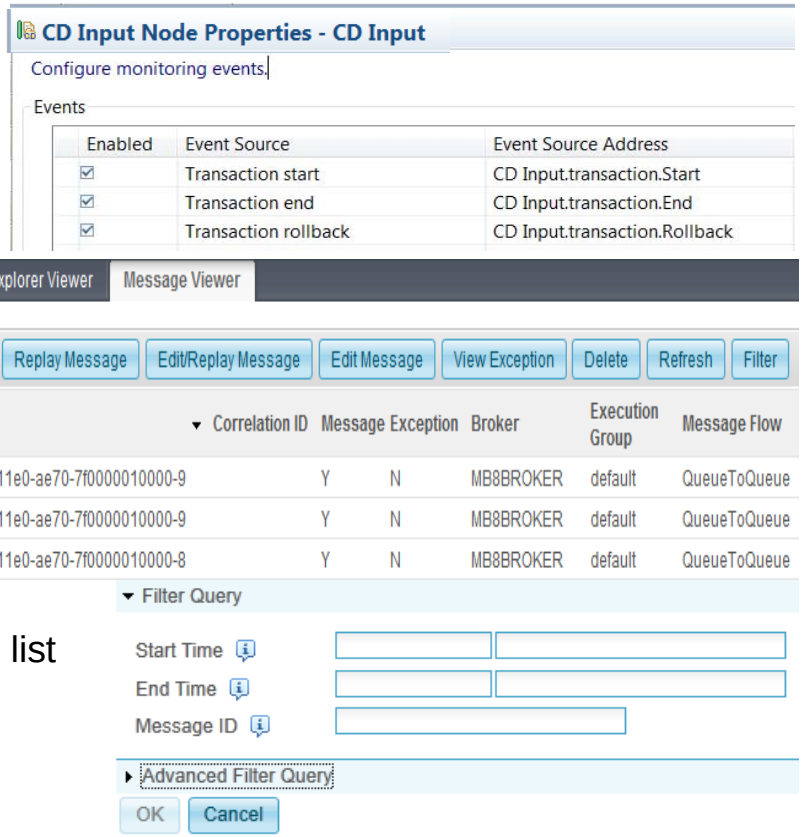
Message Tracking

- 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…

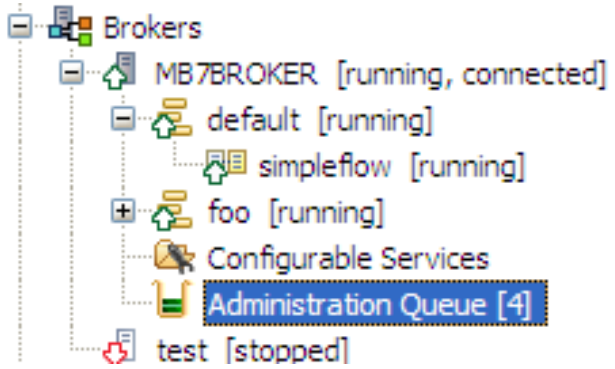
- Web Tooling to View, Query, Edit data
  - Friendly editors to view, query & edit payloads
    - Key data fields, including application data
  - Independent web admin & capture for scalability
    - Configure multiple EG listeners for web

- 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



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?



MQ Explorer - Content | MB7BROKER Administration Log

	Message	Source	Timestamp	Message Detail
i	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
i	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
i	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
i	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
i	BIP			type 'MessageFlow' on parent 'default' of typ

MQ Explorer - Content

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-c
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	DM1942c375-2
4	pending	Matt	deploy	foo	Execution Group	15-Sep-2009 11:59:55	1	DM1a81e52f-0

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 21:58:23.160163     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 '"7"'.
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='ComIbmJavaComputeNode', 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
```

Trace Status commands

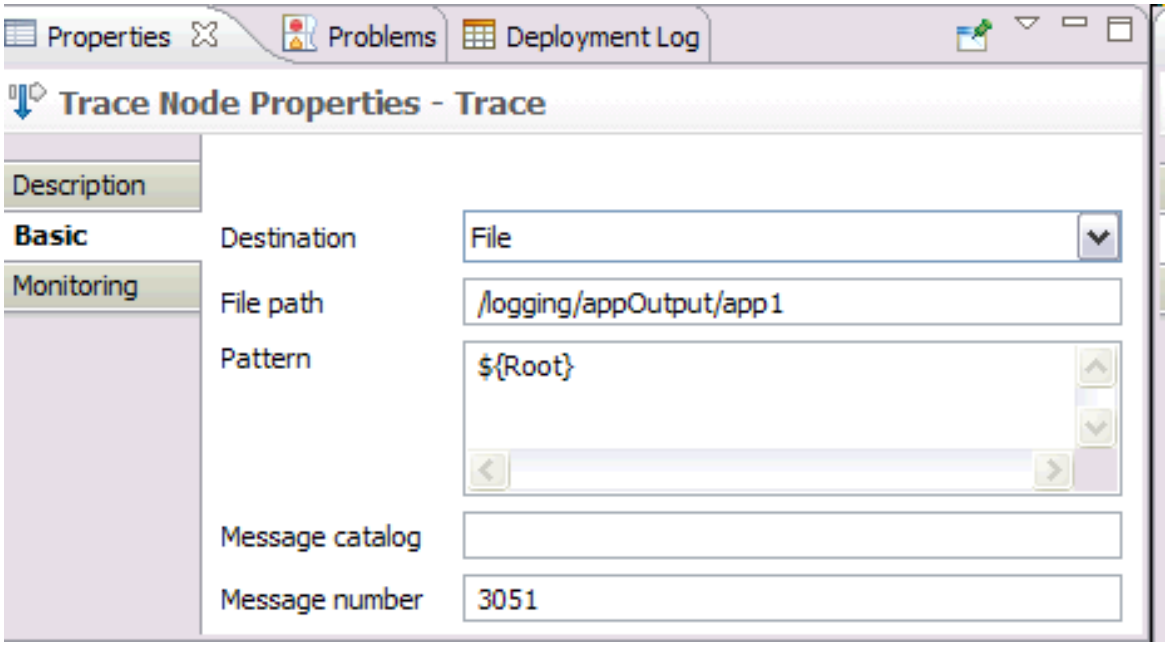
- **Non-persistent trace option** (7.0.0.3/8.0.0.0)
  - How do I find the evidence of what went wrong.
  - New ability to Enable execution group wide trace level that doesn't survive a restart
  - Helps to capture trace for abend/shutdown situations
  - Stops traces being wrapped during restart

- **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



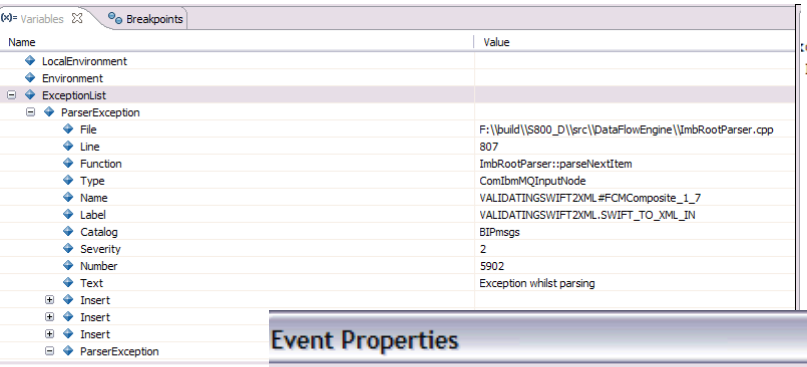
Example Trace Node Output

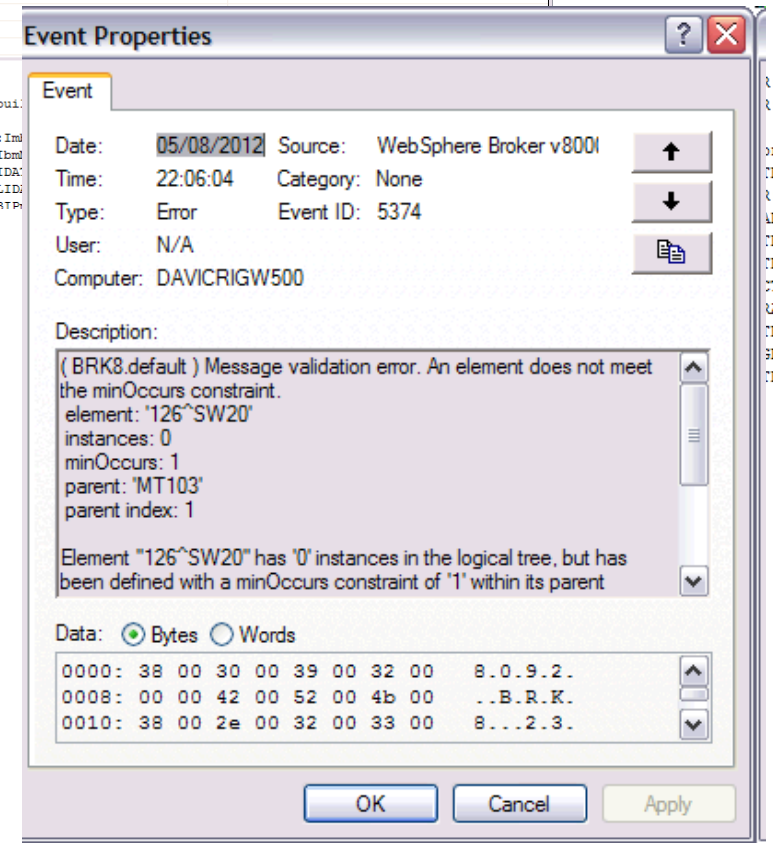
```
(0x03000000:NameValue):X-Server-Name          = 'localhost' (CHARACTER)
(0x03000000:NameValue):X-Server-Port          = '7802' (CHARACTER)
(0x03000000:NameValue):X-Query-String         = '' (CHARACTER)
(0x03000000:NameValue):X-Scheme               = 'http' (CHARACTER)
)
(0x01000000:Folder):SOAP            = ( ['SOAP' : 0x2773c388]
  (0x01000000:Folder):Context = ( ['xmlnsc' : 0x1707bc8]
    (0x03000100:Attribute  ):operation         = 'submitPO' (CHARACTER)
    (0x03000100:Attribute  ):operationType     = 'REQUEST_RESPONSE' (CHARACTER)
    (0x03000100:Attribute  ):portType          = 'SOAPNodesSampleMessageSetPortType' (CHARACTER)
    (0x03000100:Attribute  ):portTypeNamespace = 'http://com.http.orderservice' (CHARACTER)
    (0x03000100:Attribute  ):port              = 'SOAP_HTTP_Port' (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     ):Namespace         = (
      (0x03000102:NamespaceDecl)http://www.w3.org/2000/xmlns/:soapenv = 'http://schemas.xmlsoap.org/soap/envelope/' (CHARACTER)
      (0x03000102:NamespaceDecl)http://www.w3.org/2000/xmlns/:ord     = 'http://www.acmeOrders.com/OrderService' (CHARACTER)
    )
  )
  (0x01000000:Folder):Header  =
  (0x01000000:Folder):Body    = ( ['xmlnsc' : 0x1707bc8]
    (0x01000000:Folder)http://www.acmeOrders.com/OrderService:submitPORequest = (
      (0x03000000:PCDataField):partNo       = '012365' (CHARACTER)
      (0x03000000:PCDataField):partQuantity = 10 (INTEGER)
      (0x01000000:Folder     ):personName   = (
        (0x03000000:PCDataField):firstName = 'Dave' (CHARACTER)
        (0x03000000:PCDataField):lastName  = 'Crighton' (CHARACTER)
      )
      (0x01000000:Folder     ):address      = (
        (0x03000000:PCDataField):street  = 'myStreet' (CHARACTER)
        (0x03000000:PCDataField):city    = 'myCity' (CHARACTER)
        (0x03000000:PCDataField):zipCode = 'myZip' (CHARACTER)
      )
    )
```

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



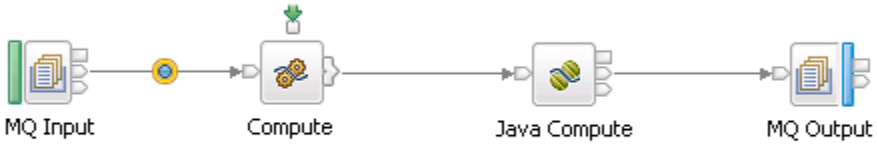


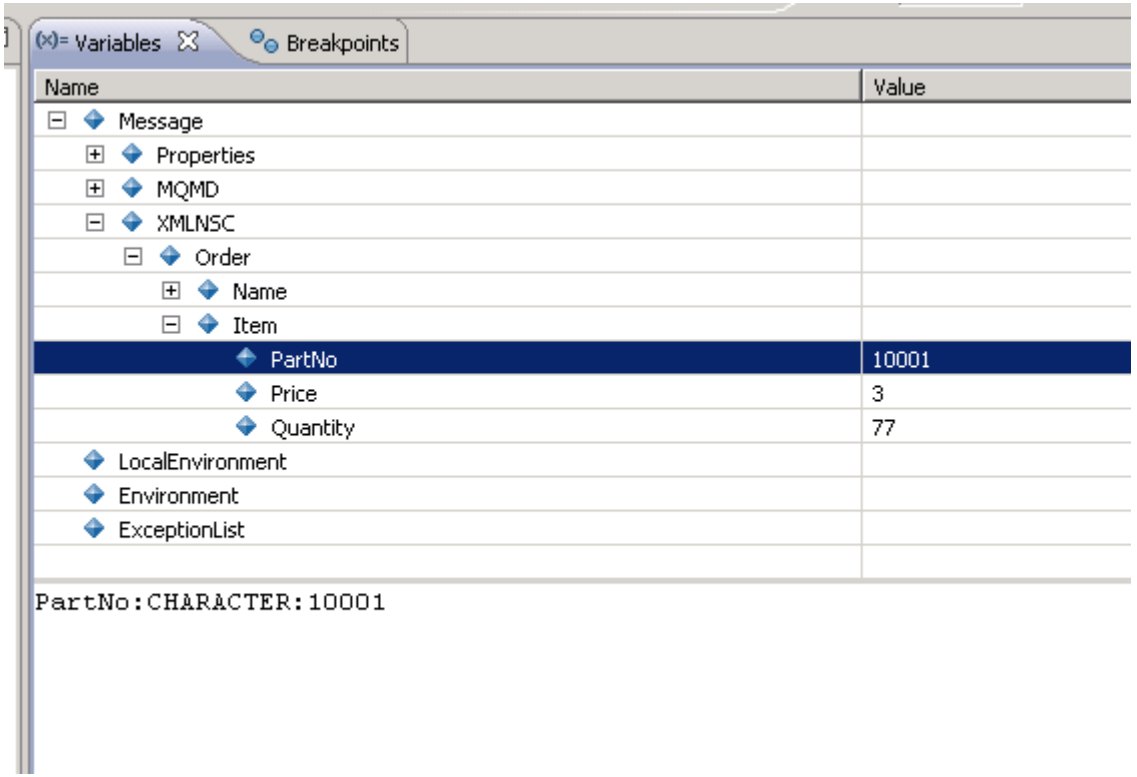

```
ExceptionList
    ParserException
            File:CHARACTER:F:\bui
            Line:INTEGER:807
            Function:CHARACTER:Iml
            Type:CHARACTER:ComIbml
            Name:CHARACTER:VALIDA
            Label:CHARACTER:VALIDA
            Catalog:CHARACTER:BIP
```

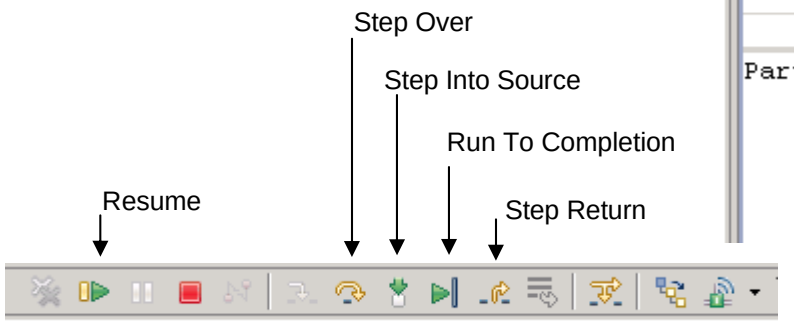
```
ceptionList
ParserException
  <SNIP>
  Severity:INTEGER:2
  Number:INTEGER:5902
  Text:CHARACTER:Exception whilst parsing
  <SNIP>
  ParserException
    <SNIP>
    Severity:INTEGER:3
    Number:INTEGER:5285
    Text:CHARACTER:ImbRecoverableException caught from worker->parseNext.
```

```
R:5421
R:TDS General Error

on
TER:F:\build\S800_D\src\MTI\MTIforBroker\MtiImbParser2\MtiImbFIHandler.cpp
R:3929
ARACTER:MtiImbFIHandler::attachDefaultElement
TER:
TER:
CTER:
RACTER:BIPmsgs
TEGER:3
GER:5374
TER:An element has been deemed complete occurring less than specified minOccurs

  Type:INTEGER:5
  Text:CHARACTER:126^SW20

  Type:INTEGER:2
  Text:CHARACTER:0

  Type:INTEGER:2
  Text:CHARACTER:1

  Type:INTEGER:5
  Text:CHARACTER:MT103

  Type:INTEGER:2
  Text:CHARACTER:1
```

Message Flow Debugger



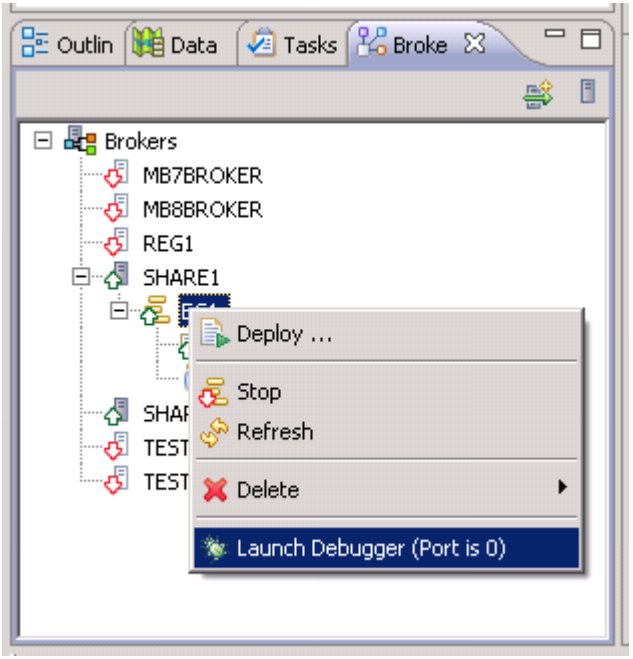




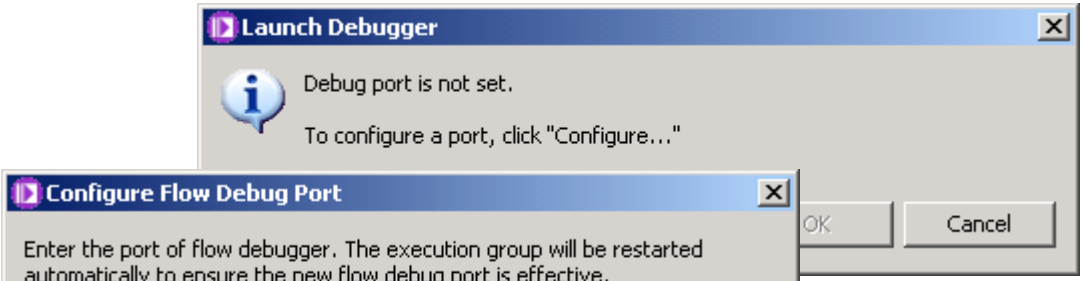
Message Flow Debugger

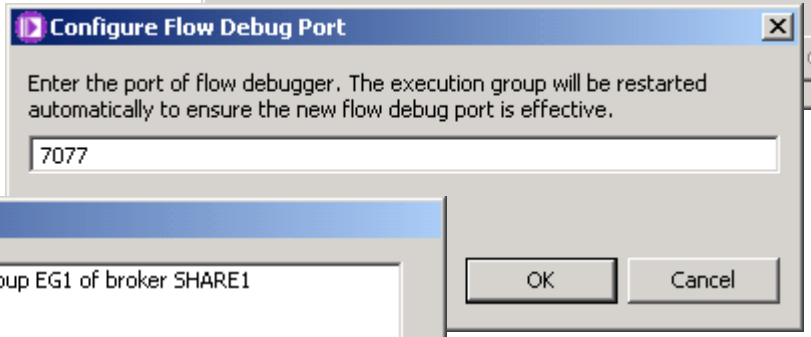
- Use the Message Flow debugger to debug your message flows
- Set breakpoints on the connections between nodes
- At each stage you can view (and edit) the Message Trees
- Step into ESQL or Java compute nodes
- Requires the enablement of the JVMDebug port on the execution group you wish to debug
  - Don’t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

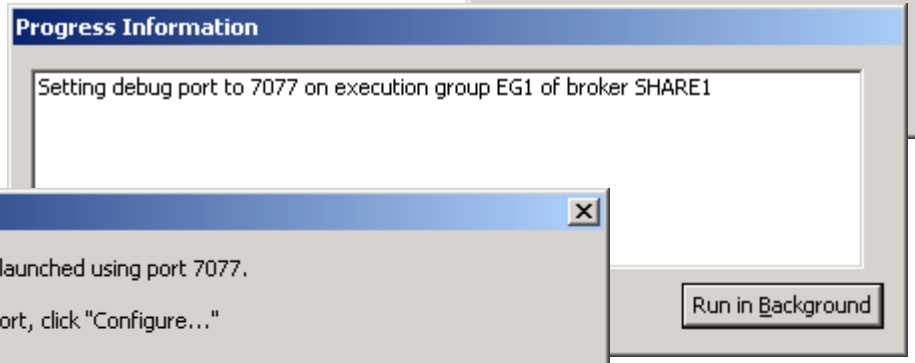
Message Flow Debugger

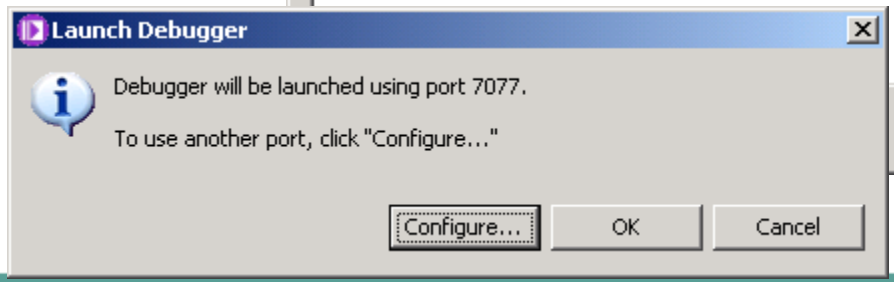


HOW TO: Enable the debugger to allow you to debug message flows from the Message Broker toolkit

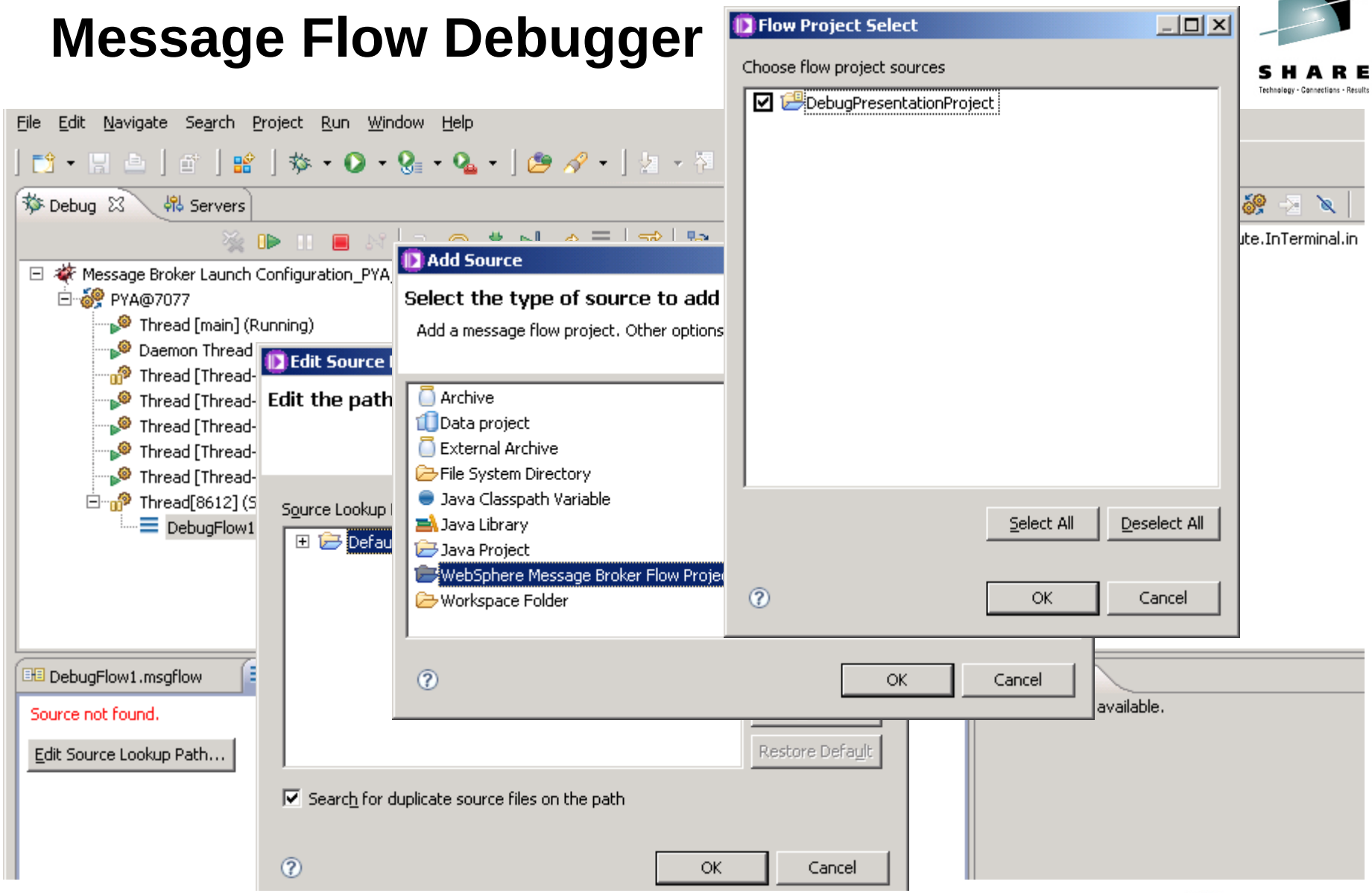








Message Flow Debugger



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

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"

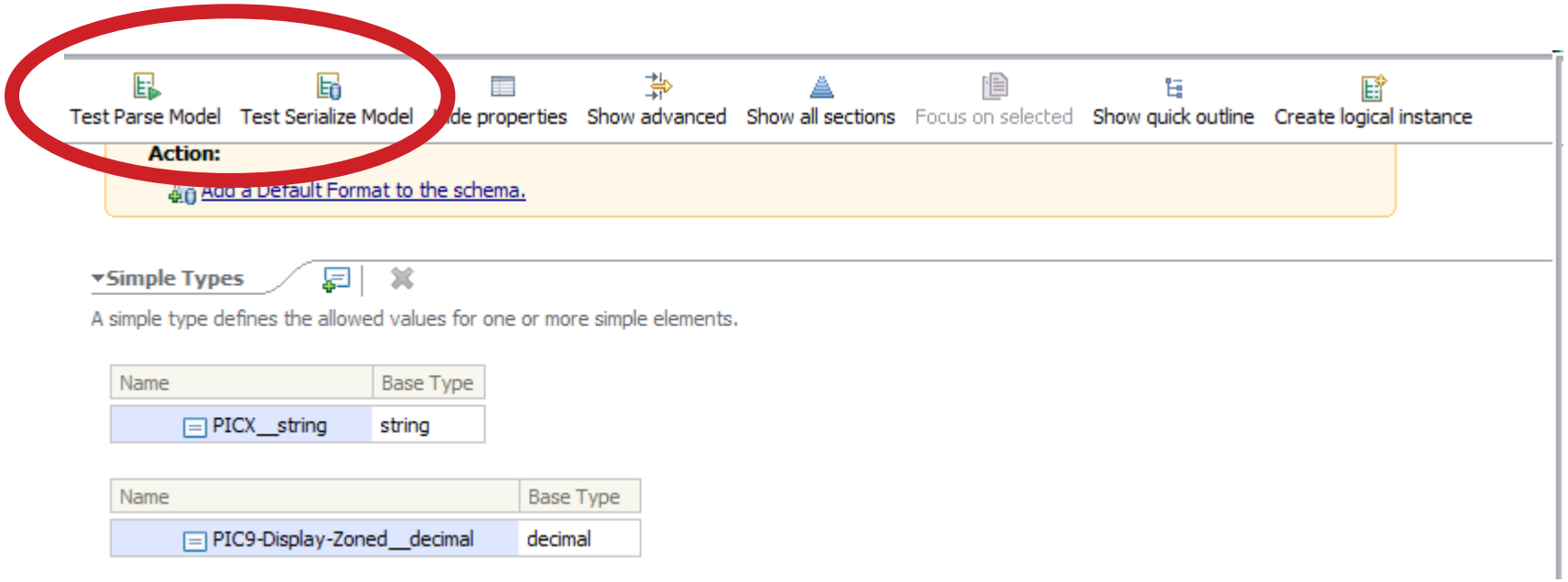
Call IBM Support or not?

- 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

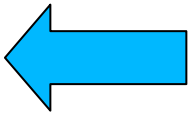
```
Traceback:
  DSA Addr  Program Unit  PU Addr   PU Offset  Entry         E Addr    E  Offset   Statement  Load Mod  Service  Status
  38F9DBD0  CEEVRONU      0707D2B8  +00001004  CEEVRONU      0707D2B8  +00001004              CEEPLPKA  HLE7730  Call
  390253A0                1DF418F8  +000000DE  ImbAbend::printStackForCurrentThread(int,bool,const 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           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           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.

DFDL Tooling



Agenda

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

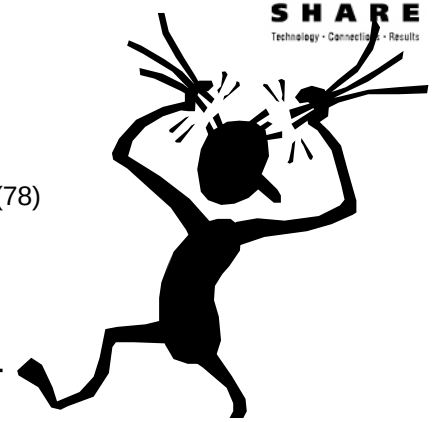
Fundamentals of WMB diagnosis

- 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



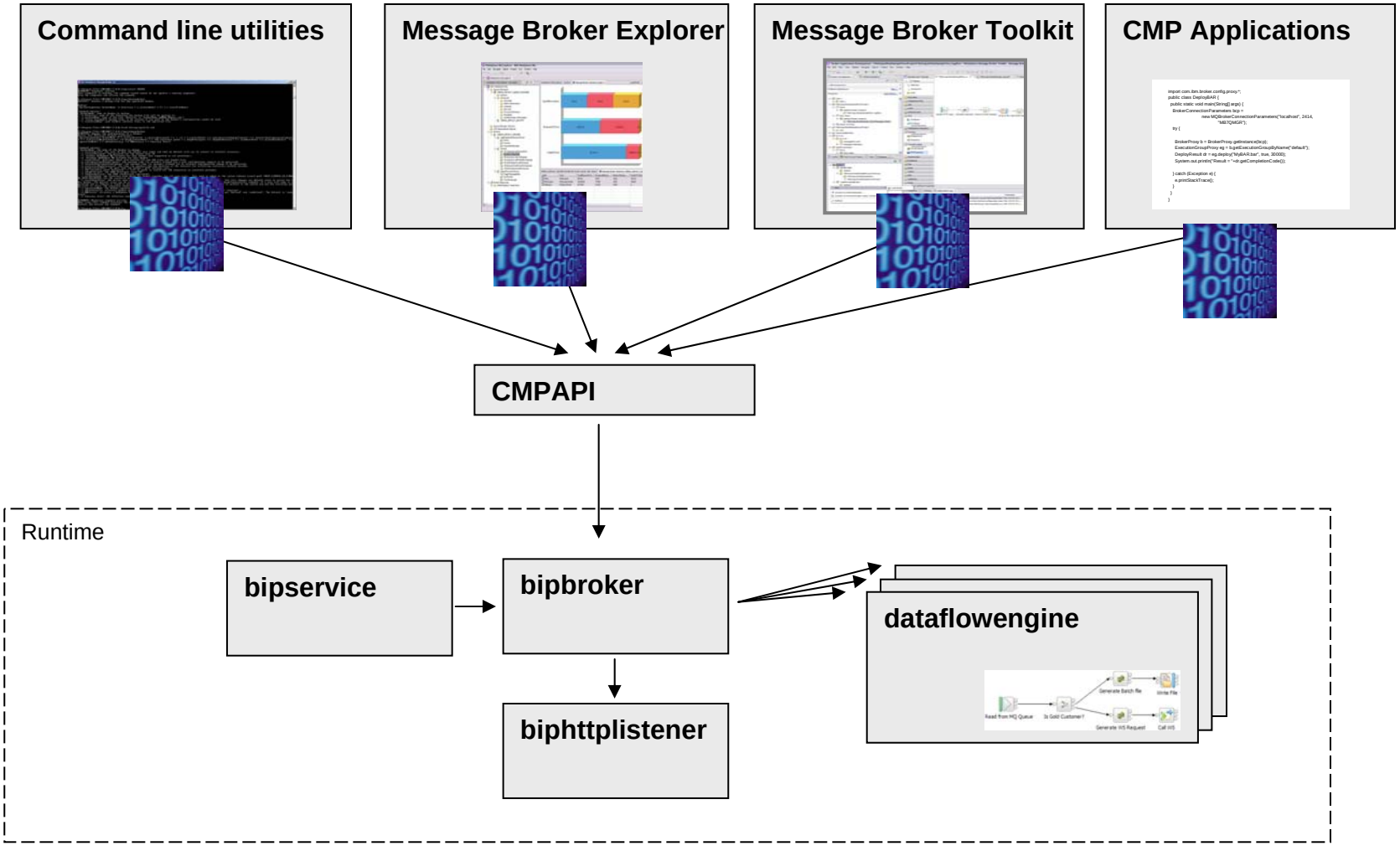
- First check the JOBLOG, Syslog or Event Viewer

+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'.
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/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.

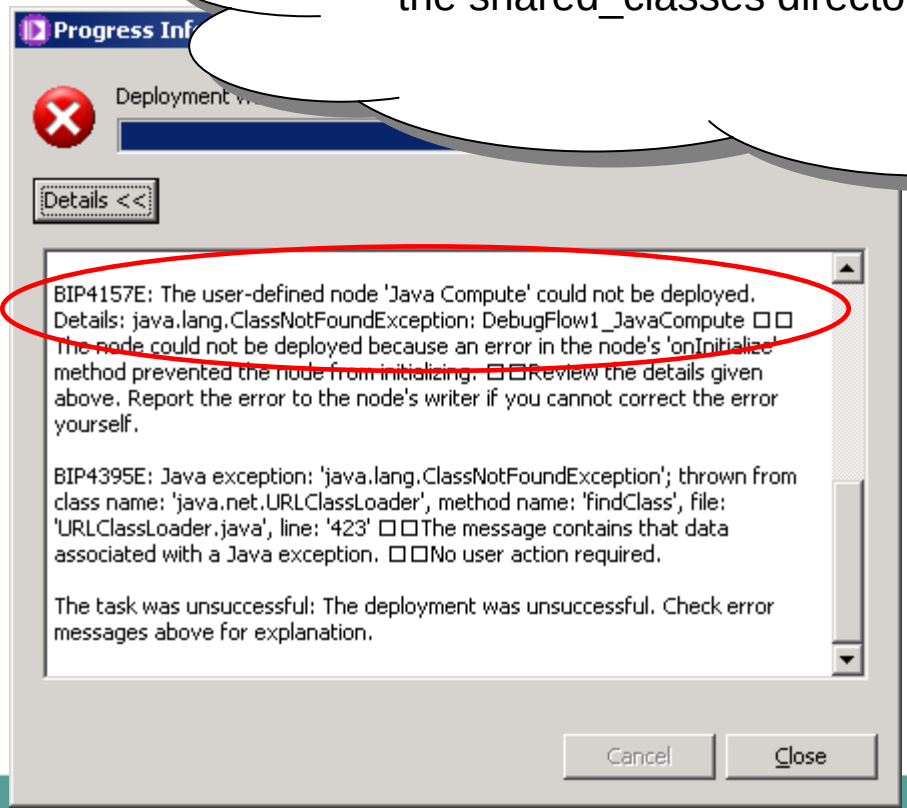
Scenario: Deploy of a Message flow fails

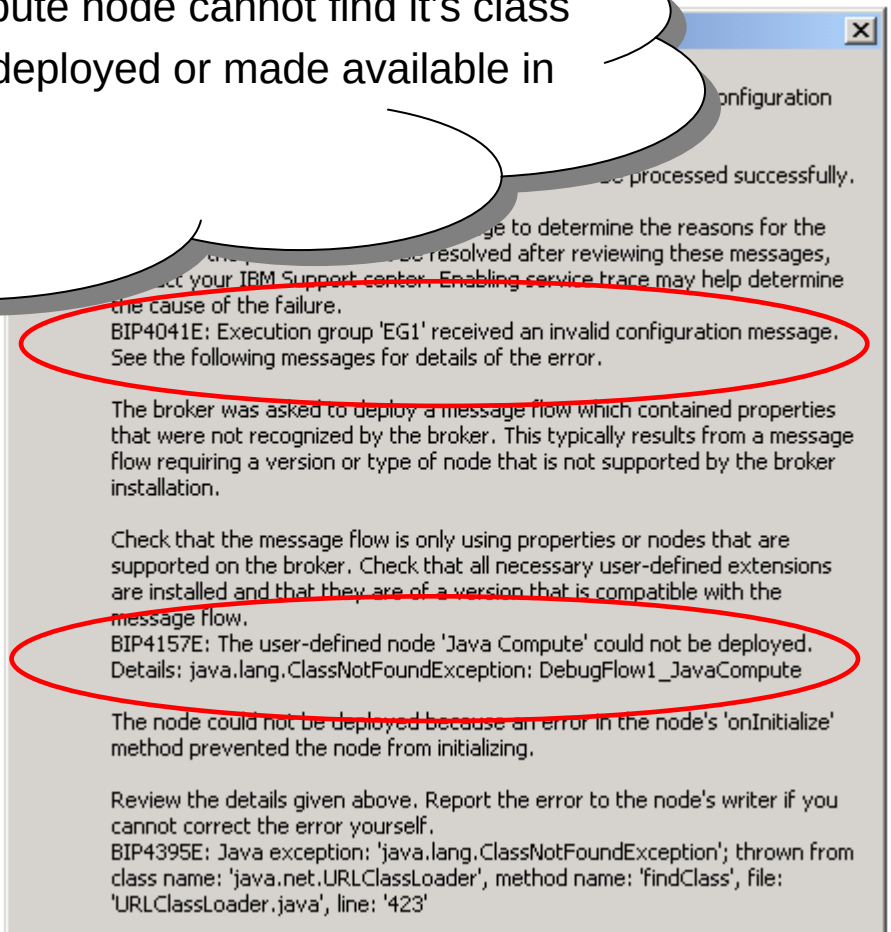


Scenario: Deploy of a Message flow fails

- 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?

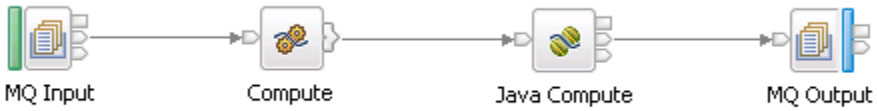




Scenario: Where's my message?

- 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

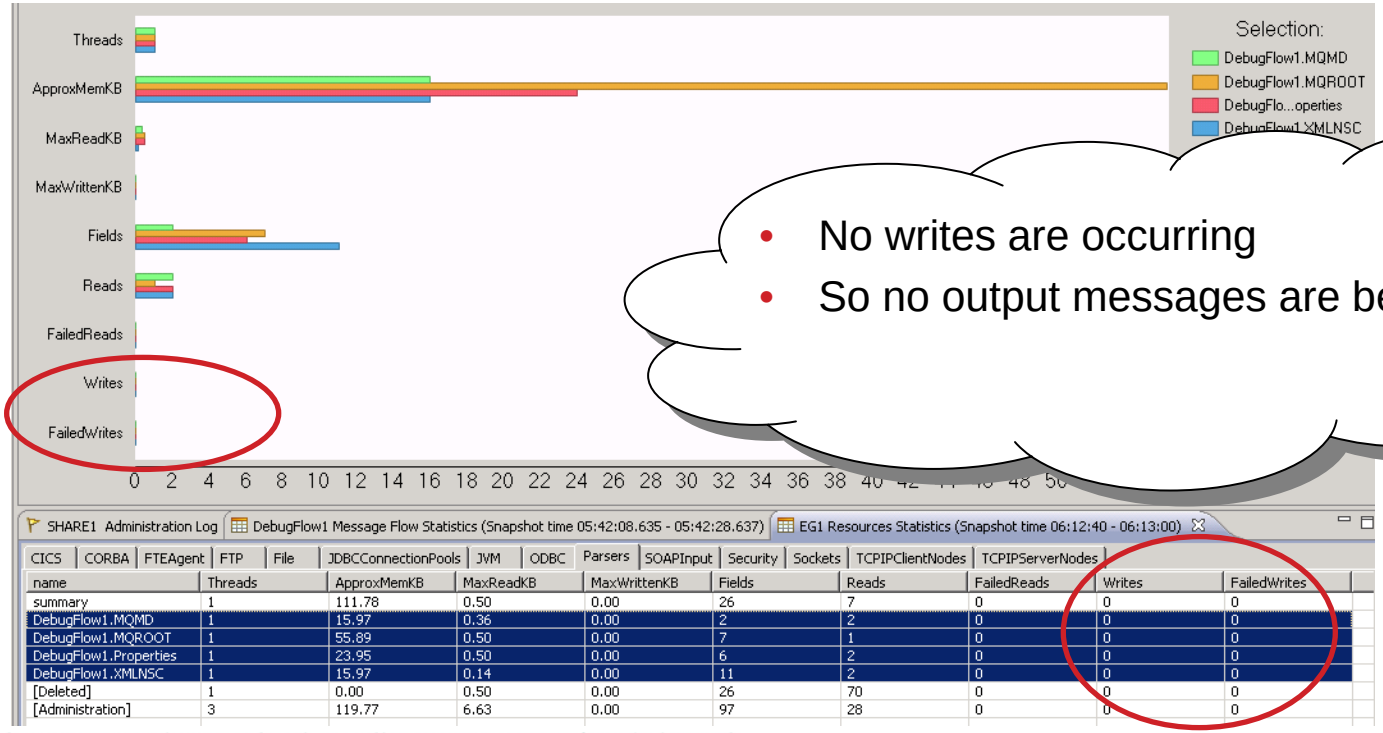
- The message flow



  - 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

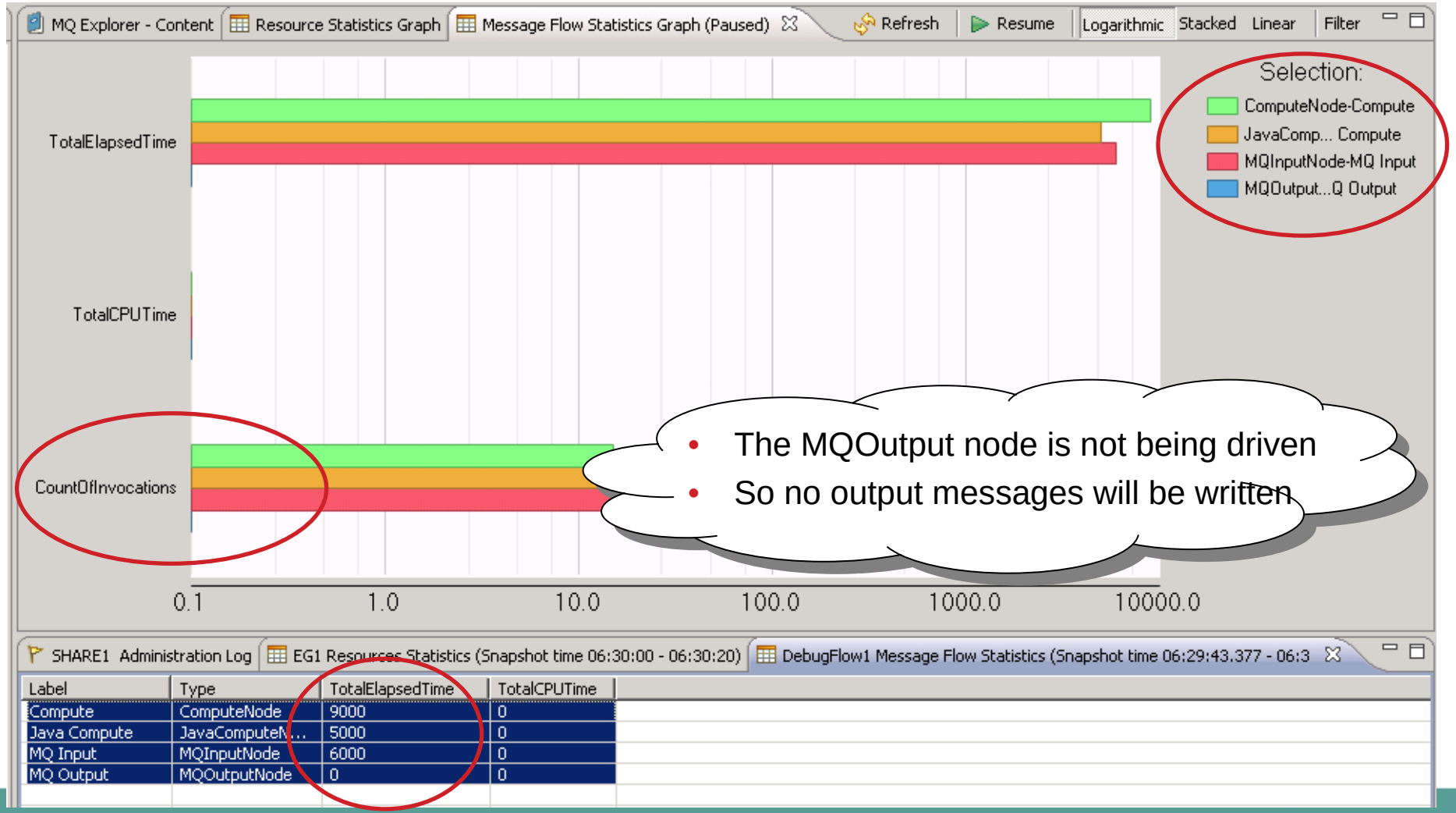


name	Threads	ApproxMemKB	MaxReadKB	MaxWrittenKB	Fields	Reads	FailedReads	Writes	FailedWrites
summary	1	111.78	0.50	0.00	26	7	0	0	0
DebugFlow1.MQMD	1	15.97	0.36	0.00	2	2	0	0	0
DebugFlow1.MQROOT	1	55.89	0.50	0.00	7	1	0	0	0
DebugFlow1.Properties	1	23.95	0.50	0.00	6	2	0	0	0
DebugFlow1.XMLNSC	1	15.97	0.14	0.00	11	2	0	0	0
[Deleted]	1	0.00	0.50	0.00	26	70	0	0	0
[Administration]	3	119.77	6.63	0.00	97	28	0	0	0

- 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?



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?

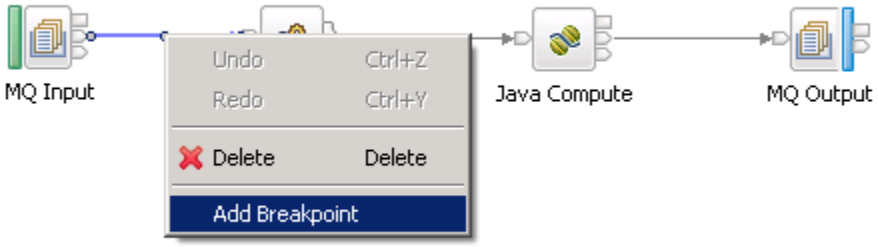
```
UserTrace   BIP2632I: Message received and propagated to 'out' terminal of MQ input node 'DebugFlow1.MQ Input'.
UserTrace   BIP6060I: Parser type "Properties" created on behalf of node 'DebugFlow1.MQ Input' to handle portion of incoming message of length 0 bytes beginning at
      offset '0'.
UserTrace   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.
UserTrace   BIP6061I: 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   BIP2537I: Node 'DebugFlow1.Compute': Executing statement   "BEGIN ... END;" at ('.DebugFlow1_Compute.Main', '2.2').
UserTrace   BIP2537I: Node 'DebugFlow1.Compute': Executing statement   "CopyEntireMessage();" at ('.DebugFlow1_Compute.Main', '3.3').
UserTrace   BIP2538I: Node 'DebugFlow1.Compute': Evaluating expression "CopyEntireMessage()" at ('.DebugFlow1_Compute.Main', '3.8').
UserTrace   BIP2537I: Node 'DebugFlow1.Compute': Executing statement   "BEGIN ... END;" at ('.DebugFlow1_Compute.CopyEntireMessage', '1.39').
UserTrace   BIP2537I: Node 'DebugFlow1.Compute': Executing statement   "SET OutputRoot = InputRoot;" at ('.DebugFlow1_Compute.CopyEntireMessage', '2.3').
UserTrace   BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "InputRoot" at ('.DebugFlo                                                    d to
      "InputRoot". The result was "ROW... Root Element Type=16777216 NameS
UserTrace   BIP2568I: Node 'DebugFlow1.Compute': Copying sub-tree from "
UserTrace   BIP2537I: Node 'DebugFlow1.Compute': Executing statement
      DECIMAL) * CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS "
UserTrace   BIP2539I: Node 'DebugFlow1.Compute': Evaluating ex
      resolved to "OutputRoot.XMLNSC.Order.Item.Price". The resu
UserTrace   BIP2539I: Node 'DebugFlow1.Compute': Evaluating expr
      ('.DebugFlow1_Compute.Main', '4.39'). This resolved to "CAST('7
UserTrace   BIP2539I: Node 'DebugFlow1.Compute': Evaluating expr                                                                    This
      resolved to "OutputRoot.XMLNSC.Order.Item.Quantity". The resul
UserTrace   BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "
      ('.DebugFlow1_Compute.Main', '4.93'). This resolved to "CAST('77' AS INTEGER)
UserTrace   BIP2539I: Node 'DebugFlow1.Compute': Evaluating expression "CAST(OutputRoot.XMLNSC.Order.                    IMAL) *
      CAST(OutputRoot.XMLNSC.Order.Item.Quantity AS INTEGER)" at ('.DebugFlow1_Compute.Main', '4.91'). This resolved to "3 * 77". The result was "231".
UserTrace   BIP2566I: Node 'DebugFlow1.Compute': Assigning value      "231" to field / variable "OutputRoot.XMLNSC.Order.Total".
UserTrace   BIP2537I: Node 'DebugFlow1.Compute': Executing statement   "RETURN TRUE;" at ('.DebugFlow1_Compute.Main', '5.3').
UserTrace   BIP4015I: Message propagated to the 'out' terminal of node 'DebugFlow1.Compute' with the following message trees: ".
UserTrace   BIP3604I: 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.
```

- The last thing the trace shows is the JavaCompute node being invoked
- This never propagates to its output terminal
- Why?

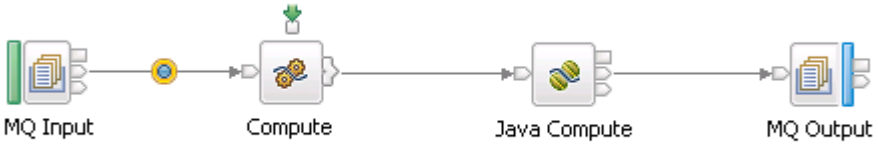
… then the trace ends ….

Scenario: Where's my message?

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



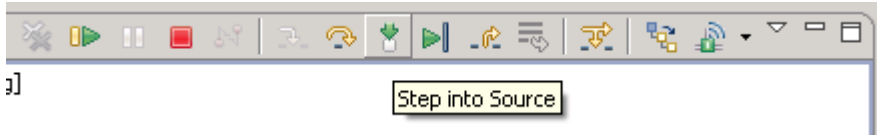
  - Fire in a message and the breakpoint triggers

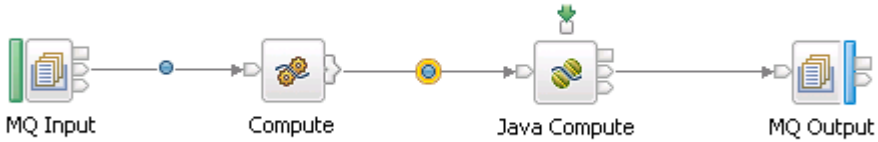


  - We can then step through the message flow and into the ESQL and Java code

Scenario: Where's my message?

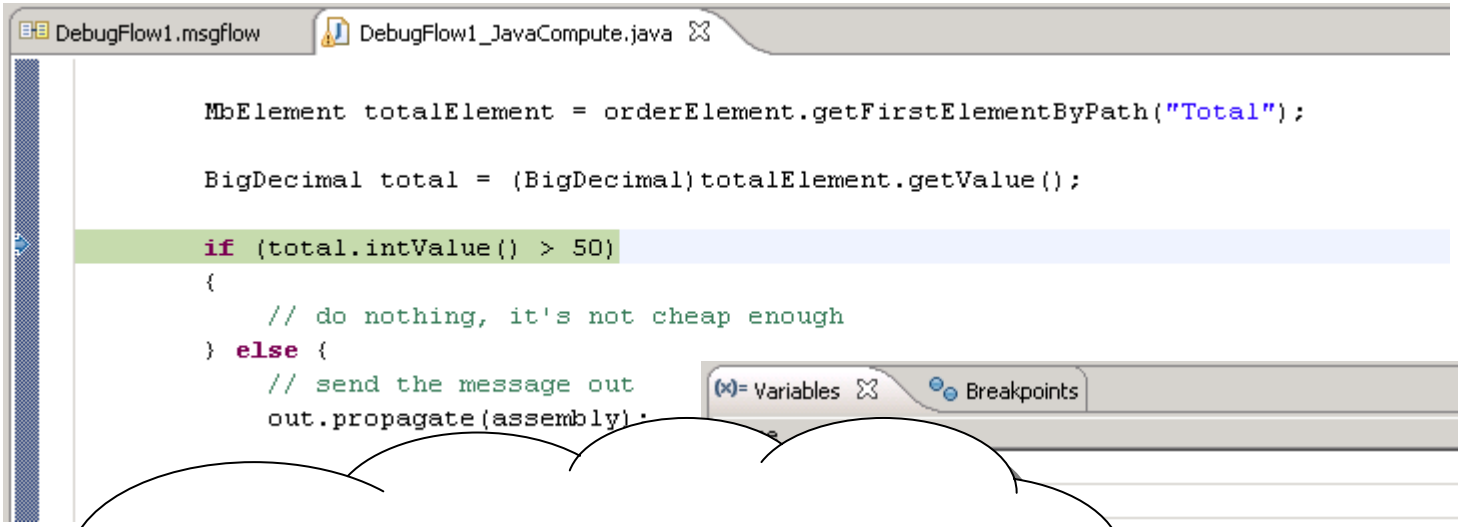
- Message Flow Debugger
  - As the message is never propagated from the JavaCompute node we need to see why
  - When the flow is paused on the connection between the compute and JavaCompute nodes we can step into the source

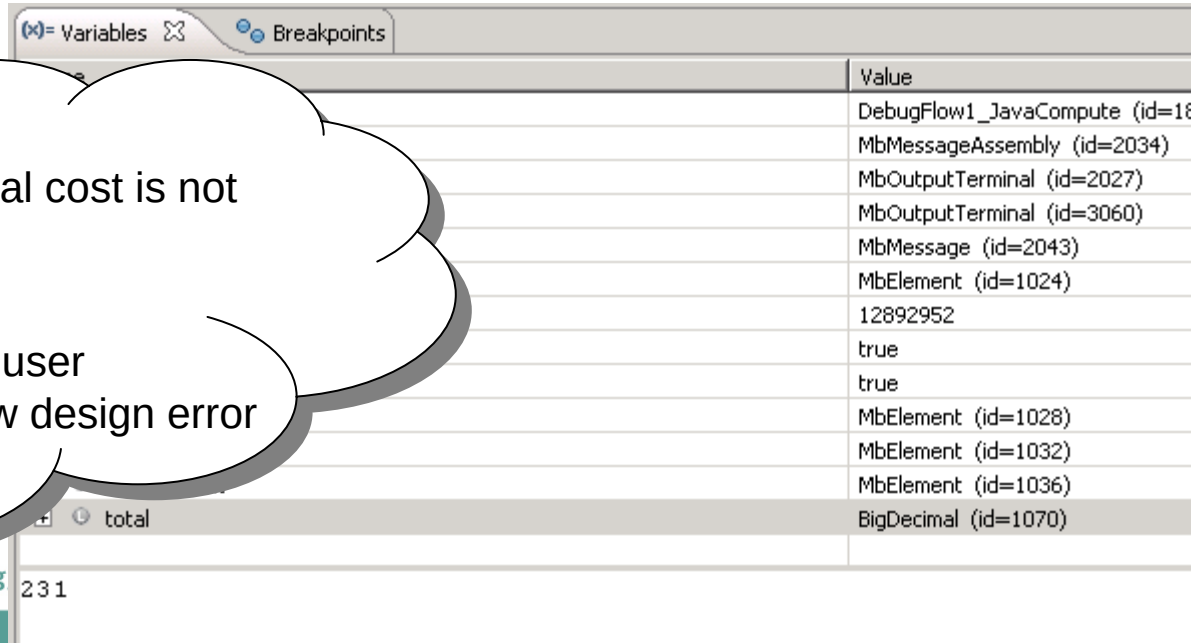




Where's my message?

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





- 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

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

```
:soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server</faultcode>
         <faultstring>BIP3113E: Exception detected in message flow WebServicesFacade.SOAP Input (broker BRK8)</faultstring>
         <detail>
            <Text>BIP3701E: A Java exception was thrown whilst calling the Java JNI method ''Axis2Requester_processResponseMessageSync''. The Java exce
        @: org.apache.axiom.om.impl.builder.StAXOMBuilder.next(StAXOMBuilder.java:252)
        @: org.apache.axiom.om.impl.llom.OMElementImpl.buildNext(OMElementImpl.java:664)
        @: org.apache.axiom.om.impl.llom.OMElementImpl.getFirstOMChild(OMElementImpl.java:681)
        @: org.apache.axiom.om.impl.llom.OMElementImpl.getFirstElement(OMElementImpl.java:1004)
        @: org.apache.axiom.soap.impl.llom.SOAPEnvelopeImpl.getBody(SOAPEnvelopeImpl.java:153)
        @: com.ibm.broker.axis2.Axis2Requester.processResponseMessageSync(Axis2Requester.java:2066)
rame : 1 javax.xml.stream.XMLStreamException: Element type "soapenv:Header" must be followed by either attribute specifications, ">" or "/>".
        @: com.ibm.xml.xlxp2.api.stax.msg.StAXMessageProvider.throwWrappedXMLStreamException(StAXMessageProvider.java:76)
        @: com.ibm.xml.xlxp2.api.stax.XMLStreamReaderImpl.produceFatalErrorEvent(XMLStreamReaderImpl.java:2006)
        @: com.ibm.xml.xlxp2.api.jaxb.JAXBXMLStreamReader.produceFatalErrorEvent(JAXBXMLStreamReader.java:316)
        @: com.ibm.xml.xlxp2.scan.DocumentScanner.reportFatalError(DocumentScanner.java:4942)
        @: com.ibm.xml.xlxp2.scan.DocumentScanner.reportFatalError(DocumentScanner.java:1205)
        @: com.ibm.xml.xlxp2.scan.DocumentScanner.scanAttributes(DocumentScanner.java:2383)
        @: com.ibm.xml.xlxp2.scan.DocumentScanner.scanStartElementCommon(DocumentScanner.java:2299)
        @: com.ibm.xml.xlxp2.scan.DocumentScanner.scanStartElement(DocumentScanner.java:2253)
        @: com.ibm.xml.xlxp2.scan.DocumentScanner.scanContent(DocumentScanner.java:1834)
        @: com.ibm.xml.xlxp2.runtime.VMContext.scanContent(VMContext.java:501)
        @: com.ibm.xml.xlxp2.scan.DocumentScanner.nextEvent(DocumentScanner.java:1276)
        @: com.ibm.xml.xlxp2.api.stax.XMLStreamReaderImpl.next(XMLStreamReaderImpl.java:579)
        @: com.ibm.xml.xlxp2.api.stax.XMLInputFactoryImpl$XMLStreamReaderProxyImpl.next(XMLInputFactoryImpl.java:183)
        @: com.ibm.xml.xlxp2.api.wssec.WSSXMLInputFactory$WSSStreamReaderProxy.next(WSSXMLInputFactory.java:55)
        @: org.apache.axiom.om.impl.builder.StAXOMBuilder.parserNext(StAXOMBuilder.java:622)
        @: org.apache.axiom.om.impl.builder.StAXOMBuilder.next(StAXOMBuilder.java:172)
        @: org.apache.axiom.om.impl.llom.OMElementImpl.buildNext(OMElementImpl.java:664)
        @: org.apache.axiom.om.impl.llom.OMElementImpl.getFirstOMChild(OMElementImpl.java:681)
        @: org.apache.axiom.om.impl.llom.OMElementImpl.getFirstElement(OMElementImpl.java:1004)
        @: org.apache.axiom.soap.impl.llom.SOAPEnvelopeImpl.getBody(SOAPEnvelopeImpl.java:153)
        @: com.ibm.broker.axis2.Axis2Requester.processResponseMessageSync(Axis2Requester.java:2066)''.
orrect the error, and if necessary redeploy the flow. : F:\build\S800_D\src\WebServices\WSLibrary\ImbSOAPRequestHelper.cpp: 2129: ImbSOAPRequestHelper
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
:/soapenv:Envelope>
```

- Error parsing the response

Scenario: Who's misbehaving

- User Trace
  - What did the remote server return?

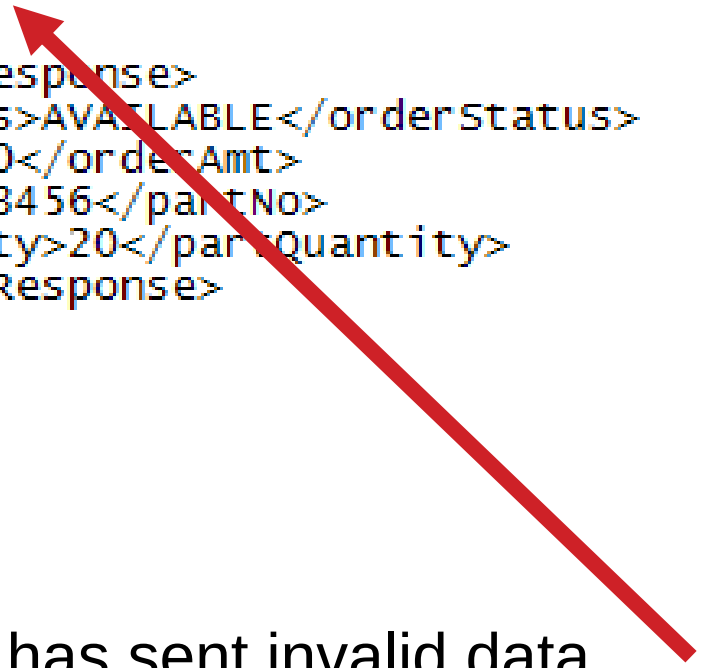
```
        See previous error messages for an indication to the cause of the errors.
7104    RecoverableException  BIP3162S: An HTTP error occurred. The HTTP Request-Line was: ''POST /acmeOrders/WADDR
        ''.
        The HTTP Request Header bitstream (if any) to be used was: 'X'436f6e74656e742d4c656e6774683a203434300d0a43
        The HTTP Request Message Body bitstream (if any) to be used was: 'X'3c736f6170656e763a456e76656c6f70652078
        The HTTP Reply Header bitstream (if any) received from the server was: 'X'485454502f312e3120323030204f4b0d
        The HTTP Reply Message Body bitstream (if any) received from the server was: 'X'3c736f6170656e763a456e7665
        Ensure that the HTTP data is valid.
        See the following messages for information pertaining to this error.
7104    RecoverableException  BIP3701E: A Java exception was thrown whilst calling the Java JNI method ''Axis2Reque
          @: org.apache.axiom.om.impl.builder.StAXOMBuilder.next(StAXOMBuilder.java:252)
          @: org.apache.axiom.om.impl.llom.OMElementImpl.buildNext(OMElementImpl.java:664)
          @: org.apache.axiom.om.impl.llom.OMElementImpl.getFirstOMChild(OMElementImpl.java:681)
```

  - 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

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelop
                  xmlns:ord="http://www.acmeOrders.com/OrderService">
   <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

This was session ???? - The rest of the week ……

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00					Free MQ! - MQ Clients and what you can do with them
09:30	Clustering – the easier way to connect your Queue Managers	MQ on z/OS – vivisection	The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation		
11:00		Diagnosing problems for Message Broker	Lock it down - WebSphere MQ Security	Using IBM WebSphere Application Server and IBM WebSphere MQ Together	Spreading the message – MQ pubsub
12:15	Highly Available Messaging - Rock solid MQ	Putting the web into WebSphere MQ: A look at Web 2.0 technologies	The Doctor is In and Lots of Help with the MQ family - Hands-on Lab		
01:30	WebSphere MQ 101: Introduction to the world's leading messaging provider	What's new in the WebSphere MQ Product Family	Extending IBM WebSphere MQ and WebSphere Message Broker to the Cloud	MQ Performance and Tuning on distributed including internals	
03:00	First steps with WebSphere Message Broker: Application integration for the messy	What's new in Message Broker V8.0	Under the hood of Message Broker on z/OS - WLM, SMF and more	The Do's and Don'ts of z/OS Queue Manager Performance	
04:30	The MQ API for Dummies - the Basics	What the **** is going on in my Queue Manager!?	Diagnosing problems for MQ	Shared Q using 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

QR