

# Using IBM WebSphere Application Server and IBM WebSphere MQ Together



Chris J Andrews  
IBM

Thursday 7th February, 2013  
Session 12611

# Agenda

- Why are we here?
- Messaging in WAS
  - What is JMS?
- Accessing WMQ from WAS
  - How to configure
  - New WMQ V7 features useful for WAS
- Configuring WMQ and WAS
  - Useful tuning parameters
  - High Availability and clusters

# Why are we here?



Some reasons for being here...

*“We currently have applications using WebSphere MQ, and now we have applications running in WebSphere Application Server that need to access the same data”*

*“We’re writing applications for WebSphere Application Server, we’re trying to choose a messaging provider and are thinking that WebSphere MQ might fit the bill”*

...

# How do we best communicate between WebSphere Application Server and WebSphere MQ?

- WebSphere Application Server is a fully compliant Java Enterprise Edition (**JEE**) Application Server
  - Provides integrated support for application messaging using Java Message Service (**JMS**) providers.
- WebSphere MQ is a fully compliant JMS provider.
- Therefore, JMS is the answer!

# Agenda

- Why are we here?
- Messaging in WAS
  - What is JMS?
- Accessing WMQ from WAS
  - How to configure
  - New WMQ V7 features useful for WAS
- Configuring WMQ and WAS
  - Useful tuning parameters
  - High Availability and clusters

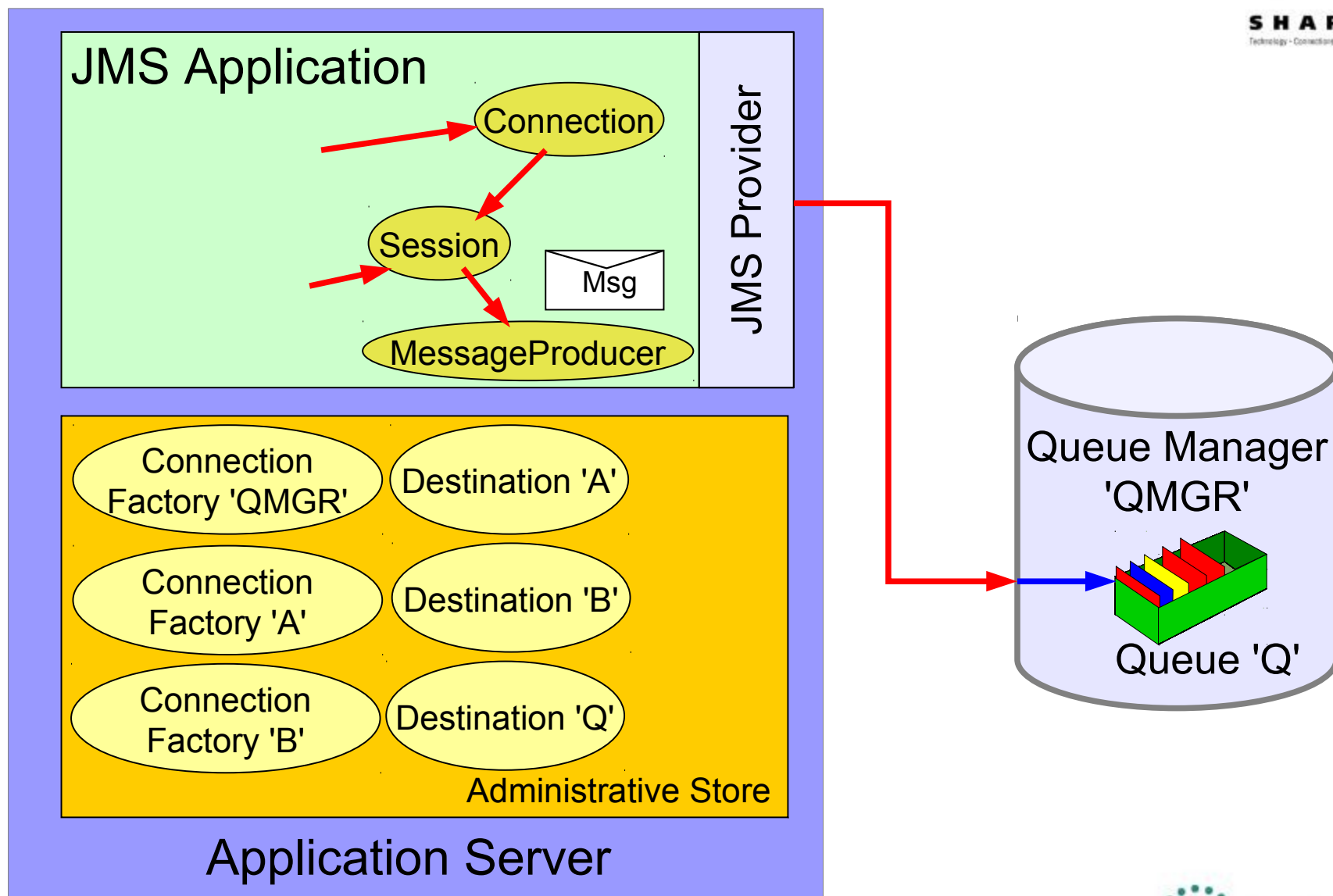
# What is JMS?

- A standardized Java API to allow applications to perform messaging
  - Applications can be unaware of implementation
  - Not a wire protocol
- Uses **administered objects** to define provider and configuration
  - Can be changed without recompiling application
- Supports point-to-point messaging, and publish/subscribe messaging

# Notes

- JEE application servers must provide support for JMS
  - JEE 1.4 and later also must support the Java Connector Architecture (JCA), which WMQ now uses within WAS 7+
- JMS is a standardised programming interface
  - Allows Java application interactions to be loosely coupled
  - JMS applications can be coded to be ignorant of the underlying JMS provider
    - *Can easily change the provider without altering the application*

# How do JMS based applications function?

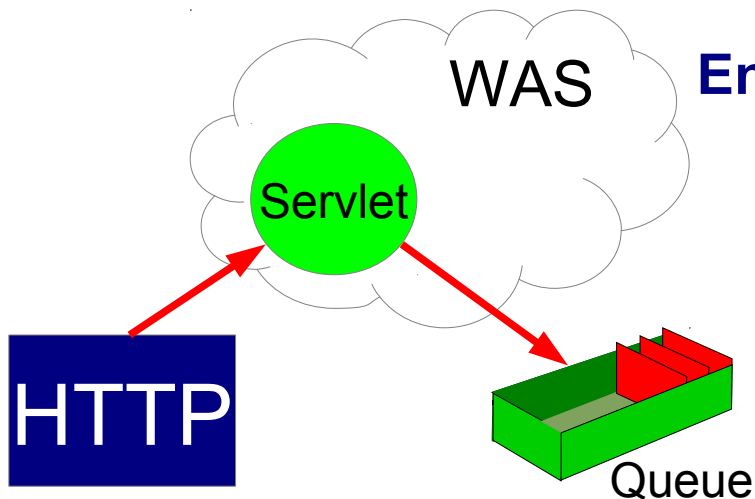
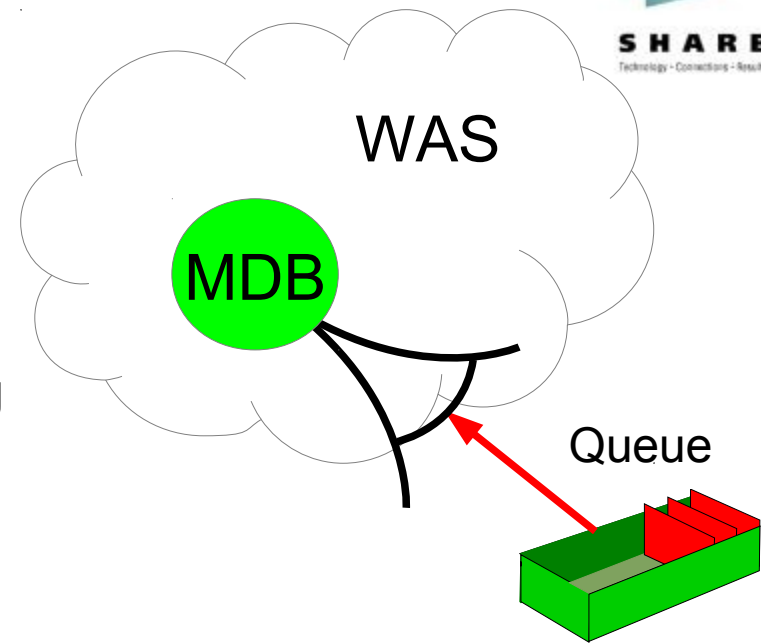




# Types of JMS applications in WAS

## Message Driven Bean (MDB):

- Specific type of EJB which has an “onMessage” Java method to process messages
- JEE Container (WAS) deals with getting the message, transactions, and running the application.
- Mechanism to get messages **into** WAS



## Enterprise Java Bean (EJB), Servlets/JSPs:

- Application creates connections and sends/receives messages to WMQ
- Container provides connection pooling and transaction management

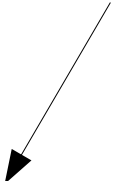
# Notes

- Provider specific configuration, such as how to connect to a messaging provider, is contained within *JMS resources* held in **JNDI**, defined at application deployment time
- JMS is not a transport protocol
  - No standardised ability to couple different JMS providers
- JMS Supports two messaging models
  - **Point-to-point** – using queues
  - **Publish/subscribe** – using topics

# Message Driven Beans dissected



Method invoked when a message is available



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

# Enterprise Java Beans (EJB)



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

Define EJB 3 Resource references that will be initialised at runtime

```
public void receiveMessage() {
    try {
        Connection conn = cf.createConnection();
        conn.start();
        Session sess =
            conn.createSession(true, Session.AUTO_ACKNOWLEDGE);
        MessageConsumer consumer = sess.createConsumer(q);
        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);
    }
}
```

More code required to get a message compared to an MDB, and less efficient

# Connecting to WMQ the bad way...

```
MQConnectionFactory cf = new MQConnectionFactory();  
cf.setQueueManager("myQMGR");  
cf.setHostName("myhost.mydomain");  
cf.setPort(1414);  
cf.setTransportType(WMQConstants.WMQ_CM_CLIENT);
```

```
Connection conn = cf.createConnection();  
conn.start();
```

```
Session sess =  
    conn.createSession(true, Session.AUTO_ACKNOWLEDGE);
```

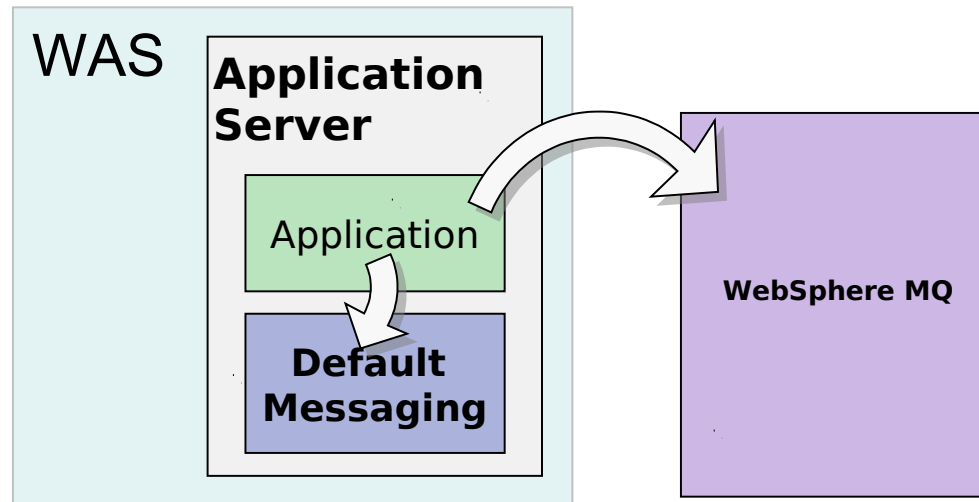
```
MQQueue q = new MQQueue("myQ");
```

```
MessageConsumer consumer = sess.createConsumer(q);
```

```
Message msg = consumer.receive(30000);
```

# WAS JMS messaging *in a slide*

- WebSphere Application Server provides first class support for two JMS messaging providers
  - (1) An embedded messaging provider, the **Default Messaging provider** (or **Service Integration Bus**)
  - (2) **WebSphere MQ**
- The nature of JMS and JEE allows easy switching between providers with little or no application changes



# WebSphere MQ in a slide

- WebSphere MQ (WMQ)
  - WMQ is IBM's flagship asynchronous messaging product
  - Queues are managed by Queue Managers
  - Applications connect to Queue Managers to access Queues
    - *Many APIs, such as JMS, are available*
  - Queue Managers can be connected together to form a network, or cluster
  - WMQ supports the point-to-point and publish/subscribe messaging models
- WMQ as the JMS provider in WAS
  - WMQ JMS messages are standard WMQ messages
  - WMQ V7 has new features to simplify and improve JMS
  - Configuration of WMQ JMS objects is fully integrated into the WAS administration tools

# WebSphere MQ messaging provider



- WMQ JMS messages are standard WMQ messages, and can therefore be exchanged with either other WMQ JMS clients, or any other WMQ application, as long as the format of the message is understood.
- WAS applications connect to a queue manager using:
  - Bindings (shared memory) – when application and queue manager are on the same server
  - Client (TCP/IP) – when application server and queue manager are on separate, networked, servers
- In WAS V6.0 and V6.1, the WebSphere MQ Client came with WAS as part of the lib directory structure and was referenced with the use the WebSphere environment variable MQ\_INSTALL\_ROOT. This allowed the version of the client to be changed to one installed locally on the machine. Changing the install location is required when running in bindings mode as it is also the method used to locate the WebSphere MQ native libraries
- In WAS V7, the client has been replaced with the WebSphere MQ V7 JCA 1.5 RA and is installed as a resource adapter. The MQ\_INSTALL\_ROOT variable is no longer used (except for migration purposes and the client container), instead the native library path for the RA is used to identify the location of the required native libraries
- Version of WMQ shipped with WAS:
  - <http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg21248089>
- Ensuring the correct WMQ version is used in your WAS profile:
  - <http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/t>



# Service Integration Bus Overview



- Service Integration Bus (SIBus)
  - SIBus is a native component of WAS, providing JMS messaging
- The “bus” is a logical entity
  - Messaging engines in application servers manage the “physical” messages
- SIBus as the JMS provider in WAS
  - SIBus supports the JMS programming interface for Java applications.
  - SIBus has features that allow it to connect to and be part of a WMQ messaging solution.

# Notes

- SIBus forms the asynchronous messaging platform for application server based products
- Physically:
  - *Application servers can be made a member of a bus*
    - *Destinations are assigned to bus members*
  - *Bus members run messaging engines that manage messages*
  - *Messages are routed from any messaging engine in the bus to the application*
  - *A WAS cell contains application servers, or clusters of application servers*
    - *These can optionally be made a member of a bus*
  - *Destinations are assigned to bus members*
  - *Bus members run messaging engines within the application server JVMs*
  - *Messaging engines manage the runtime and persistent state of messages in the bus*
  - *Messaging applications form a connection to a messaging engine*
  - *Messages are routed from any messaging engine in the bus to the application*
- Logically:
  - *A bus is a logical entity that contains destinations (e.g. queues/topic spaces)*
  - *Messaging applications connect to the bus to access the destinations*
  - *A bus is location transparent, all destinations and their messages are available from anywhere in the bus*

# Agenda



- Why are we here?
- Messaging in WAS
  - What is JMS?
- Accessing WMQ from WAS
  - How to configure
  - New WMQ V7 features useful for WAS
- Configuring WMQ and WAS
  - Useful tuning parameters
  - High Availability and clusters

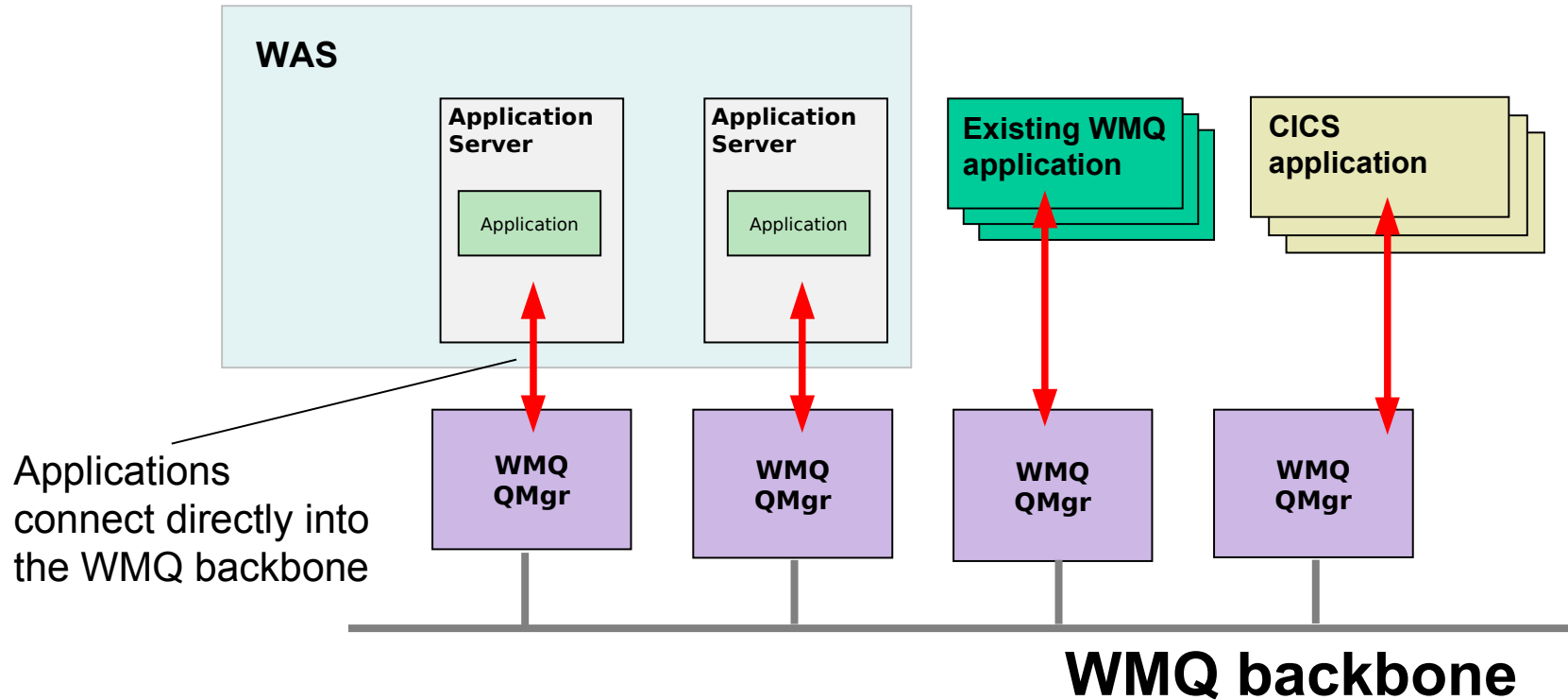
# Ways of accessing WMQ from within WAS

## ■ Three key options:

1. Use WMQ as the JMS provider in WAS
2. Use the service integration bus in WAS and connect that messaging system with your WMQ backbone
3. Use a mixture of both
  - *SIBus for intra-WAS messaging*
  - *WMQ for inter-system messaging*

## Option 1:

### WebSphere MQ as the JMS provider in WAS

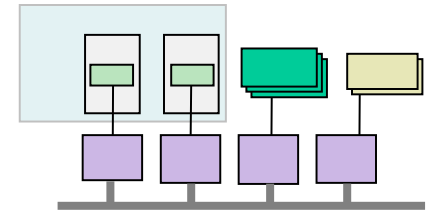


# Option 1: WMQ as the JMS provider in WAS



## • Pros

- The most **direct** route to WMQ queues
  - *Best performing way to get messages on and off a WMQ queue*
  - *Least moving parts*
    - *Least things to configure*
    - *Least places where things can go wrong*
- A single messaging system to maintain rather than two



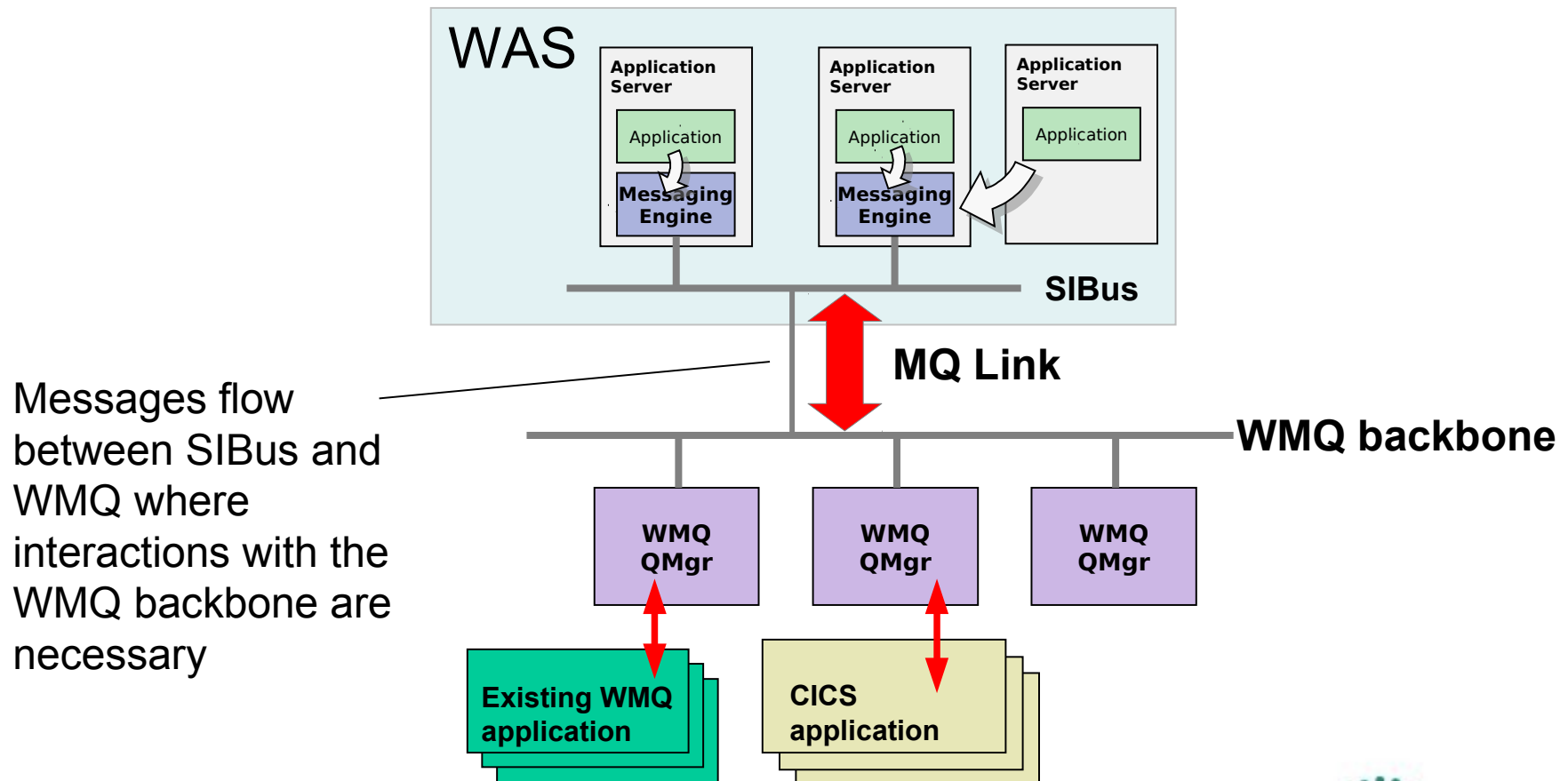
## • Cons

- WAS application deployment depends on knowledge of WMQ topology
  - *e.g. the WMQ queue location is required when connecting applications*
- WAS applications dependent on two systems being simultaneously available
- Can be complex to exploit WAS and WMQ high availability and scalability features

## Option 2:

### SIBus for WAS application messaging

SIBus interacts with WMQ to access the messaging backbone



# Notes



- WAS manual describes this in more detail:
  - <http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.web>
- Can connect using either the “WebSphere MQ Link”, or using a WMQ client connection
  - MQ Link is a server-server connection, WMQ appears as a SIBus “foreign bus”
  - WMQ client connection makes the queue manager appear as a member of the SIBus



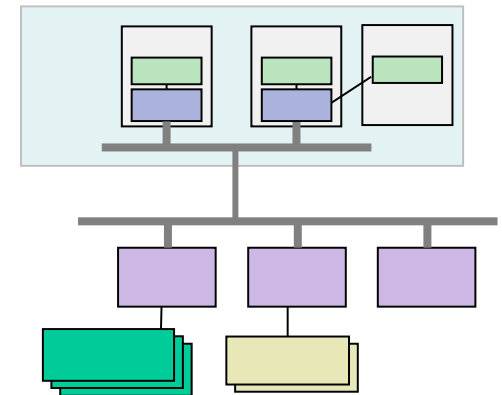
# Option 2: SIBus messaging system

## • Pros

- Messaging runtime is embedded into the same runtime as the applications
- Ability to asynchronously queue messages in the WAS system while connectivity with the WMQ backbone is unavailable
- JMS resource configuration relates to WAS-defined SIBus configuration objects
  - *WMQ-aware configuration in WAS is minimised*
- Applications can be unaware of actual queue locations

## • Cons

- Two messaging systems to configure and maintain
- Additional complexity in configuring interactions between SIBus and WMQ
- Performance overhead when passing from one messaging system to another



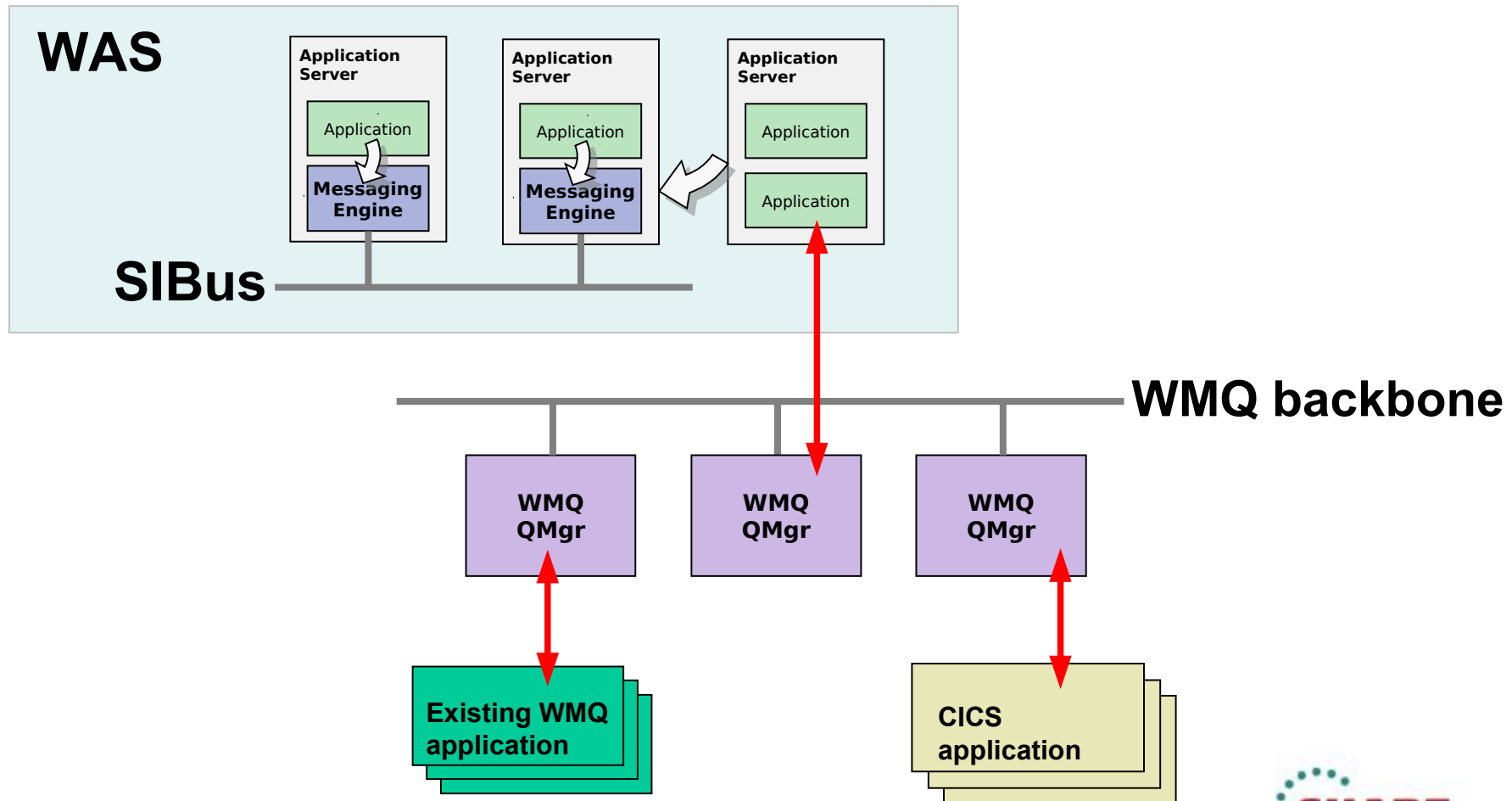
# Notes



- Translating messages between WMQ and WAS formats introduces an overhead
- As both WMQ and SIBus allow for asynchronous messaging, if one end of the link is down, messages will be queued up until the other is available.
- Potentially allows applications to continue operating if one of the systems goes down.

# Option 3 : Mixture of both

SIBus for WAS-WAS application messaging,  
WMQ for WAS-*'other'* messaging



# WebSphere MQ as the messaging provider



- Pre-deployed and ready to go
- Fully integrated into WAS administrative tools
  - Can even do limited WMQ administration from WAS
- Access to WMQ via TCP or shared memory (“bindings”)
- Supports connections to variety of WMQ queue manager versions
- Interfaces with WAS using the well-defined JCA interface
  - WebSphere MQ provided as a single unit “Resource Adapter”
- WMQ Resource Adapter is a component of WAS
  - Updates applied using WAS process, not WMQ

# Notes



- WMQ support is built directly into WAS
  - WMQ already deployed to WAS, no need to deploy WMQ
  - When creating a JMS resource, you are asked to choose between “default” messaging and WMQ messaging.
  - Maintenance of the WMQ Resource Adapter (RA) is through WAS service channels, such as WAS interim fixes/fix packs
- WMQ JMS code can connect to any version of WMQ queue manager
  - No tight coupling between WMQ JMS client version and QM version.
  - WMQ JMS version in use detailed here: <http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg21248089>
- The WMQ transport can be either TCP, or shared memory/bindings mode
  - Bindings mode available when queue manager and application server on the same machine
    - *Uses JNI interface, requires native library to be loaded into the WMQ Resource Adapter.*
    - *Need to configure the native library using the RA configuration panel*
- JCA Interface, well defined interface for application servers to access Enterprise Information Systems.
  - Same WMQ interface used in other JEE application servers
  - Same connector interface as used by SIB and other providers in WAS
- MQ\_INSTALL\_ROOT variable no longer used to locate WMQ JMS code
  - *Can still be used to locate native libraries used by WMQ.*

# Configuring for WMQ : Activation Specifications



Resources > JMS > Activation Specifications > [New]

Activation specifications > AS

WebSphere MQ Activation Specification

Configuration

## General Properties

### Administration

Scope

Node=localhostNode03,Server=server1

Provider

WebSphere MQ Resource Adapter

\* Name

AS

\* JNDI name

jms/AS

Description

### Connection

Queue manager

MyQM

Transport

Bindings, then client

☒ Enter host and port information in the form of separate hostname and port values

\* Hostname

localhost

Port

1414

☐ Enter host and port information in the form of a connection name list

Connection name list

- Activation Specs are the standardised way of delivering messages to an MDB
- The WebSphere MQ messaging provider in **WAS V7** adds support for Activation Specifications
  - Listener Port functionality is stabilized.
- Activation Specs combine the configuration of connectivity, the JMS destination to be processed and the runtime characteristics of the MDB itself
- Can be tested during creation to verify they work.
- Activation specifications can be defined at all WAS configuration scopes, as can be done for ConnectionFactories and Destinations.

# Activation Specs



- In WAS V7, the introduction of the WebSphere MQ V7 JCA 1.5 adapter enables MDB's to use activation specs to connect to queue managers rather than using listener ports.
- Activation specs are the strategic direction for delivering JMS messages to MDBs
- Activation specs are easier to administer as they are definable at any scope where as listener ports are defined on a per server basis
- A new wizard has been created which assists in the migration of existing listener ports to activation specs, note that this only creates a new activation spec with the same configuration as the listener port, it does not modify application deployments to use the newly created activation spec

# Configuring for WMQ : Connection Factories

Resources > JMS > Connection factories > [New]

Connection factories > MyCF

A unified JMS connection factory can be used to create JMS connections to both queue and topic destination

Configuration

## General Properties

### Administration

Scope

Node=localhostNode03,Server=server1

Provider

WebSphere MQ messaging provider

\* Name

MyCF

\* JNDI name

jms/MyCF

Description

### Connection

Queue manager

MyQM

Transport

Bindings, then client

☐ Enter host and port information in the form of separate hostname and port values

Hostname

localhost

Port

1414

☒ Enter host and port information in the form of a connection name list

\* Connection name list

localhost(1414)

Server connection channel

SYSTEM.DEF.SVRCONN

- Specify how an application connects to a WMQ queue manager
- Typically requires:
  - Queue manager name
  - Hostname and port
  - Channel name
  - Other parameters such as:
    - Transport type (client or bindings)*
    - Use of SSL*
- Or, can use WMQ **client channel definition table** (CCDT) URL



# Configuring for WMQ : Destinations

Resources > JMS > Queues / Topics > [New]

**Queues** > Q

Queue destinations provided for point-to-point messaging by the WebSphere MQ destinations for the WebSphere MQ messaging provider.

Configuration

---

**General Properties**

---

**Administration**

Scope  
Node=localhostNode03,Server=server1

Provider  
WebSphere MQ messaging provider

\* Name  
Q

\* JNDI name  
jms/Q

Description

---

**WebSphere MQ Queue**

\* Queue name  
Q

Queue manager or Queue sharing group name

Apply OK Reset Cancel

- Defines references to the resources in WMQ that a JMS application will use
  - The WMQ resources must be created using WMQ administration
- **Queues**
  - Identifies the actual queue in WMQ
    - *Can be used to set properties such as persistence, priority, etc.*
- **Topics**
  - Defines the WMQ publish/subscribe destination

# Configuration for SIB to WebSphere MQ

**Buses**

[Buses](#) > [MyBUS](#) > **Bus members**

Bus members are the servers, WebSphere MQ servers and clusters that have been added to the bus.

⊕ Preferences

Add Remove

⊞ ⊞ ⊞ ⊞

Select	Name	Type
You can administer the following resources:		
<input type="checkbox"/>	<a href="#">SIMON</a>	<a href="#">WebSphere MQ server</a>
<input type="checkbox"/>	<a href="#">localhostNode03:server1</a>	<a href="#">Server</a>
Total 2		

- **Queue manager as a bus member**
  - WMQ Queues linked as bus destinations
  - No asynchronous delivery, if QM is unavailable, destinations cannot be used
  - Can easily read and write messages to queues

**Buses** > [MyBUS](#) > **Foreign bus connections**

A foreign bus connection allows communication with another bus. The foreign bus connection to another foreign bus.

⊕ Preferences

New... Delete Test connection

⊞ ⊞ ⊞ ⊞

Select	Name	Routing type
You can administer the following resources:		
<input type="checkbox"/>	<a href="#">QMSIMON</a>	Direct, WebSphere MQ link

- **Queue manager as a foreign bus (MQLink)**
  - WMQ Queues linked as foreign destinations
  - Messages stored on SIBus if QM is unavailable
  - More complicated to read messages from WMQ Queues

# SIB → MQ link



- Queue Manager as a bus member
  - First define the queue manager as a WebSphere MQ server in the “Servers” section
- In the Service Integration → Buses → *BusName* → Bus members view, Add the queue manager
- Add the destinations you want to make available on the bus, for example in the Service Integration → Buses → *BusName* → Destinations view, add a new queue destination.
  - Specify the appropriate WebSphere MQ bus member, and use the queue filter to locate the existing WMQ queue
- Create JMS connection factories and JMS destinations, and then use as normal
- Can read and write messages directly from WMQ Queues, with confidence that messages arrive at their destination.
- Queue Manager as a foreign bus
  - In the Service Integration → Buses → *BusName* → Foreign Bus connections view, click New, and create a Direct connection to a WebSphere MQ queue manager
    - Define unique Foreign Bus name and MQLink name
    - Specify sender/receiver channels on WMQ to use (and ensure they exist on WMQ!)
    - The WMQ port used by WAS is defined by the port value SIB\_MQ\_ENDPOINT\_ADDRESS (e.g.5558)
  - Create JMS destinations that reference the foreign bus destination (Connection Factories still connect to the local bus)
  - Uses WMQ Sender/receiver channels, and so messages are sent to existing queues. Return path uses remote queue definitions/transmission queues. Undelivered messages sent to dead-letter queues.
- QM as a bus member is strongest for incoming messages, QM as a foreign bus is strongest for outgoing (it can queue up messages when the QM is unavailable).
  - Can use a combination of methods.

# Agenda

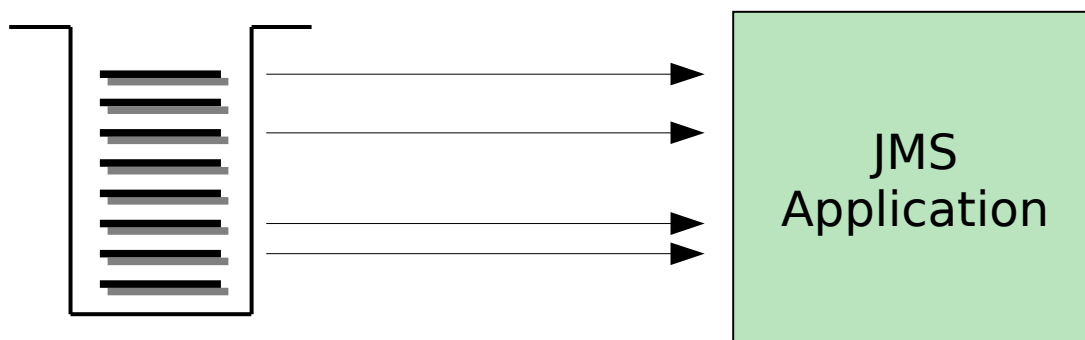
- Why are we here?
- Messaging in WAS
  - What is JMS?
- Accessing WMQ from WAS
  - How to configure
  - New WMQ V7 features useful for WAS
- Configuring WMQ and WAS
  - Useful tuning parameters
  - High Availability and clusters

## New functional areas in WMQ V7 of interest to JMS

- Message Properties + Selectors
- Asynchronous Consumption of messages
- Simplified Browse + Co-operative Browse
- Asynchronous Put Response
- Read-ahead of messages
- Connection changes
- Multi-instance Queue Managers (WMQ 7.0.1)

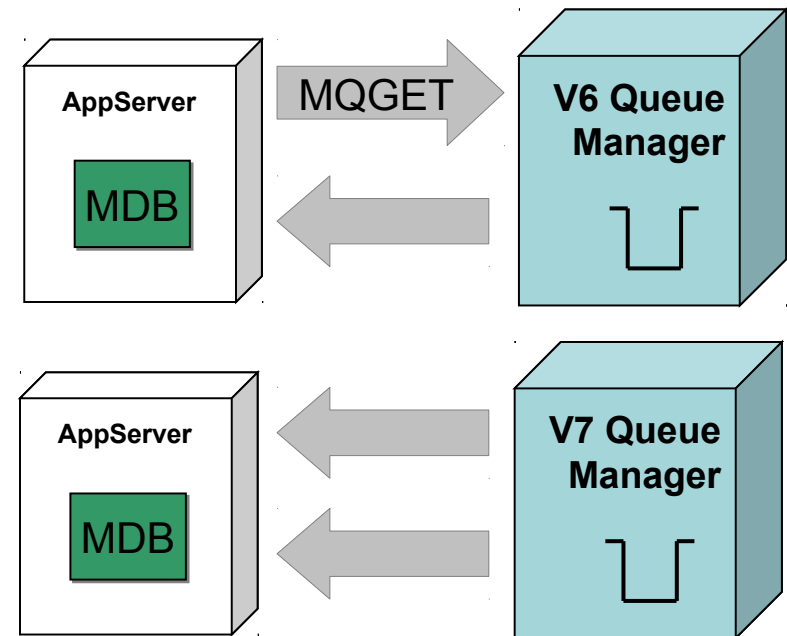
# Message properties and Selectors

- JMS Selectors allow messages from a destination to be filtered
  - Filter on system or user message properties
- WMQ V7 queue manager understands JMS message properties
- Queue manager evaluates JMS selector
  - Only transmits matching messages to the client
  - Efficient delivery of messages



# Asynchronous Consume

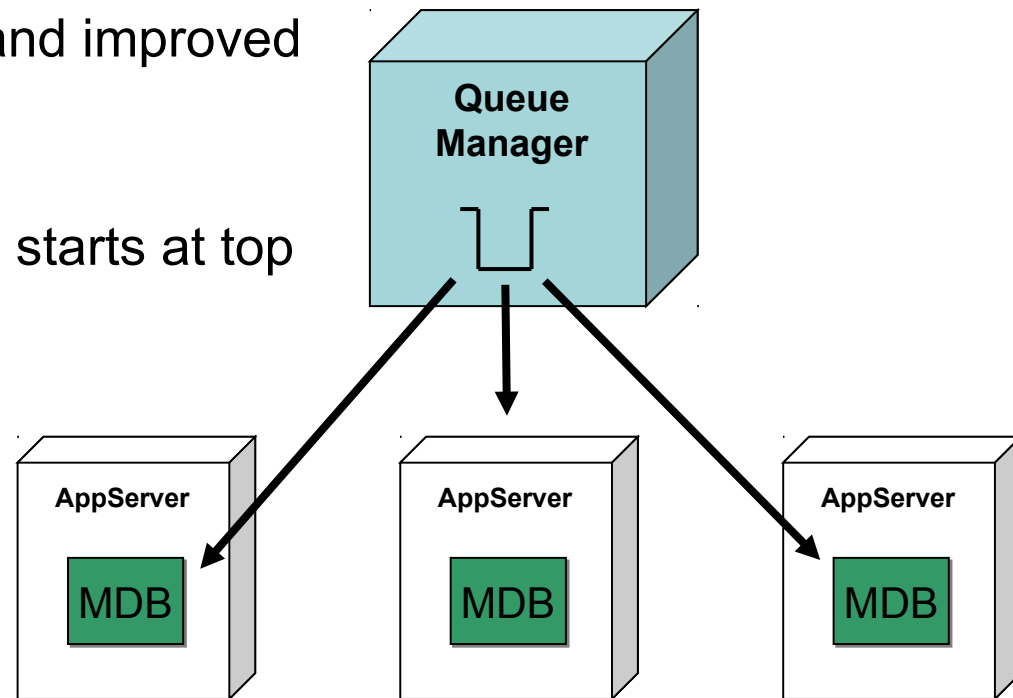
- JMS has had concept of asynchronous consumer since inception
  - MessageListener / onMessage code
- Event driven processing 'natural' for the Java Environment
- WMQ V7 introduced ability of queue manager to drive consumer when suitable message available
  - Avoids need for 'polling thread'
  - More efficient message delivery, especially with selectors.
- With proper support in WMQ – JMS implementation simplified
- Less client CPU overhead of this busy thread



# Asynchronous Consumer and Browse-with-mark



- Multiple application servers will contend for messages when processing the same queue.
  - Can result in failed message delivery and wasted resources
- Browse-with-mark allows the application server to hide messages it has browsed and intends to process
  - No contention for messages, and improved throughput.
- The MDB browse thread always starts at top
  - Message priority honoured





# Browse - Mark



- Prior to WMQ v7 – only z/OS WMQ had a feature of ‘Browse-Mark’
  - Now all MQ v7 Queue Managers have browse mark
- Let a browser mark messages when it had browsed them
- Other browsers can request to not see marked messages
  - Messages marked for a period of time defined on the queue manager
  - Browsers that do not opt to hide marked messages can see all messages
    - *V6 style JMS clients can still remove messages and cause message contention*
- This lets asynchronous message delivery be more efficient
- Less chance of contention when scaled up to multiple JVMs
  - Contention can still occur if application servers take a long time (> message browse mark interval) time to process message.
  - JMSSCC0108 error reports this condition in WAS SystemOut.log
    - *Indication that server needs to be tuned, e.g. increase MDB sessions, number of MDB processing threads, or decrease MDB processing time.*
- Reduction in failed MQGETs and rolled back transactional work in WAS gives significant performance benefit

# Agenda

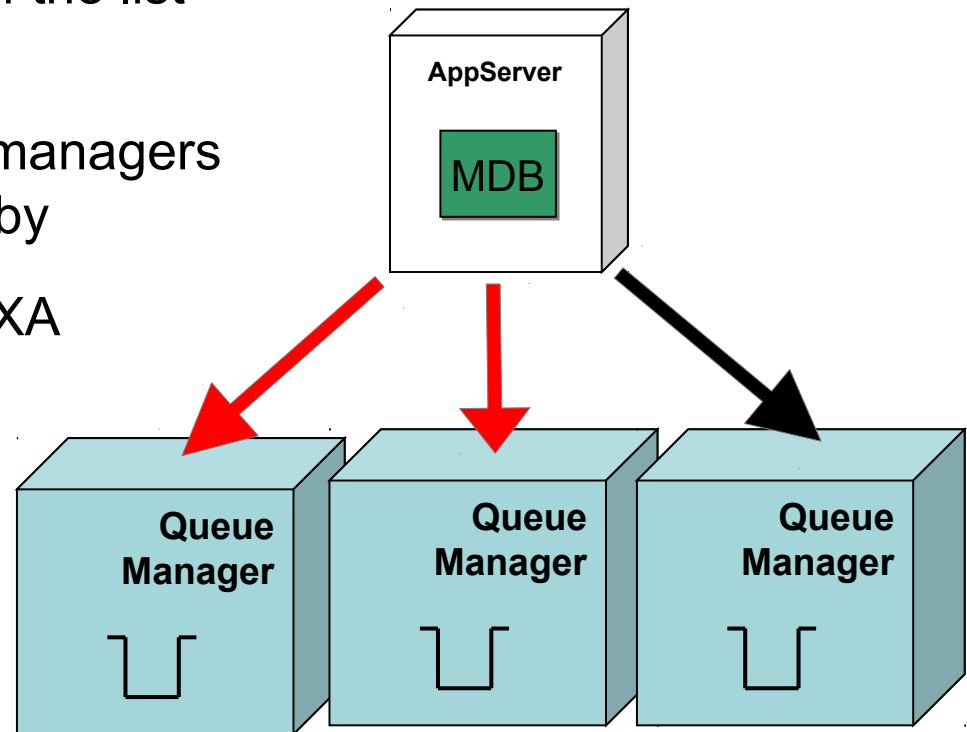
- Why are we here?
- Messaging in WAS
  - What is JMS?
- Accessing WMQ from WAS
  - How to configure
  - New WMQ V7 features useful for WAS
- **Configuring WMQ and WAS**
  - Useful tuning parameters
  - High Availability and clusters

## WMQ Connection Factory Properties: Provider Version

- The version of WMQ that the queue manager this Factory is pointing to is running on.
  - The client is optimized to connect to Version 7 queue managers, and makes use of Version 7 functionality, such as asynchronous message delivery.
- The WMQ JMS client can be used to connect to Version 5.1 queue managers and above.
  - This uses a slightly modified version of the WMQ Version 6 JMS Client, which is embedded in the Version 7 code.
  - No tight coupling between WMQ JMS client and queue manager versions
- “Pre-warning” the WMQ JMS code it is connecting to a Version 6 or earlier queue manager speeds up connection time.

# Connection Name List

- This property specifies a list of hostnames and ports to attempt to connect to.
  - Comma-separated list of “hostname(port)” entries
- Entries in the list are tried until a working connection is found, or the end of the list is reached.
- Used with multi-instance queue managers automatically reconnect to standby
- **Must** be same QM data if using XA
- Standard property in WAS V8
  - WAS V7 custom property

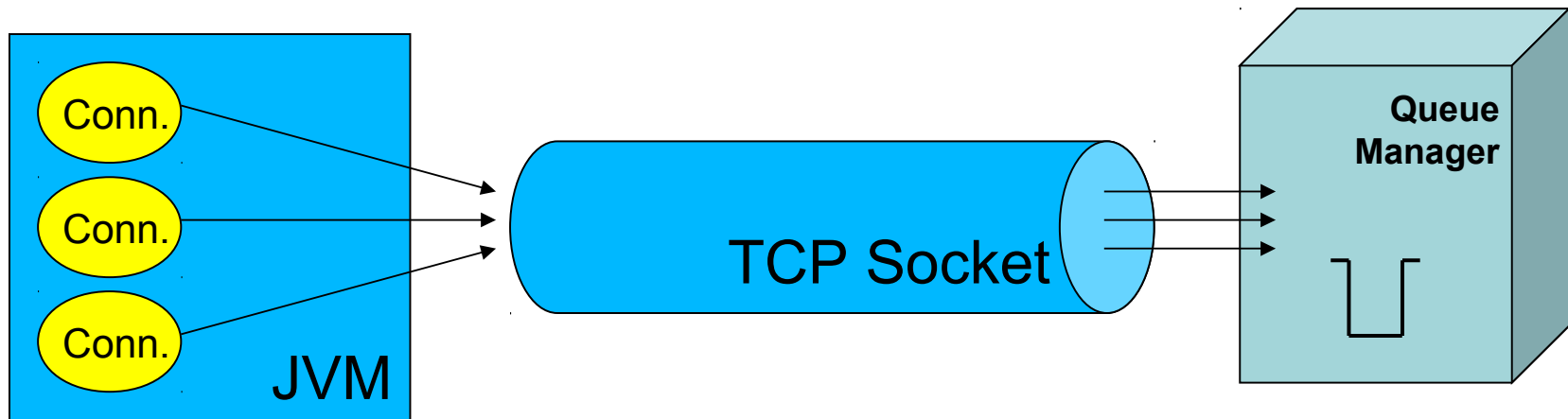


# Connection Name List Notes

- Similar to a CCDT with multiple entries
  - Does not do connection balancing that is possible with CCDTs
- If applications are transacted, the entries in the list must refer to the same QM data (such as the active and standby queue manager instances)
- WAS transaction manager will be unaware that connection is to a different hostname/port, and will expect the same transactions to be visible when connecting to the ConnectionFactory.
- If different QMs are referenced, XA transaction recovery will not work, and transactions will require manual resolution.
- Same restriction applies to CCDTs
- Setting the custom property:
  - Configuring custom WAS properties
    - <http://www-01.ibm.com/support/docview.wss?uid=swg27020700&myns=swgws&mynp=OCSSFKSJ&mynp=OCSSEQTP&mync=R>
  - Connection Factories:
    - *Set a custom property called XMSC\_WMQ\_CONNECTION\_NAME\_LIST to the list of host/port names that you wish to connect to. For example: host1(port1),host2(port2)*
  - Activation Specs:
    - *Set a custom property called connectionNameList on the activation spec with the same format as above: host1(port1),host2(port2)*

# 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

# Shared Conversation Allowed Notes



- Possible Values
  - Yes
    - *JMS applications running in the same Java Virtual Machine can use this Factory and share their connections to the queue manager. The amount of sharing is controlled by the Server Connection Channel*
  - No
    - *Every JMS application that uses this Factory will create a new connection to the queue manager.*
- JMS applications tend to use more QM connections, as there is one per JMS connection and JMS session. Historically, this has led to the QM Max Channels limit being reached more easily, and requiring the limit to be increased. Shared Conversations mitigate this.

# Using WMQ JMS with existing WMQ applications



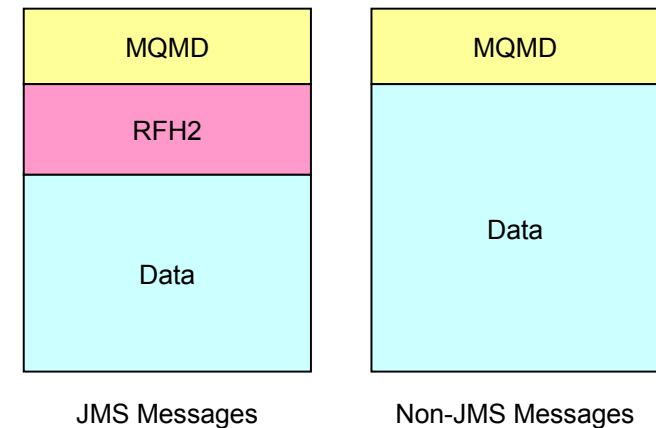
- MQMD Read/Write enabled
  - Determines if MQMD properties can be set via JMS get/set message property methods.
  - Allows full access to the MQMD header values
  - Useful for sending or receiving messages from MQ applications that use specific header properties.
  - JMS message property names begin with “JMS\_IBM\_MQMD...”

**Message descriptor**
☐ MQMD read enabled  
☐ MQMD write enabled

**Additional**
MQMD message context  

DEFAULT

- Target Client (JMS/MQ)
  - Indicates what the receiving application is
  - Removes the RFH2 header from **sent** messages WMQ applications may have issues with





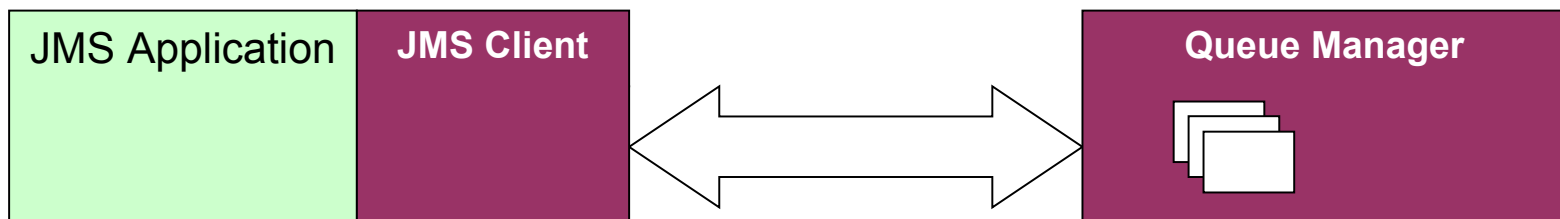
# MQMD Read/Write & Target Client Notes



- These are destination properties, set on the JMS Queue/Topic in use
- WMQ Manual page on MQMD Read/Write:
  - <http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.cs>
  - Note that MQMD Message Context also needs to be set to allow some properties to be set on the message
  - Care should be taken with direct access to the MQMD, as it is possible to violate the JMS specification

# Read Ahead Allowed

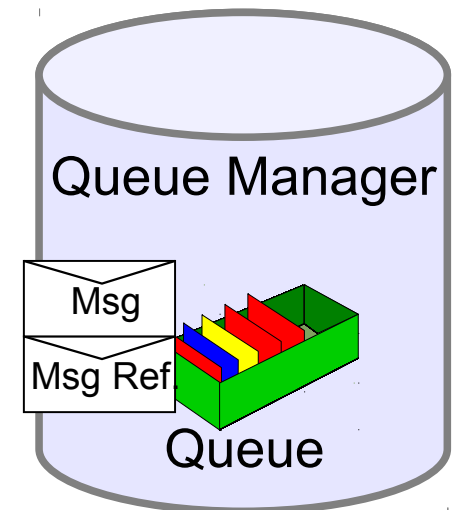
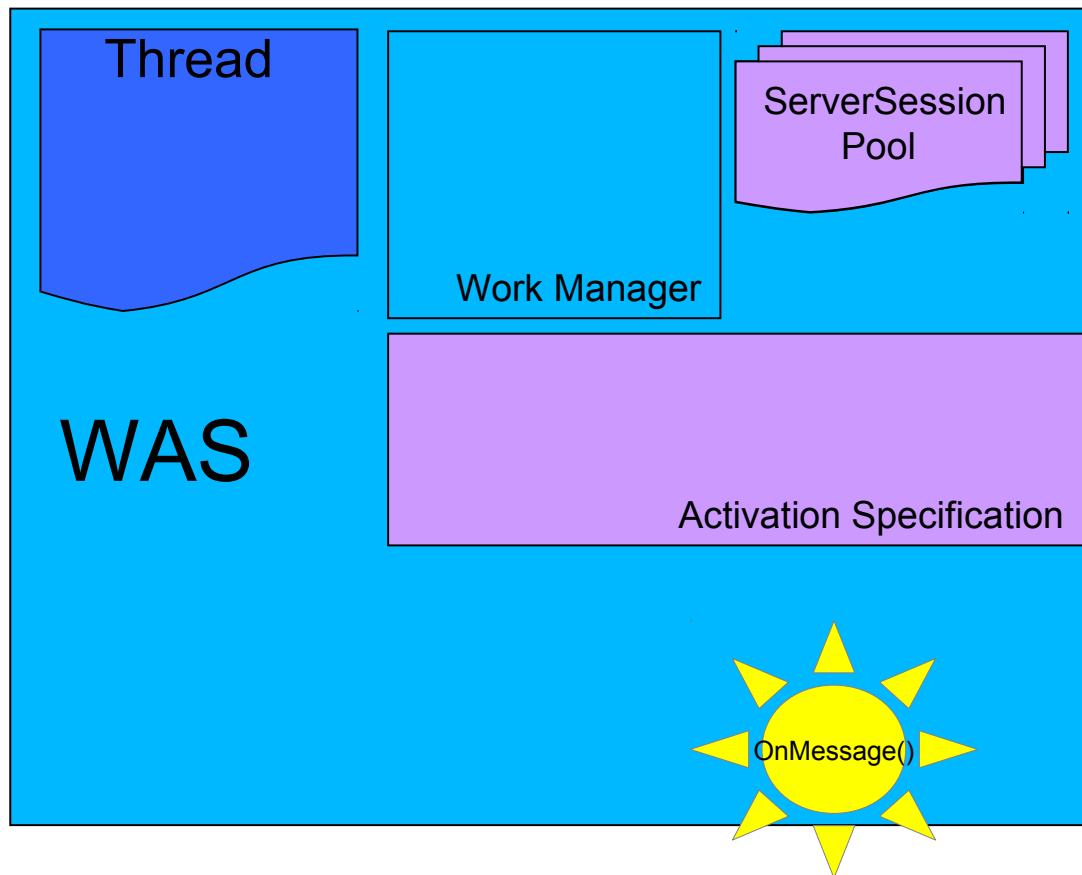
- In general, messages are sent to JMS applications one at a time.
- The Read Ahead Allowed property tells the queue manager whether non-persistent messages can be streamed to the client application in preparation for them being consumed.
  - Messages are stored in a buffer on the client.
  - If the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



# WAS Configuration: Activation Specifications

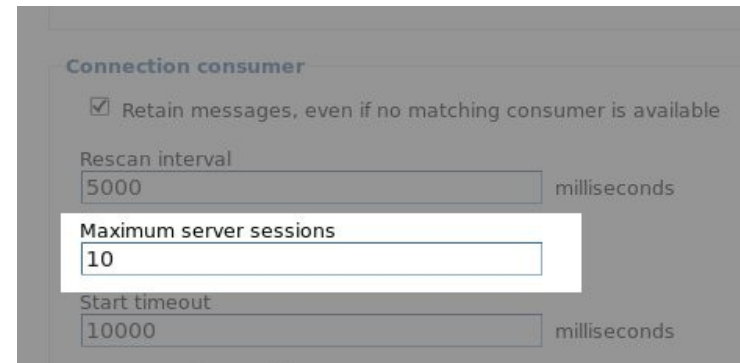


- How are messages destined for MDBs processed?



# Activation Specifications - Threads

- To process a message, a ServerSession **and** a thread is required
- Activation Specification parameter **Maximum server sessions** configures ServerSession pool, default 10
  - ServerSession pool per MDB
- Application Server Thread pool WMQJCAResourceAdapter used, default **Maximum size** 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



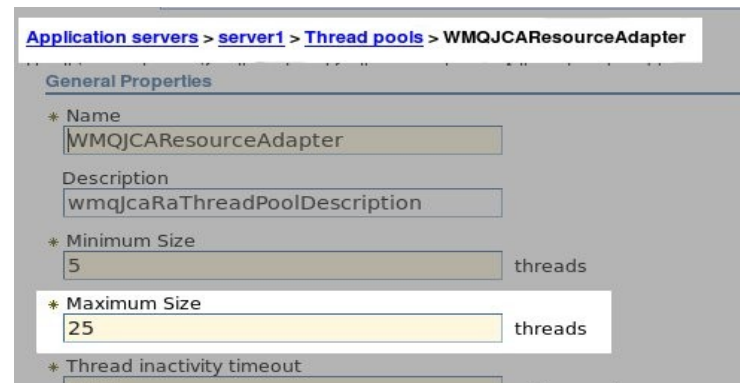
Connection consumer

☒ Retain messages, even if no matching consumer is available

Rescan interval  
5000 milliseconds

Maximum server sessions  
10

Start timeout  
10000 milliseconds



Application servers > server1 > Thread pools > WMQJCAResourceAdapter

General Properties

\* Name  
WMQJCAResourceAdapter

Description  
wmqjcaRaThreadPoolDescription

\* Minimum Size  
5 threads

\* Maximum Size  
25 threads

\* Thread inactivity timeout  
5000 milliseconds

# 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.
- If the Activation Specification stops, this is **only** reported in Application Server SystemOut.log logs.

# Activation Specifications - Recovery



- To configure WMQ RA settings, go to Resources → Resource Adapters, select a scope to configure, and in the Preferences, select “Show built-in resources”, and Apply.
- Select the “WebSphere MQ Resource Adapter”
- Select custom properties, and modify the “**reconnectionRetryCount**” and “**reconnectionRetryInterval**” properties
- Upon first failure, the RA will try an immediate reconnection attempt, and if that fails, retry at the configured interval.
- The properties are configured on the RA, and so affect **ALL** Activation Specifications using that RA

## Preferences

Maximum rows

20

☐ Retain filter criteria

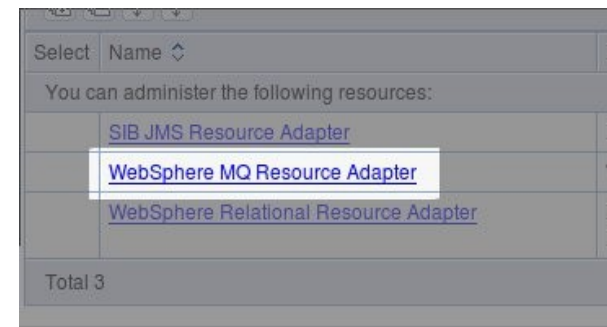
Show items at the following autl

All Roles ▾

☒ Show built-in resources

Apply

Reset



# JMS User authentication



`createConnection(user,password)`

- In JEE, JMS application-specified user and passwords are not necessarily used.
  - “Container-managed” authentication
- Activation Specifications / Connection Factories can be associated with authentication data
  - JAAS – J2C Authentication Data defines username and password details
  - The application needs to use JEE resource-references to access the connection factory
  - The *authenticationType* parameter needs to be set to container for container-managed authentication.
- As for other WMQ clients, security exits are required to validate passwords, WMQ only checks user id
  - User ids more important with WMQ 7.1 channel authentication

## Security settings

Select the authentication values for this resource.

Authentication alias for XA recovery

localhostNode03/MyUserID

Mapping-configuration alias

DefaultPrincipalMapping

Container-managed authentication alias

localhostNode03/MyUserID

## Related Items

- [JAAS - J2C authentication data](#)

# JMS User Authentication



- 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>application</res-auth>`, or not using resource-references means the application is responsible for any authentication information
- Resource references have other benefits, such as decoupling application configuration names from real JNDI names.
- As for other WMQ clients, security exits are required to validate passwords, WMQ only checks user id
  - User ids more important with WMQ 7.1 channel authentication
- If nothing specified, by default WMQ JMS passes a blank id
  - Custom property “com.ibm.mq.jms.ForceUserID=true” will pass the JVM process id.
  - <http://www-01.ibm.com/support/docview.wss?uid=swg21470801>
- ejb-jar.xml resource-ref values need a res-auth of container to use the CF security settings.
- Using annotations, will need to use the `@Resource` authenticationType

```
<resource-ref id="ResRef_1">
  <description></description>
  <res-ref-name>jms/CF</res-ref-name>
  <res-type>javax.jms.ConnectionFactory</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

```
@Resource(authenticationType = AuthenticationType.CONTAINER)
private ConnectionFactory cf = null;
```



# Agenda

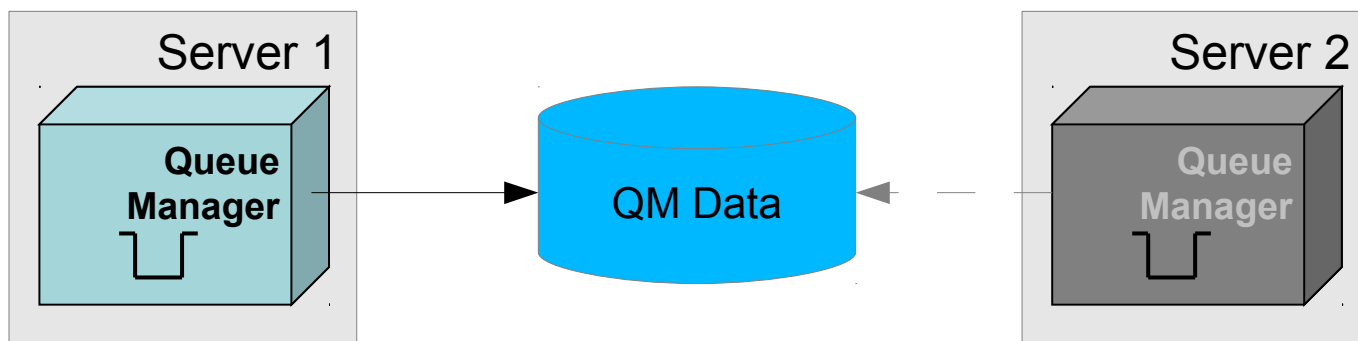
- Why are we here?
- Messaging in WAS
  - What is JMS?
- Accessing WMQ from WAS
  - How to configure
  - New WMQ V7 features useful for WAS
- Configuring WMQ and WAS
  - Useful tuning parameters
  - High Availability and clusters

# High Availability

- Treat WAS and WMQ as separate resources
- WMQ V7.0.1 features make it easier for WAS to reconnect during failover
  - Quicker detection of broken connections
  - Connection Name Lists
- OS-level HA solutions (HACMP etc.)
  - Active – Passive failover
  - Transparent as resources such as IP addresses are moved over
  - Can include WMQ as well as other resources in the same failover
  - Third party solution, introduces expense

# Multi-instance Queue Managers

- WMQ's implementation of Active – Passive failover
  - Only moves WMQ resources
  - Can be faster than OS-level HA
- Needs data to be stored in external storage with locking, such as NFS V4
- One active queue manager, with other standby instance
- Standby instance can detect failure and take over
  - Can also trigger failover
- Same data is shared, so it's the SAME queue manager.



# Multi-instance Queue Managers



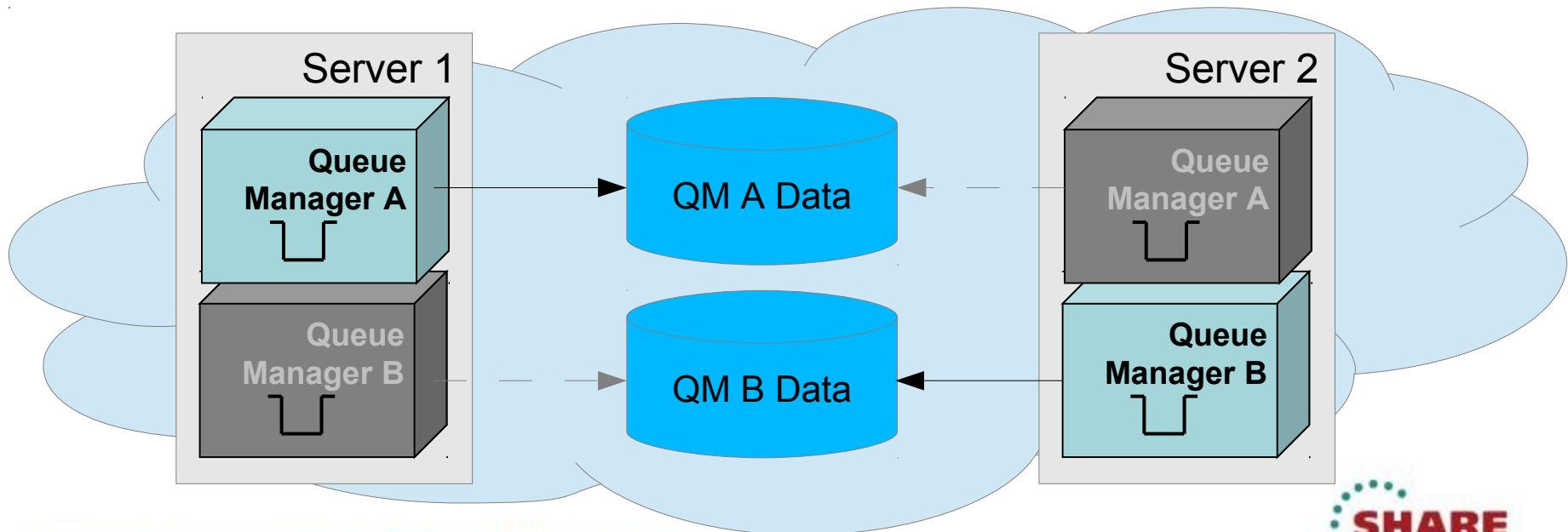
- Basic failover support without HA coordinator
  - Faster takeover: fewer moving parts
  - Cheaper: no specialised software or administration skills needed
  - Windows, Unix, Linux platforms
  - Needs WMQ 7.0.1 or later
- Queue manager data is held in networked storage
  - NAS, NFS, GPFS etc so more than one machine sees the queue manager data
  - Improves storage management options: formal support for these even without failover config
- Multiple instances of a queue manager on different machines
  - One is “active” instance; other is “standby” instance
  - Active instance “owns” the queue manager’s files and will accept app connections
  - Standby instance does not “own” the queue manager’s files and apps cannot connect
    - If active instance fails, standby performs queue manager restart and becomes active
- Instances share data, so it’s the SAME queue manager

# Multi-instance and Java EE

- Connection to Multi-Instance Queue Managers is supported
  - Minimum of WAS 7.0.0.13 required (contains WMQ 7.0.1.3)
  - Can be configured using CCDT
  - Can be configured using Connection Name Lists
- Automatic Client Reconnect **is not supported** (in EJB/Web container)
  - MDB based applications use ListenerPort/Activation Specification based reconnection
  - Automatic Client Reconnect can be used in unmanaged/client environments

# High Availability continued...

- Active – Active WMQ
  - Multiple active running queue managers
  - Provides service resilience and load balancing
    - *e.g. WMQ Clusters, z/OS Queue Sharing Groups (QSG)*
  - Can have stranded messages problems if queue manager becomes unavailable.
    - *Can use HACMP/Multi-instance queue managers to avoid*



# Useful Information: Product Connectivity Scenarios



- Scenario-based instructions for implementing a solution in a business context
- Providing a thorough, complete, and single end-to-end main path from the business need (high level) to the tasks (low level)
- Optional features help you learn as you progress through each task
  - Information about *why* you are instructed to do something
  - Information about *what else* you might do, or want to learn about, related to what you are reading in the main window
- Includes samples that you can try out and adapt

<http://publib.boulder.ibm.com/infocenter/prodconn/v1r0m0/index.jsp>

# Useful information: Product Connectivity Scenarios



- Two **Connecting WebSphere Application Server to WebSphere MQ** scenarios are already available:
  - The *Getting Started* scenario guides you through the task steps that are required to allow a Java Message Service (JMS) application running on WebSphere Application Server to connect to WebSphere MQ.
  - Starting with an existing WebSphere Application Server and WebSphere MQ Version 6 installation, the *Migrating from Version 6 to Version 7* scenario leads you through the key tasks required to migrate to Version 7.



# Further Information

- WAS product information : <http://www-306.ibm.com/software/webservers/appserv/was/>

- WAS Information Centers :

- 6.0 <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>
- 6.1 <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>
- 7.0 <http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp>
- 8.0 <http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp>
- 8.5 <http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp>

- Product Connectivity Information Center

<http://publib.boulder.ibm.com/infocenter/prodconn/v1r0m0/index.jsp>

- Using WebSphere MQ Java Interfaces in J2EE/JEE Environments

<http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg21266535>

- IBM developerWorks : <http://www.ibm.com/developerworks>

(Searching on “Service Integration Bus” returns a number of interesting articles)

- [http://www.ibm.com/developerworks/websphere/techjournal/0901\\_leming/0901\\_leming.html](http://www.ibm.com/developerworks/websphere/techjournal/0901_leming/0901_leming.html) - WASV7
- [http://www.ibm.com/developerworks/websphere/techjournal/0601\\_ratnasinghe/0601\\_ratnasinghe.html](http://www.ibm.com/developerworks/websphere/techjournal/0601_ratnasinghe/0601_ratnasinghe.html) - Security
- [http://www.ibm.com/developerworks/websphere/techjournal/0601\\_smithson/0601\\_smithson.html](http://www.ibm.com/developerworks/websphere/techjournal/0601_smithson/0601_smithson.html) - Security

- IBM RedBooks : <http://www.redbooks.ibm.com>

- *WebSphere Application Server V7: Messaging Administration Guide* SG24-7770-00
- *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708-00
- *WebSphere Application Server V7: Technical Overview*, REDP-4482-00
- *WebSphere Application Server V6.1: JMS Problem Determination*, REDP-4330-00
- *WebSphere Application Server V6.1: System Management & Configuration*, SG24-7304-00
- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392-00
- *WebSphere Application Server V6.1 Security Handbook*, SG24-6316-01
- *WebSphere Application Server V6.1: Technical Overview*, REDP-4191-00
- *WebSphere Application Server V6.1: Planning and Design*, SG24-7305-00
- *WebSphere Application Server V6.1: Installation Problem Determination*, REDP-4305-00

## Further Information (2)

- IBM RedBooks
  - <http://www.redbooks.ibm.com>
    - *WebSphere Application Server V7: Messaging Administration Guide SG24-7770-00*
    - *WebSphere Application Server V7: Concepts, Planning and Design, SG24-7708-00*
    - *WebSphere Application Server V7: Technical Overview, REDP-4482-00*
    - *WebSphere Application Server V6.1: JMS Problem Determination, REDP-4330-00*
    - *WebSphere Application Server V6.1: System Management & Configuration, SG24-7304-00*
    - *WebSphere Application Server V6 Scalability and Performance Handbook, SG24-6392-00*
    - *WebSphere Application Server V6.1 Security Handbook, SG24-6316-01*
    - *WebSphere Application Server V6.1: Technical Overview, REDP-4191-00*
    - *WebSphere Application Server V6.1: Planning and Design, SG24-7305-00*
    - *WebSphere Application Server V6.1: Installation Problem Determination, REDP-4305-00*

# Agenda

- Why are we here?
- Messaging in WAS
  - What is JMS?
- Accessing WMQ from WAS
  - How to configure
  - New WMQ V7 features useful for WAS
- Configuring WMQ and WAS
  - Useful tuning parameters
  - High Availability and clusters

# This was session 12611 - The rest of the week .....



	Monday	Tuesday	Wednesday	Thursday	Friday
08:00					Are you running too many queue managers or brokers?
09:30		What's New in WebSphere Message Broker			Diagnosing Problems for MQ      CICS and WMQ - The Resurrection of Useful
11:00		Extending IBM WebSphere MQ and WebSphere Message Broker to the Cloud	WMQ - Introduction to Dump Reading and SMF Analysis - Hands-on Lab	BIG Data Sharing with the cloud - WebSphere eXtreme Scale and WebSphere Message Broker integration	Getting the best availability from MQ on z/OS by using Shared Queues
12:15					
01:30	Introduction to MQ	MQ on z/OS – Vivisection	Migration and maintenance, the necessary evil	The Dark Side of Monitoring MQ - SMF 115 and 116 Record Reading and Interpretation	
03:00	First Steps With WebSphere Message Broker: Application Integration for the Messy	BIG Connectivity with WebSphere MQ and WebSphere Message Broker	WebSphere MQ CHINIT Internals	<b>Using IBM WebSphere Application Server and IBM WebSphere MQ Together</b>	
04:30	WebSphere MQ application design, the good, the bad and the ugly	What's New in the WebSphere MQ Product Family	MQ & DB2 – MQ Verbs in DB2 & Q-Replication	WebSphere MQ Channel Authentication Records	
06:00			Clustering - The Easier Way to Connect Your Queue Managers		



# Copyright and Trademarks



© IBM Corporation 2013. All Rights Reserved.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).