#SHAREorg

MQ Java Zero to Hero

Chris Matthewson (cmatth@uk.ibm.com)
IBM Hursley Park

March 14th, 2012
Session 10537

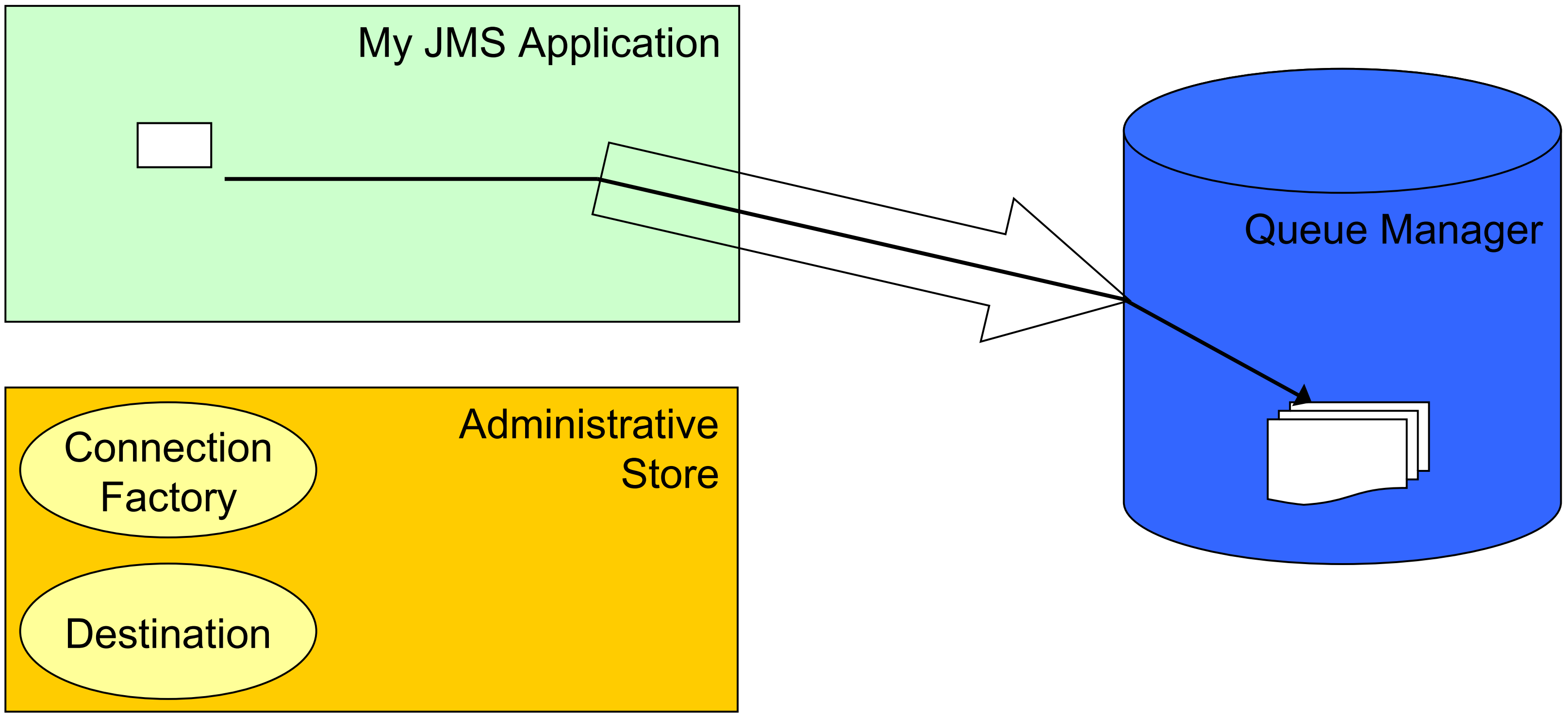
Agenda

- What is WMQ Java and JMS?
- **WMQ JMS in standalone J2SE Environments**
  - JNDI or programmatic configuration
  - Administration tools
  - Running your application
- **WMQ JMS in WebSphere Application Server Environments**
  - The extras, or why to use WAS
  - MDBs, EJBs, Servlets…
  - Deploying your application
- **Changing parameters and tuning**
  - WMQ parameters
  - WAS parameters

What is WMQ Java and JMS?

- **WebSphere MQ Classes for Java**
  - Pre-dates JMS API
  - Resembles WMQ API
  - Offers programmatic access to WMQ features
- **WebSphere MQ Classes for JMS**
  - Java Message Service
  - API that provides a common way for Java applications to send and receive messages
  - Provider Agnostic
  - Often used in J2EE Application Servers
  - Developed by Sun with input from IBM and others

How does a JMS application work?



How do Connection Factories and Destinations get created?

Agenda

- **What is WMQ Java and JMS?**
- **WMQ JMS in standalone J2SE Environments**
  - JNDI or programmatic configuration
  - Administration tools
  - Running your application
- **WMQ JMS in WebSphere Application Server Environments**
  - The extras, or why to use WAS
  - MDBs, EJBs, Servlets…
  - Deploying your application
- **Changing parameters and tuning**
  - WMQ parameters
  - WAS parameters

JNDI or programmatic configuration

- Recommended way to define JMS resources is to use a JNDI (Java Naming and Directory Interface) store
  - Requires a JNDI store to be created
  - Easy to change properties, or even JMS provider!
  - Easy to share definitions across enterprise
  - Encourages code re-use.
- Can also programmatically define resources
  - Requires the application to set all properties
  - Needs recompilation to alter/add properties
  - Each application needs to set properties

Sample code for each style

- JNDI:

```
Hashtable environment = new Hashtable();
environment.put(Context.INITIAL_CONTEXT_FACTORY,
  "com.sun.jndi.fscontext.RefFSContextFactory");
environment.put(Context.PROVIDER_URL, "file:/IBM/JNDI/");
Context ctx = new InitialDirContext(environment);
cf = (ConnectionFactory) ctx.lookup("MyCF");
```

- Programmatic:

```
MQConnectionFactory cf = new MQConnectionFactory();
cf.setQueueManager("MyQM");
cf.setTransportType(JMSC.MQJMS_TP_CLIENT_MQ_TCPIP);
cf.setHostName("myhost");
cf.setPort(14140);
…
…
```

WMQ JMS Administration Tools: JNDI Repositories

- The WMQ JMS Administration Tools need access to a JNDI Repository to store connection factory and destination definitions.
- Two main types of JNDI Repository are supported.

  - LDAP Directories
    - Heavyweight.
    - Offers high levels of security.
    - Easy to share connection factory and destination objects between multiple JMS clients running on different machines.
    - Can be difficult to set up.

  - File system
    - Very lightweight.
    - Easy to set up and get running.
    - Not very secure.
    - Hard to share repository between multiple JMS clients.

  - WebSphere MQ SupportPac ME01 allows the queue manager to be used as a JNDI store, as well as auto-define existing queues.

WMQ JMS Administration Tools: JMSAdmin

- Command line tool.
- Supported on all platforms.
- Installed into <WMQ_HOME>\java\bin.
- To run it:
  - Edit JMSAdmin.config file to point to the JNDI repository that will be used, and optionally pass in authentication credentials.
  - Then enter JMSAdmin
- Can be used to run scripts
  - For example “jmsadmin < myscript.txt”

WMQ JMS Administration Tools: JMSAdmin

- JMSAdmin.config contains three properties
  - INITIAL_CONTEXT_FACTORY
    - The class used by the JNDI repository to store and retrieve objects.
    - Possible values are:

      com.sun.jndi.ldap.LdapCtxFactory
      Used for LDAP Repositories on distributed platforms.
      com.ibm.jndi.LDAPCtxFactory
      Used to connect to an LDAP repository from JMS applications running on z/OS.
      com.sun.jndi.fscontext.RefFSContextFactory
      Used for file system Repositories

WMQ JMS Administration Tools: JMSAdmin

- PROVIDER_URL
  - An address used by the JMSAdmin tool to access the JNDI Repository.
  - Possible values are:

ldap://<hostname>/<contextname>

The hostname and port that the LDAP server is listening on, followed by the top level directory context where the objects will be stored.

file:<drive>/<pathname>

The path to the directory where the administered object definitions will be saved. The directory must exist before JMSAdmin can be run.

WMQ JMS Administration Tools: JMSAdmin

- SECURITY_AUTHENTICATION
  - Used when connecting to a secure LDAP JNDI Repository.
  - Possible values are:

    none

    The JMSAdmin tool does not pass any security information to LDAP.

    simple

    The JMSAdmin tool sends an LDAP distinguished name and password to the Server for authentication during startup.

    CRAM-MD5

    The JMSAdmin tool flows an LDAP distinguished name and password (encrypted as an MD5 hash) when connecting to the LDAP server.
- The distinguished name and password can be specified, using the PROVIDER_USERDN and PROVIDER_PASSWORD properties, or JMSAdmin will prompt

WMQ JMS Administration Tools: JMSAdmin

- JMSAdmin expects commands to be in the format
  - Verb noun(value) noun(value)…..
- Useful verbs are:
  - DEFINE
  - ALTER
  - DISPLAY
- Nouns include:

CF	- JMS Connection Factory
QCF	- JMS Queue Connection Factory
TCF	- JMS Topic Connection Factory
Q	- JMS Queue
T	- JMS Topic
QMGR	- Queue Manager Name
QU	- Queue Name
TO	- Topic Name

WMQ JMS Administration Tools: JMSAdmin

- To create a Queue Connection Factory for the queue manager QM1, enter the command:
  - DEFINE QCF(testQCF) QMGR (QM1)
    - The Queue Connection Factory will be stored in the JNDI Repository with the name testQCF.

- To create a JMS Queue Destination that points to the queue “test”, enter:
  - DEFINE Q(testQ) QU(test)
    - The object will be stored in JNDI with the name testQ.

WMQ JMS Administration Tools: MQ Explorer

- Graphical version of the JMSAdmin tool.
- To create JMS Administered Objects in MQ Explorer:
  - Set up a context
    - Directory-like structure where the objects will be stored.
  - Select the type of JNDI repository that will be used
    - File system
    - LDAP
    - Another JNDI repository
  - Specify the address of the JNDI repository
    - For file system contexts, select the directory where the Administered Objects will be stored
    - For LDAP repositories, enter the URL of the LDAP server
  - Optionally enter the username and password used to connect to the JNDI repository
  - And that's it!

WMQ JMS Administration Tools: MQ Explorer

- MQ Explorer wizards provide step-by-step instructions for creating JMS Administered Objects.

- MQ Explorer also allows Queue or Topic Destination Administered Objects at the same time as creating the actual Queue or Topic.

- MQ Queues and Topics can also be created from Destination Administered Objects.

Running your WMQ JMS Application

- JVM Classpath needs to include:
  - com.ibm.mqjms.jar
  - jms.jar
  - JNDI libraries, such as fscontext.jar, if using JNDI
- If using bindings mode, JVM java.library.path needs to include the WMQ Java lib directory.
- If on a different machine to the queue manager, use SupportPac MQC7 to obtain the WMQ JMS client libraries

- That's it, run your Java Application!

If it goes wrong...

- Runtime errors are reported as JMSExceptions, which include the MQException (if any) that caused it
  - Contains the MQ reason code of the problem
  - For example RC=2059 MQRC_Q_MGR_NOT_AVAILABLE

```
Caused by: com.ibm.mq.MQException: JMSCMQ0001: WebSphere
  MQ call failed with compcode '2' ('MQCC_FAILED') reason
  '2059' ('MQRC_Q_MGR_NOT_AVAILABLE').
```

  - Not all MQRC's reported as JMSExceptions, such as RC=2033 MQRC_NO_MSG_AVAILABLE
- mqjms.log and FFDC files generated with serious errors

Agenda

- **WMQ JMS in standalone J2SE Environments**
  - JNDI or programmatic configuration
  - Administration tools
  - Running your application
- **WMQ JMS in WebSphere Application Server Environments**
  - The extras - why to use WAS
  - MDBs, EJBs, Servlets…
  - Deploying your application
- **Changing parameters and tuning**
  - WMQ parameters
  - WAS parameters

The extras – why use WMQ JMS with WAS?

- WebSphere Application Server provides an environment to run JMS applications in.
  - Built-in JNDI repository
  - Web based administration, and scripted administration tools integrated closely with WMQ
  - WMQ JMS client installed and configured for use with WAS
- Less coding to achieve enterprise class applications
  - MDBs – potential to code a single method that results in multi-instance message processor
  - EJBs – easy access/re-use of code
  - Servlets/JSPs – web access to MQ
- Transaction management and coordination
- Resource management, such as connection pooling
- Easy integration with other JEE applications

Message-driven beans

- Message-driven beans (MDBs) are JMS applications that get called when a message arrives on a given destination.
  - Similar to WMQ triggered applications.

- Recommended way of getting WMQ messages into WAS.

- Application developer only has to worry about the business logic required to process the message.
  - Application server handles the actual detection and delivery of the message.

Message-driven beans

- MDBs must implement a method called onMessage().
  - This is called when a message is detected on the specified destination.
  - Message is passed into the method.
  - onMessage() simply needs to contain the code to process it.
  - Application Server handles all transaction management.
- Application server handles concurrent processing to facilitate scaling
- IBM Rational tooling provides wizards for creating MDBs.

Message-driven beans

```
public void onMessage(Message message) {
     try {
     if (message instanceof TextMessage) {
             TextMessage textMsg = (TextMessage)message;
             System.out.println("Message text is " + textMsg.getText());
         }
     } catch (JMSException ex) {
         System.out.println("JMSException occurred : " + ex);
     }
  }
```

Deploying message-driven beans: Activation Specifications

- WAS 7 and newer access WMQ using the WebSphere MQ Resource Adapter (RA)
  - Based on the J2EE Connector Architecture (JCA) 1.5 standard.
- Standard mechanism for listening for messages on JMS destinations.
- Contain information to create a connection to a specified queue or topic on a queue manager.
- Provides a common way for all JEE 1.4 compliant application servers to connect to JMS providers.

Activation Specifications

- To create an Activation Specification in WAS:
  - Specify the JMS Destination to listen on.
  - Enter details of the queue manager where the Destination resides.
  - Optionally, specify a JMS Message Selector.
    - SQL expression.
    - Only messages that match the Selector will be delivered to applications using this Activation Specification.
  - A handy wizard takes you through all of the necessary steps, and checks it works too!.
- When deploying the MDB application, specify the Activation Specification to use.

Enterprise Java Beans/Servlets

- Java applications that run inside of WAS.
- EJBs and Servlets need to create their own connections to WMQ and get (or send) messages themselves.
  - EJBs and Servlets use the standard JMS API, in a similar way to standalone JMS applications.
  - Can be easier than MDBs when handling responses in request-reply messaging
  - Application server still handles transaction management, based on values specified in the application's deployment descriptor.

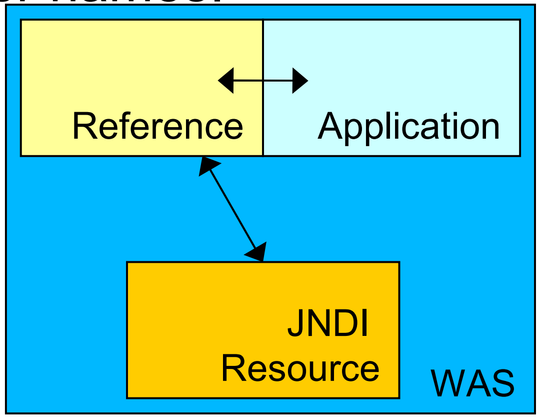
Application Development
Enterprise Java Beans

```
@Resource()
private ConnectionFactory cf;
@Resource()
private Queue d;

public void receiveMessage() {
    try  {
      Connection conn = cf.createConnection();
      conn.start();
      Session sess = conn.createSession(true,Session.AUTO_ACKNOWLEDGE);
      MessageConsumer consumer = sess.createConsumer(d);
      Message msg = consumer.receive(30000);
      if (msg instanceof TextMessage) {
        System.out.println("Message received:" + ((TextMessage) msg).getText());
      }
    conn.close();
    } catch (Exception ex) {
   System.out.println("Exception :  " + ex);
    }
 }
```

Enterprise Java Beans/Servlets Deploying

- Define required Connection Factories and Destinations
  - Web based administration console allows most properties to be configured
  - Other properties can be set as custom properties, using JMSAdmin names and values.
- Applications should use resource-references to decouple application connection factory/destination names from server names.
  - Allows reconfiguration without recompilation.
- During deploy, references link to real resources



- EJB 3 introduces annotations
  - Remove reliance on XML configuration files
  - Resource references defined in the application code, for ease of use.

Agenda

- **WMQ JMS in standalone J2SE Environments**
  - JNDI or programmatic configuration
  - Administration tools
  - Running your application
- **WMQ JMS in WebSphere Application Server Environments**
  - The extras, or why to use WAS
  - MDBs, EJBs, Servlets…
  - Deploying your application
- **Changing parameters and tuning**
  - WMQ parameters
  - WAS parameters
- **Demo**

WMQ Connection Factory Properties: Connection Name List

- This property specifies a list of hostnames and ports to attempt to connect to.
  - Comma-separated list of “hostname(port)” entries
  - Similar to a CCDT with multiple entries.
  - Can be used with client reconnection options and client reconnection timeout to allow automatic reconnection to a standby queue manager.
- JMSAdmin Property Name: CONNECTIONNAMELIST / CNLIST

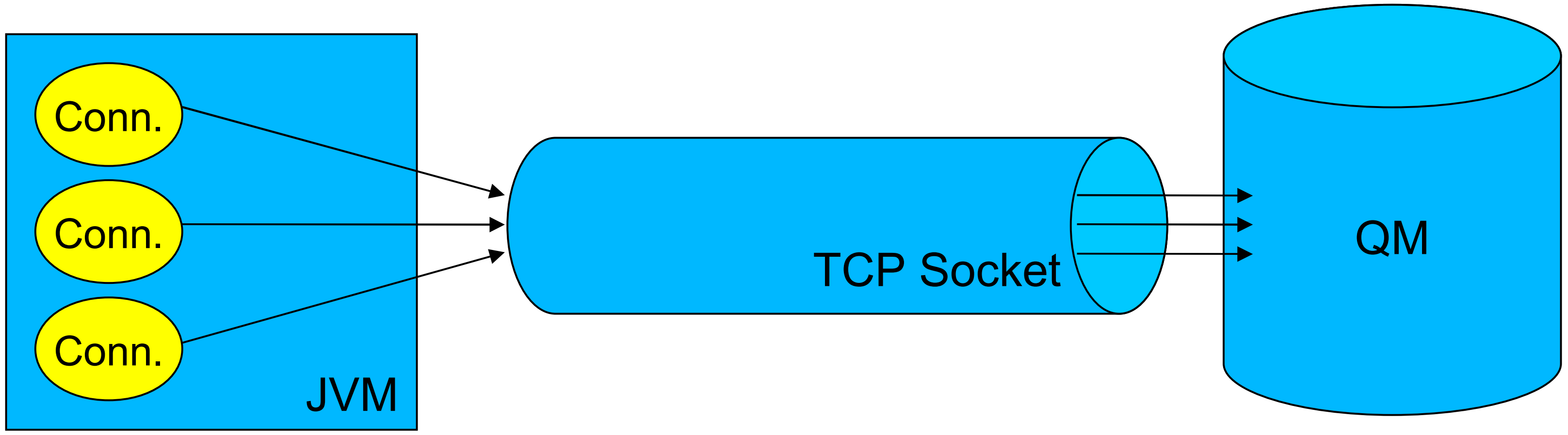
WMQ Connection Factory Properties: Automatic Client Reconnection

- Determines whether the JMS client should reconnect on failure
- Used in conjunction with connection name list
- Allows failover to multi-instance queue manager
- Not for use with WAS
- Reconnects happen until reconnection timeout value is reached.

- JMSAdmin property : CLIENTRECONNECTOPTIONS / CLIENTRECONNECTTIMEOUT

WMQ Connection Factory Properties: Shared Conversation Allowed

- This property specifies whether JMS applications that use this Factory can share their connection to a Version 7 queue manager.



- Useful to reduce the number of network connections to a queue manager.
- Can have slight performance impact.
  - Multiple JMS applications will be sending data to a queue manager and waiting for a response over the same channel.
  - Set server connection channel SHARECNV to 1 for maximum performance
- JMSAdmin Property Name: SHARECONVALLOWED / SCALD

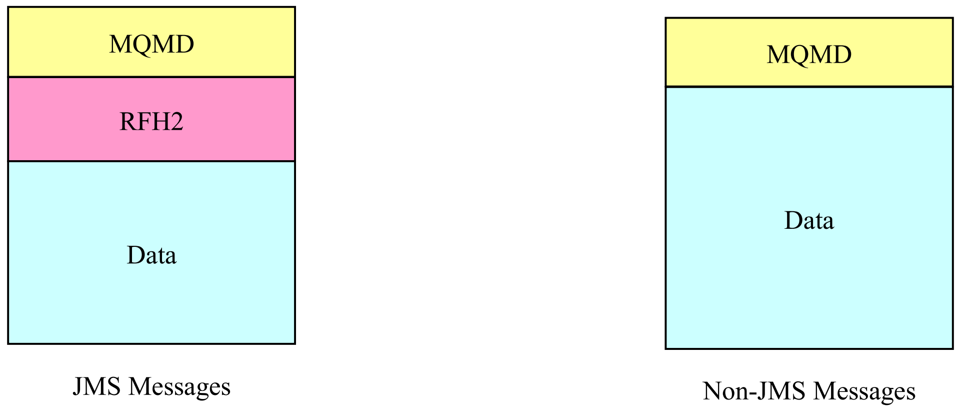
WMQ Connection Factory Properties: MQMD Read/Write enabled

- Determines if MQMD properties can be set via JMS get/set message property methods.
  - Allows full access to the MQMD header values
  - Useful for sending or receiving messages from MQ applications that use specific header properties.
  - JMS message property names begin with "JMS_IBM_MQMD…"
- MSGCTX

- JMSAdmin Property Name: MDREAD / MDR & MDWRITE / MDW

WMQ Destination Properties: Target Client

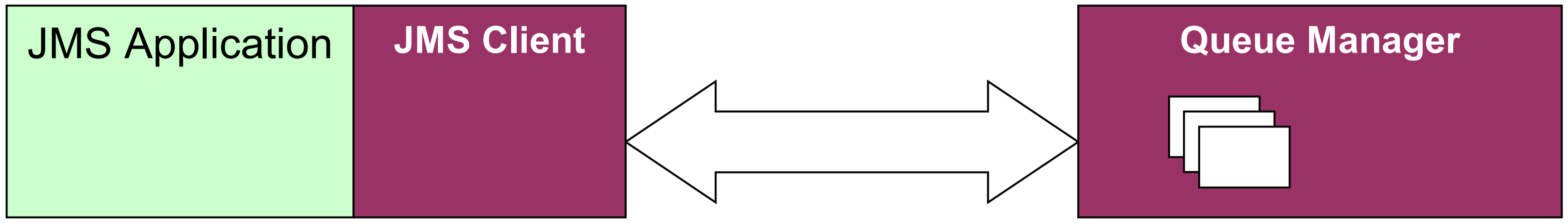
- Indicates whether messages sent to this destination are for other JMS applications, or for non-JMS WMQ applications.
  - WMQ JMS messages, by default, have an RFH2 header containing JMS specific information.
  - If a message is for a non-JMS application, this header may cause problems, so can be turned off.



JMSAdmin Property Name: TARGCLIENT / TC

WMQ Destination Properties: Read Ahead Allowed

- In general, messages are sent to JMS applications one at a time.
- The Read Ahead Allowed property tells the queue manager whether non-persistent messages can be streamed to the client application in preparation for them being consumed.
  - Messages are stored in a buffer on the client.
  - If the client application terminates unexpectedly, all unconsumed non-persistent messages are discarded.



- JMSAdmin Property Name: READAHEADALLOWED / RAALD

WMQ Destination Properties: Asynchronous Puts

- Sending equivalent of Read Ahead Allowed
- Allows messages to be sent, but the JMS client does not wait for queue manager acknowledgement.
- Improves performance of sending applications
- JMS client checkpoints during commit or according to SENDCHECKCOUNT for non-transacted sessions.

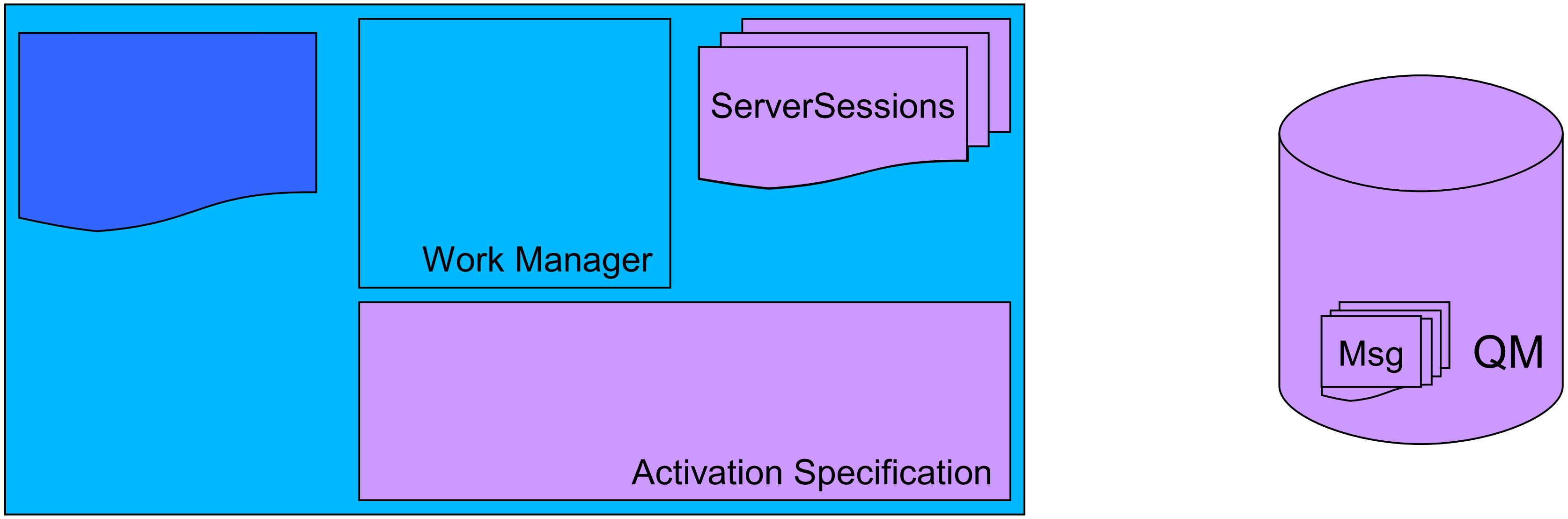
- JMSAdmin property names: PUTASYNCALLOWED / SENDCHECKCOUNT

Agenda

- **WMQ JMS in standalone J2SE Environments**
  - JNDI or programmatic configuration
  - Administration tools
  - Running your application
- **WMQ JMS in WebSphere Application Server Environments**
  - The extras, or why to use WAS
  - MDBs, EJBs, Servlets…
  - Deploying your application
- **Changing parameters and tuning**
  - WMQ parameters
  - WAS parameters

WAS Configuration: Activation Specifications

- How are messages destined for MDBs processed?



WAS Configuration
Activation Specifications - Threads

- To process a message, a ServerSession and thread are required
- Activation Specification parameter Maximum Sessions configures ServerSession pool, default 10
  - ServerSession pool per MDB
- Application Server Thread pool WMQJCAResourceAdapter used, default max 25
  - Thread pool used by all MDBs.
- So, by default, 3 MDBs could exhaust the threads
  - Will cause slower processing
  - Recommendation is that thread pool maximum accounts for all MDB maximums

WAS Configuration
Activation Specifications - Recovery

- What happens if the connection to a queue manager used by Activation Specifications is broken?
- The Activation Specification has recovery features to allow it to attempt several reconnection attempts before stopping.
- Default is to try 5 times in 5 minute intervals.
  - Configured using reconnectionRetryCount / reconnectionRetryInterval (ms) on the Resource Adapter, for all Activation Specifications.
  - Accessed through Resources -> Resource Adapters, but need to “Show built-in Resources”
- If the Activation Specification stops, this is only reported in Application Server logs.

WAS Configuration
JMS User authentication

```
createConnection(user,password)
```

- WAS has built-in user credential repositories that can be defined with username and password details
  - JAAS – J2C Authentication Data
- Activation Specifications allow this to be configured in the Security parameters
- Connection Factories also allow this to be configured, although application configuration determines if it will be used.
  - The application needs to use JEE resource-references to access the connection factory
  - The res-auth parameter needs to be set to “container” for container managed authentication.
- res-auth of application, or not using resource-references means the application is responsible for any authentication information
- As for other WMQ clients, security exits are required to validate passwords, WMQ only checks user id.

Agenda

- **WMQ JMS in standalone J2SE Environments**
  - JNDI or programmatic configuration
  - Administration tools
  - Running your application
- **WMQ JMS in WebSphere Application Server Environments**
  - The extras, or why to use WAS
  - MDBs, EJBs, Servlets…
  - Deploying your application
- **Changing parameters and tuning**
  - WMQ parameters
  - WAS parameters

Any questions?

- If you have any questions, or ideas for future topics, feel free to email me at **cmatth@uk.ibm.com**

This was session 10537 - 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 The Doctor is in. Hands-on Lab and Lots of Help with the MQ Family	Message Broker administration	The Do’s and Don’ts of z/OS Queue Manager Performance	
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	

SHARE Technology · Connections · Results