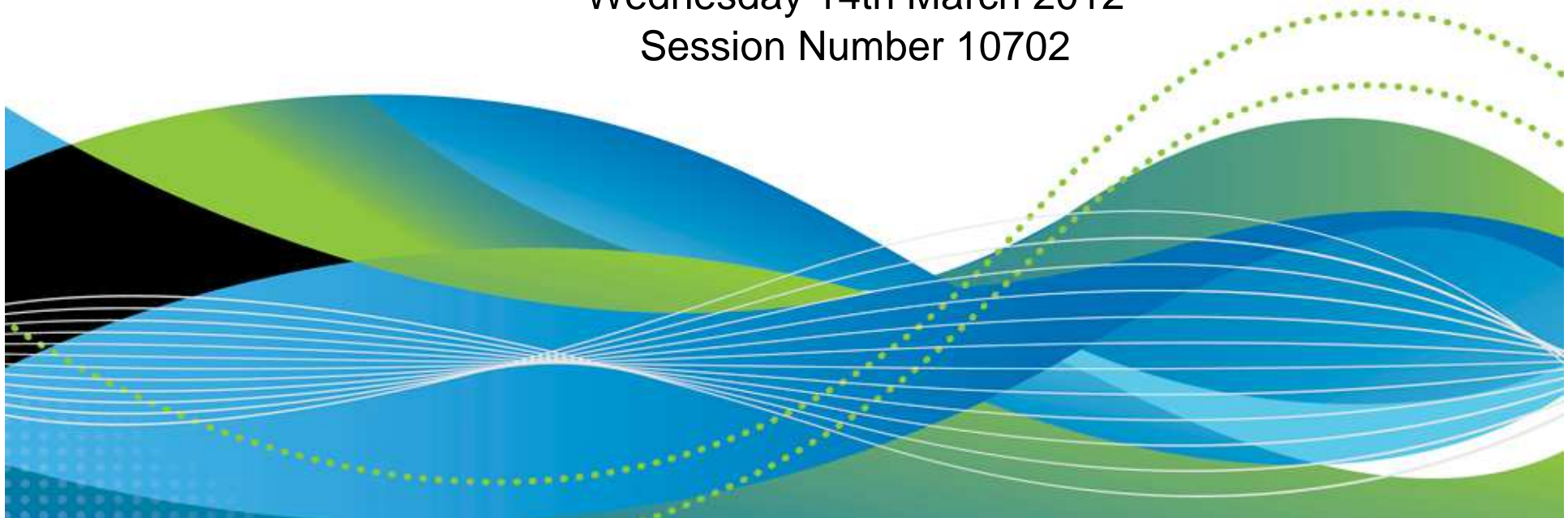


Message Broker Administration

David Coles – WebSphere Message Broker Level 3 Technical Lead,
IBM Hursley – dcoles@uk.ibm.com

Wednesday 14th March 2012
Session Number 10702

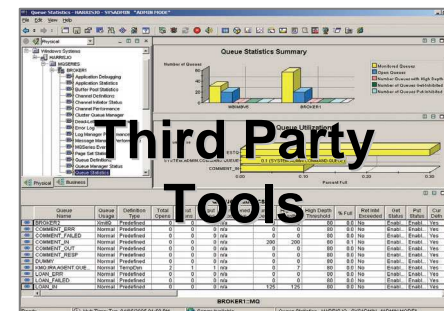
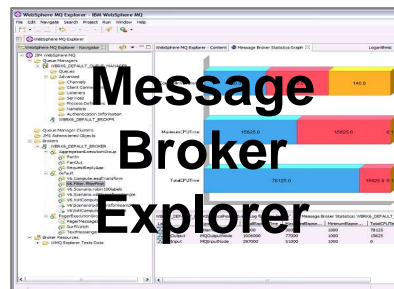
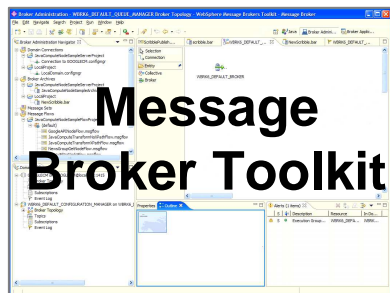


Agenda

- Administering your broker
 - Connecting to a Broker
 - Message Broker Toolkit
 - Message Broker Explorer
 - Command Line
 - Message Broker Administration API (CMPAPI)
- WMB Development Lifecycle and Environments
- Common Administrative Tasks
 - Bringing a new broker online
 - WMB and the Cloud
 - Administrative Security
 - HA & DR
 - Broker Backup/Restore
 - Managing what's deployed
 - Understanding broker behaviour
 - Optimizing and tuning
 - Migration
 - Maintenance
- Broker details
 - Configuration Data
 - Runtime Environment
 - Userids
- Where to look when it all goes wrong!

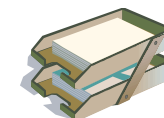
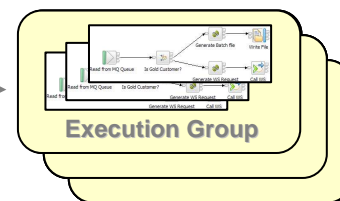


Administering your broker (V7/V8)



```
// instantiate an object that describes the connection
// characteristics to the broker.
BrokerConnectionParameters brokerConnParams =
    new MQBrokerConnectionParameters(brokerHostName, brokerPort, brokerQmgr);
BrokerProxy b = null;
```

Message Broker API (CMP)



Notes: Administering your broker (V7/V8)

- A broker is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.
- Administering brokers and associated broker resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.
- Administration of brokers includes the following tasks:
 - Managing brokers
 - Managing execution groups
 - Managing message flows
 - Developing applications that use the Administration API
 - Accessing Administration log information
 - Changing the location of the work path
 - Managing resources used by brokers
 - Backing up resources
 - Administering Java applications
- These tasks can be performed by using one, or more, of the administrative techniques supported by WebSphere Message Broker:
- Administer the broker by using the WebSphere Message Broker Explorer, the Broker view in the WebSphere Message Broker Toolkit, or the product commands. Alternatively, you can write your own programs to use the Message Broker Administration API (also known as the CMP API).
- Manage the application resources of the broker, which include message flows and message sets, by using the WebSphere Message Broker Toolkit or WebSphere Message Broker Explorer; these two applications connect to the broker by using a WebSphere MQ server connection, which is defined to the broker queue manager when you create the broker.
- The picture on the previous page shows the relationship between the resources that exist at run time, and how they interact with the WebSphere Message Broker Explorer and WebSphere Message Broker Toolkit.
- As well as administering your broker you also have to consider the external resources that your broker requires and that the message flows use such as a Queue Manager, File System, Databases and other flow resources.

Connecting to a broker

- You need to connect to the broker's Queue Manager to perform administration actions
 - MQ Bindings connect to local brokers
 - MQ Client connect to local or remote brokers
- Connecting to a local broker
 - Just the broker name required
 - As it's a local broker we can look everything else up
 - Graphical tools will automatically show local brokers
- Connecting to a remote broker
 - Hostname, port and broker name required
 - More advanced options available
 - SVRCONN channel name if not using the default (SYSTEM.BKR.CONFIG)
 - The class name and JAR file location of a Java security exit if one is required for the channel
 - If using SSL on the channel then the following options are also required:
 - *Cipher Suite, Distinguished Names, CRL Name List, Key Store, and Trust Store*

Brokers View for Application Developers



The screenshot displays the WebSphere Message Broker Toolkit interface. The main window shows a message flow diagram with the following steps: Handle HTTP Input, Calculate response, Remove SOAP Headers, Write to file, and Add SOAP headers. The left-hand side contains a tree view of resources, with the 'Brokers' folder expanded to show the 'FileOutputNodeSampleFlow' project. A red dashed arrow points from the 'FileOutputNodeSampleFlow.msgflow' file in the tree to the 'New' button in the 'Brokers' view. A red circle highlights the 'New' button and the 'Local Broker ...' option. The bottom right corner shows a 'Details' log with the following entries:

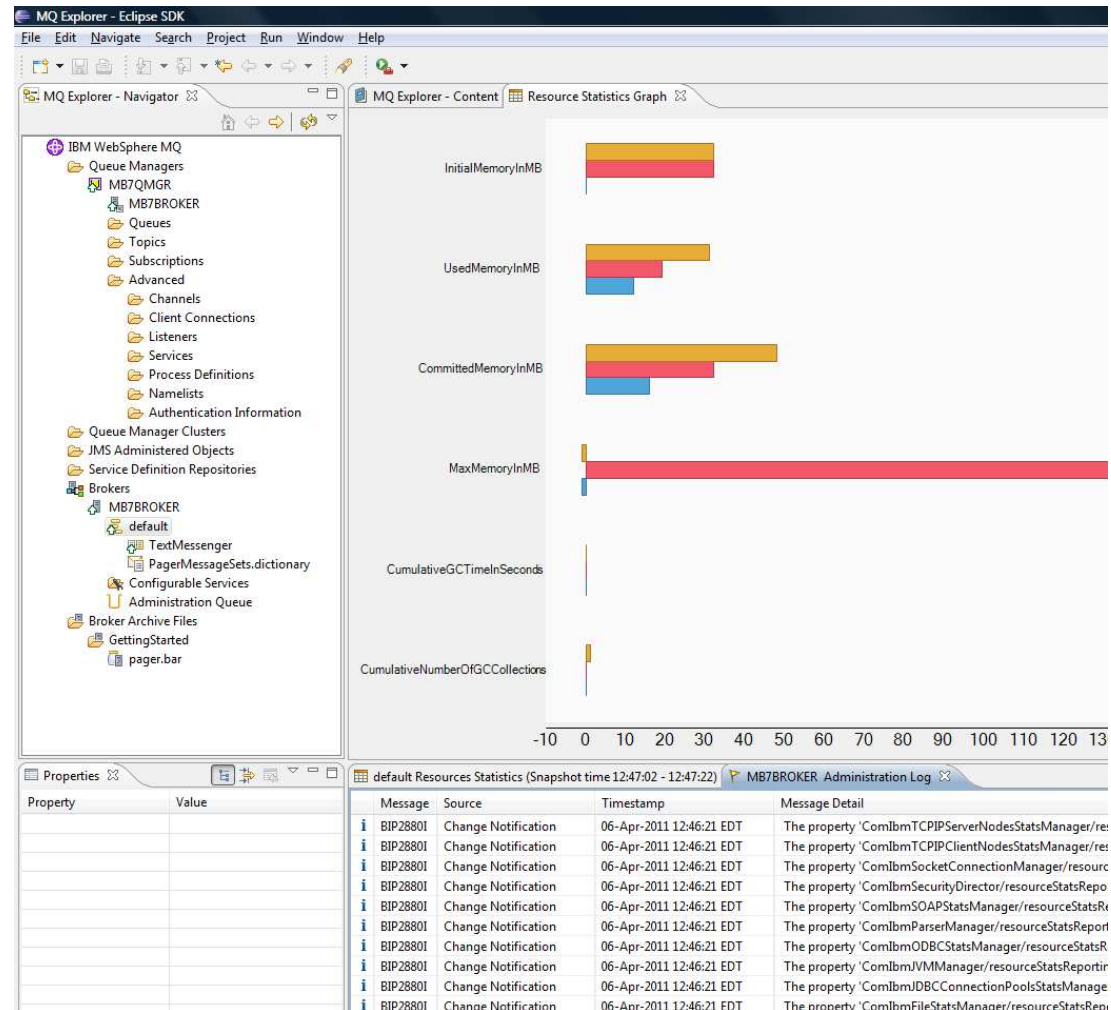
Timestamp	Message
Mon Oct 05 16:1...	[FileOutputNodeSampleFlow.msgflow] has been deployed to execution group FileOutputNodeS:
Mon Oct 05 15:1...	BIP2056I: Broker MB7BROKER successfully processed the entire internal configuration mess
Mon Oct 05 15:1...	BIP4040I: The Execution Group 'FileOutputNodeSampleExecutionGroup' has processed a co

Notes: Message Broker Toolkit

- Use the Brokers view to create and work with brokers in the WebSphere® Message Broker Toolkit.
- The brokers view offers a limited set of administration actions and is primarily aimed at developers who want to deploy and test their message flows.
- By default, the Brokers view is displayed at the bottom of the Broker Application Development perspective in the WebSphere Message Broker Toolkit. If the Brokers view is not displayed, you can show it by clicking Window > Show View > Other > Broker Runtime > Brokers.
- Brokers that are created on the local system are automatically displayed in the Brokers view. You can add remote brokers to the Brokers view. When you open or switch to the Brokers view, the WebSphere Message Broker Toolkit attempts to connect to brokers on the local system, and any remote brokers that have been defined. Warnings and errors might be displayed if the WebSphere Message Broker Toolkit cannot connect to brokers, for example, if the broker is stopped, or the queue manager listener is not running.
- Right-click the Brokers folder in the Brokers view to display the following options:
 - New Local Broker
 - Connect to a Remote Broker
 - If you specify a keystore or truststore in the remote connection information, you are prompted to enter a password for the keystore or truststore when you connect to the remote broker.
 - Connect to a Remote Broker Using *.broker File
 - Refresh

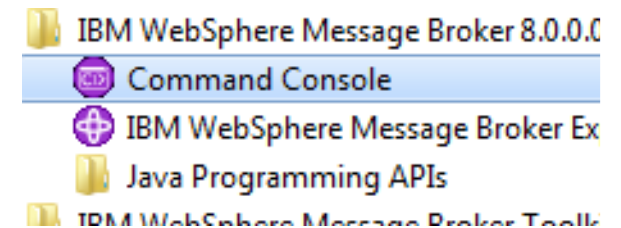
Message Broker Explorer (MBX)

- 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
 - Activity Log
 - Security & Policy Set editors



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.

Notes: Command-line

- Message Broker ships with commands for performing configuration and administration actions which complement and extend our graphical administration options
- On distributed platforms you need to apply the mqsiprofile to be able to run the commands
- On z/OS the commands are available as jobs, console commands, or both
 - During broker customization you should copy the sample jobs from the SBIPSAMP/SBIPPROC libraries to the broker's component dataset
 - The commands run by jobs are run as the user submitting the job
 - Unless a USER=<user> statement is added to the JCL
 - Console commands are run by the broker userid and are run inside the main broker started task address space
- Two types of commands:
 - Java based ones which used to be the commands that talked to the Config Manager
 - These use the CMP API
 - Can work with local and remote brokers
 - Eg: mqsideploy, mqsiexecuteexecutiongroup, mqsisstartmsgflows
 - You get the full brokerSpec options for connecting
 - Older commands which just work with local brokers
 - Eg: mqsisstart, mqsisstop, mqsisreload, mqsichangeproperties

Command-line examples



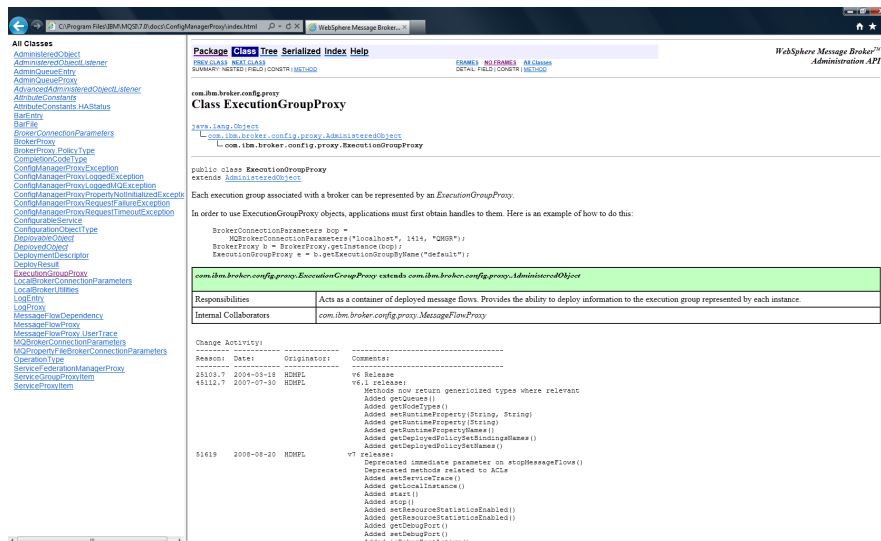
- `mqsilist`
 - Displays a list of local brokers
 - Displays detailed information about brokers and their deployed resources via `-d` option
 - This works with local and remote brokers

```
BIP1288I: Message flow 'simpleflow' on execution group 'ello' is running.
Additional thread instances: '0'
Deployed: '24/07/09 16:37' in Bar file 'C:\My Documents\BAR Files\test.bar'
Last edited: '08/08/07 17:42'
User-defined property names:
Keywords:
  Author = 'Matt'
  Information = 'This flow simply removes messages from SYSTEM.DEFAULT.LOCAL.QUEUE'
  Usage = 'This usage is buried inside the CMF' VERSION = 'v1.1'
```

- `mqsistart / mqsistop`
 - Use to stop and start brokers
- `mqsireload`
 - Use to restart a restart a broker, or a single execution group to pickup configuration changes

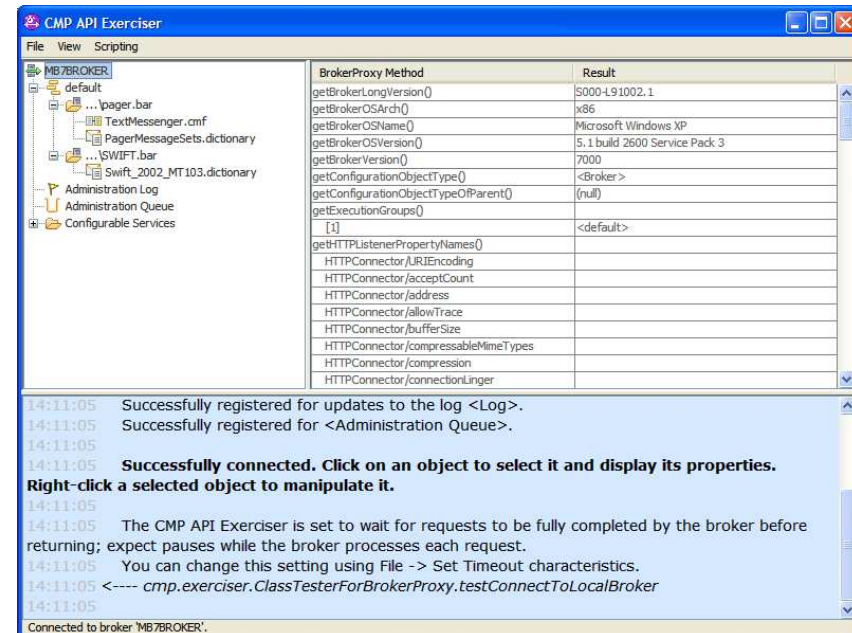
Message Broker API (CMP)

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



The screenshot shows the Java IDE documentation for the `com.ibm.broker.config.proxy.ExecutionGroupProxy` class. It includes a package tree, class hierarchy, and a table of responsibilities and collaborators.

Responsibilities	Internal Collaborators
Acts as a container of deployed message flows. Provides the ability to deploy information to the execution group represented by each instance.	<code>com.ibm.broker.config.proxy.MessageFlowProxy</code>



The screenshot shows the CMP API Exerciser application. It displays a tree view of the MB7BROKER and a table of BrokerProxy methods and their results.

BrokerProxy Method	Result
<code>getBrokerLongVersion()</code>	5000-L9.1002.1
<code>getBrokerOSArch()</code>	x86
<code>getBrokerOSName()</code>	Microsoft Windows XP
<code>getBrokerOSVersion()</code>	5.1 build 2600 Service Pack 3
<code>getBrokerVersion()</code>	7000
<code>getConfigurationObjectType()</code>	<Broker>
<code>getConfigurationObjectTypeOfParent()</code>	(null)
<code>getExecutionGroups()</code>	
[1]	<default>
<code>getHTTPListenerPropertyNames()</code>	
<code>HTTPConnector/JURIEncoding</code>	
<code>HTTPConnector/acceptCount</code>	
<code>HTTPConnector/address</code>	
<code>HTTPConnector/allowTrace</code>	
<code>HTTPConnector/bufferSize</code>	
<code>HTTPConnector/compressableMimeTypes</code>	
<code>HTTPConnector/compression</code>	
<code>HTTPConnector/connectionLinger</code>	

Log messages:

```

14:11:05 Successfully registered for updates to the log <Log>.
14:11:05 Successfully registered for <Administration Queue>.
14:11:05
14:11:05 Successfully connected. Click on an object to select it and display its properties. Right-click a selected object to manipulate it.
14:11:05
14:11:05 The CMP API Exerciser is set to wait for requests to be fully completed by the broker before returning; expect pauses while the broker processes each request.
14:11:05 You can change this setting using File -> Set Timeout characteristics.
14:11:05 <--- cmp.exerciser.ClassTesterForBrokerProxy.testConnectToLocalBroker
14:11:05
Connected to broker 'MB7BROKER'.
  
```



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

CMPAPI Example



- This simple example connects to a broker on the local machine and deploys a bar file and displays the result

```
import com.ibm.broker.config.proxy.*;
public class DeployBAR {
    public static void main(String[] args) {
        BrokerConnectionParameters bcp =
            new MQBrokerConnectionParameters("localhost", 2414, "MB7QMGR");
        try {

            BrokerProxy b = BrokerProxy.getInstance(bcp);
            while(!b.hasBeenPopulatedByBroker()) {}
            ExecutionGroupProxy eg = b.getExecutionGroupByName("default");
            DeployResult dr = eg.deploy("MyBAR.bar", true, 30000);
            System.out.println("Result = "+dr.getCompletionCode());

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

CMP API Exerciser



The screenshot shows the 'CMP API Exerciser' application window. The title bar reads 'CMP API Exerciser'. The menu bar includes 'File', 'View', and 'Scripting'. The left pane shows a tree view of the broker's structure, with 'MB7BROKER' selected. Under 'MB7BROKER', there is a 'default' folder containing several objects: '... \pager.bar' (with sub-objects 'TextMessenger.cmf' and 'PagerMessageSets.dictionary'), '... \SWIFT.bar' (with sub-object 'Swift_2002_MT103.dictionary'), 'Administration Log', 'Administration Queue', and 'Configurable Services'. The right pane displays a table of properties for the selected 'MB7BROKER' object.

BrokerProxy Method	Result
getBrokerLongVersion()	S000-L91002.1
getBrokerOSArch()	x86
getBrokerOSName()	Microsoft Windows XP
getBrokerOSVersion()	5.1 build 2600 Service Pack 3
getBrokerVersion()	7000
getConfigurationObjectType()	<Broker >
getConfigurationObjectTypeOfParent()	(null)
getExecutionGroups()	
[1]	<default>
getHTTPListenerPropertyNames()	
HTTPConnector/URIEncoding	
HTTPConnector/acceptCount	
HTTPConnector/address	
HTTPConnector/allowTrace	
HTTPConnector/bufferSize	
HTTPConnector/compressableMimeTypes	
HTTPConnector/compression	
HTTPConnector/connectionLinger	

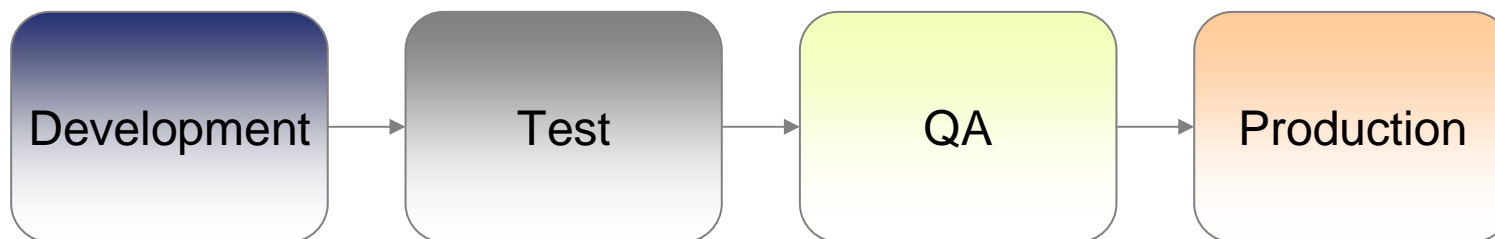
Below the table, a log window displays the following messages:

```
14:11:05 Successfully registered for updates to the log <Log>.
14:11:05 Successfully registered for <Administration Queue>.
14:11:05
14:11:05 Successfully connected. Click on an object to select it and display its properties.
Right-click a selected object to manipulate it.
14:11:05
14:11:05 The CMP API Exerciser is set to wait for requests to be fully completed by the broker before
returning; expect pauses while the broker processes each request.
14:11:05 You can change this setting using File -> Set Timeout characteristics.
14:11:05 <---- cmp.exerciser.ClassTesterForBrokerProxy.testConnectToLocalBroker
14:11:05
```

At the bottom of the window, a status bar indicates: 'Connected to broker 'MB7BROKER'.'

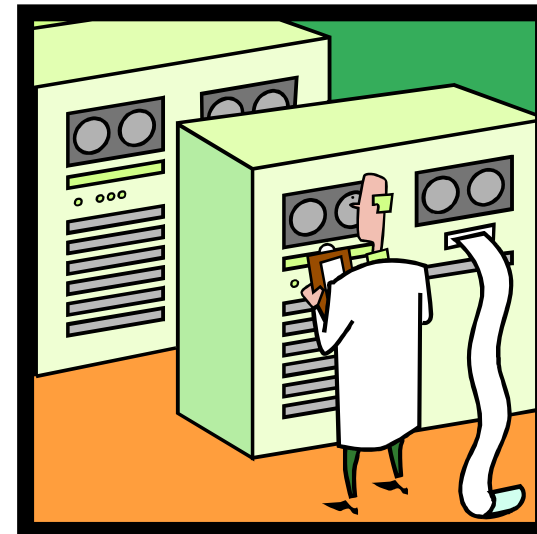
WMB Development Lifecycle and Environments

- Application Developer
 - Develops message flows, message models etc.
 - Unit Tests on local machine
 - Creates Broker Archive (BAR) files containing required artefacts
- Administrator
 - Customizes BAR for target environment (message flow properties including queues, database names etc.)
 - Deploys BAR to target broker
 - Broker management and operational control




Common Administrative Tasks

- Planning and configuration
 - Bringing a new broker online
 - Securing a broker
 - Making a broker highly available
 - Planning for disaster recovery
- Managing brokers
 - Managing what's deployed
 - Understanding broker behaviour
 - Optimizing and tuning
 - Migration
 - Maintenance

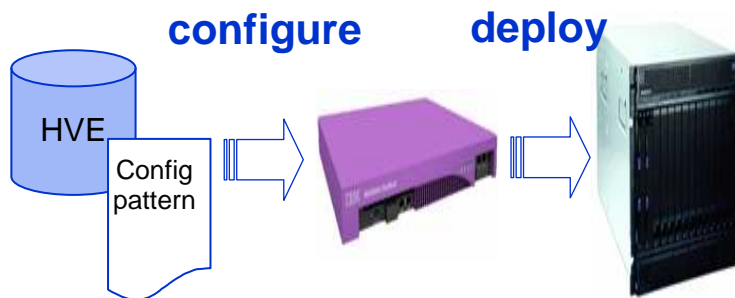
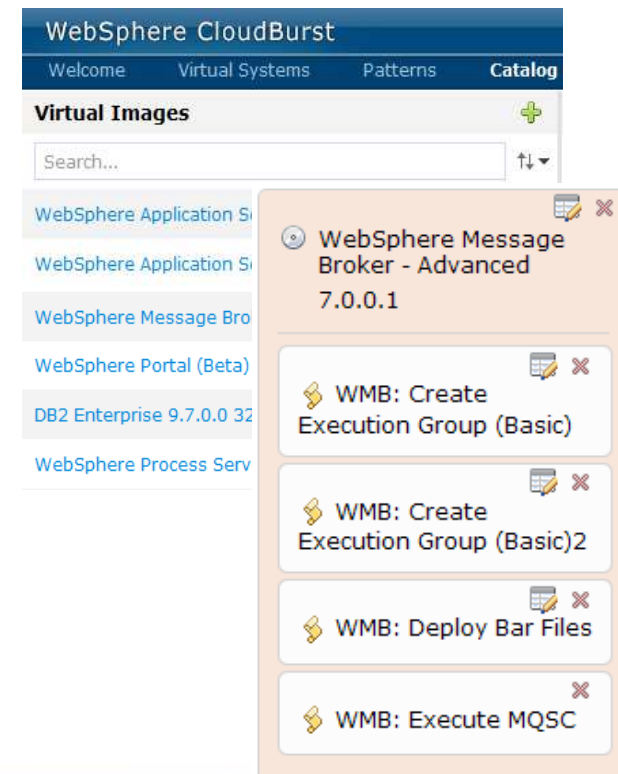


Bringing a new broker online

- Typical steps required
 - Preconfiguration (e.g. OS, userids)
 - Installation (MQ, MB, additional software)
 - Creating the broker
 - Creating execution groups
 - Deploying BAR files
 - Additional configuration (e.g. security, userids)
- Ensure that the environment is reproducible 
 - Documentation
 - Scripting
 - Virtualization

WMB and the Cloud

- Consider virtualization technologies for new environments
 - Makes it easy to provision new systems (and restore known state!)
 - WMB supports the use of them
 - Understand maintenance and performance aspects
- WMB Hypervisor Edition
- IBM Workload Deployer appliance

Securing a broker

- Administrative security in V7/V8 allows 3 levels of authorization:
 - Reading
 - Writing
 - Executing (i.e. starting and stopping)
- On two object types:
 - Broker
 - Execution Group
- Administrative Security is not enabled by default
- Access controlled using MQ queues on the broker's queue manager
- For those migrating from 6.x there is guidance provided for migration from CM ACLs
 - Though there is not a one-to-one mapping

Security Queues

SYSTEM.BROKER.AUTH
SYSTEM.BROKER.AUTH.<egname>

The screenshot displays the 'MB7QMGR - SYSTEM.BROKER.AUTH - Manage Authority Records' window. A 'New Authorities' dialog box is open, showing configuration for a 'User' entity named 'MyBrokerUser' with 'Queue' object type under the 'SYSTEM.BROKER.AUTH' profile. The 'MQI' section has 'Inquire', 'Put', and 'Set' checked. A red circle highlights these three options, with an arrow pointing to a legend: '+inq = Read', '+put = Write', and '+set = Execute'. Another red circle highlights the 'Inquire', 'Put', and 'Set' columns in the background table, which also have green checkmarks.

Name	Browse	Change	Clear	Delete	Display	Get	Inquire	Put	Set
Matt@LUCAS	✓	✓	✓	✓	✓	✓	✓	✓	✓

Legend:
+inq = Read
+put = Write
+set = Execute

Required Task Authorizations



Tasks	Queue Names	
	SYSTEM.BROKER.AUTH	SYSTEM.BROKER.AUTH.EG
Set broker properties	R+W	
View broker properties	R	
Create or delete configurable services	R+W	
Set configurable services properties	R+W	
View configurable services properties	R	
Create or delete execution groups	R+W	
Rename execution groups	R+W	
List execution groups	R	
Start or stop execution groups	R X ¹	X ¹
Set execution group properties	R	W
View execution group properties	R	R
Start or stop resource statistics collection	R	X
Report resource statistics	R	R
Deploy	R	W
List message flows and other deployed objects	R	R
Start or stop message flows	R	X
Delete resources from an execution group	R	W

Note: X¹ Execute access is required on the broker or on an individual execution group

http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/bp43530_.htm

Required Command Authorizations

Command	Queue Names	
	SYSTEM.BROKER.AUTH	SYSTEM.BROKER.AUTH.EG
mqsichangeresourcestats	R	X
mqsicreateexecutiongroup	R+W	
mqsdeleteexecutiongroup	R+W	
mqsdeploy	R	W
mqsilist	R	R ¹
mqsimode	R (to display) R+W (to change)	
mqsireloadsecurity	R	W
mqsireportresourcestats	R	R
mqsistartmsgflow	R	X
mqsistopmsgflow	R	X


Note: R¹ You require read access on any execution groups for which you wish to display information

http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/bp43540_.htm

Making the broker highly available

- How do I ensure that the broker is continually processing messages?
 - Active/Active vs. Active/Passive
 - Agree SLAs with the business (% uptime)
- The broker includes restart recovery, but this is usually not sufficient on its own
- Two main options
 - Third-party solutions (e.g. VCS)
 - Multi-instance queue managers and brokers
 - Run your broker as an MQ service

Planning for disaster recovery

- What would you do if your primary WMB location goes down?
- Distribute WMB to multiple sites if possible
 - This introduces data replication and latency concerns
- Keep DR concerns separate from HA! 
 - HA: Systems at a single site with a single configuration
 - DR: Systems at multiple sites with replicated configurations
 - An HA failover can be a planned activity; DR is unplanned.



Backing Up

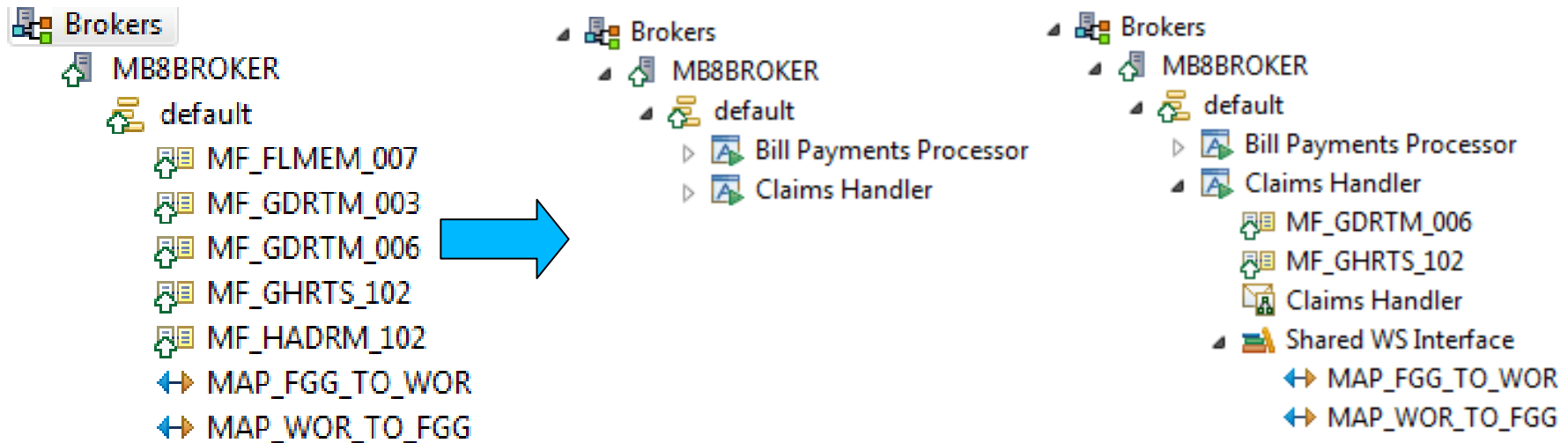


- Regularly use *mqsibackupbroker* to backup the broker's configuration
 - Ideally, after all configuration changes
 - Backup an active broker as long as it's not processing configuration changes
- Consider required resources, for example:
 - Database tables
 - Source artefacts (message flows, BAR files)
- Ensure regular restore testing
 - Use *mqsirestorebroker* to restore a broker's configuration
 - Use the backup file that is created to restore a broker in an identical operating environment
 - The operating system must be at the same level, and the broker and queue manager names must be identical.
- WMB V7 pattern to reconstitute message flows from a running broker





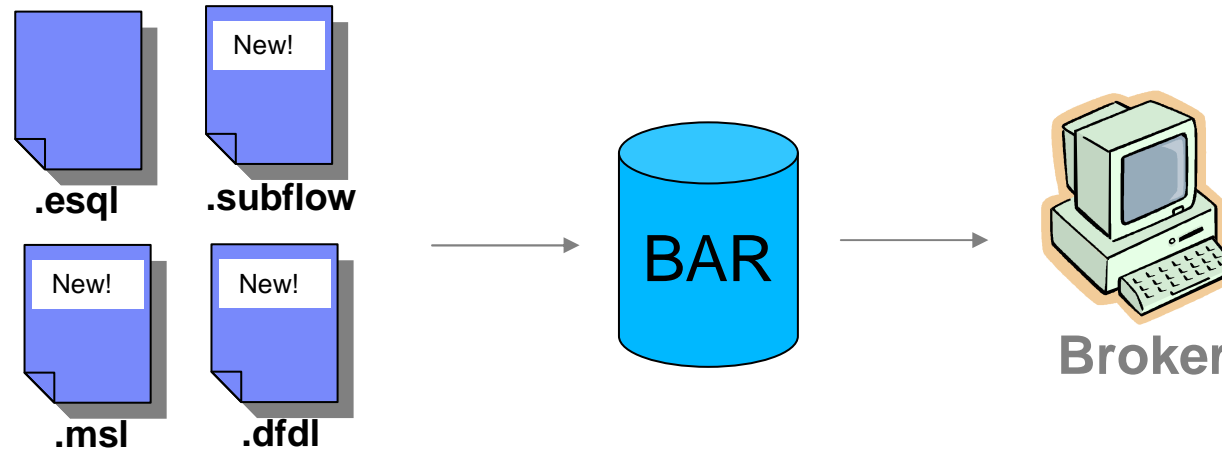
Managing what's deployed



- It can often be difficult to understand what individual deployed resources are used for
- Applications and libraries can allow you to understand precisely why each file is there
 - Application: Encapsulates a single use case or scenario
 - Library: Promotes re-use of a shared set of files
- Concepts are shared between WMB developers and administrators
 - The WMB developer chooses to create an application or library; the collection is then carried all the way through to the broker runtime



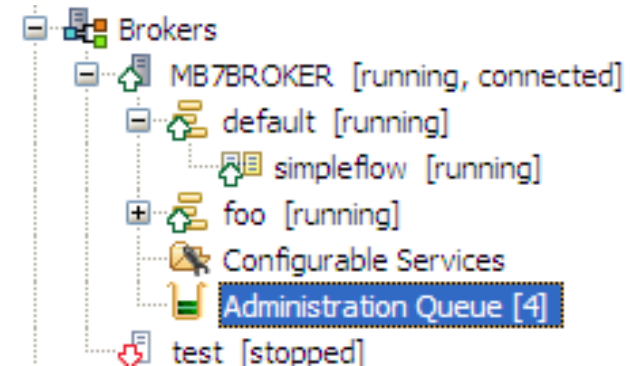
Managing shared resources



- V8 also allows you to deploy source files directly to the broker runtime
 - Subflows (.subflow), ESQL (.esql) ... in addition to CMF etc.
- This means that you can manage subflows and ESQL independently of the message flows that use them
- Update a shared service once only per execution group
- After deployment of a new version, all affected message flows pick up any changes with the next message

Understanding broker behaviour

- The tools include a lot of information that is useful to the administrator, for example:
 - Administration queue and log, Message flow and resource statistics
 - Warnings when features that affect performance are enabled



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 BIP2871I	Administration Request	09-Oct-2009 10:17:10	The request made by user 'Matt' to 'stop' the resource 'TextMessenger' of type 'MessageFlow' on parent 'default' of typ

PagerExecutionGroup Resources Statistics (Snapshot time 16:13:31 - 16:13:51)

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	0.54	0.43	93	24	0	16				
TextMessenger.MQMD		1	15.97	0.36	0.43	2	8	0	4				
TextMessenger.MOROOT		1	55.89	0.54	0.00	25	4	0	4				

- Use this information to understand recent configuration changes, how the broker is performing, connected endpoints, etc.

Resource Statistics XML



- Based on existing accounting and statistics framework
 - Sample XML published to `$$SYS/Broker/<broker>/ResourceStatistics/<eg>` :

```
<ResourceStatistics brokerLabel="STRESS1" brokerUUID="1e9e4ba1-020a-4f91-ab07-7f2006e94e5b" executionGroupName="eg.RDS.SOCKET.1" executionGroupUUID="15cccf2f-2401-0000-0080-8e68043b2078" startDate="2009-10-08" startTime="10:29:26.198" endDate="2009-10-08" endTime="10:29:46.270">
  <ResourceType name="Sockets">
    <resourceIdentifier name="summary" TotalMessages="854" MinimumMessagesPerMinute="1179" MaximumMessagesPerMinute="1383" AverageMessagesPerMinute="5328" TotalBytesSent="1066547" MinimumBytesSentPerSecond="11571" MaximumBytesSentPerSecond="41756" AverageBytesSentPerSecond="42334" TotalBytesReceived="282058" MinimumBytesReceivedPerSecond="11455" MaximumBytesReceivedPerSecond="11455" AverageBytesReceivedPerSecond="9264" MinimumBytesSentPerMessage="835627" MaximumBytesSentPerMessage="835627" AverageBytesSentPerMessage="2506891" MinimumBytesReceivedPerMessage="54338" MaximumBytesReceivedPerMessage="54338" AverageBytesReceivedPerMessage="163014"/>
    <resourceIdentifier name="localhost.7800" TotalMessages="461" MinimumMessagesPerMinute="1383" MaximumMessagesPerMinute="1383" AverageMessagesPerMinute="1383" TotalBytesSent="231422" MinimumBytesSentPerSecond="11571" MaximumBytesSentPerSecond="11571" AverageBytesSentPerSecond="11571" TotalBytesReceived="229117" MinimumBytesReceivedPerSecond="11455" MaximumBytesReceivedPerSecond="11455" AverageBytesReceivedPerSecond="11455" MinimumBytesSentPerMessage="502" MaximumBytesSentPerMessage="502" AverageBytesSentPerMessage="502" MinimumBytesReceivedPerMessage="497" MaximumBytesReceivedPerMessage="497" AverageBytesReceivedPerMessage="497"/>
    <resourceIdentifier name="localhost.7900" TotalMessages="393" MinimumMessagesPerMinute="1179" MaximumMessagesPerMinute="1179" AverageMessagesPerMinute="1179" TotalBytesSent="835125" MinimumBytesSentPerSecond="41756" MaximumBytesSentPerSecond="41756" AverageBytesSentPerSecond="41756" TotalBytesReceived="53841" MinimumBytesReceivedPerSecond="2692" MaximumBytesReceivedPerSecond="2692" AverageBytesReceivedPerSecond="2692" MinimumBytesSentPerMessage="2125" MaximumBytesSentPerMessage="2125" AverageBytesSentPerMessage="2125" MinimumBytesReceivedPerMessage="137" MaximumBytesReceivedPerMessage="137" AverageBytesReceivedPerMessage="137"/>
  </ResourceType>
</ResourceStatistics>
```



Understanding broker behaviour (V8)



The screenshot shows the MB8BROKER interface. On the left, a tree view shows the 'default' message flow selected, with a context menu open showing 'Open Activity Log' as the active option. On the right, the 'Request Activity Log' window displays a table of activity messages.

Message ...	Timestamp	Message Summary
BIP12001I	20-Jun-2011 06:28:07.03...	Connected to JMS provider 'WebSphere_MQ'
BIP12002I	20-Jun-2011 06:28:07.06...	Created a 'Transaction_None' session for JMS provider 'WebSphere_MQ'
BIP12004I	20-Jun-2011 06:28:07.17...	Created JMS producer for destination 'ACTIVITYLOG_JMS_REC'
BIP12014E	20-Jun-2011 06:28:07.24...	Failed to send message to 'ACTIVITYLOG_JMS_REC'
BIP12004I	20-Jun-2011 06:28:07.17...	Created JMS producer for destination 'ACTIVITYLOG_JMS_REC'
BIP12014E	20-Jun-2011 06:28:07.25...	Failed to send message to 'ACTIVITYLOG_JMS_REC'
BIP12001I	20-Jun-2011 06:28:09.77...	Connected to JMS provider 'WebSphere_MQ'
BIP12004I	20-Jun-2011 06:28:09.80...	Created JMS producer for destination 'ACTIVITYLOG_JMS_REC'
BIP12014E	20-Jun-2011 06:28:09.81...	Failed to send message to 'ACTIVITYLOG_JMS_REC'
BIP12004I	20-Jun-2011 06:28:07.20...	Created JMS producer for destination 'ACTIVITYLOG_JMS_REC'
BIP12014E	20-Jun-2011 06:28:07.24...	Failed to send message to 'ACTIVITYLOG_JMS_REC'
BIP12004I	20-Jun-2011 06:28:09.81...	Created JMS producer for destination 'ACTIVITYLOG_JMS_REC'
BIP12014E	20-Jun-2011 06:28:09.82...	Failed to send message to 'ACTIVITYLOG_JMS_REC'

- The new Activity Log shows you all recent activity on a message flow or resource manager
 - For example, “Show me all recent JMS activity”
- Visible in MBX (through the CMP) and/or written to a file
- Customisable rotation rules (based on age or size) and content

Optimizing and tuning

- The tools allow you modify the broker's configuration operationally
- These tweaks more efficient to make than modifying message flows
- Encourage developers to create message flows that enable operational tweaks to be made, e.g.
 - Configurable Services
 - User-defined properties
 - User-defined configurable services
- Support migration of your flows from test & development through QA into production



Configurable Service

Create a new Configurable Service and set its attributes

*Name MY_FTP_CONFIGURABLE_SERVICE

*Type FtpServer

Template None

Key	Value
serverName	ftp.example.com

File Input Node Properties - File Input

Remote Transfer

Transfer protocol FTP

Server and port MY_FTP_CONFIGURABLE_SERVICE
e.g. ftp.server.com:21 (if port not specified 21 is assumed for FTP, 22 for SFTP)

Connectivity Identity

Notes: Configurable Services



- Allows separation of flow design from the details about external services
 - eg: SMTP server or a JMS provider
- Flow developer configures nodes with the configurable service names
- Broker administrator creates and configures the configurable service with appropriate values
- Restart the execution group for changes to take effect
- Configure using Message Broker Explorer or using the commands
 - `mqsicreateconfigurableservice` – create a new configurable service

```
mqsicreateconfigurableservice <brokerName> -c <cs name> -o <cs name>
mqsicreateconfigurableservice <brokerName> -c <cs name> -o <cs name> -n <property>,<property2> -v
<value>,<value2>
```
 - `mqsichangeproperties` – change an existing configurable service

```
mqsichangeproperties <brokerName> -c <cs type> -o <cs name> -n <property> -v <value>
```
 - `mqsireportproperties` – report available configurable services and their attributes

```
mqsireportproperties <brokerName> -c <cs type> -o <cs name> -r
```
 - `mqsideleteconfigurableservice` – delete a configurable service

```
mqsideleteconfigurableservice <brokerName> -c <cs name> -o <cs name>
```


Configurable Services



- Extensive list available

- Aggregation
- CICSCONNECTION
- Collector
- CORBA
- EmailServer
- EISProviders
- FtpServer
- IMSCONNECT
- JavaClassLoader
- JDBCProviders
- JDEdwardsConnection
- JMSProviders
- MonitoringProfiles
- PeopleSoftConnection
- PolicySets
- PolicySet Bindings
- Resequence
- SAPConnection
- SecurityProfiles
- Service Registries
- SiebelConnection
- SMTP
- TCPIPClient
- TCPIPServer
- Timer
- UserDefined

MQ Explorer - Content

Configurable Service CICSCONNECTIONTemplate

Properties QuickView:

Name	CICSCONNECTIONTemplate
Type	CICSCONNECTION
cicsServer	
clientApplid	
clientQualifier	
connectionTimeoutSecs	30
gatewayURL	local:
requestTimeoutSecs	120
securityIdentity	

Notes: Configurable Services - example



```
mqsicreateconfigurableservice MB7BROKER -c JDBCProviders -o DB2EXTRA -n connectionUrlFormat  
-v "jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=[password];"
```

```
mqsichangeproperties MB7BROKER -c JDBCProviders -o DB2EXTRA -n maxConnectionPoolSize -v 20
```

```
mqsireportproperties MB7BROKER -c JDBCProviders -o DB2EXTRA -r
```

JDBCProviders

DB2EXTRA

```
connectionUrlFormat='jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=[password];'  
connectionUrlFormatAttr1=""  
connectionUrlFormatAttr2=""  
connectionUrlFormatAttr3=""  
connectionUrlFormatAttr4=""  
connectionUrlFormatAttr5=""  
databaseName='default_Database_Name'  
databaseType='default_Database_Type'  
databaseVersion='default_Database_Version'  
description='default_Description'  
environmentParms='default_none'  
jarsURL='default_Path'  
maxConnectionPoolSize='20'  
portNumber='default_Port_Number'  
securityIdentity='default_User@default_Server'  
serverName='default_Database_Server_Name'  
type4DatasourceClassName='default_Type_Four_Datasource_Class_Name'  
type4DriverClassName='default_Type_Four_Driver_Class_Name'
```

BIP80711: Successful command completion.

```
mqsdeleteconfigurableservice MB7BROKER -c JDBCProviders -o DB2EXTRA
```



Migrating between versions 1/2

- Message Broker supports same-machine coexistence of different versions
 - Install new version alongside your previous version
 - However, must use V7+ tools for v7+ brokers, v6.x tools for v6.x brokers/ConfigMgrs
 - Single command to migrate brokers between versions (forwards and backwards)
 - Migrate directly to V7 from V6.0 or V6.1; to V8 from V6.1 or V7
 - Use the same MB Toolkit version as the target runtime (e.g. V7 toolkit for V7 flows)
- Migration Process (to V7 or V8)
 - Ensure you're at MQ V7.0.1.x
 - If you're using V6.x MB pub/sub, migrate subscriptions to MQ pub/sub using *migmbbrk*
 - Migrate brokers using *mqsimigratecomponents*, or create and deploy new brokers
 - Load up existing artefacts in the new toolkit; branch a new dev stream if necessary

Migrating between versions 2/2

- After migration
 - Graphical tools automatically show local brokers
 - If you wish to manage remote brokers, connect to them (IP/Port/QMgr)
 - Remove CM and DB if no longer required
- Message Broker Explorer (MBX) considerations
 - MBX is a new component in V7; updated in V8
 - MQ does not support co-existence, therefore only one copy of MQ Explorer on a given machine
 - Both V7 and V8 MBX can manage V7 and V8 brokers
 - However, V8 MBX recommended to take advantage of new V8 features (e.g. applications)

Maintenance



- Schedule regular maintenance windows
 - IBM recommends that you are on the latest maintenance level
 - Plan exactly what will be applied and when
 - A highly available environment ensures that there is no downtime
- WMB has an option to enable/disable new capability in fixpacks
 - New nodes and parsers are disabled by default

```
mqsichangebroker MB7BROKER -f 7.0.0.2
```

Where does the configuration data go?

- As of Message Broker v7+ all configuration data is stored on the file system
 - No database pre-req
 - So no extra admin overhead when not using databases for message flow applications
- Default location for the data depends on the platform
 - Windows
 - C:\Documents and Settings\All Users\Application Data\IBMMQSI
 - Unix
 - /var/mqsi
 - z/OS
 - It depends!
 - *Chosen by the user when customizing a z/OS Broker*
 - *Configured by the ++COMPONENTDIRECTORY++ JCL variable*
- This location is generally referred to as the workpath and/or registry

Configuration Data Directories Explained



<workpath>

/common

/errors

- *Abend/error files are written here – always worth monitoring!*

/log

- *Internal binary trace files*

/profiles

- *Additional user profile scripts*

/components/<broker name>

- Internal configuration data for a given broker

/config/<broker name>/<eg name>

- Execution group specific command environment scripts

/registry/<broker name>

- Internal configuration data for a given broker

/shared-classes

- Non deployed JAR files

/XSL

- Non deployed stylesheets used by the XSLT node

Moving the Configuration Data

- On z/OS you can use a different component directory per broker to store each broker's configuration data in a different location
 - Specify a per broker location for the ++COMPONENTDIRECTORY++ JCL variable when customizing
- On all platforms you can specify mqsicreatebroker options to move certain configuration data
 - -w workPath
 - The directory in which working files for this broker are stored
 - *common/errors*
 - *common/log*
 - *components/<broker name>*
 - *components/shared-classes*
 - -e sharedWorkPath (Unix / Windows only)
 - Primarily used for enabling multi-instance broker support
 - Can also be used to move most broker internal configuration data to another location for HA purposes
 - *components/<broker name>*
 - *-e overrides -w setting*
 - *registry/<broker name>*
 - Only 1 file is then stored in the default location pointing at the 2nd location

Broker Runtime Environment

- Broker requires certain environment variables to run
- On distributed platforms mqsiprofile provides the defaults
 - Present in the bin directory of the installation
 - Needs to be applied before running any commands
 - Unix:
 - Broker is started in the same environment as mqsisstart is run in
 - Windows:
 - mqsisstart kicks off the service definition which creates a new shell and applies the profile
 - If you need to set additional variables then create a new profile in the <workpath>/common/profiles directory
 - Any scripts found in the profiles directory are run after the broker profile
 - Any edits to mqsiprofile will be overwritten when a fix pack is applied
- On z/OS the ENVFILE sets up the runtime environment
 - Generated during broker customization in the broker's home directory
 - BIPBPROF member contains the default values
 - Edit BIPBPROF and submit BIPGEN to generate the ENVFILE
 - BIPBPROF can be used for user settings

Per Execution Group Profiles

- Extend or change the environment for a specific execution group
- Distributed
 - Add a script (or scripts) to the appropriate directory
 - Windows:
 - `<MQSI_WORKPATH>\config\<my_broker_name>\<my_eg_label>\profiles`
 - Linux & Unix
 - `<MQSI_WORKPATH>/config/<my_broker_name>/<my_eg_label>/profiles`
 - Scripts are run after mqsiprofile and any scripts in the common/profiles directory are run
- z/OS
 - Customize BIPPROF as normal for all execution group parameters
 - Copy and customize BIPEPROF for each appropriate execution group
 - Edit BIPGEN adding an additional step for each new BIPEPROF
 - Submit BIPGEN
 - ENVFILE & ENVFILE.<eg name> generated in broker's home directory

Runtime UserIDs

- Windows:
 - Runs as the userid of the services definition
 - mqsicreate/changebroker -i -a to set/update
 - Can also use the LocalSystem account
- Unix
 - Runs as the userid who issues the mqsisstart command
- z/OS
 - Runs as the user defined in the started task definition
 - The user requires an OMVS segment with a home directory
- Windows/Unix: The userid that starts the broker no longer requires mqm authority
 - But it is required to create a broker
- No Database UserID and Password required from v7
 - Use mqsisetdbparms to control default ODBC and JDBC access control
 - Any v6.x defaults are migrated

Runtime Resources UserIDs

- You can control the credentials used by the broker to connect to external resources by using mqsisetdbparms
- Associate credentials, normally username/password with a resource name
- Resource name is referenced from a flow or configurable service definition
- mqsisetdbparms used to create, alter & delete credentials
- Credentials can be set for nearly all external resources which broker can connect to
 - CICS, ODBC/JDBC databases, Email POP/IMAP/SMTP, FTP, IMS, JMS/JNDI, Kerberos Key Distribution Center (KDC), SFM, keystores, EIS providers, WSRR
- After updating any userid/password definitions you must restart the relevant execution group/broker to pick up the changes

Where to look when it all goes wrong!

- Local Error Logs
 - Message Broker components use the local error log to record information about major activities
 - Actual local error logs vary by platform
 - *Windows – Windows Event Log (Application View)*
 - *Unix/Linux – syslog*
 - *z/OS – JOBLLOG & system console log*
- *stdout/stderr*
 - *Useful place to look for errors / debugging and always worth checking for exceptions if problems are occurring*
 - *Can be useful for flow developers who use Java and code `system.out.println` statements for debugging*
- *coredumps / SVC dumps*
 - *In the unlikely event that Message Broker encounters a problem that results in an abend you need to be aware of where to look for dumps*
 - *Check error logs for location or look in the broker's home dir*
 - *On z/OS check the system dump datasets*

Notes: Local Error Logs

- Key information point for an administrator to monitor
- Message Broker components use the local error log to record information about major activities
- Actual local error logs vary by platform
 - Windows – Windows Event Log (Application View)
 - Unix/Linux – syslog
 - z/OS – JOBLLOG & system console log
- When an error occurs, check the local error log first
- Often requested by support
 - Windows
 - The event log fills up so check the size is sufficient or that circular logging is enabled
 - Unix/Linux
 - Make sure you configure the syslog daemon

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*
C:\Documents and Settings\All Users\Application Data\IBMMQSI\components\
<brkName>\console.txt
 - *Execution group*
C:\Documents and Settings\All Users\Application Data\IBMMQSI\components\
<brkName>\<egUUID>\console.txt
 - Linux/Unix
 - *Admin Agent*
/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

Notes: Coredump



- In the unlikely event that Message Broker encounters a problem that results in a coredump you need to be aware of where to look for dumps
- Windows
 - BIP2111 error message (message broker internal error).
 - The error message contains the path to the MiniDump file in your errors directory
- Linux/UNIX
 - BIP2060 error message (execution group terminated unexpectedly)
 - Look in the directory where the broker was started, or in the service user ID's home directory, to find the core dump file
 - Check your ulimits
 - We recommend an unlimited hard & soft limit for corefile size
 - Ensure you have enough disk space

Notes: SVC dump (z/OS)

- Message Broker on z/OS should always produce an SVC dump
- Dump dataset is written based on the system defined setup
 - Use the “display dump” command to display the naming options
 - BIP2060 error message (execution group ended unexpectedly) from the main Broker Address Space.
 - Message should be accompanied by one of the following messages and dump
 - *IEF450I message in the syslog, or component's joblog, showing an abend code followed by a reason code, for example:*

```
IEF450I MQ83BRK DEFAULT - ABEND=S2C1 U0000 REASON=000000C4
```
 - *Look in the system's dump dataset hlq for the dump dataset, or search the syslog for the appropriate IEA611I message to find out the dump dataset name.*
- In extreme cases you may see a coredump instead
 - In these cases you will see an IEA993I message in the syslog
 - Look in the started task user's directory for the coredump.pid file, as specified in the syslog:

```
IEA993I SYSMDUMP TAKEN TO coredump.00500319
```
- If a dump is not produced then look for a reason why in the JOBLOG and system console log
 - Check both as errors are not always repeated
 - A dump might have been suppressed by DAE

Summary

- We have discussed a number of different ways that help ensure that WMB administration is trouble-free
- Top Tips
 - Always ensure your environment is reproducible
 - Treat DR and HA separate
 - Ensure regular backups
 - Encourage developers to create message flows that enables operational tweaks to be made
 - Schedule regular maintenance windows
 - Know where to look when it all goes wrong!

This was session 10702 - 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. Other company, product, or service names may be trademarks or service marks of others.