Getting WMQ messages into WebSphere Application Server (including from WebSphere Message Broker)

Session 8714

Simon Gormley – WebSphere MQ JMS Level 3 Service,
IBM Hursley, sgormley@uk.ibm.com

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

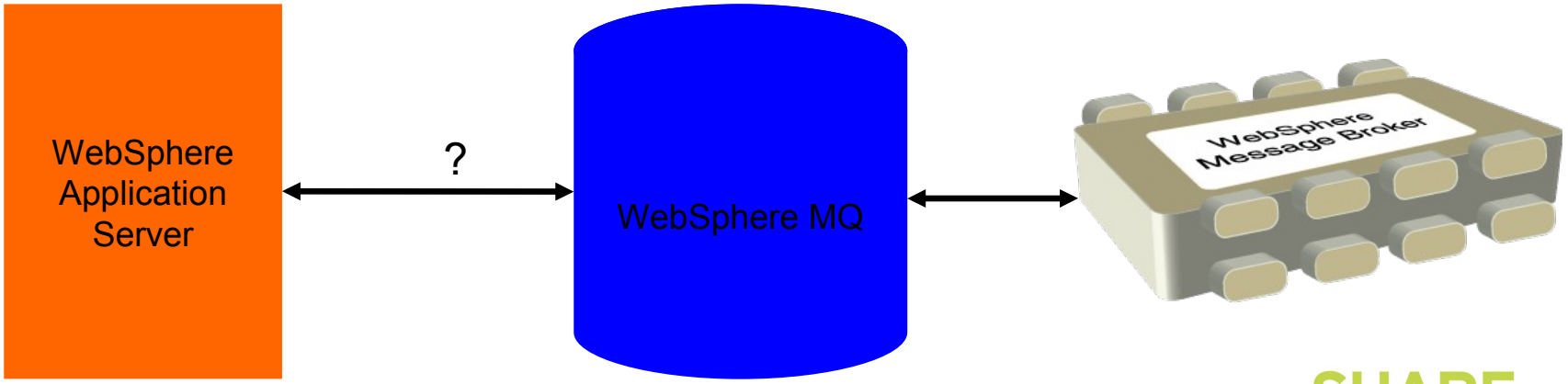
Thursday 3rd March 2011

Agenda

- **The Scenario**
- Initial Setup
  - Queue Manager
  - Connection Factories
  - Destinations
  - Activation Specifications
- Application Development
  - Message-driven beans (MDBs)
  - Enterprise Java Beans (EJBs)
- Application Deployment
- JMS in WebSphere Message Broker

The Scenario

- Imagine your enterprise is an existing WebSphere MQ user or sends messages across other transports.
- Maybe WebSphere Message Broker is pulling in data from another source
- Another part of the organisation has started using WebSphere Application Server, and now wants to use the data that already exists in other applications.
- How do you do this?



The Scenario

- WebSphere Application Server is a fully compliant Java Enterprise Edition (JEE) Application Server.
  - Provides integrated support to connect to Java Message Service (JMS) providers.
- WebSphere MQ is a fully compliant JMS provider.
- WebSphere Message Broker can route messages across any JMS 1.1 compliant provider
- Therefore, JMS is the answer!

Agenda

- The Scenario
- **Initial Setup**
  - **Queue Manager**
  - Connection Factories
  - Destinations
  - Activation Specifications
- Application Development
  - Message-driven beans (MDBs)
  - Enterprise Java Beans (EJBs)
- Application Deployment
- JMS in WebSphere Message Broker

Initial Setup
Queue Manager

- The good news is that there isn't much setup required on the queue manager.

- If the application server is on a different machine to the queue manager, setup as for other client applications :
  - Start a listener
  - Create a SVRCONN channel.

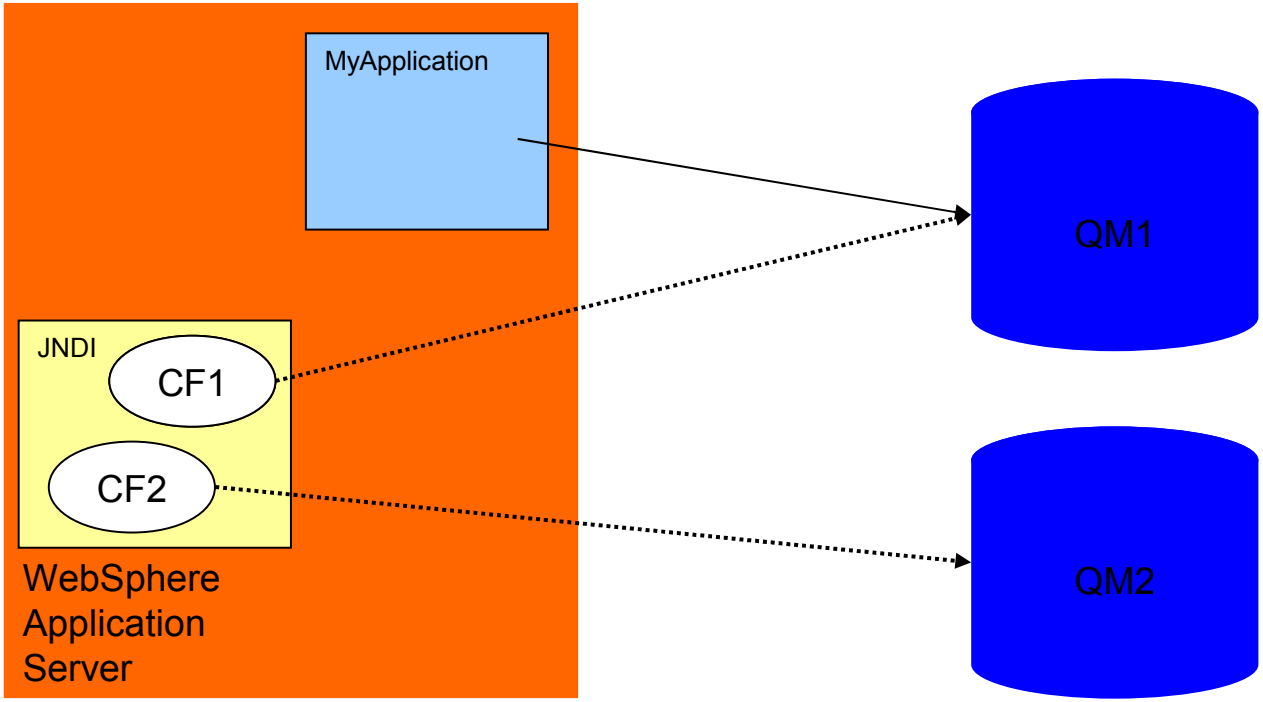
- If the application server is on the same machine as WMQ, you don't need to do anything!

Agenda

- The Scenario
- **Initial Setup**
  - Queue Manager
  - **Connection Factories**
  - Destinations
  - Activation Specifications
- Application Development
  - Message-driven beans (MDBs)
  - Enterprise Java Beans (EJBs)
- Application Deployment
- JMS in WebSphere Message Broker

Initial Setup
Connection Factories

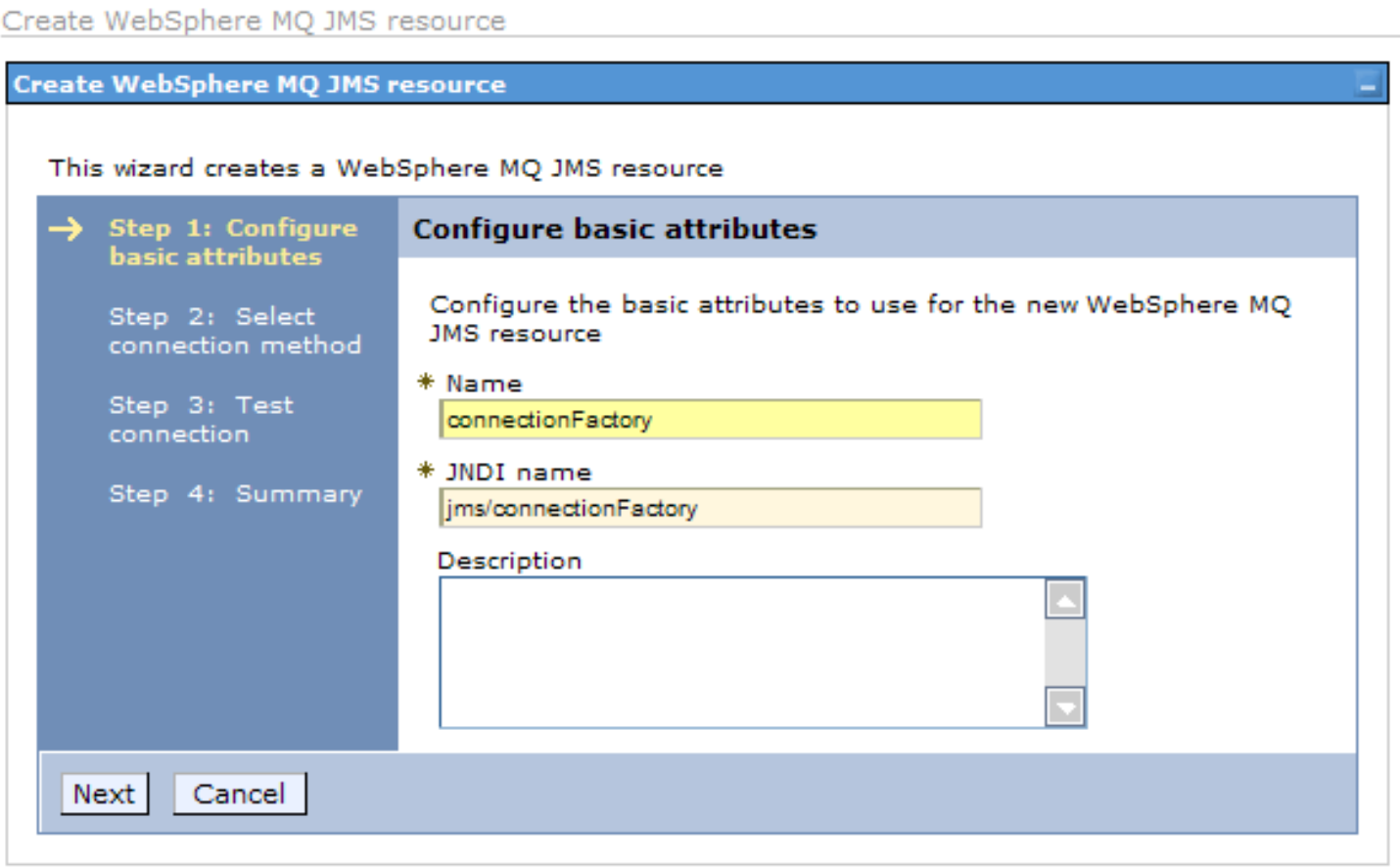
- Contains information about how to connect to a queue manager.



Initial Setup
Connection Factories

- Need to be defined using WebSphere Administrative Console.
- A handy wizard helps through the creation process.
  - Also provides the ability to verify the Connection Factory has been defined correctly, by trying to connect to the specified queue manager.

Initial Setup
Connection Factories



Initial Setup
Connection Factories

- WebSphere Application Server supports three types:
  - Queue Connection Factory
    - Used by applications that are going to be sending and receiving messages to and from queues (point-to-point messaging).
  - Topic Connection Factory
    - Used by publish/subscribe applications.
  - Unified Connection Factory
    - Can be used by either point-to-point or publish/subscribe applications.

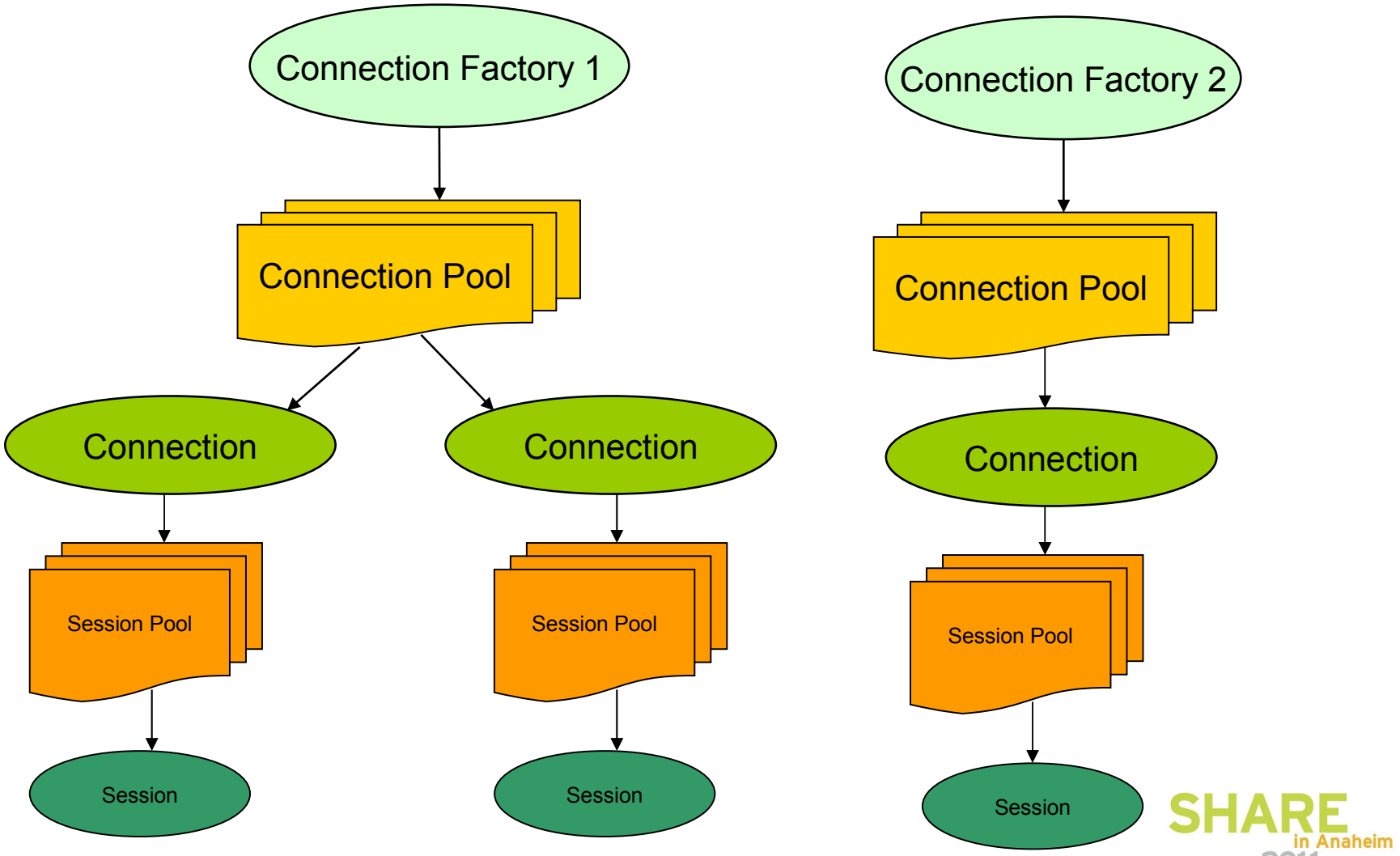
Initial Setup
Connection Factories

- Important properties:
  - Queue Manager Name
  - Transport Type
    - BINDINGS
      - *Used when WebSphere Application Server is on the same machine as the queue manager.*
    - CLIENT
      - *Used when WebSphere Application Server is on a different machine to the queue manager.*
    - BINDINGS_THEN_CLIENT
      - *Special option useful when you are not sure if the application is running on the same machine as the queue manager or not.*
  - Hostname
  - Server Channel Name

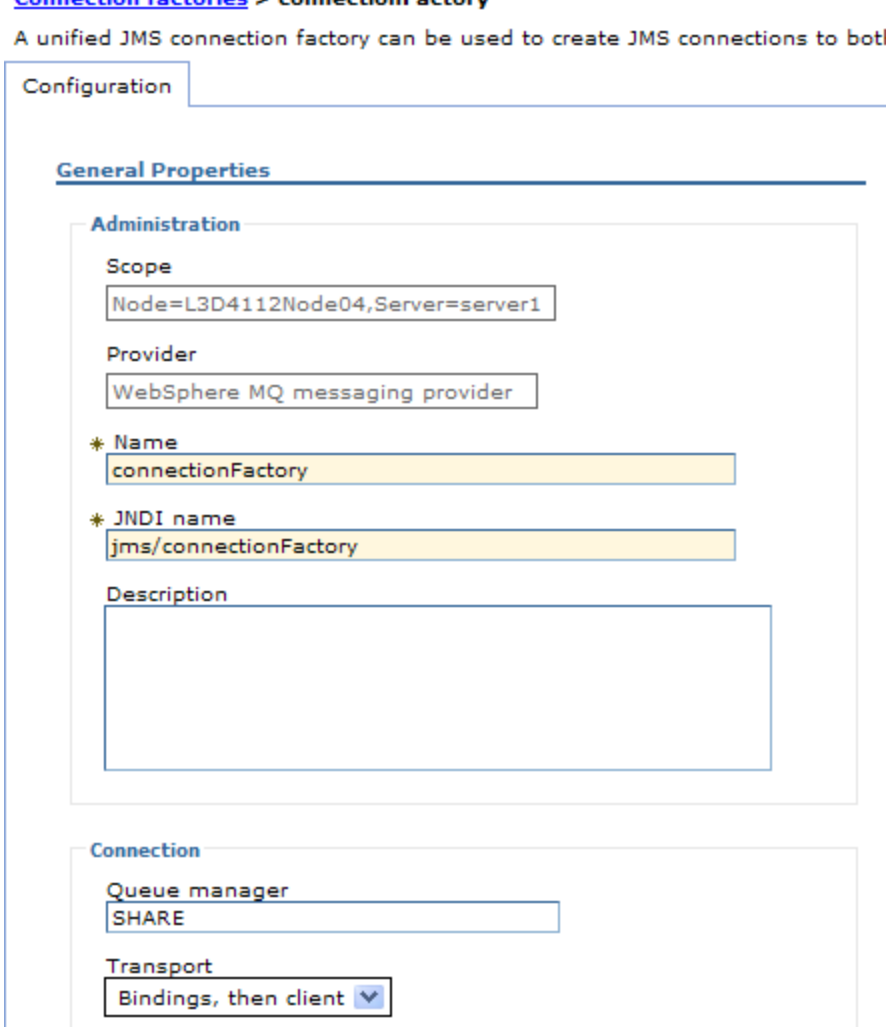
Initial Setup
Connection Factories

- WebSphere Application Server also provides Connection Pooling.
  - One Connection Pool per Connection Factory.
  - Used when an application creates a JMS Connection from a Connection Factory.
  - By default, only 10 connections can be created to a queue manager from a given Connection Factory.

Initial Setup
Connection Factories



Initial Setup
Connection Factories

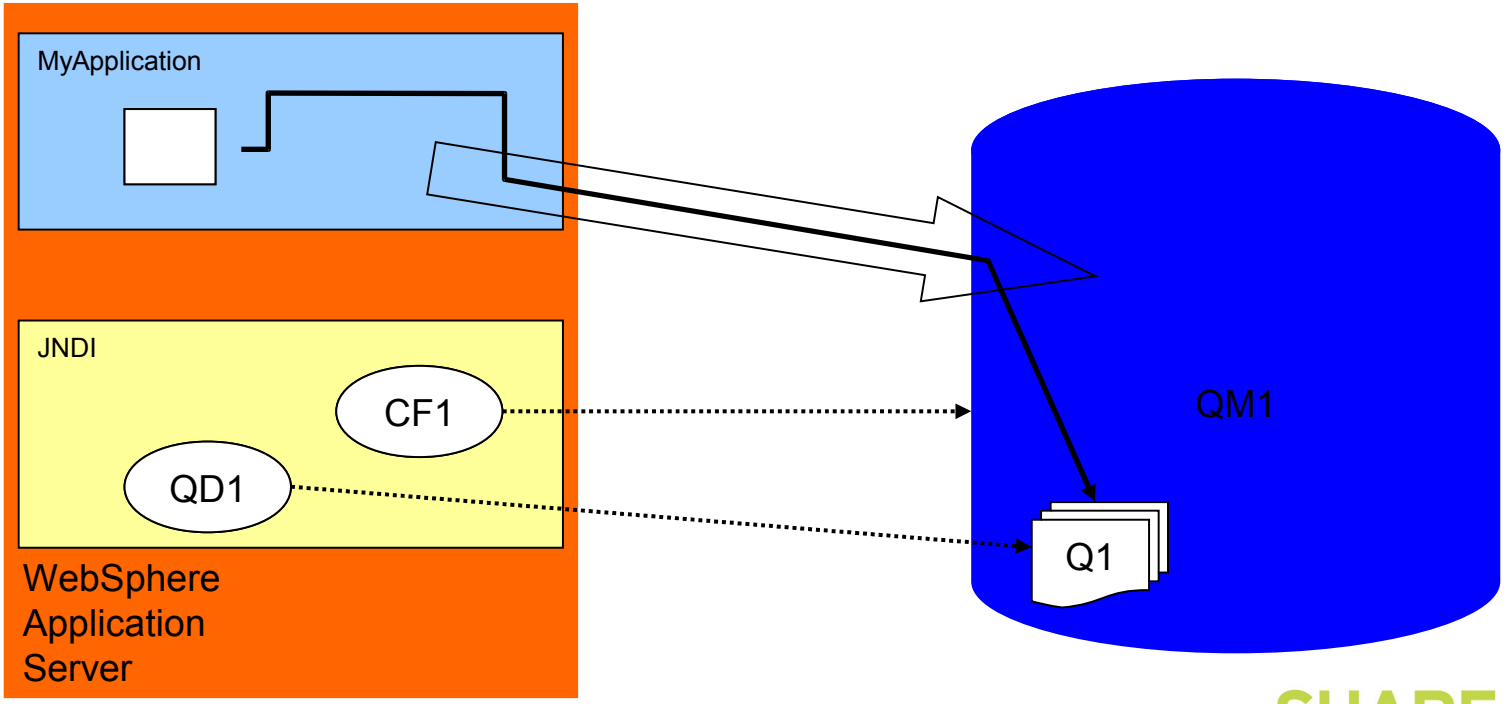


Agenda

- The Scenario
- **Initial Setup**
  - Queue Manager
  - Connection Factories
  - **Destinations**
  - Activation Specifications
- Application Development
  - Message-driven beans (MDBs)
  - Enterprise Java Beans (EJBs)
- Application Deployment
- JMS in WebSphere Message Broker

Initial Setup Destinations

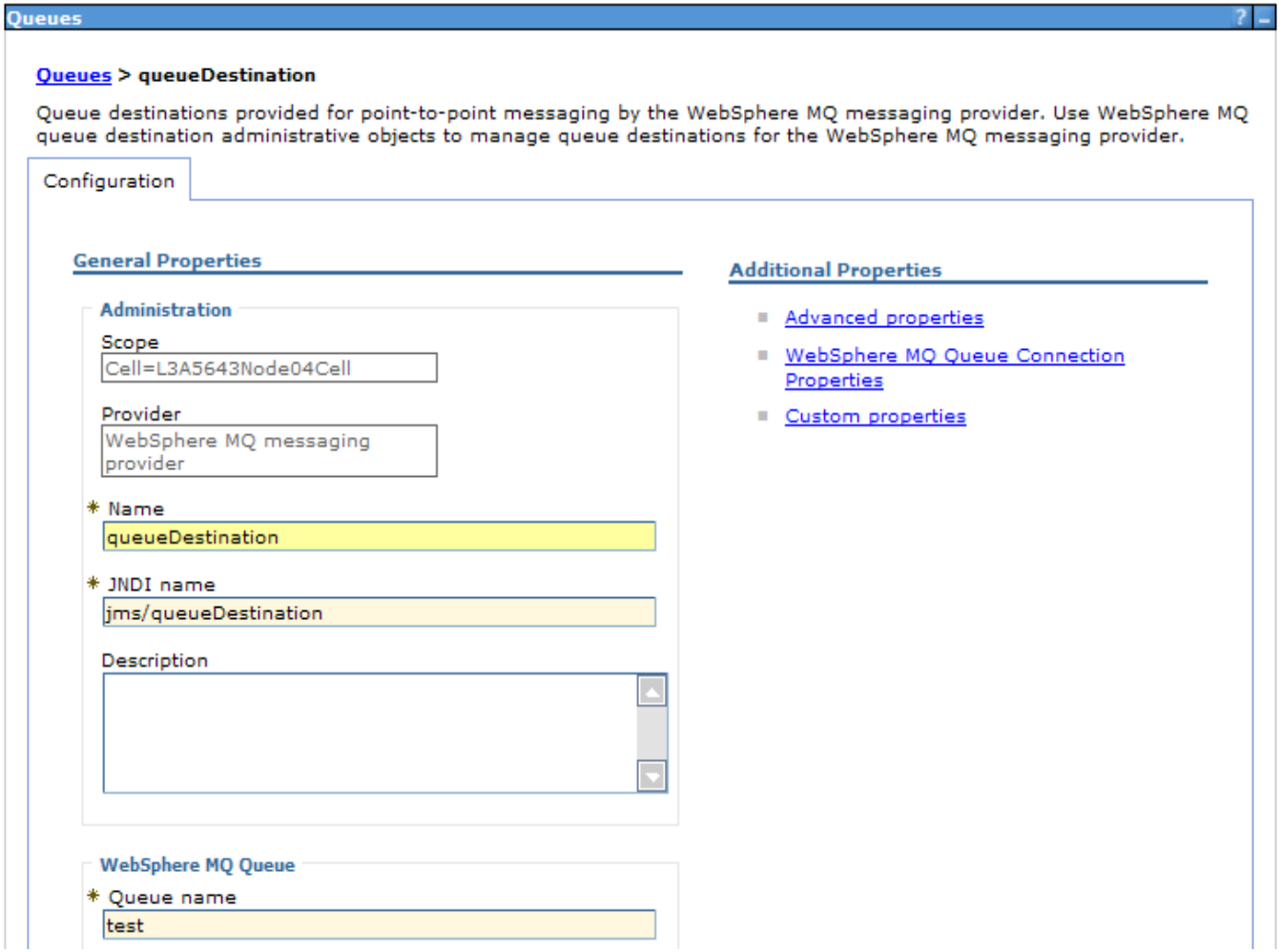
- JMS Destinations map to either queues or topics.



Initial Setup Destinations

- Need to be defined using WebSphere Administrative Console.
- Should have one Destination definition for every queue or topic used by applications running inside of the application server.
- Important properties:
  - Queue Name
  - Topic Name

Initial Setup Destinations



Agenda

- The Scenario
- **Initial Setup**
  - Queue Manager
  - Connection Factories
  - Destinations
  - **Activation Specifications**
- Application Development
  - Message-driven beans (MDBs)
  - Enterprise Java Beans (EJBs)
- Application Deployment
- JMS in WebSphere Message Broker

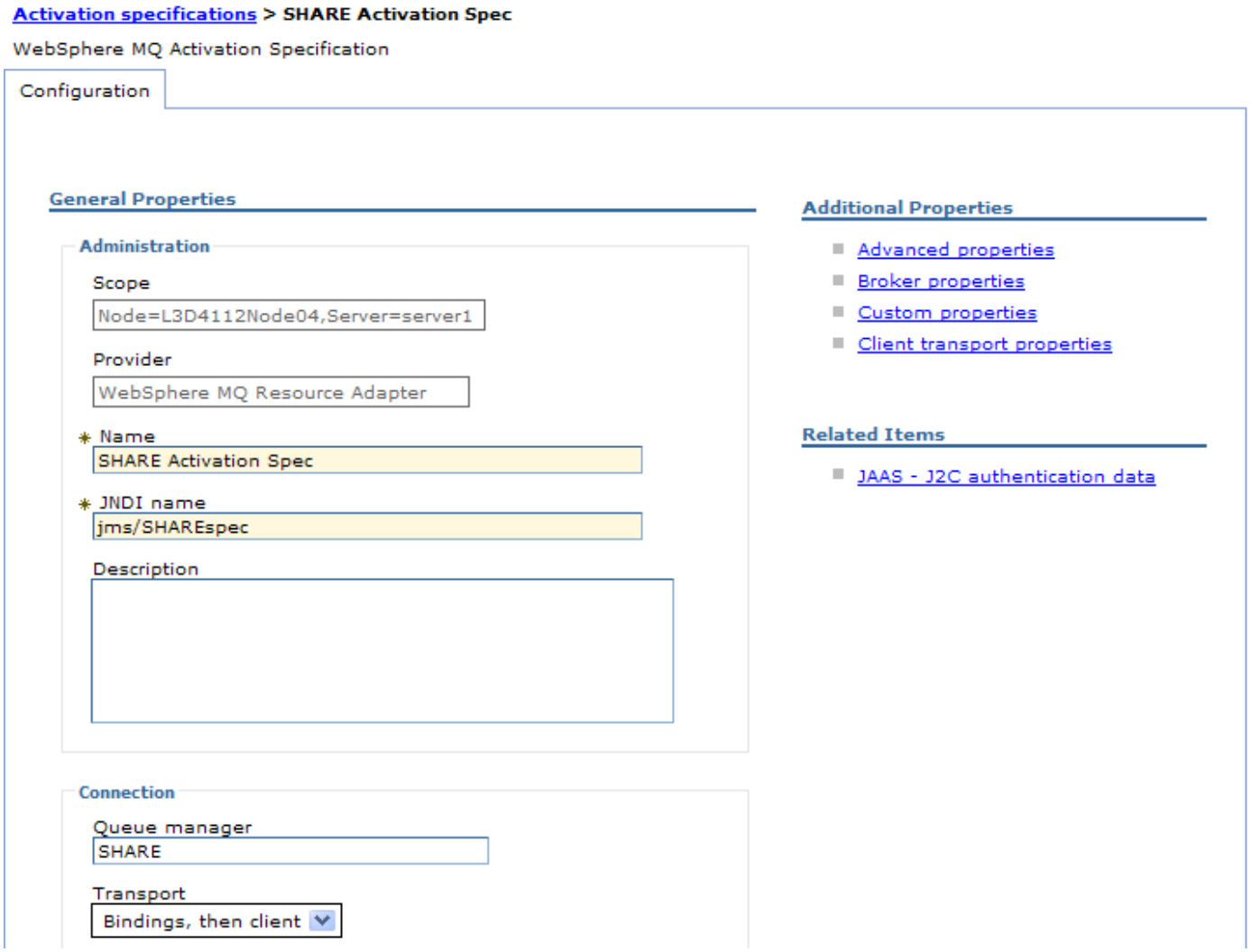
Initial Setup
Activation Specifications

- 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.
- Based on the J2EE Connector Architecture (JCA) 1.5 standard.
- Provides a common way for all JEE 1.4 compliant application servers to connect to JMS providers.

Initial Setup
Activation Specifications

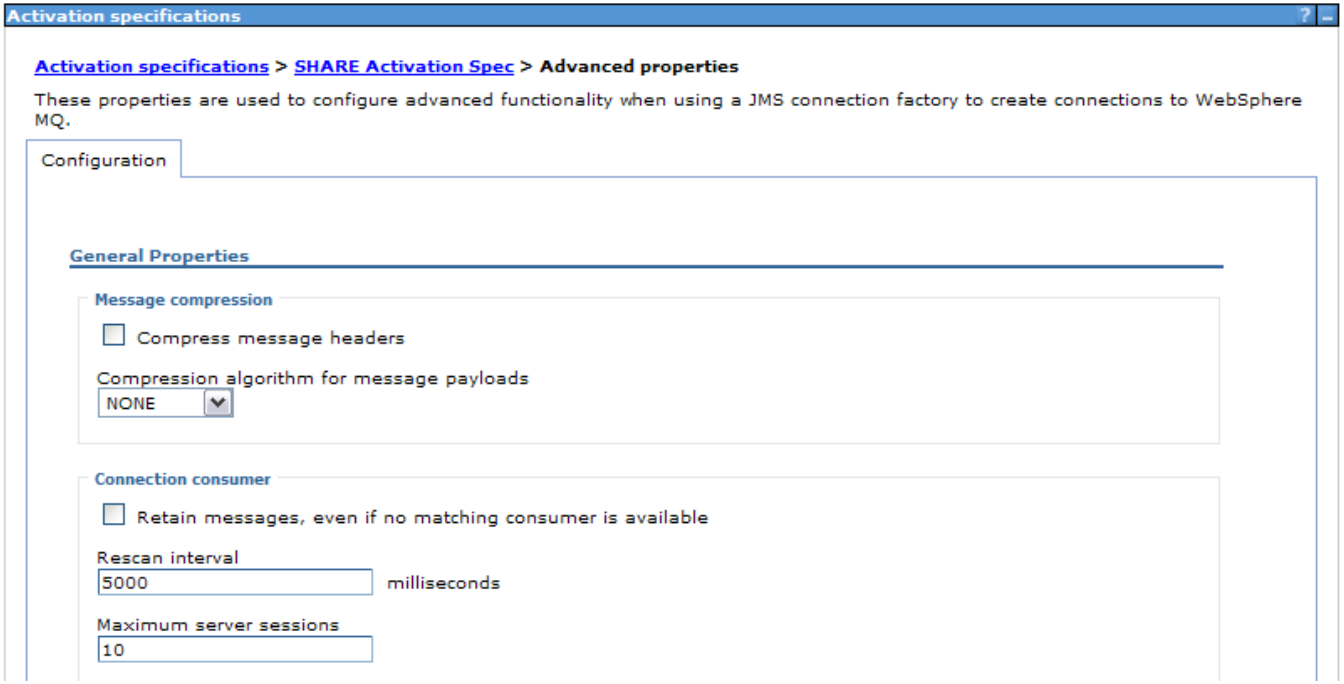
- To create an Activation Specification.
  - Specify the JMS Destination to listen on.
  - Enter details of the queue manager where the Destination resides.
  - Optionally, specify a JMS Message Selector.
    - SQL expression.
    - Only messages that match the Selector will be delivered to applications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

Initial Setup
Activation Specifications



Initial Setup
Activation Specifications

- By default, applications that use the Activation Specification will process 10 messages concurrently.
  - To change this, modify the Activation Specification Advanced Property Maximum server sessions.



Agenda

- The Scenario
- **Initial Setup**
  - Queue Manager
  - Connection Factories
  - Destinations
  - Activation Specifications
- **Application Development**
  - **Message-driven beans (MDBs)**
  - Enterprise Java Beans (EJBs)
- Application Deployment
- JMS in WebSphere Message Broker

Application Development

- JEE applications are known as Enterprise Applications, and are stored in Enterprise Application Archives (EARs).

- Can consist of multiple parts:
  - Message-driven beans
  - Enterprise Java Beans
  - Servlets
  - Static web pages

Application Development
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.

Application Development
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.

- IBM Rational tooling provides wizards for creating MDBs.

Application Development
Message-driven beans


```
public void onMessage(Message message) {
    try
    {
       System.out.println("In onMessage()");

       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);
    }
}
```

Agenda

- The Scenario
- **Initial Setup**
  - Queue Manager
  - Connection Factories
  - Destinations
  - Activation Specifications
- **Application Development**
  - Message-driven beans (MDBs)
  - **Enterprise Java Beans (EJBs)**
- Application Deployment
- JMS in WebSphere Message Broker

Application Development Enterprise Java Beans

- Java applications that run inside of WAS.

- EJBs need to create their own connections to WMQ and get (or send) messages themselves.
  - EJBs use the standard JMS API.
  - 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


```
public void receiveMessage()
{
   try
   {
      InitialContext ctx = new InitialContext();
      ConnectionFactory cf = (ConnectionFactory) ctx.lookup("connectionFactory");
      Connection conn = cf.createConnection();
      conn.start();
      Session sess = conn.createSession(true, Session.AUTO_ACKNOWLEDGE);

      Queue d = (Queue) ctx.lookup("queueDestination");
      MessageConsumer consumer = sess.createConsumer(d);
      int timeout = 30000;
      Message msg = consumer.receive(timeout);
      if (msg instanceof TextMessage)
      {
         TextMessage textMessage = (TextMessage) msg;
         System.out.println("Message received:" + textMessage.getText());
      }
      consumer.close();
      sess.close();
      conn.close();
   }
   catch (Exception ex)
   {
      System.out.println("Error getting the connection factory");
      System.out.println("Exception : " + ex);
   }
}
```

Agenda

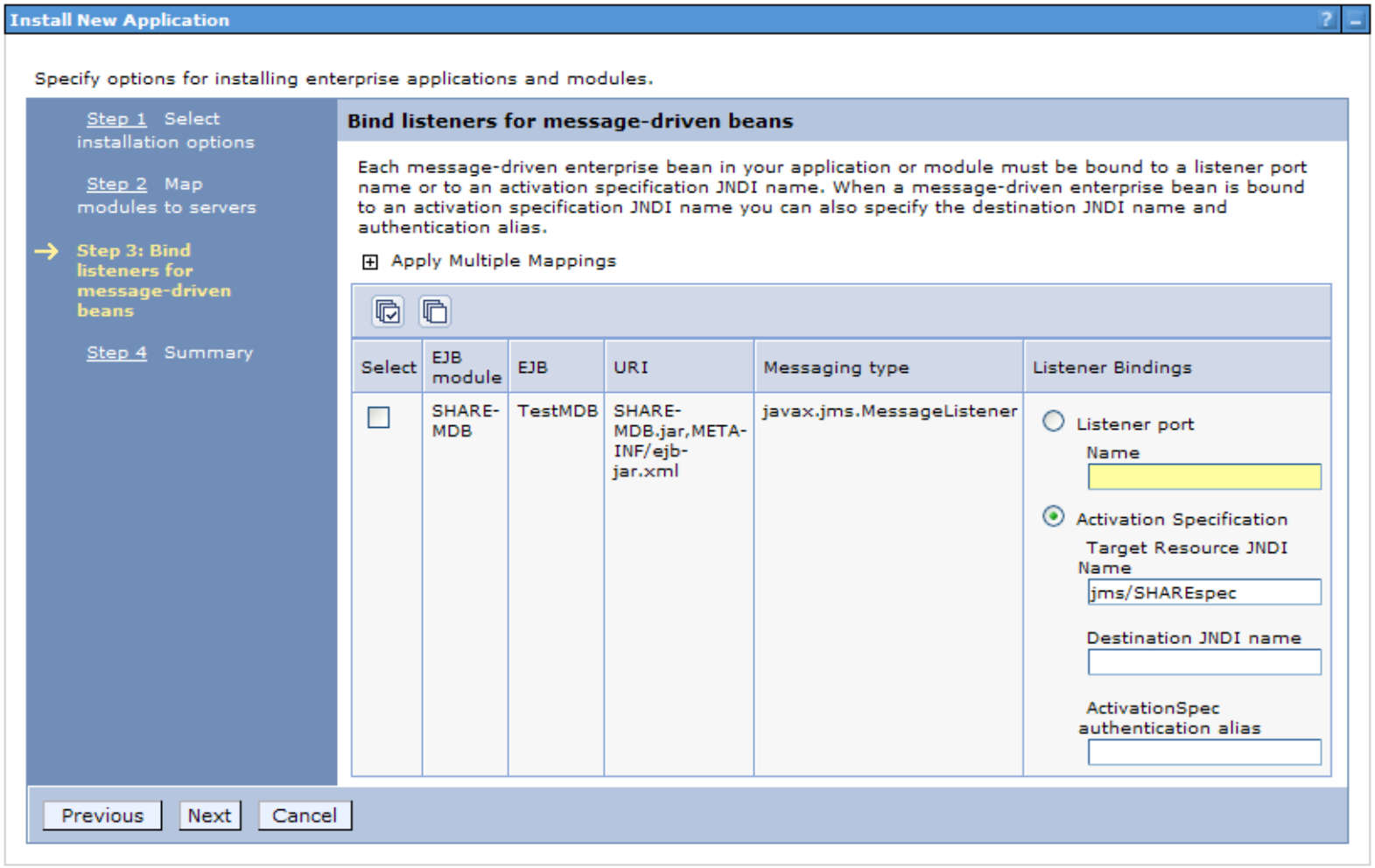
- The Scenario
- Initial Setup
  - Queue Manager
  - Connection Factories
  - Destinations
  - Activation Specifications
- Application Development
  - Message-driven beans (MDBs)
  - Enterprise Java Beans (EJBs)
- **Application Deployment**
- JMS in WebSphere Message Broker

Application Deployment

- The final step is to deploy the application into the application server.

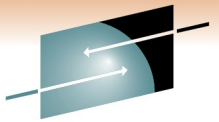
- As part of the message-driven bean deployment process, you need to tell the application which Activation Specification to use, which determines the queues and queue managers to monitor.

Application Deployment



Application Deployment

- If the application has been deployed against an Activation Specification, simply start the application to start processing messages.
- If deployed against a Listener Port, a full application server restart might be required!
- And that's it!



Getting WMQ messages into WebSphere Application Server (including from WebSphere Message Broker)

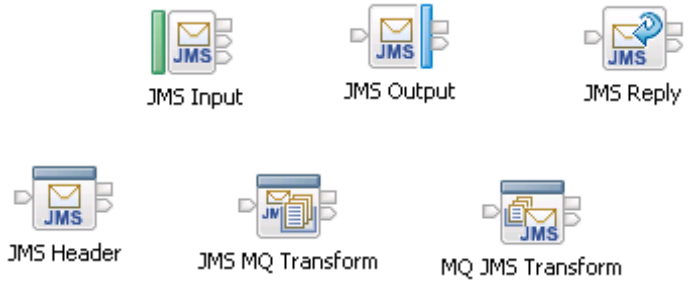
Simon Gormley – WebSphere MQ JMS Level 3 Service,
IBM Hursley, sgormley@uk.ibm.com

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

Thursday 3rd March 2011

JMS in Message Broker

- Support for any JMS 1.1 compliant provider not just MQ
- 6 JMS specific nodes for JMS
  - JMSInput
  - JMSOutput / JMSReply
  - JMSMQTransform
  - MQJMSTransform
  - JMSHeader



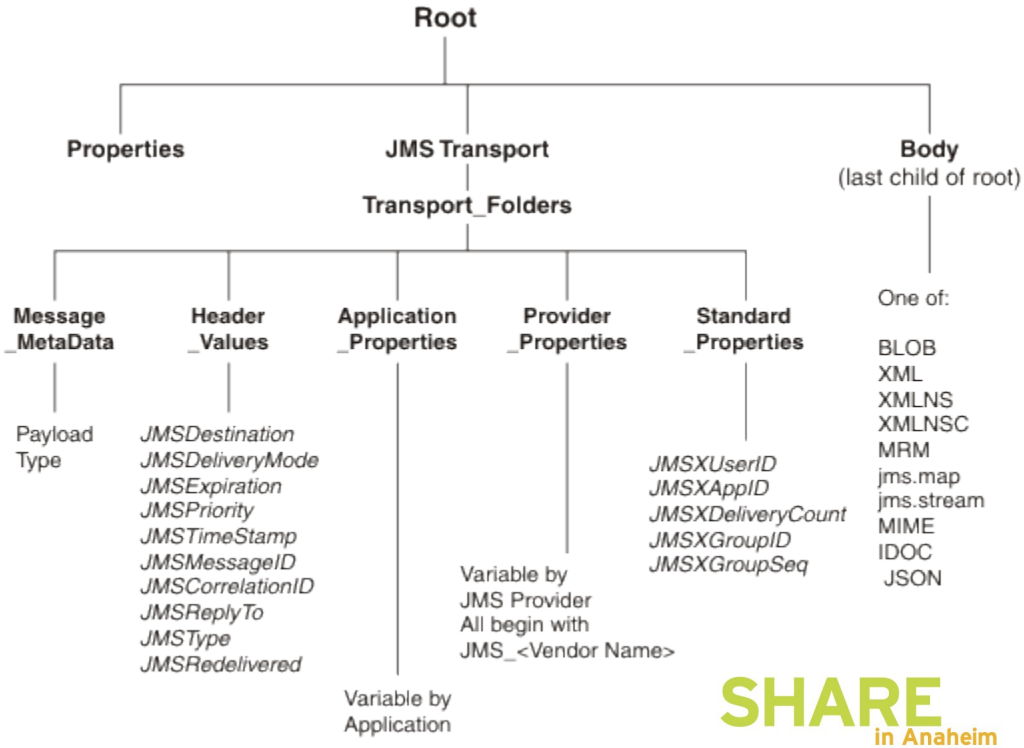
- Also support for JMS on SOAP nodes for SOAP over JMS
- Point to Point + Publish Subscribe

JMS Message Structure

- JMS Messages are not like bit stream wire format messages
- JMS Messages are Java objects

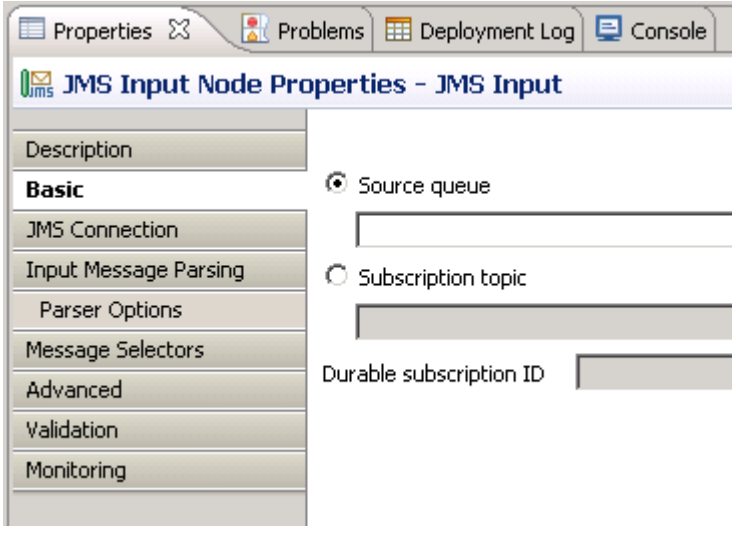
- Message Broker translates the Java object into our logical message tree

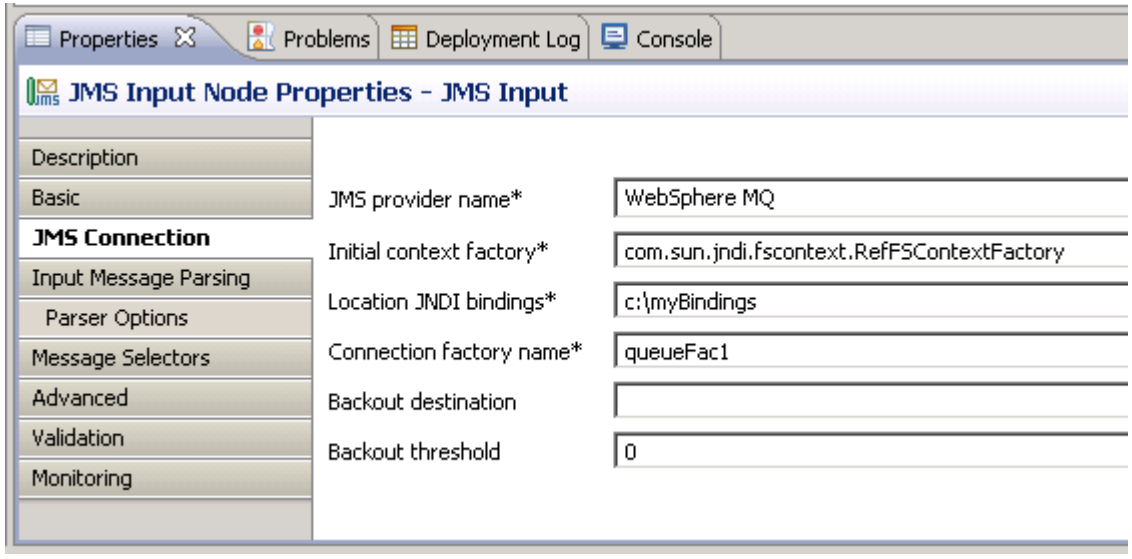
- The payload is stored in the body in the same way as with wire format messages



Configuring JMS Input Node

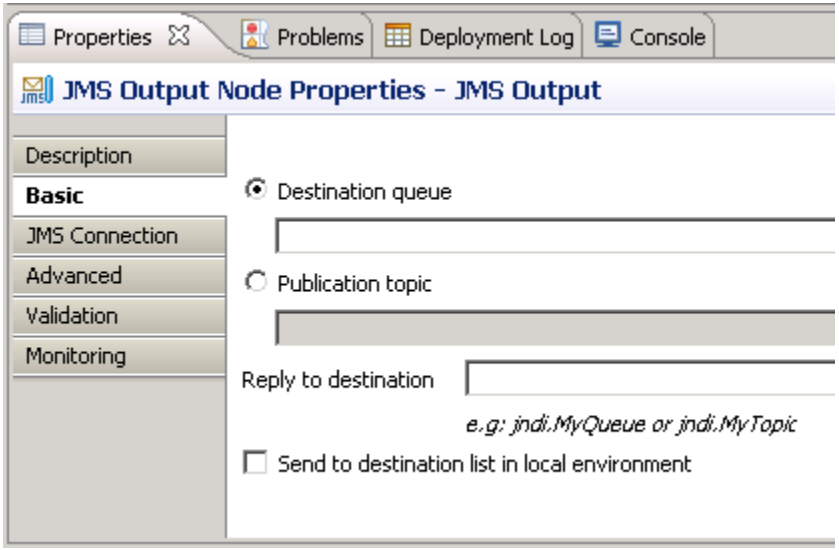
- Point 2 Point or Publish Subscribe message consumer
- Responsible for creating JMS tree from JMS input message
- Hands payload to appropriate broker parser
- Connection Properties
  - Initial Context factory
  - Location JNDI bindings
    - Points to JNDI administered objects
    - LDAP required userId and password
  - ConnectionFactory identified
- Backout Destination & Threshold – must be configured
- Message selectors available if filtering required

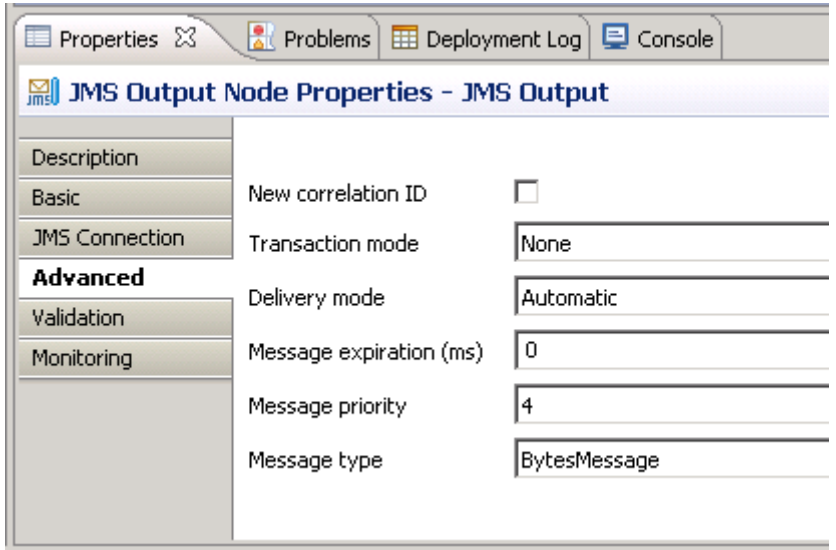




Configuring the JMS Output/Reply Node

- Point to Point or Publish Subscribe message producer
- Responsible for creating JMS output message from JMS tree
- JMS Connection values similar to JMSInput
  - No backout details
  - If you have a JMSInput node in the same the JMSOutput/Reply node does not have to use the same provider
- JMSReply node just sends its message to the destination in the JMSReplyTo header
- JMSOutput node can either send to a queue or a publish to a topic
  - Reply to destination settable for request messages

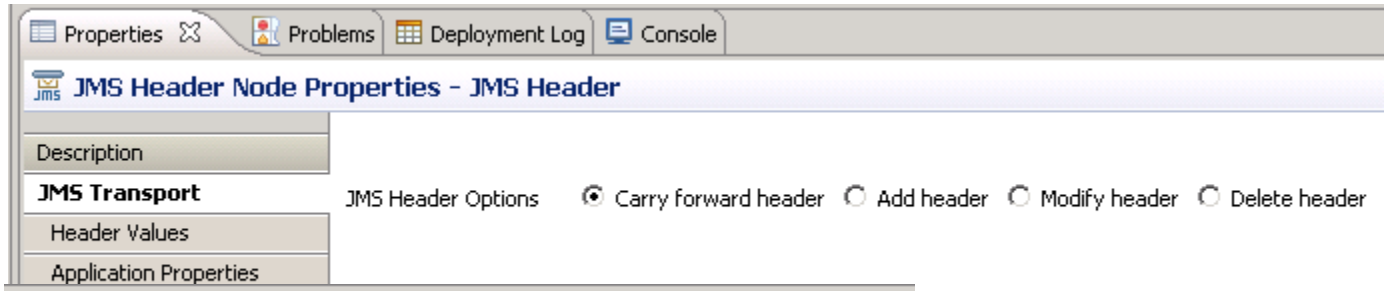


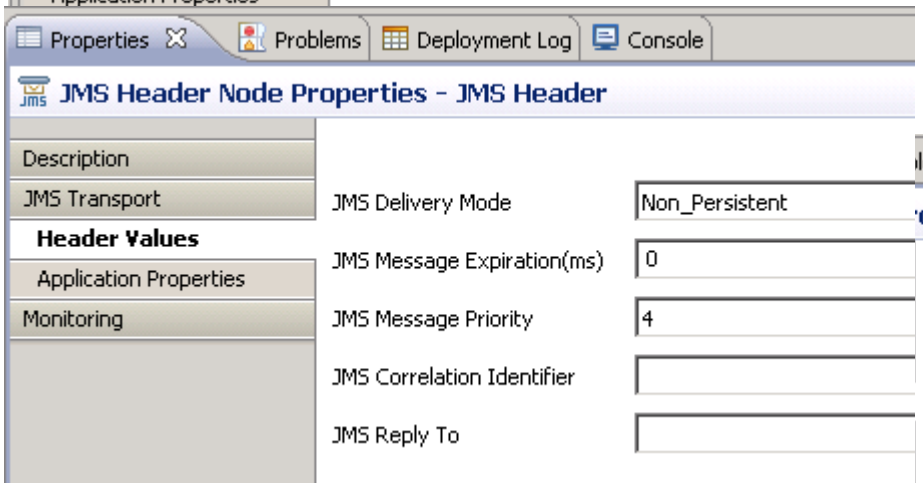


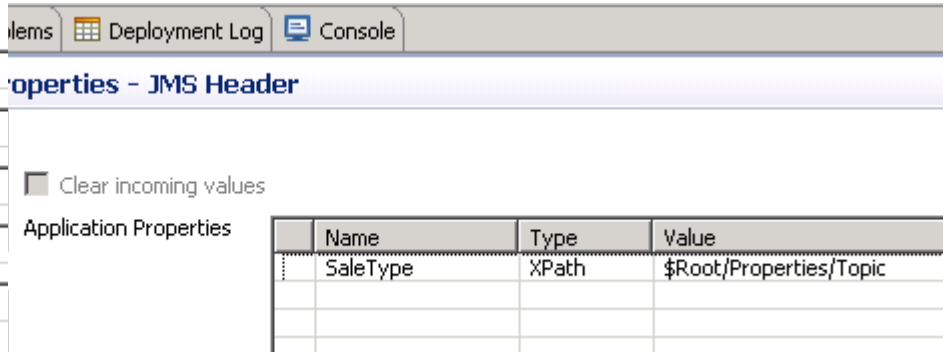
JMSHeader node

- Code-free way to create/update/delete JMS headers
- Subset of common values can be changed
- Application properties can be added or deleted
  - Static or dynamic values using XPath

JMS Header



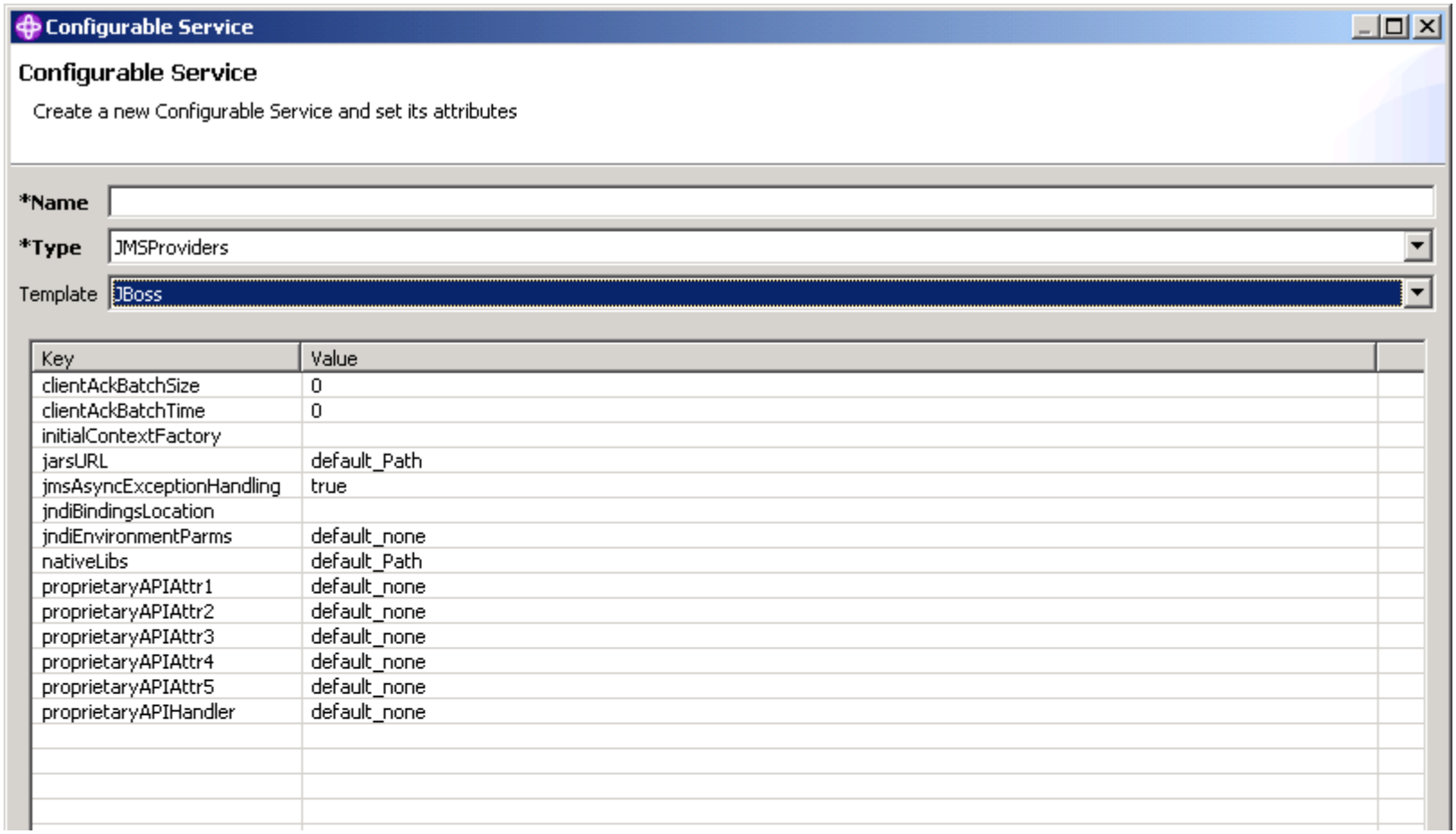




Configuration steps for any JMS provider

- Ensure JMS Provider jars available to the broker
  - Either place them in the shared-classes directory under the broker's workpath
    - Linux/Unix default: /var/mqsi/shared-classes
    - Windows default: C:\Documents and Settings\All Users\Application Data\IBM\MQSI\shared-classes
    - z/OS: <broker component directory>/shared-classes
  - Or use the jarsURL property on the JMSProvider configurable service
- Add any JMS Provider native libraries to Broker's LIBPATH
- You can secure your JMS connections and JNDI lookups
  - JMS connection:
    - mqsisetdbparms <broker Name> -n jms::<Connection Factory Name> -u myuserid -p secret
  - JNDI lookup
    - mqsisetdbparms <broker Name> -n jndi::<Initial Context Factory> -u myuserid -p secret
- Use JMSProvider configurable service to override the JMS connection properties on the node
  - Also useful to enable provider specific JMS options as not all JMS 1.1 compliant supplier implement the spec in the same way
    - JBOSS uses asynchronous exception handling
    - BEAWeblogic requires extra parameters for XA

JMSProviders configurable service



WMB JMS Extras

- SOAP over JMS
  - SOAP over Java Message Service 1.0 is a specification that describes how SOAP can bind to a messaging system that supports the Java Message Service
  - Use WSDL definitions with JMS Bindings

```
<soap:address location="jms:jndi:REPLYTOQ2?
  jndiConnectionFactoryName=QCF&amp;
  jndiInitialContextFactory=com.sun.jndi.fscontext.RefFSContextFactory&amp;
  jndiURL=file:/C:/mqsi6/webservices/SOAP/JMS/JNDI&amp;
  targetService=SOAPJMSGenMessageSetSOAP_JMS_Service&amp;
  timeToLive=30000"/>
```

  - Supported across all WMB SOAP Nodes

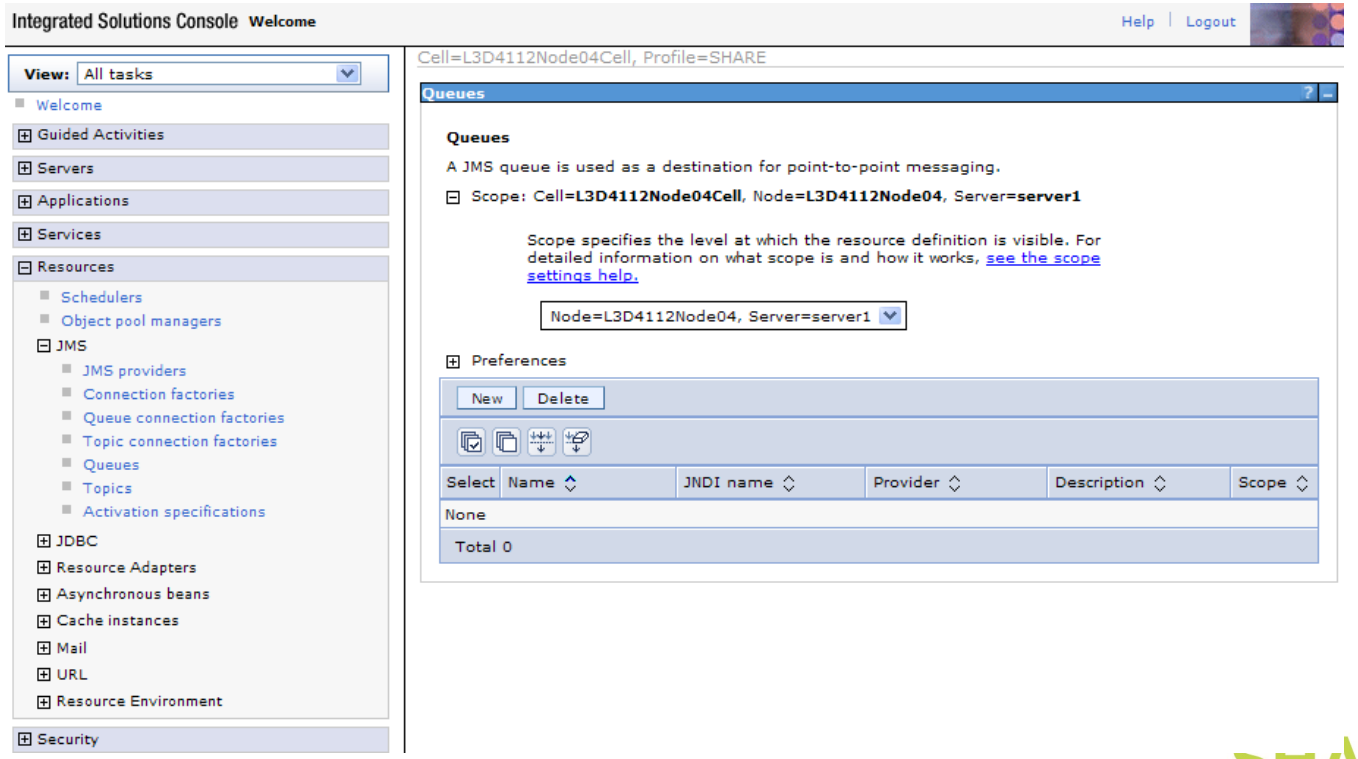
- Coordinated transaction support
  - XA on distributed
    - Only if the JMS Provider supports the XA/Open interface
  - RRS on z/OS when using the MQ Java Client and connecting in Bindings mode

Any questions?

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

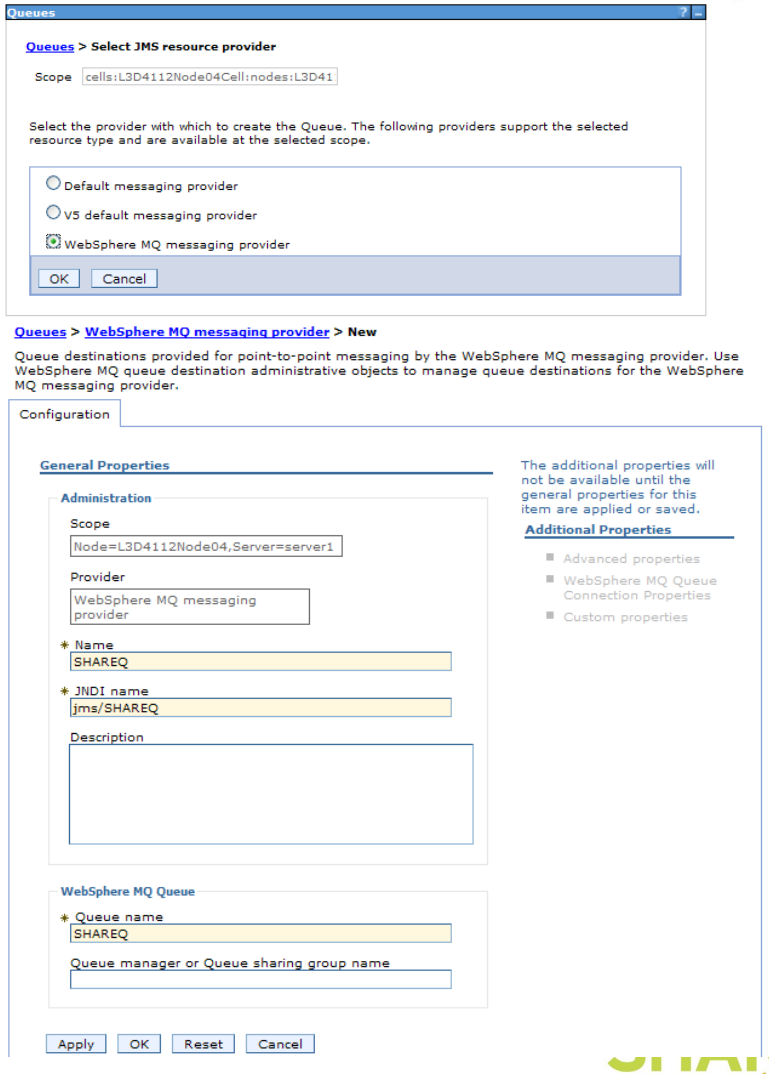
Demo Slides

- With an existing MQ Queue manager, and WebSphere Application Server instance running, we start by defining a JMS queue resource via the WebSphere Administration Console



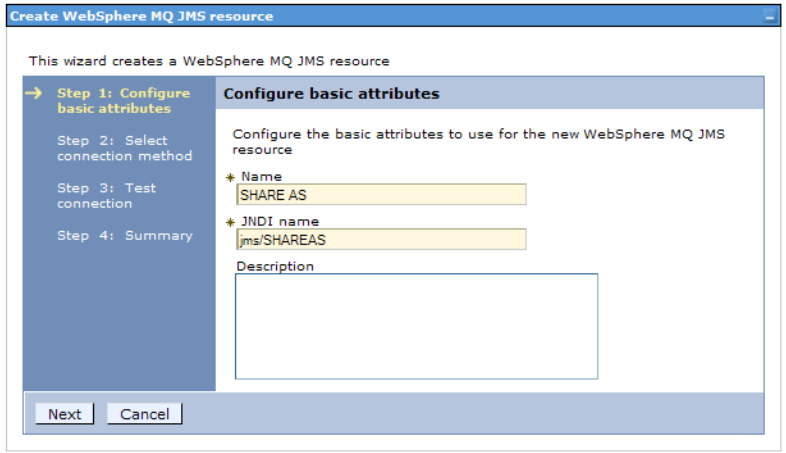
Demo Slides

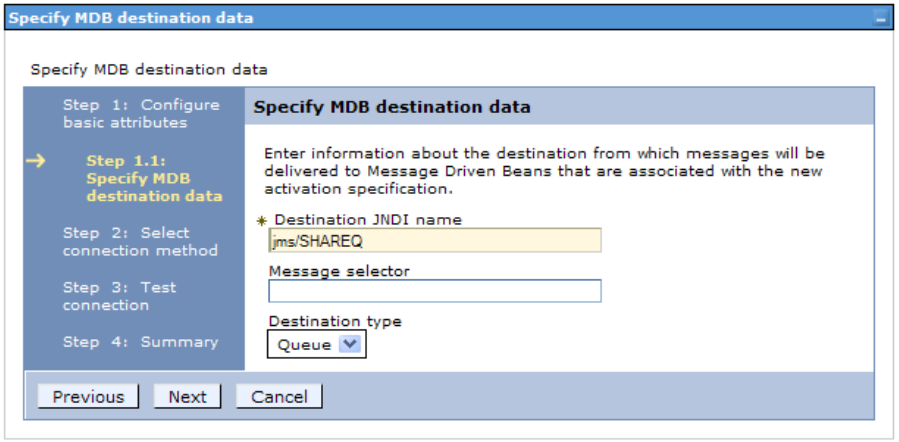
- Click on New, then select the WMQ JMS provider, and click OK
- Fill in the destination details, and click OK.



Demo Slides

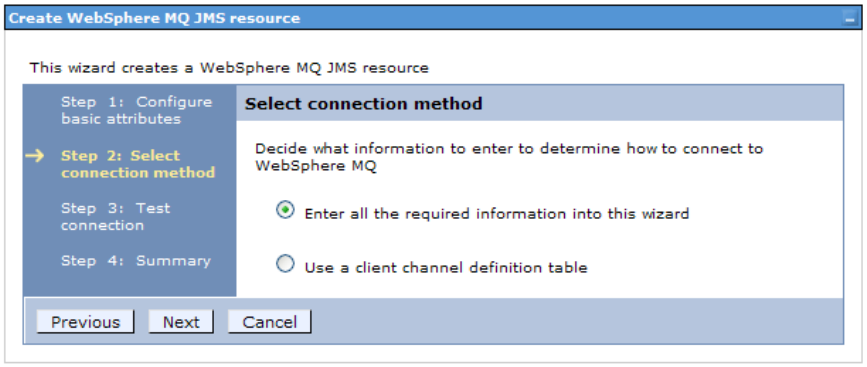
- Create a new WMQ JMS Activation Specification. Click on JMS, Activation Specifications, then New, and select the WMQ JMS Provider. Fill in the name details, and click Next.
- Select the JMS destination to use, for example, the destination just created. The message selector (filter) can also be specified here.



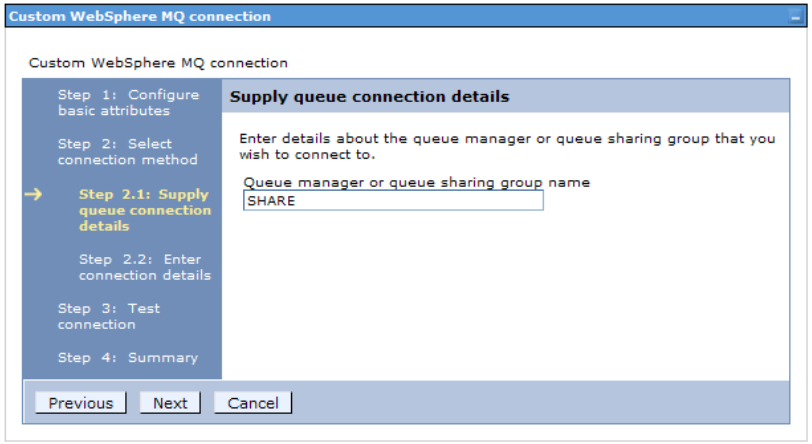


Demo Slides

- Select the connection method, for example, entering all the information to the wizard.

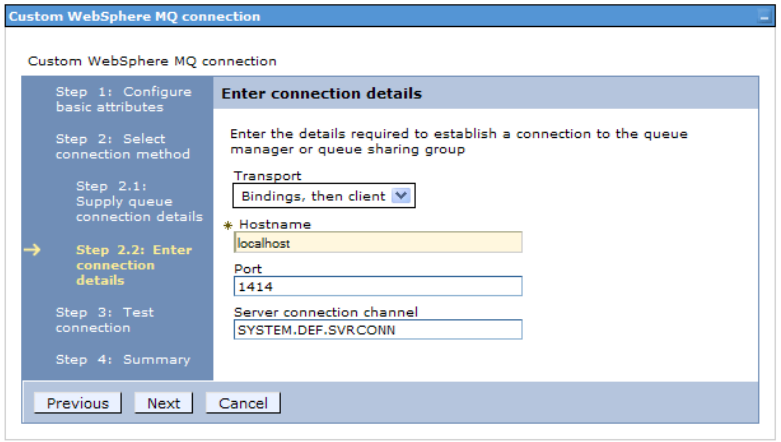


- Enter the Queue Manager name

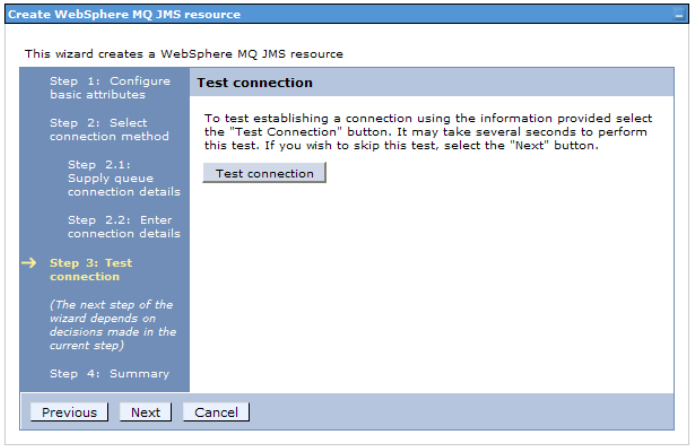


Demo Slides

- Enter the connection details



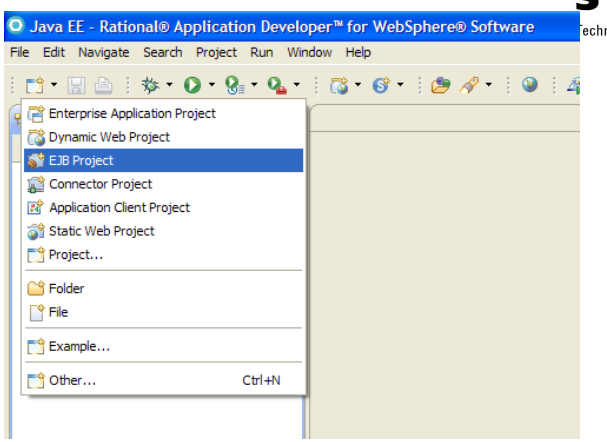
- Test the connection, which should return successfully!
- Click Next, Finish and then Save to store the information

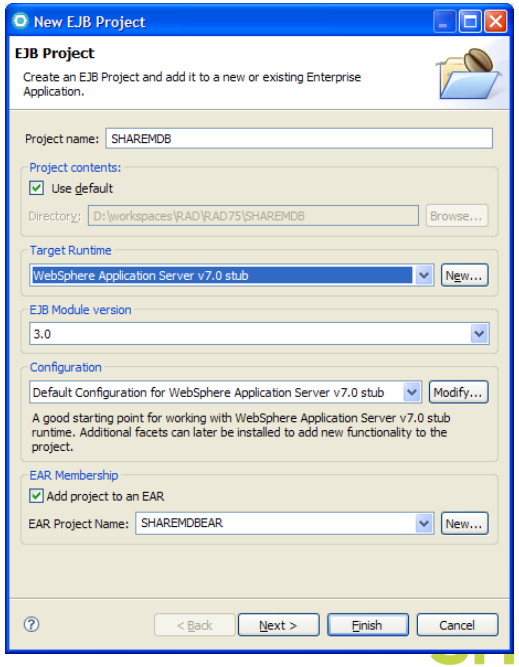


Demo Slides

- In Rational Application Developer, create a new EJB Project

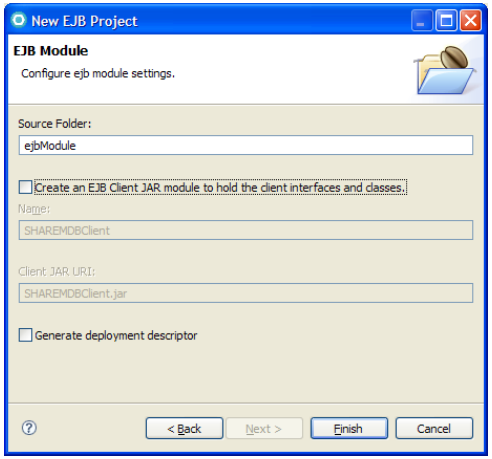
- Specify a project name, the target application server, and ensure a EAR project is also specified/created.



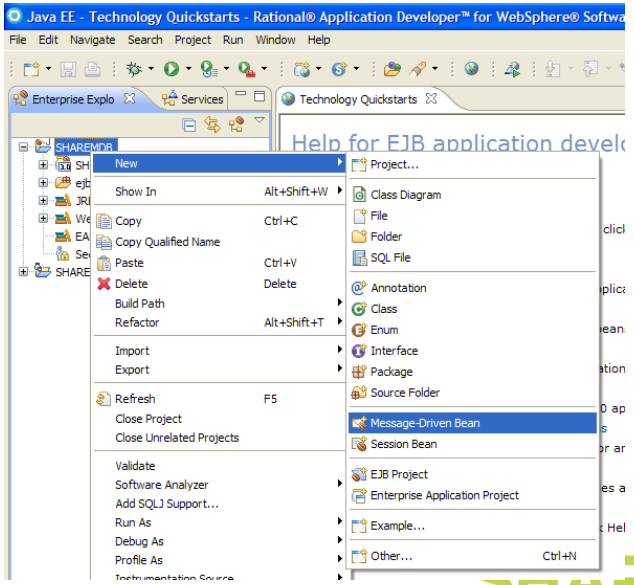


Demo Slides

- There's no need to create a EJB Client. Click Finish.

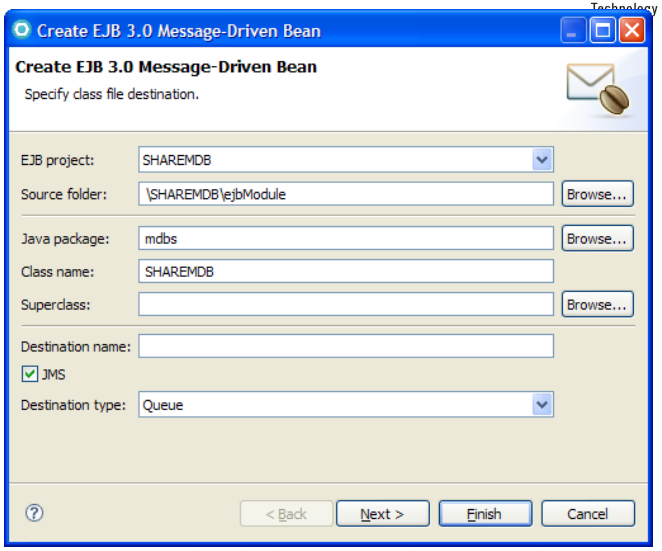


- Right click on the EJB project, and create a new Message-Driven Bean.



Demo Slides

- Enter the MDB Java package name and class name to be created. Click Finish.



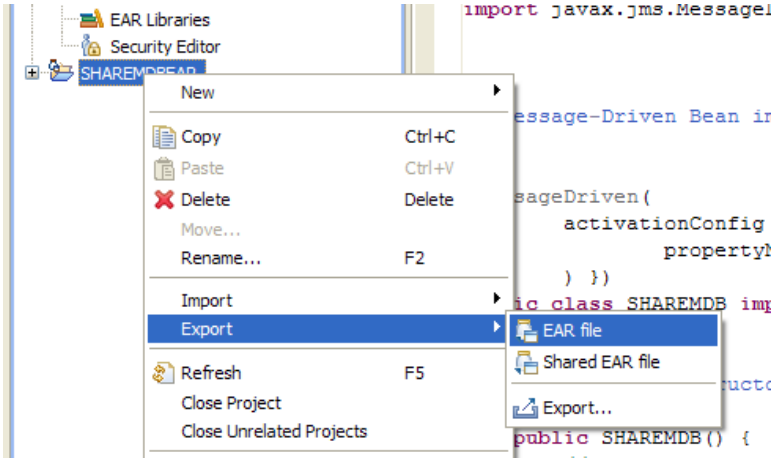
- In the MDB, enter the business logic code into the onMessage method. (This code would require the following Java import statements:

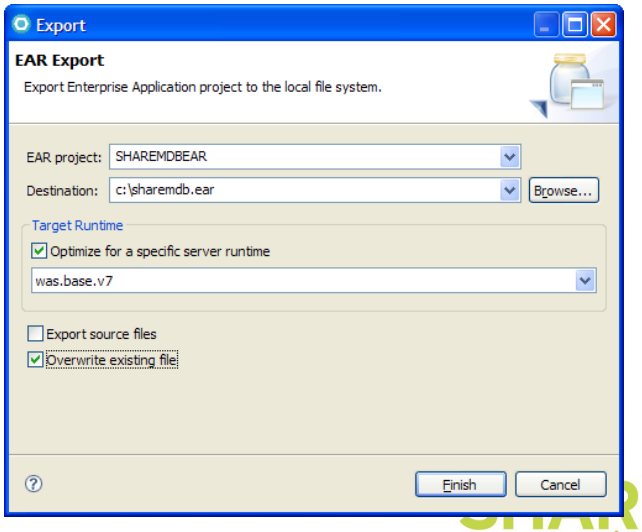
import javax.jms.JMSException;
import javax.jms.TextMessage;

```
public void onMessage(Message message) {
    if (message instanceof TextMessage) {
        try {
            System.out.println("Received text message: "
                    + ((TextMessage) message).getText());
        } catch (JMSException e) {
            e.printStackTrace();
        }
    } else {
        System.out.println("Received message: " + message);
    }
}
```

Demo Slides

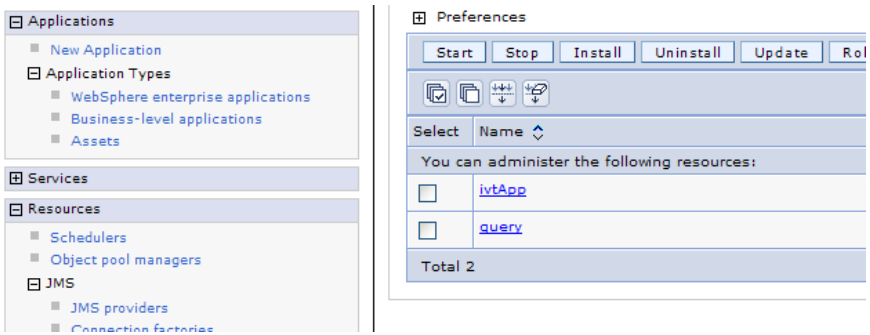
- Right click on the EAR project, and export the EAR file, ensuring the file ends with ".ear".

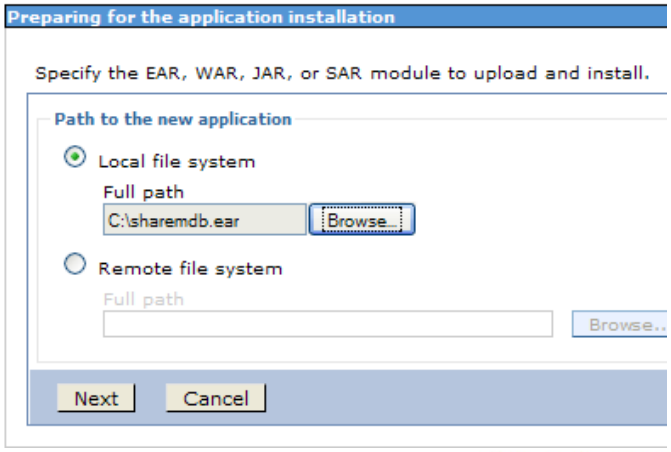




Demo Slides

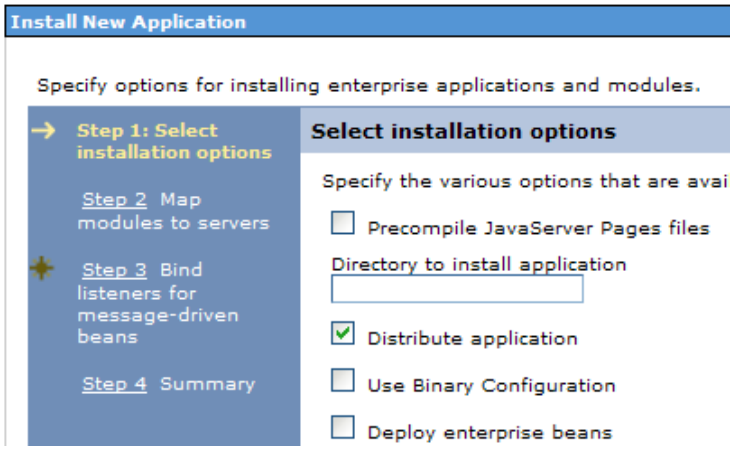
- In the WAS Administration Console, click Applications, Application Types, WebSphere enterprise applications, and then the “Install” button
- Locate the newly created .ear file, and click Next. Then choose “Fast Path”



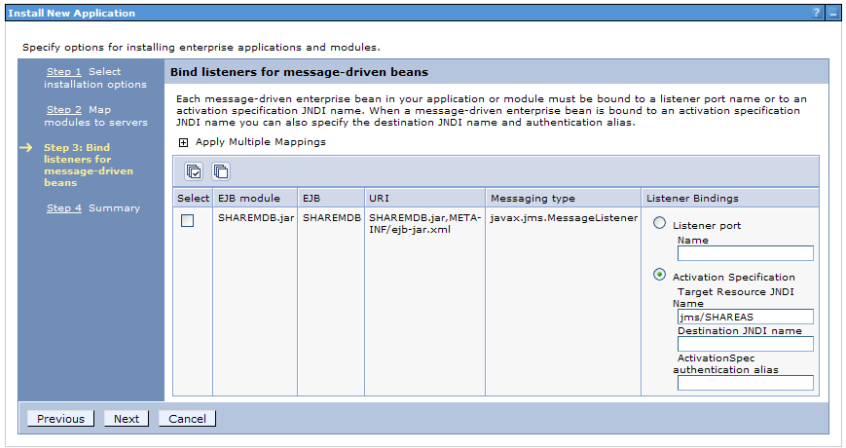


Demo Slides

- It is possible to skip over Steps 1 and 2, and click on Step 3.



- Enter the Activation Specification created earlier, and click Next, then Finish.
- Then save the application to the master configuration.



Demo Slides

- The application is now deployed! Select it, and click the Start button to get it running.
- You can now put a test message to the queue, and for this application, see the result in the Application Server SystemOut.log.