Guided Exercise: Configuring Messaging Resources

In this lab work, you will configure JMS resources required by an application.

Resources	
Files:	**/home/student/JB248/labs/messaging-resources**
App URL:	**http://localhost:9990**
	http://localhost:8080/messaging-client

Results

You must be able to create a JMS ConnectionFactory and Queue and implement applications that use them.

before you start

This guided exercise uses two simple test applications:

1. **messaging-client.war is a web application that connects to the JNDI address java:/jms/CustomCF JMS ConnectionFactory and publishes messages to the JNDI address java:/jms/queue/JB248TestQueue JMS Queue. Accepts text from the user to use as message properties and the message body.**

2. **messaging-mdb.jar is an application with a Message Driven Bean (MDB) that connects to the same JMS ConnectionFactory and consumes messages from the same JMS Queue as the previous application. Displays the message body and properties in the EAP server log.**

Both applications will be deployed on a separate server instance created in the first guided exercise on data sources.

Before beginning the guided exercise, run the following command to verify that EAP is installed to /opt/jboss-eap-7.0 and that no EAP instances are running, and to download the files for the exercise:

```
[student@workstation ~]$ lab messaging-resources setup
```

1. Start a standalone EAP 7 server instance.

Start an instance of EAP using the /home/student/JB248/labs/ standalone folder as the base directory, but using the standalone-full.xml configuration file.

Use the following commands:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./standalone.sh --server-config=standalone-full.xml \ -Djboss.server.base.dir=/
home/student/JB248/labs/standalone/
```

Leave this terminal running the standalone EAP server instance.

2. Create the JMS ConnectionFactory using the JNDI name expected by the Applications.

2.1. Open another terminal window and start the CLI:

```
[student@workstation ~]$ /opt/jboss-eap-7.0/bin/jboss-cli.sh --connect
```

2.2. Create a pooled-connection-factory that refers to the ActiveMQ in-vm connector default and use the JNDI name java:/jms/CustomCF.

Run the following command:

```
[standalone@localhost:9990 /] cd /subsystem=messaging-activemq/server=default
[standalone@localhost:9990 server=default] ./pooled-connection-factory=\
custom:add(connectors=[in-vm], entries=[java:/jms/CustomCF])
```

Leave this terminal running the CLI and where it is connected. We will return to this topic later in this exercise.

3. Create the JMS Queue using the JNDI name expected by the applications.

3.1. Access the web browser and access the administration console at http://localhost:9990

3.2. Click on Settings in the top level menu bar, and follow Subsystems > Messaging - ActiveMQ > Default. Next, click Queues/Topics in the combo box to enter the Messaging Destinations page.

3.3. Click Add to open the Create JMS Queue form and complete it as follows:

- **Name: TestQueue**
- **Number of JNDI: java:/jms/queue/JB248TestQueue**
- **Durable: verified.**
- **Selector: empty.**

Click Save to create the JMS Queue.

Leave the Admin Console browser tab open, as we will return to this topic a few times during this exercise.

4. Deploy the message producer's test application.

4.1. Click Back in the administration console to leave the Messaging Destinations page. Then click Deployments on the main level menu bar.

4.2. Click Add to enter the Add Deployment wizard. Select Upload a new deployment and click Next.

4.3. Click Browse and choose the messaging-client.war file from the /home/student/JB248/labs/messaging-resources folder. Click Next.

4.4. DO NOT change any fields and click Finish to upload and activate the producer's test application.

4.5. Go back to the terminal where EAP was started and check the log messages, confirming that the deployment was successful.

You should see messages similar to the following:

```
13:08:15,111 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 77) WFLYUT0021:
  Registered web context: /messaging-client 13:08:15,128 INFO [org.jboss.as.server]
(XNIO-1 task-7) WFLYSRV0010: Deployed
  "messaging-client.war" (runtime-name : "messaging-client.war")
```

use

Note that the implementation can succeed even if you made a mistake in the previous steps, and the JMS resources were created using different JNDI names than expected by the application.

Also leave the application's browser tab open, as we will return to this topic a few times during this exercise.

5. Post two test messages.

5.1. Open a browser tab and enter the producer's test application using the following URL: http://localhost:8080/messaging-client

5.2. Complete the web form as follows:

- **Message count: 2**
- **Message label: test**
- **Message to send: testing message sending**

Click Send Message to generate (or send) the message.

5.3. The web form should update and show the following message in green: Messages sent successfully.

6. Verify that the messages have been placed within the EAP.

6.1. Go back to the browser tab where the administration console is open.

Click Runtime on the top level menu bar. Then go to Standalone Server > Subsystems > Messaging - ActiveMQ and click View to enter the Messaging Statistics page.

6.2. Click View in the default messaging provider table row to view the JMS Queue Metrics page.

6.3. Click the TestQueue row and notice that the Counters for Message Count and Messages Aggregate are two. All other counters must be at zero.

The following figure illustrates what you should see:

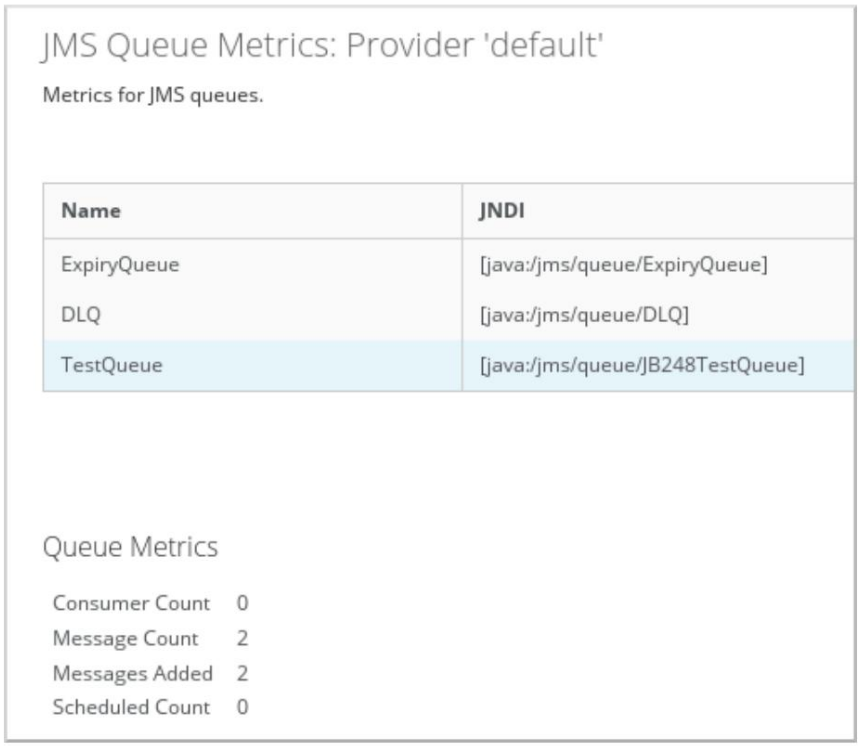

Figure 8.4: JMS Queue Metrics page after generating (or sending) two messages.

If you see different values in the counters, they should reflect other actions you took using the producer's test application. But if your actions are NOT reflected by the counters, an error occurred in a previous step.
For example: if you post a message but the Aggregated Messages counter is NOT incrementing, this means that the publisher application is NOT being able to post messages, perhaps due to incorrect JMS resource configuration.

use

If you created the JMS ConnectionFactory or JMS Queue using different JNDI names than expected by the test application, sending a message will throw a NameNotFoundException in the EAP server log. For example:

```
16:07:49,811 INFO [stdout] (default task-7)
  java.lang.RuntimeException:
  java.lang.RuntimeException: javax.naming.NameNotFoundException: jms/ CustomCF --
service jboss.naming.context.java.jms.CustomCF
```

If you see errors like the above, go back and correct the JNDI names for the JMS resources. Then update the app in the browser tab. There should be no need to redeploy the test application from the producer.

7. Inspect the headers and properties of the messages in the queue.

7.1. Go back to the terminal where the EAP CLI is connected and check the counters of queue messages.

Run the following command:

```
[standalone@localhost:9990 server=default] ./jms-queue=TestQueue\ :read-
resource(include-runtime=true)
```

The expected result is similar to:

```
{
    "outcome" => "success",
    "result" =>
        { "consumer-count" => 0,
        "dead-letter-address" => "jms.queue.DLQ", "delivering-
        count" => 0, "durable" => true,
        "entries" => ["java:/jms/
        queue/JB248TestQueue"], "expiry-address" =>
        "jms.queue.ExpiryQueue", "legacy-entries" => undefined,
        "message-count" => 2L, "messages-
        added" => 2L, "paused" =>
        false, "queue-address" =>
        "jms.queue.TestQueue",
        "scheduled-count" => 0L, "selector" => undefined,
        "temporary" => false


    }
}
```

The values for consumer-count, messages-count, messages-added, and scheduled-count must match those returned by the management console.

7.2. Inspect the headers and properties for messages published but still not consumed.

Run the following command:

```
[standalone@localhost:9990 server=default] ./jms-queue=TestQueue:list-messages
```

The expected result is similar to:

```
{
    "outcome" => "success", "result"
    => [ {

                "JMSPriority" => 4,
                "JMSMessageID" => "ID:c3811a30-12e4-11e6-ae18-597e323e15a8", "address" =>
                "jms.queue.TestQueue", "JMSExpiration" => 0,
                "__AMQ_CID" =>
                "c37cfb7a-12e4-11e6-ae18-597e323e15a8", "Copy" => 1, "JMSTimestamp" =>
                1462468442322L,
                "Label" => "test", "messageID" => 17,
                "JMSDeliveryMode" =>
                "PERSISTENT"

        },
        {
                "JMSPriority" => 4,
                "JMSMessageID" => "ID:c3836421-12e4-11e6-ae18-597e323e15a8", "address" =>
                "jms.queue.TestQueue", "JMSExpiration" => 0,
                "__AMQ_CID" =>
                "c37cfb7a-12e4-11e6-ae18-597e323e15a8", "Copy" => 2, "JMSTimestamp" =>
                1462468442339L,
                "Label" => "test", "messageID" => 18,
                "JMSDeliveryMode" =>
                "PERSISTENT"

        }
    ]
}
```

Notice that each message is enclosed in braces { ... } and, for each message, the Copy and Labels attributes match those entered by the user using the publisher's test application. The message body is not visible using the list-messages operation.

8. Deploy the message consumer test application.

8.1. Go back to the browser tab where the administration console is open.

Click Return to leave the JMS Queue Metrics page, and then click Implementations on the top level menu bar.

8.2. Click Add to enter the Add Deployment wizard. Select Upload a new deployment and click Next.

8.3. Click Browse and choose the messaging-mdb.war file from the /home/student/JB248/labs/messaging-resources folder. Click Next.

8.4. DO NOT change any fields and click Finish to load and activate the application consumer test.

8.5. Go back to the terminal where EAP was started and check the log messages, confirming that the deployment was successful.

You should see a log entry similar to the following:

```
15:38:35,555 INFO [org.jboss.as.server] (ServerService Thread Pool -- 76)
  WFLYSRV0010: Deployed "messaging-mdb.jar" (runtime-name : "messaging-mdb.jar")
```

use

Note that the implementation can succeed even if you made a mistake in the previous steps, and the JMS resources were created using different JNDI names than expected by the application.

For example, if the JDNI name of the JMS ConnectionFactory is incorrect, a NameNotFoundException will be seen in the server logs, but the deploy operation will NOT fail. The server logs appear as follows:

```
16:07:35,914 WARN [org.apache.activemq.artemis.ra] (default  threads - 2)
AMQ152005: Failure in broker activation




org.apache.activemq.artemis.ra.inflow.ActiveMQActivationSpec( ra=org.apache.activemq.artemis.ra.ActiveMQResourceAdapter@78
```

After an error like the one above, the consumer test application will NOT be able to consume any messages and will need to be implemented again after the JNDI names are fixed.

9. Verify that all messages in queues have been consumed.

9.1. The consumer test app includes an MDB that should start immediately to consume messages, and will display the body and properties of each message in the server log.

The expected log entries are similar to:

```
15:38:35,767 INFO [class com.redhat.training.messaging.mdb.MessageReceiver]
  (Thread-1 (ActiveMQ-client-global-threads-427178549)) Message Properties: Copy #1 [test]

15:38:35,767 INFO [class com.redhat.training.messaging.mdb.MessageReceiver]
  (Thread-1 (ActiveMQ-client-global-threads-427178549)) Message Body: testing message sending
```

```
15:38:35,777 INFO [class com.redhat.training.messaging.mdb.MessageReceiver]
  (Thread-0 (ActiveMQ-client-global-threads-427178549)) Message Properties: Copy #2 [test]

15:38:35,778 INFO [class com.redhat.training.messaging.mdb.MessageReceiver]
  (Thread-0 (ActiveMQ-client-global-threads-427178549)) Message Body: testing message sending
```

9.2. Go back to the browser tab where the administration console is open.

Click Runtime on the top level menu bar. Then go to Standalone Server > Subsystems > Messaging - ActiveMQ and click View to enter the Messaging Statistics page.

9.3. Click View in the default messaging provider table row to view the JMS Queue Metrics page.

9.4. Click the TestQueue row and notice that the Added Messages counter is still is at two, but the Message Count counter is at zero because there are no messages waiting to be consumed.

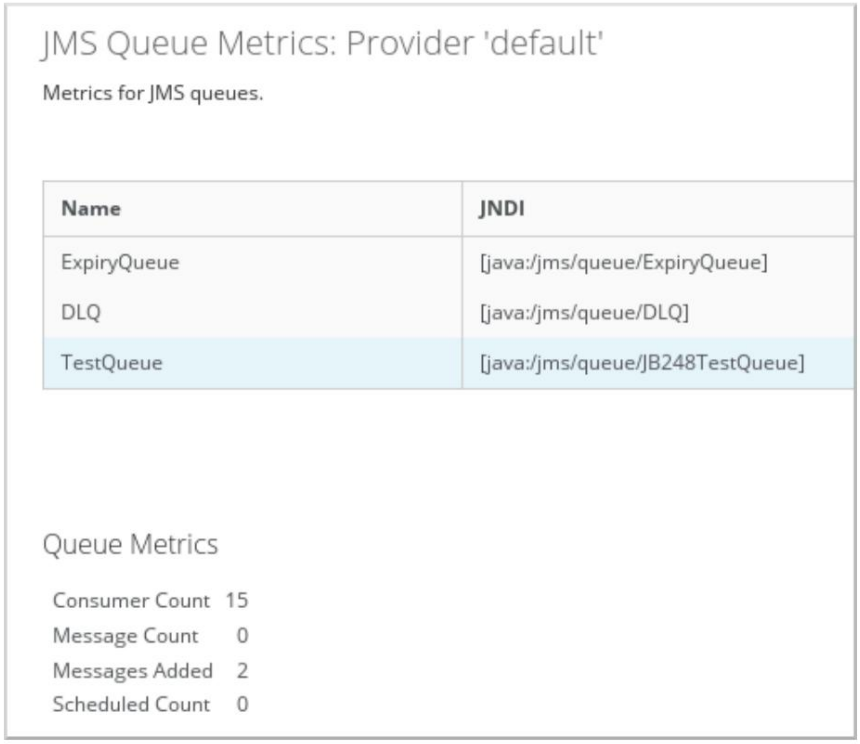

Figure 8.5: JMS Queue Metrics page after consuming two messages.

use

The Message Count counter at 15 means that EAP started 15 copies of the consumer MDB to process messages in parallel. Check the EAP 7 product documentation for information on MDB configuration settings, such as the number of instances created.

10. Post a third test message.

10.1. Go back to the browser tab where the test application of the producer.

10.2. Complete the form as follows:

- **Message count: 1**
- **Message label: third**
- **Message to send: another test message**

Click Send Message to post the message.

10.3. The form must update and show the following message in green: Messages sent correctly.

11. Verify that the message has been consumed immediately.

11.1. Go back to the terminal where the EAP standalone server instance is being running.

It should display log messages similar to the following:

```
18:25:45,583 INFO [class com.redhat.training.messaging.mdb.MessageReceiver]
  (Thread-2 (ActiveMQ-client-global-threads-427178549)) Message Properties: Copy #1 [third]

18:25:45,583 INFO [class com.redhat.training.messaging.mdb.MessageReceiver]
  (Thread-2 (ActiveMQ-client-global-threads-427178549)) Message Body: another test message
```

11.2. Go back to the terminal where the CLI is running and check the counters queue messages again.

Run the following command: (Tip: Use the arrow keys to run the history command again.)

```
[standalone@localhost:9990 server=default] ./jms-queue=TestQueue\ :read-
resource(include-runtime=true)
```

The expected result is similar to:

```
{
        "outcome" => "success",
        "result" => {
```

```
                "consumer-count" => 15, "dead-
                letter-address" => "jms.queue.DLQ", "delivering-count"
                => 0, "durable" => true, "entries"
                => ["java:/jms/queue/
                JB248TestQueue"], "expiry-address" => "jms.queue.ExpiryQueue",
                "legacy-entries" => undefined, "message-count" => 0L,
                "messages-added" => 3L, "paused" =>
                false, "queue-address" =>
                "jms.queue.TestQueue",
                "scheduled-count" =>
                0L, "selector" => undefined, "temporary" => false




        }
}
```

Notice that the message-count attribute is still at zero, but the messages-added counter increased by three.

12. Stop the standalone EAP server instance, but leave the EAP applications test and JMS resources, as they will be required by the next guided exercise.

Use the CLI to stop the server instance:

```
[standalone@localhost:9990 server=default] /:shutdown
```

This concludes the guided exercise.