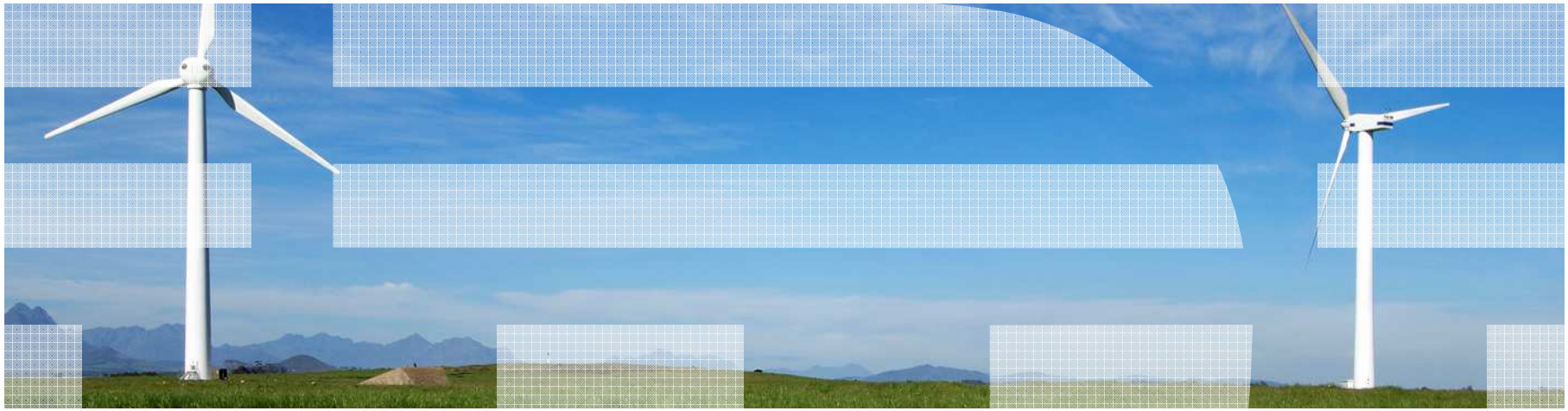
Integration of WebSphere MQ with JavaEE

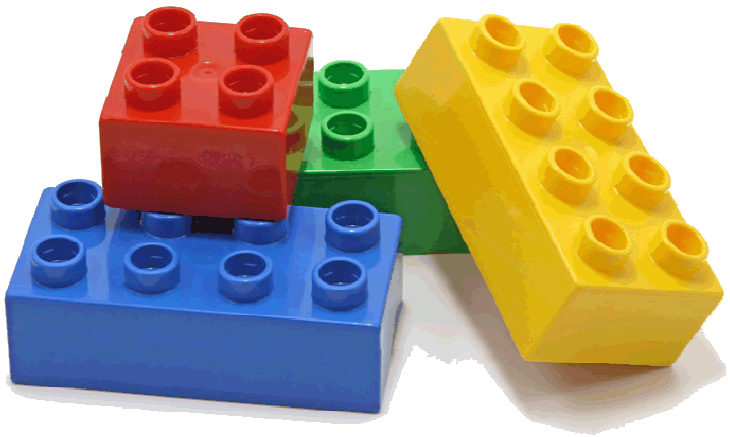
Matthew B White
IBM WebSphere MQ Development



Agenda

- How does WebSphere MQ really work when used as a messaging backbone in an Application Server Environment?
- What are the differences between WMQ v6 and WMQ v7 when deployed within WAS?
- What considerations are there with deploying within other App Servers?

The basics



WebSphere MQ *in a slide*

- WebSphere MQ (WMQ)
  - WMQ is IBM's flagship asynchronous messaging product
  - Queues are managed by **queue managers**
  - Queue Managers can be connected together to form a network, or *cluster*
  - Messaging applications connect to queue managers to access queues
    - Applications can send messages to any queue in the network from their queue manager
    - Applications can receive messages from queues on their queue manager
  - WMQ supports the point-to-point and publish/subscribe messaging models

- WMQ as the JMS provider in WAS
  - WMQ supports the JMS programming interface for Java applications.
  - Applications in WAS can use JMS to interact with WMQ queues (and topics)
    - WAS V6 and V6.1 contains a WMQ JMS client to enable this
    - WAS V7 contains the WMQ V7 JCA resource adaptor
  - WAS applications connect to a queue manager using:
    - **Bindings** – when application and queue manager are on the same server
    - **Client** – when application server and queue manager are on separate, networked, servers - using a TCP/IP socket

WAS messaging *in a slide*

- WebSphere Application Server (WAS) is a JEE application server, JEE includes support for asynchronous messaging using **JMS** (Java Message Service)
- Allows Java application interactions to be loosely coupled
  - Temporal and transactional decoupling of applications
- A standardised programming interface
  - Implemented by a *JMS provider*
  - JMS applications can be coded to be ignorant of the underlying JMS provider
    - 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

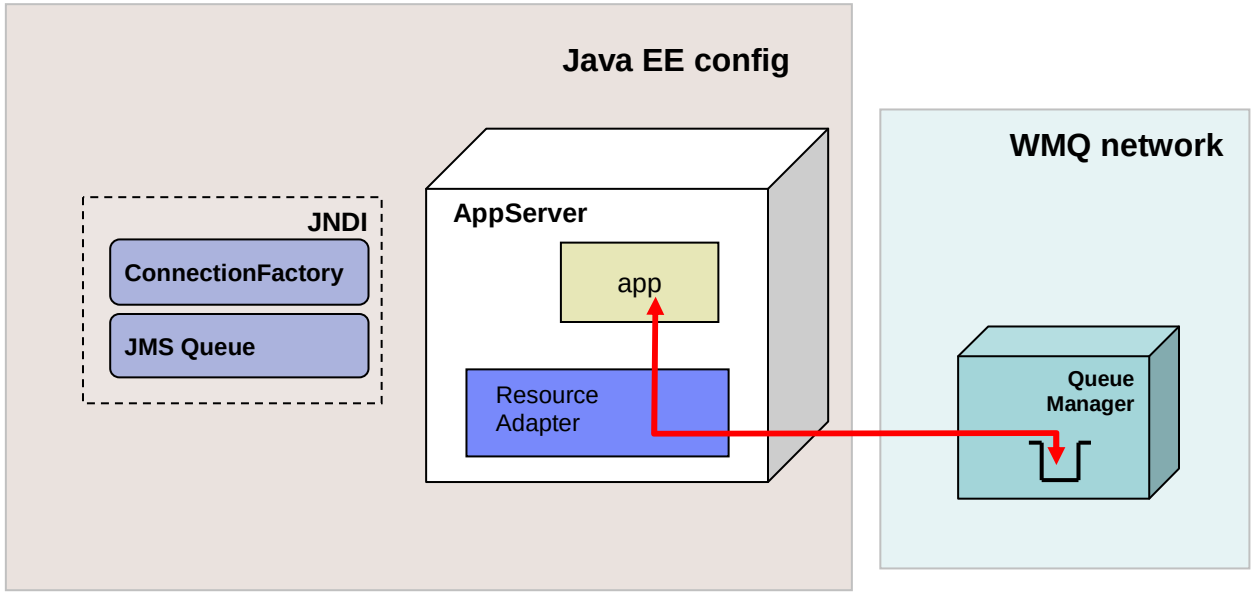
  JMS Supports two messaging models
  - **Point-to-point** – using queues
  - **Publish/subscribe** – using topics
- WebSphere Application Server provides first class support for two JMS messaging providers
  - An embedded messaging provider, the ***Default Messaging provider*** (or **Service Integration Bus**)
  - **WebSphere MQ** as an external messaging provider

Service integration bus *in a slide*

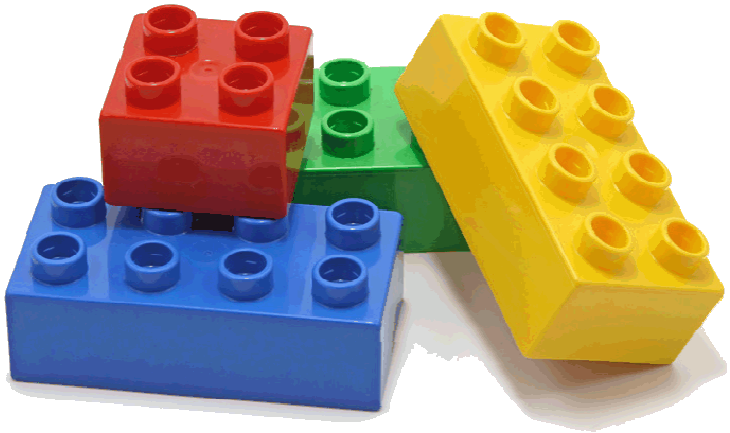
- Service Integration Bus (SIBus)
  - SIBus is a native component of WAS, forming the asynchronous messaging platform for application server based products
  - Logically:
    - A ***bus*** is a logical entity that contains ***destinations*** (e.g. queues/topic spaces)
    - Messaging applications connect to the bus to access the destinations
    - A bus is **location transparent**, all destinations and their messages are available from anywhere in the bus
  - Physically:
    - 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
    - SIBus as the JMS provider in WAS
  - SIBus supports the JMS programming interface for Java applications.
    - Natively supporting both point-to-point and publish/subscribe messaging models
  - SIBus is the default JMS provider in WAS – hence the ***"Default messaging provider"***

WMQ Resource Adapter *in a slide*

- The resource adapter allows applications running in an application server to access WebSphere MQ resources. It supports inbound and outbound communication.
- The Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) provides a standard way of connecting applications running in a Java EE environment to an Enterprise Information System (EIS) such as WebSphere® MQ or DB2®. The WebSphere MQ resource adapter implements the JCA 1.5 interface



The basics in a little more detail



WMQ and WAS – the architectural relationship

- Within WAS there is the ‘WebSphere MQ messaging provider’

- WAS V6.0 and V6.1
  - The WebSphere MQ Client came with WAS
  - Referenced with the use the WebSphere environment variable MQ_INSTALL_ROOT.
  - 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

- WAS V7
  - 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)
  - Native library path for the RA is used to identify the location of the required native libraries
  - Service updates to RA now solely by use of the WAS FixPacks

WMQ and Java EE in general

- The WMQ Resource Adapter – JCA 1.5
- Allows any application server that supports JCA 1.5 to access WMQ resources
- Some of these Application Servers can interact via generic JMS or have WMQ specific Adapters
- Recommended route is now to consider WMQ Resource Adapter for all integration requirements

What's the relationship with ASF

- ASF – Application Server Facilities
- Optional portion of the JMS specification intended to allow integration of JMS to JavaEE
- Provides concurrent asynchronous delivery of messages

- The Resource Adapter utilizes the features of ASF to provide the same level of asynchronous delivery
- Resource Adapter does not provide the non-ASF mode previously found in WAS

New in WMQ v7 and v7.0.1

- WMQ V7 and V7.0.1 added some notable new features.
- How do these affect the integration into the application server?
- How do you exploit these within the application server?

WebSphere MQ Version 7 – in general

- Central requirement was to improve JMS implementation
  - More applications being written to use this API
  - Underpins many SOA/ESB solutions needing access to messaging
- Lead to designs involving features such as
  - Ease-of-use
  - Performance
- But it also leads to enhancements for ALL applications
  - Not just JMS users
- Extension of publish/subscribe capabilities
  - Designed with Message Broker in mind
- Easier programming in any environment
  - Some features suggested by JMS requirements are useful in MQI
- Administration model and APIs natural evolution of existing interfaces

Main new functional areas in WMQ v7

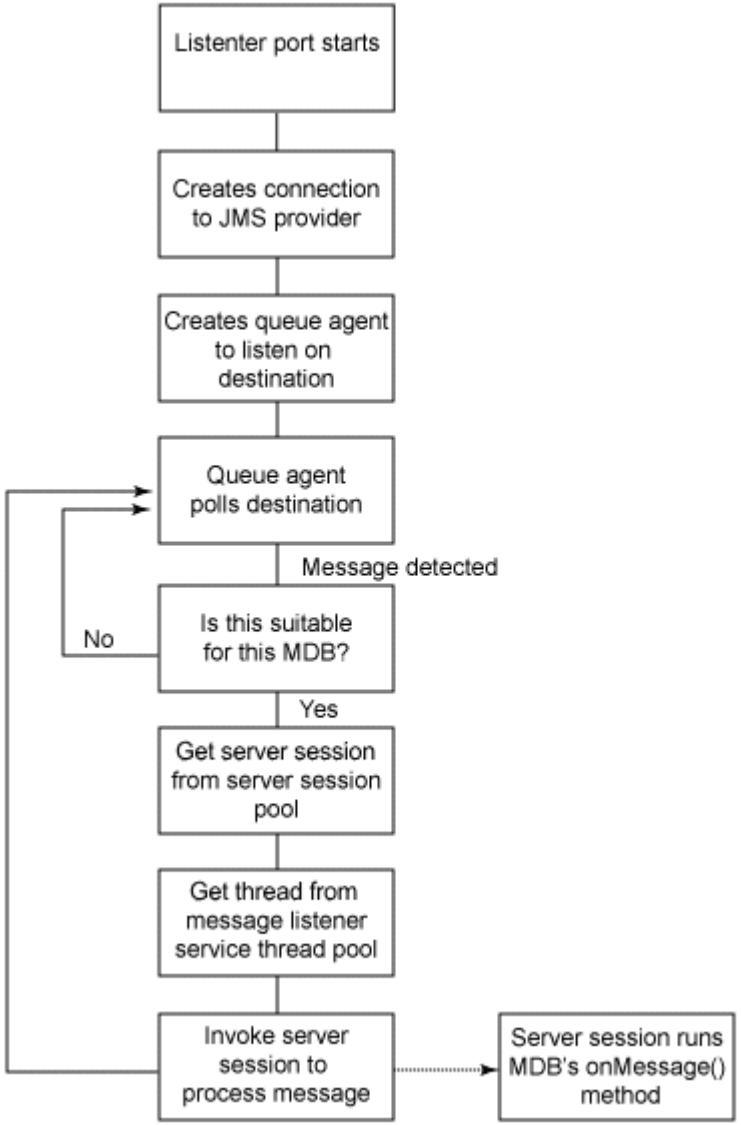
- Asynchronous Consumption of messages
- Asynchronous Put Response
- Read-ahead of messages
- Connection changes
- Message Properties + Selectors
- Simplified Browse + Co-operative Browse
- Multi-instance QueueManagers (WMQ 7.0.1)

… of interest to Java EE

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

Refresher on ASF Mode

- Two main parts
  - Browse loop looking for messages
  - Destructive get
- Issues
  - Browse loop is busy loop
  - Browse loop had the potential to miss messages either arriving with differing priorities or at the end of the queue
  - If multiple MDBs potential to race to destructively get the message
- *Search on line for 'Titheridge ASF' for more info*



Asynchronous Consume

- JMS has had concept of asynchronous consumer since inception
- Event driven processing ‘natural’ for the Java Environment

- WMQ only support Synchronous Gets – message listener processing therefore had to be implemented with a ‘Polling Thread’

- With proper support in WMQ – JMS implementation simplified
- Less client CPU overhead of this busy thread

Browse - Mark

- Prior to WMQ v7 – zOS WMQ had a feature of ‘Browse-Mark’
- Let a browser mark messages when it had browsed them
- This prevented, for a fixed time, other browsers from seeing the message

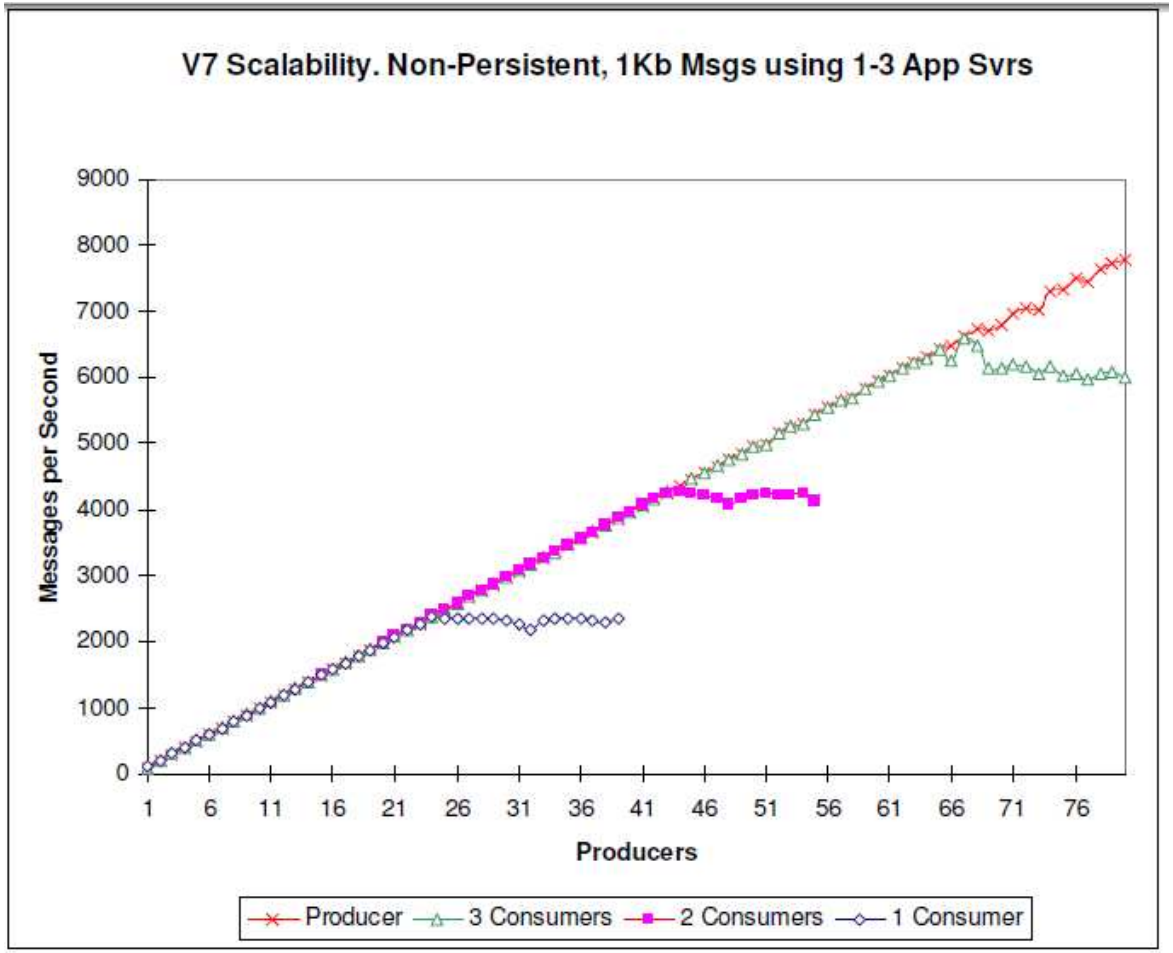
- This let the ASF destructive get more more efficient
- Less chance of contention

- Reduction in failed gets significant benefit

Asynchronous Consumer and Browser-Mark

- Not needing a polling loop anymore has big advantages
- The Browse thread will honour messages with high priority arriving on the queue
- No risk of the polling thread being stuck at the end of a queue by constantly arriving messages

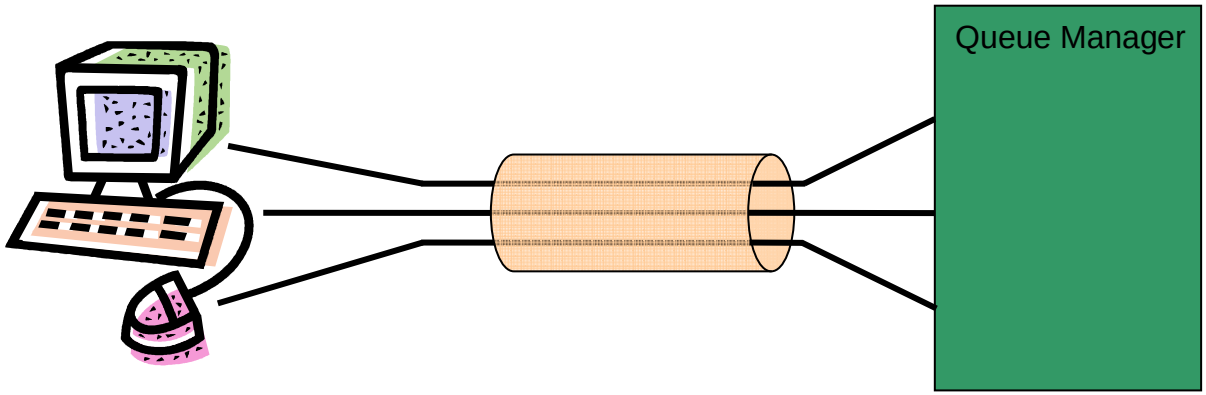
Some graphs…



- Source WebSphere MQ JMS v7 Performance Evaluation ftp://public.dhe.ibm.com/software/integration/support/supportpacs/individual/mp07.pdf

Connection Changes

- In WMQ v6 every single JMS Connection and JMS Session would need to have a separate socket back to the QM
- These were not duplex connections – all instigated by the client
- WMQ v7 JMS can
  - Multiplex multiple connections down the same socket
  - Detect failures much quicker due to fully duplex communications



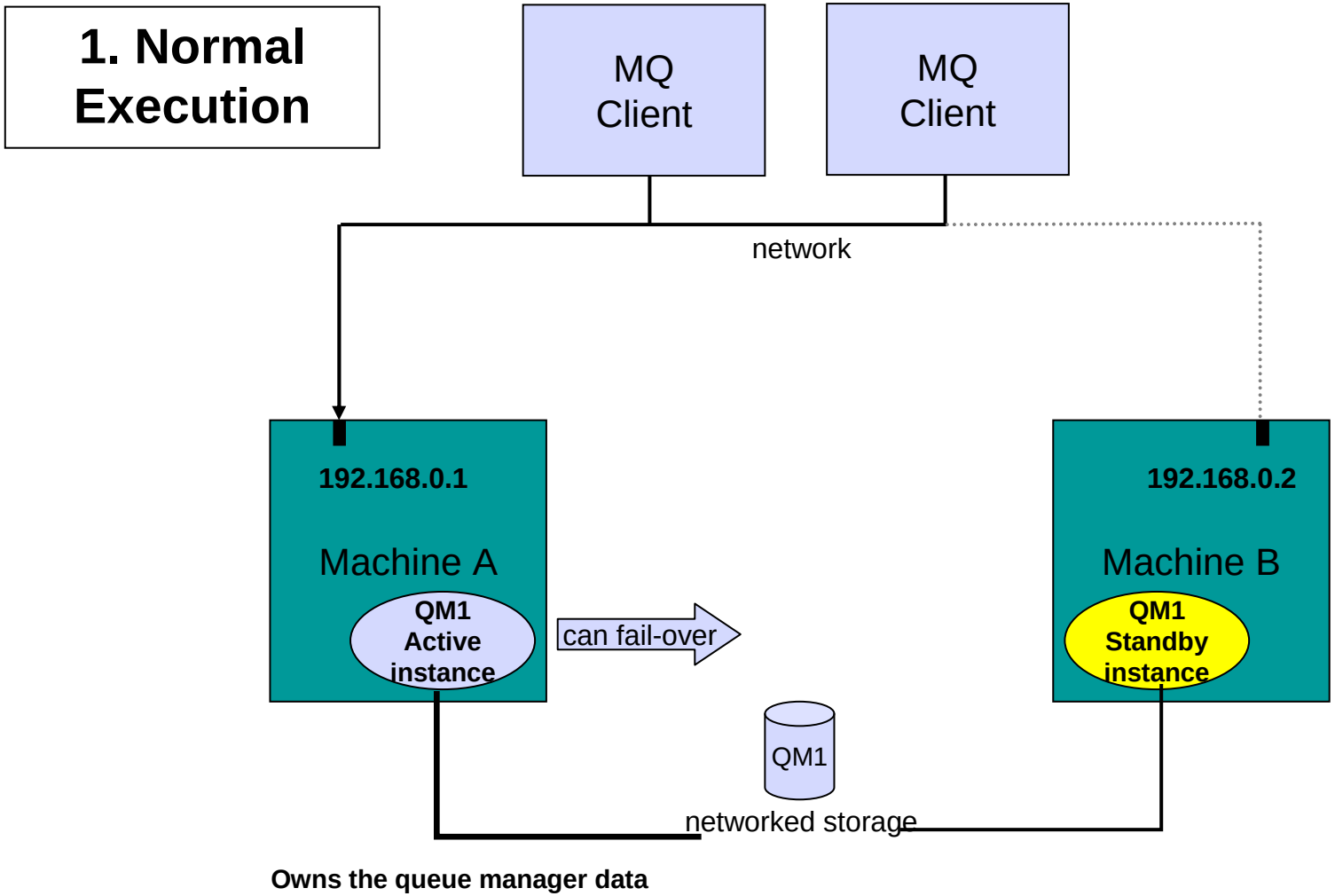
Distributed Platforms: Multi-instance Queue Managers

- Basic failover support without HA coordinator
  - Faster takeover: fewer moving parts
  - Cheaper: no specialised software or administration skills needed
  - Windows, Unix, Linux platforms

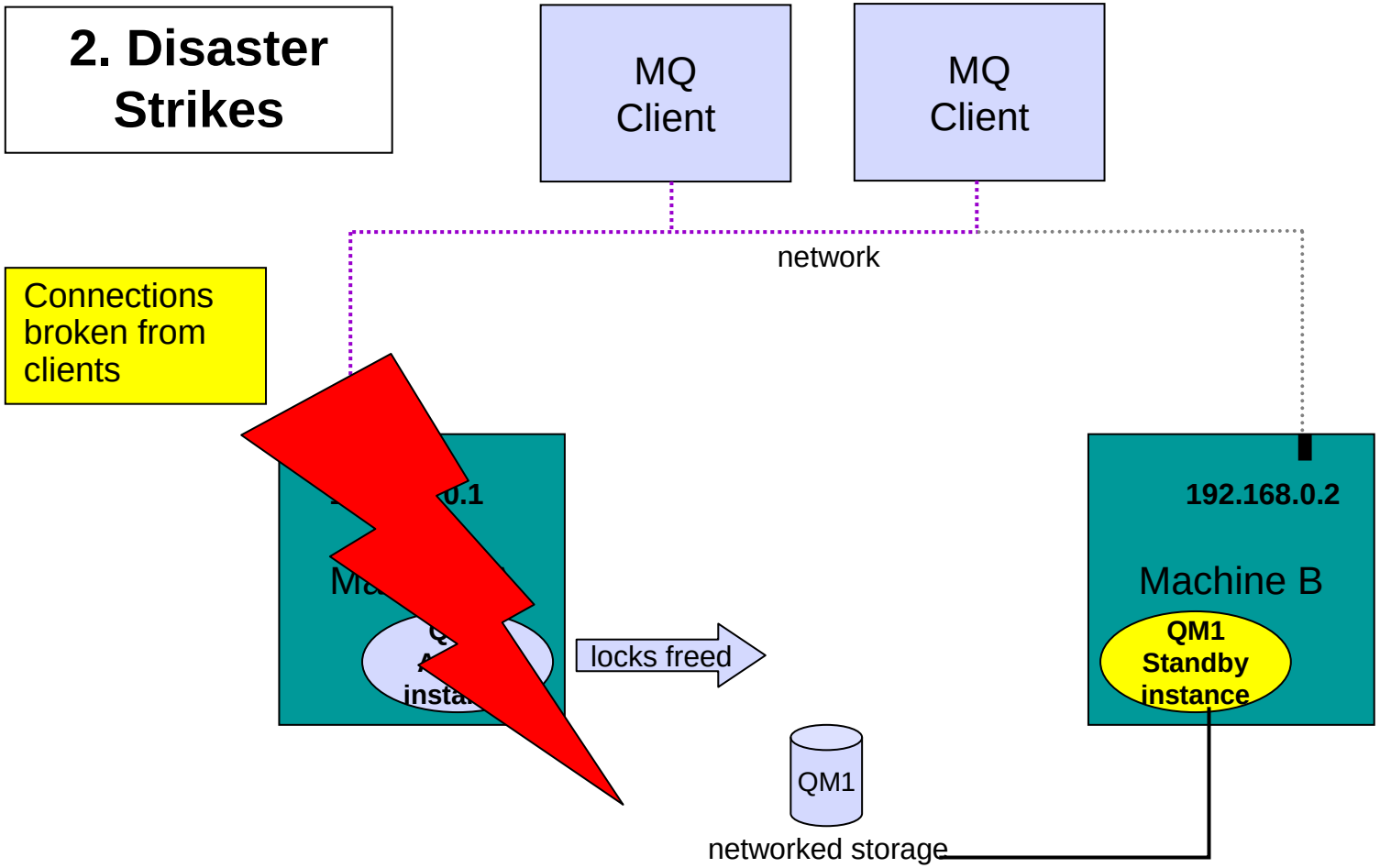
- Queue manager data is held in networked storage
  - NAS, NFS, GPFS etc so more than one machine sees the queue manager data
  - Improves storage management options: formal support for these even without failover config
- Multiple (2) instances of a queue manager on different machines
  - One is “active” 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 Queue Managers

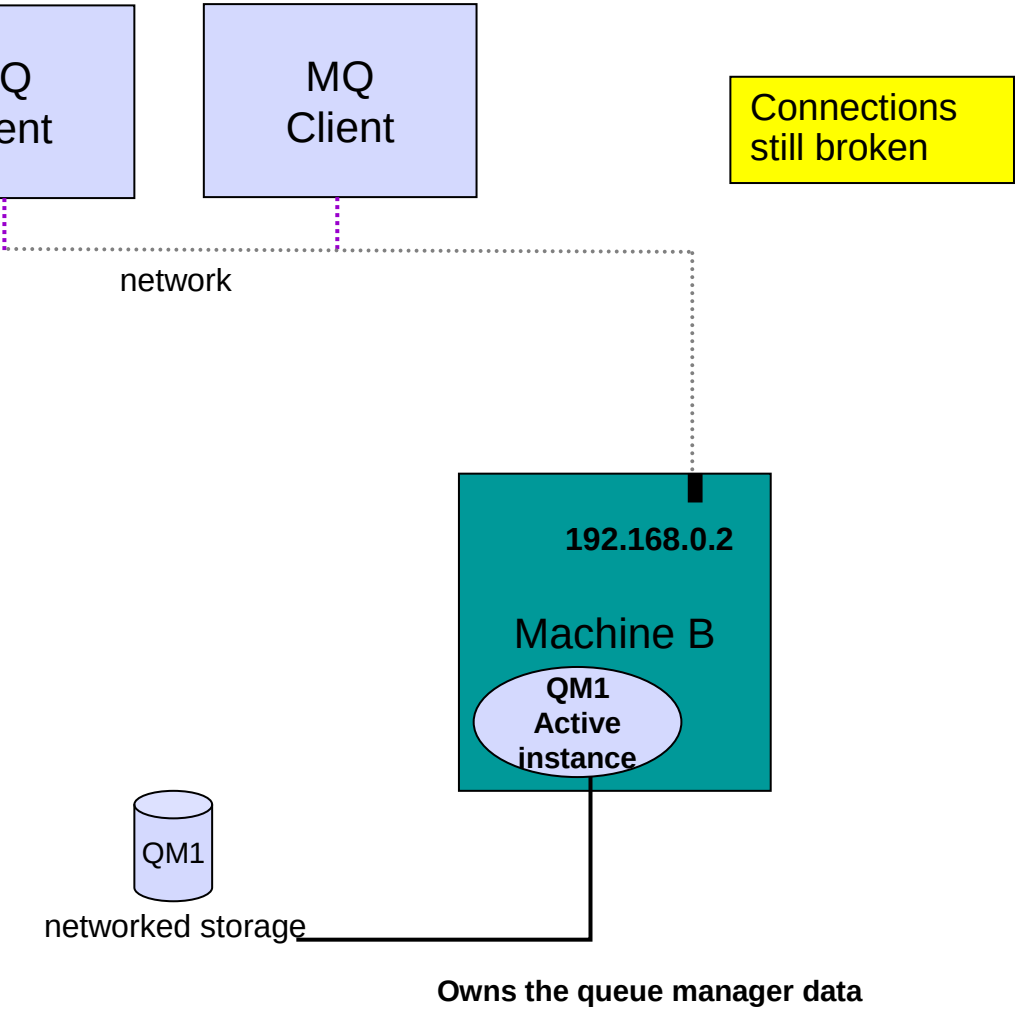


Multi-instance Queue Managers



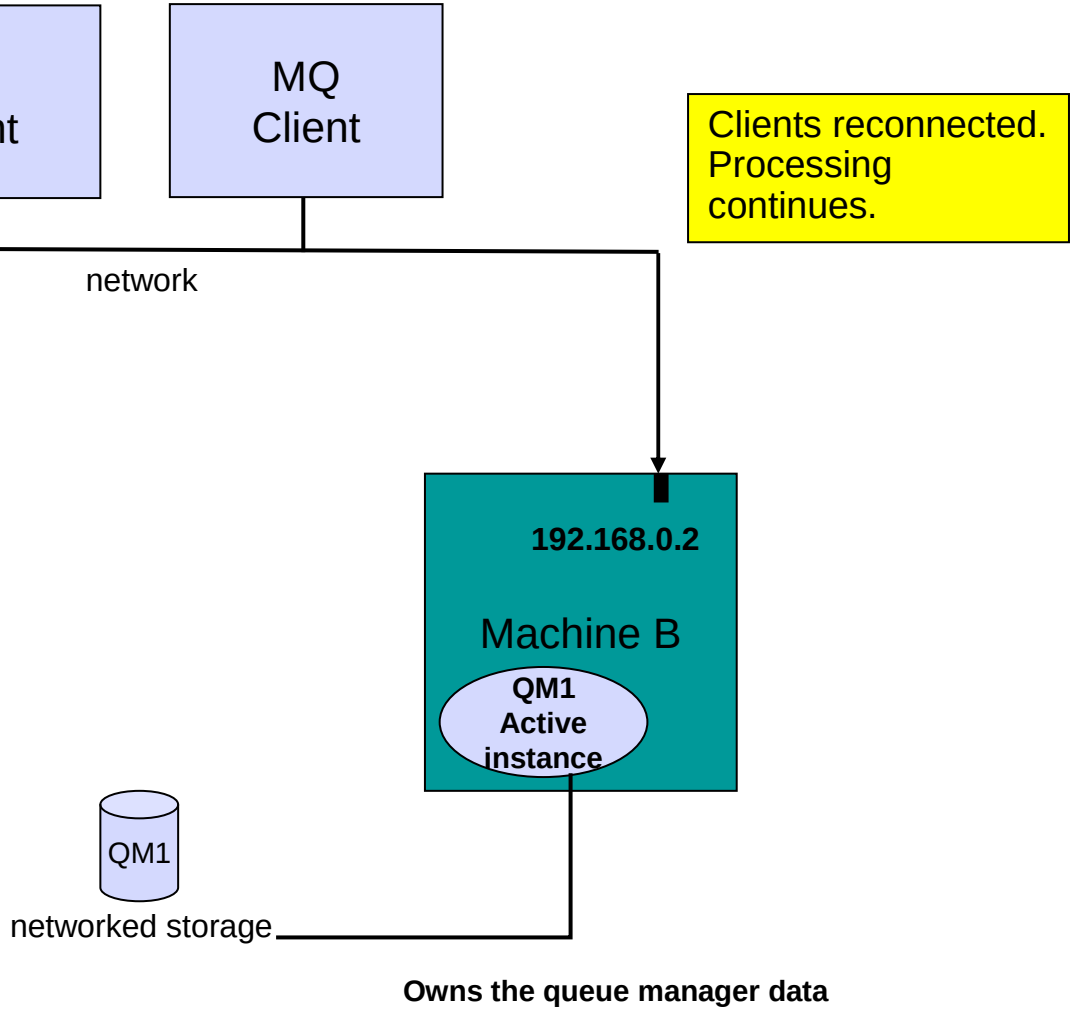
Multi-instance Queue Managers

3. Standby Comes to Life



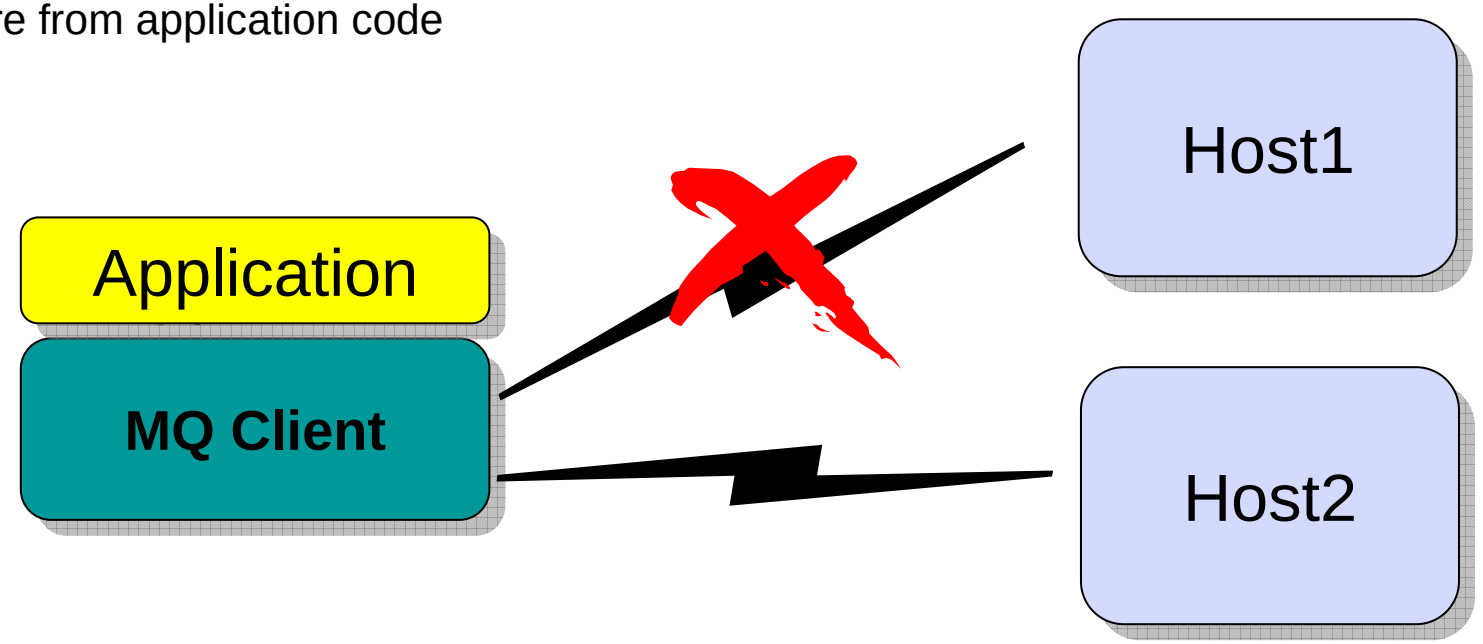
Multi-instance Queue Managers

4. Recovery Complete



Automatic Client Reconnection

- Client library provides necessary reconnection logic on detection of a failure
- Hides failure from application code



Within an Application Server

- The Resource Adapter implements the client connection reconnection logic
- Together with the application servers pooling; the RA therefore does not support Automatic Client Reconnection

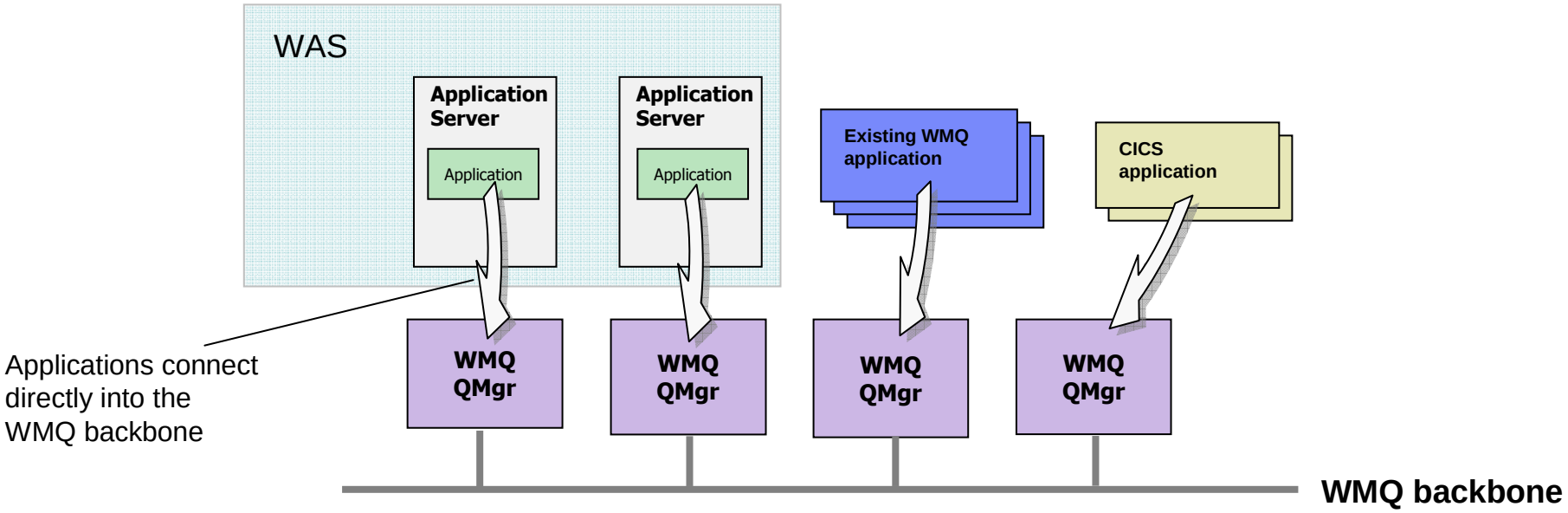
- Recommend use of WMQ 7.0.1.3/WAS7.0.0.13 (October 25th or PM19399)

- Configuration requires the use of ‘ConnectionNameList’ Connection Factory Property
- Within WAS configure using Customer Properties

    - `CONNECTIONNAMELIST=host1(1414),host2(1415)`

Performing the integration in practice

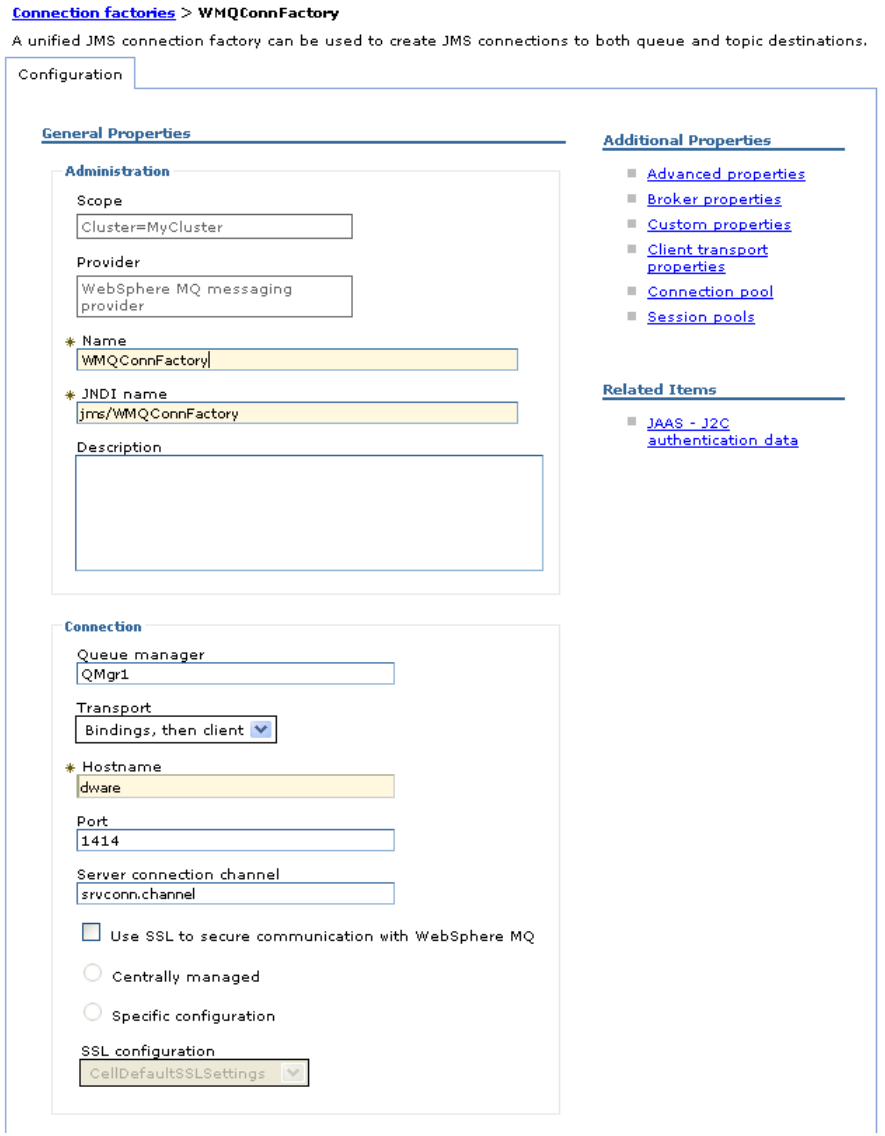
WebSphere MQ as WAS JMS Provider



WMQ Connection Factories

- 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
- WAS V7 allows the identification of a WMQ **client channel definition table** (CCDT) URL
  - This removes the need for most of the above configuration
  - Prior to V7, a CCDT can be specified through custom properties
- WAS V7 added the option to try ***'bindings, then client'*** as a transport type

Resources > JMS > Connection factories > [New]



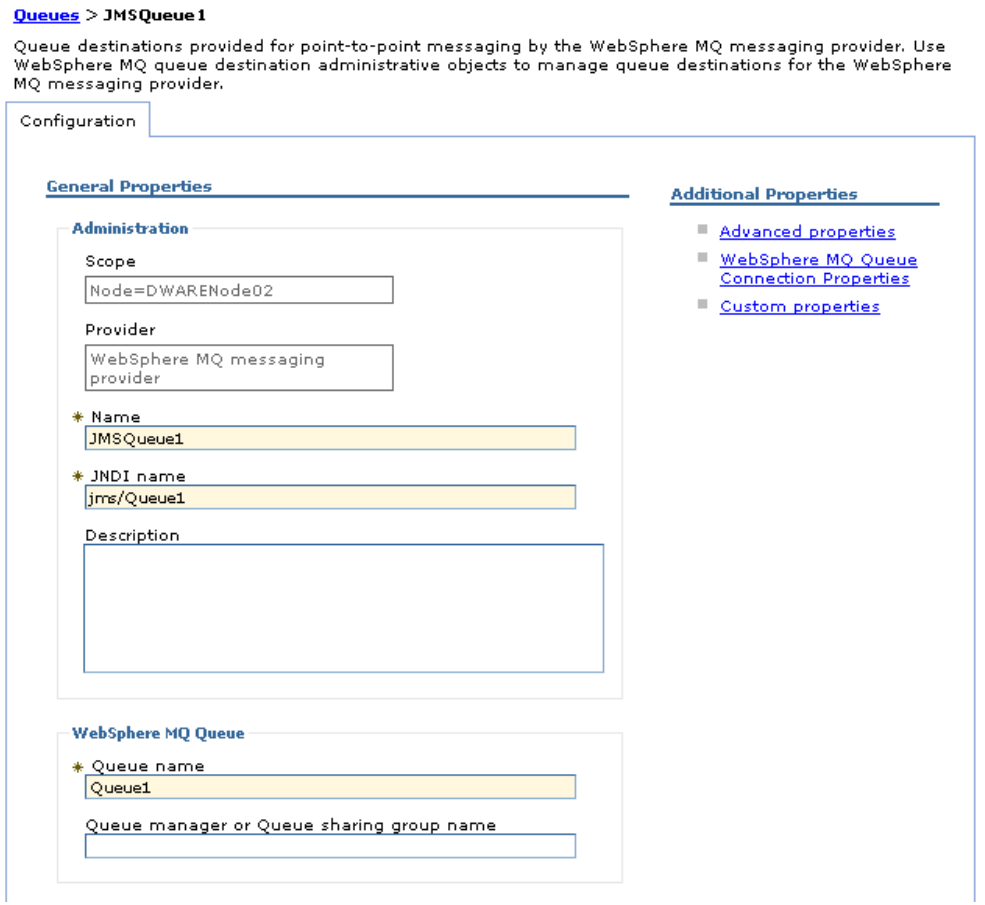
WMQ Destinations

- 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 properties
    - Can include details of a durable subscription
    - Requires knowledge of WMQ pubsub broker configuration

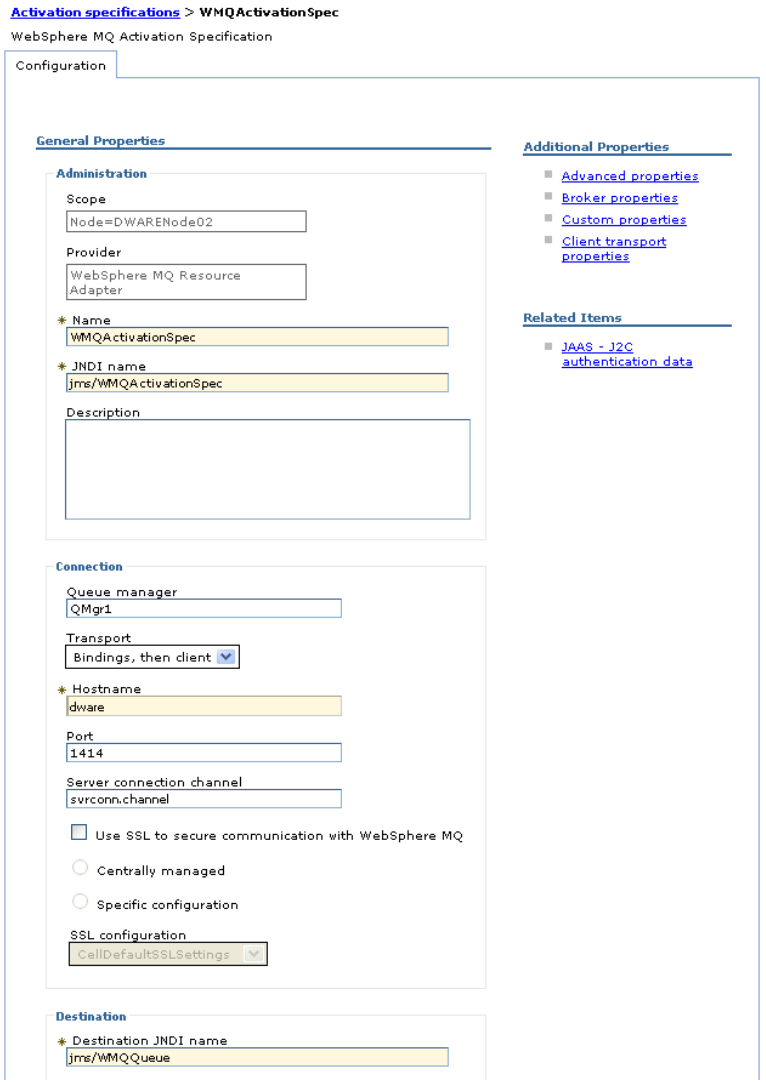
Resources > JMS > Queues / Topics > [New]



WMQ Activation Specifications

- **Message-driven beans** (**MDBs**) are a special class of EJB, they enable JEE applications to asynchronously process messages, with WAS managing the transactionality and concurrency of the application.
- Activation specs are the standardised way of defining and activating an MDB
- The WebSphere MQ messaging provider in **WAS V7** adds support for activation specs
- Activation specs combine the configuration of connectivity, the JMS destination to be processed and the runtime characteristics of the MDB itself
- Activation specs can be defined at all WAS configuration scopes, as can be done for ConnectionFactories and Destinations.

Resources > JMS > Activation Specifications > [New]



Listener Ports

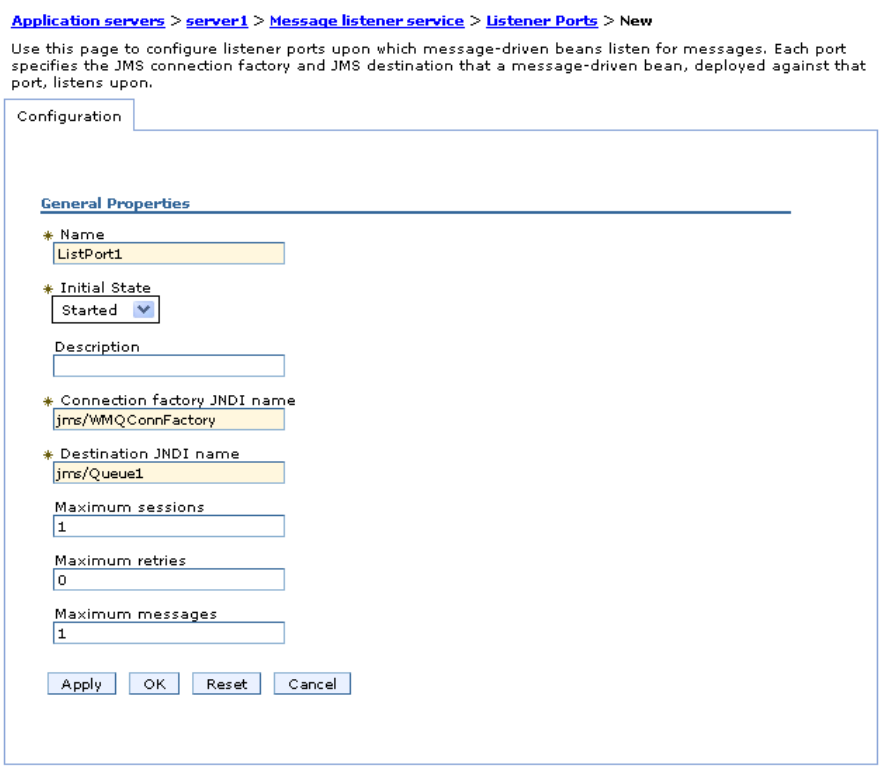
Servers > Server Types > WebSphere application servers > *serverName* > Message listener service > Listener ports > [New]

- Prior to V7, activation specifications are not available when using WMQ. Instead, MDBs are enabled by an application server's **message listener service** and its associated **listener ports**

- Just like an activation spec, a **listener port** is bound to the JMS destination that an MDB wishes to receive messages from
  - It also identifies a ConnectionFactory to use for connecting to WMQ

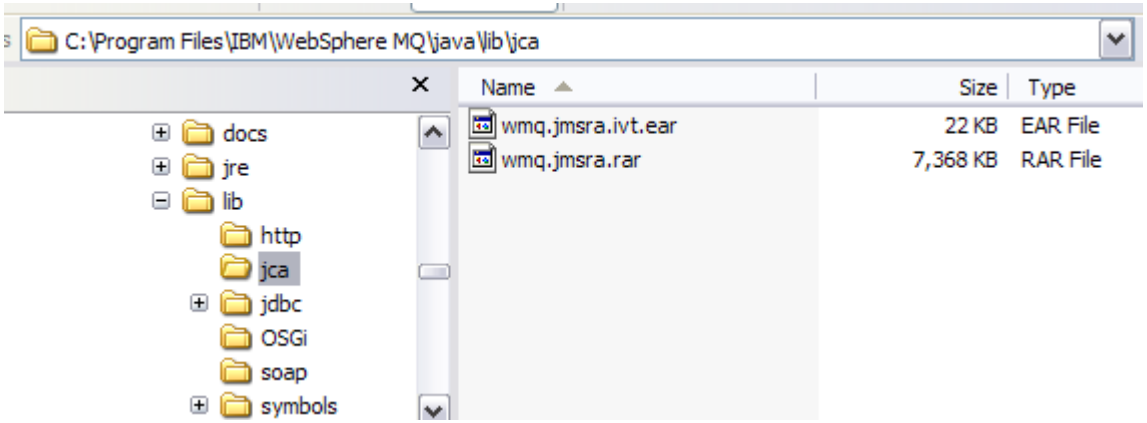
- Listener ports can only be defined at server scope.

- WAS V7 provides a wizard to migrate listener ports to activation specs



WMQ and Java EE

- WebSphere MQ V6.0.2.1 and later includes a J2EE Connector Architecture (JCA) V1.5 resource adapter.
- Can be used in application servers that are certified as having passed the J2EE 1.4 (or later) Compatibility Test Suite to access the resources of a WebSphere MQ queue manager
- WMQ RA is supplied with the WMQ JMS Client
  - Copy is available with either the normal WMQ install or via SupportPac MQC7



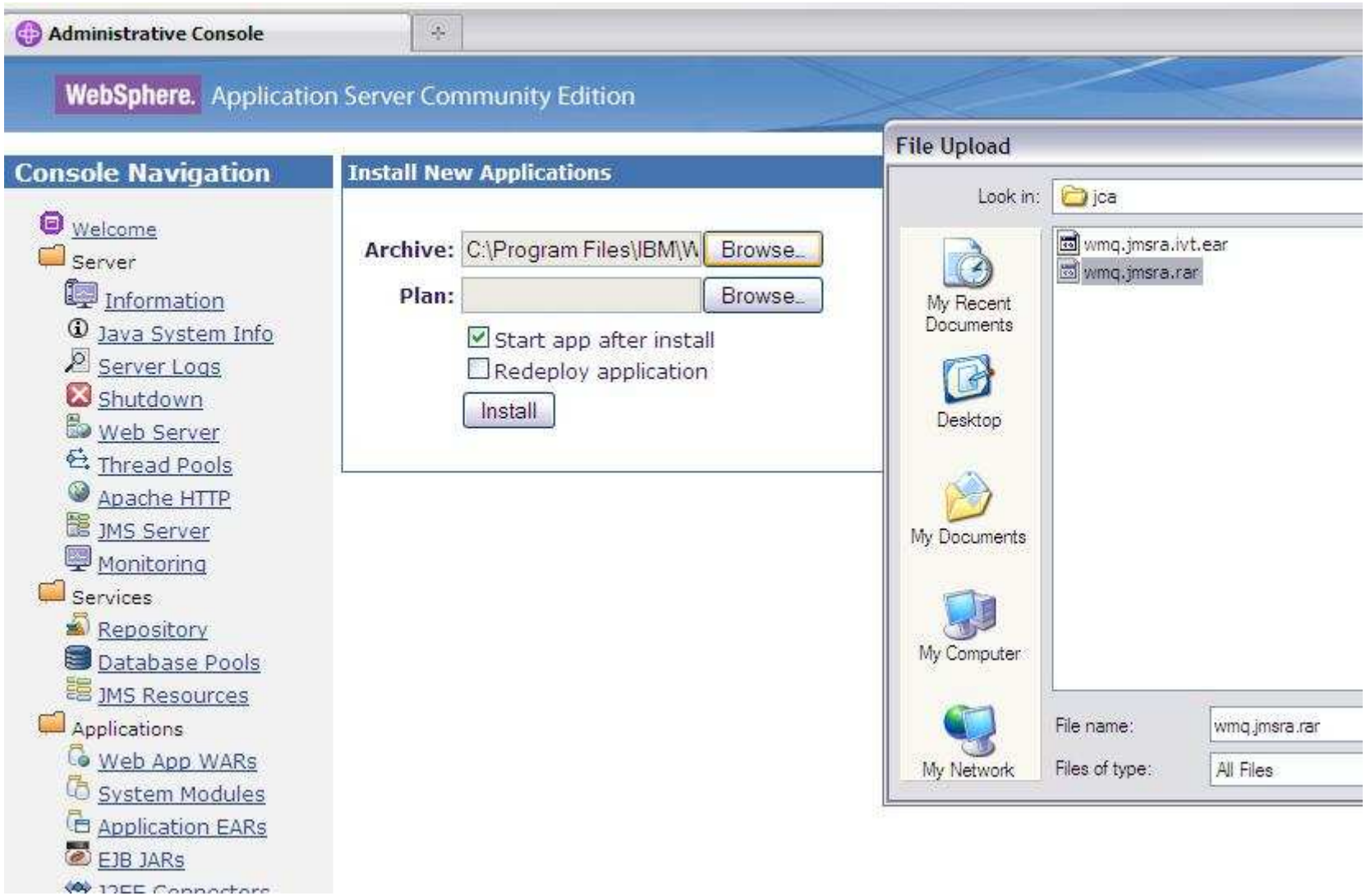
Considerations

- Transactional Client
  - If you wish to use XA Transactions over Client (TCP/IP) connection need Transactional Client License
- All other functionality is fully featured

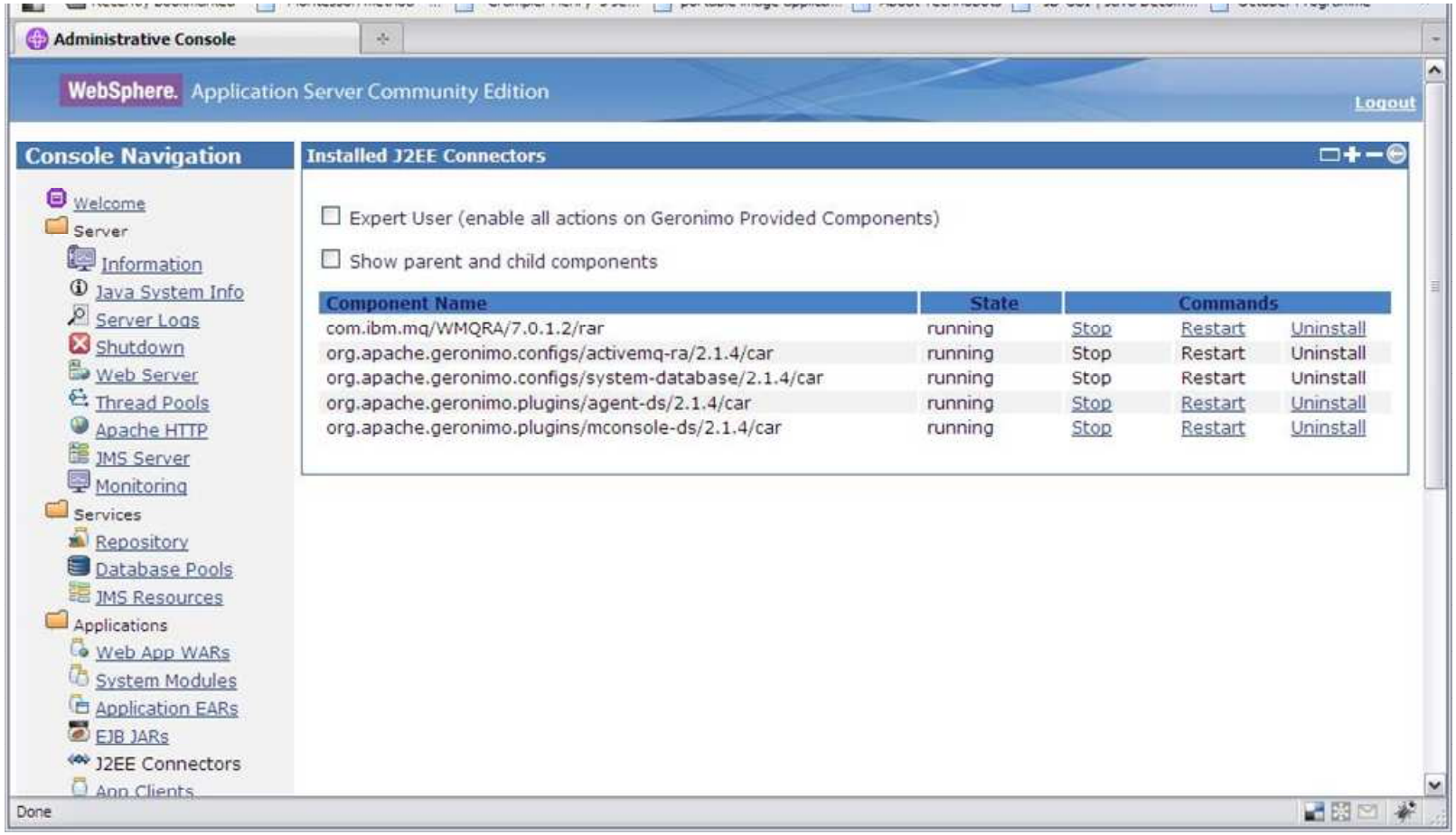
- Support Position
  - Need JCA 1.4
  - Need to confirm IVT installation and execution

- Tracing
  - JCA Tracing interface is limited
  - Recommend that use ‘logWriterEnabled=false’ to RA sends trace to local disk
  - But turn on trace, trace goes via JMS Client mechanisms
- *http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg27009524*

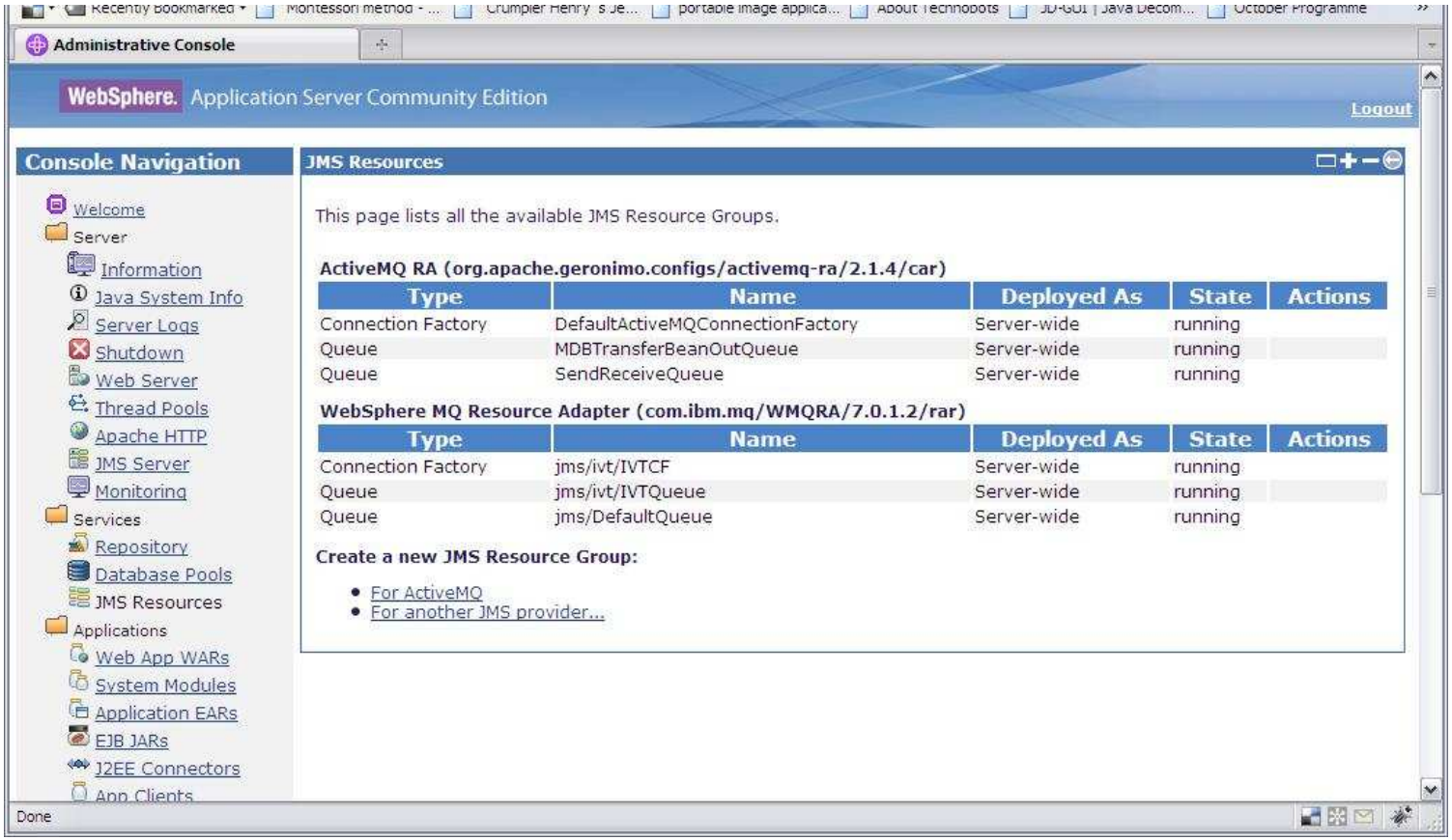
WAS CE – RA Installation



WebSphere Application Server Community Edition



WebSphere Application Server Community Edition



Questions….?

- Contact Information
  - Matthew White
  - whitemat@uk.ibm.com

Further Information

- WebSphere MQ Resource Adapter Support Statement
  - http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg27009524
- When to use ASF and non-ASF modes
  - http://www.ibm.com/developerworks/websphere/library/techarticles/0611_titheridge/0611_titheridge.html
- Evaluating Open Source Application Servers
  - https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=sw-app&S_PKG=ossred&S_TACT=109J63DW&S_CMP=web_ibm_ws_appinfra_bot_wascefeat

Further Information

- WAS product information
  - http://www-306.ibm.com/software/webservers/appserv/was/
- WAS Info Center
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp
- Using WebSphere MQ Java Interfaces in J2EE/JEE Environments
  - http://www-01.ibm.com/support/docview.wss?rs=171&context=SSFKSJ&dc=DB520&uid=swg21266535&loc=en_US&cs=UTF-8&lang=en&rss=ct171websphere
- 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
- 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 - WASV7
    - 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 - Security

WMQ and WAS Default Provider ‘Side by side’

Service integration bus	WebSphere MQ
Fully Integrated with WAS runtime, configuration and console	Own runtime, admin domain and explorer interface
Included with WAS (c. 2005) growing customer base limited tooling support	Separate product (c. 1993) huge install base widespread tooling support
Underpins WESB, WPS, …	Underpins WMB, FTE, …
Provides point-point & publish-subscribe messaging models	
Supports JMS	Supports multiple programming languages, including JMS
Optimised for WAS-based messaging performance	Optimised for heterogeneous messaging performance