Using a Java Messaging Service (JMS) with Informatica Cloud Application Integration
Abstract

This article provides detailed instructions on how to create and deploy business processes with Informatica Cloud Application Integration Process Server running on an Informatica Cloud Secure Agent and a Java Messaging Service provider. It assumes you have some basic product experience but does not require specific knowledge about JMS.

Supported Versions

- Informatica Cloud Application Integration

Table of Contents

JMS Overview .................................................................................................................................. 2
Before You Start .............................................................................................................................. 3
  Install a JMS Provider .................................................................................................................. 3
  Install the Informatica Secure Agent ............................................................................................ 3
Define the Dequeue Process ........................................................................................................... 4
  Create a Process Object .............................................................................................................. 4
  Create a Process ......................................................................................................................... 4
Define and Set Debugging for a Messaging Service ................................................................... 5
Create the Enqueue Process ....................................................................................................... 8
Create a Cloud Process to Invoke the JMS Service .................................................................. 11
Test the New Service .................................................................................................................... 12
  Execute the Process Using a REST Client ................................................................................ 12
View the Process Execution in the Process Console ................................................................. 12

JMS Overview

A Java Message Service (JMS) enables messaging between loosely-coupled processes. “Loosely-coupled”—for purposes of messaging—means the messages are independent of one another. The interaction between the processes can be synchronous or asynchronous. However, this article shows only the synchronous use case. (Asynchronous messages require that you also set message events.)

If the process sends a message while the message consumer is not running, the message waits in the JMS provider's queue until the message consumer is available. The contents of these messages can vary, but the data within the message is usually expressed as XML.

Taking advantage of the data flow that occurs with JMS, you can eliminate the need to invoke a service such as a database management system that runs on-premises.

Follow the examples below to create Process Objects and enable messaging with systems outside Process Designer. The examples illustrate how to move data from one variable to another.

See this article to learn more: http://www.activevos.com/developers/howto#JMS. The details on Process Server (ActiveVOS) are specific to on-premises deployments but apply generally as well.
Before You Start

To complete the tasks outlined in this article, you first need to:

1. Install or obtain information about the JMS Provider with whom you want to integrate
2. Install the Informatica Secure Agent

You can then learn how to:

1. Create a process to dequeue a message. This process receives the message data from the JMS provider and assigns it to a field. The message data is contained in a process object.
2. Define a message service in the Process Console. This enables you to view the activity when it executes.
3. Create a process to hold the enqueue message. This process takes the message data as input and assigns it to the fields created in a process object.
4. Create a process that runs in the cloud and invokes the enqueue and dequeue processes.

Install a JMS Provider

If you have already installed a JMS provider, skip this section. You can obtain and install a JMS to run on your local machine from many providers such as Apache ActiveMQ, HornetQ, IBM WebSphere MQ, and others.

Our illustrations use the Apache ActiveMQ system. Be sure to define a queue in the JMS provider, as you need to specify it in the process later. In this example, the queue is called “local-queue”:

Install the Informatica Secure Agent

The Process Designer requires that the JMS provider be running on-premises.

After you install the Secure Agent, the JMS provider must reside in your Secure Agent Directory, as shown in this example path:

<Informatica Cloud Secure Agent installation directory>\data\process-engine-custom-lib\activemq-all-5.9.0.jar
Define the Dequeue Process

The JMSDequeue process used in this example acts as a JMS consumer or producer and assigns the data it receives to a specific field.

Create a Process Object

To create JMSDequeue, first create a simple Process Object to contain the data structure. While the data could be sent as JSON or text, it is more common to send it as XML.

1. From the Design Home page, select New > Process Object.
2. Enter a Name for the process object. Description and Tags are optional fields.
3. Enter the Object Fields for the process object, a simple Purchase Order in this example. For this example, the PO contains “name” and “desc” fields, both of type Text:

   ![Process Object Fields](image)

4. Save the Process Object.

   The example has only a name and description. You can include other object fields when you define your process object.

Create a Process

To assign data to a field, add the process object to a new process and define an Assignment step. In the Process properties, add the Process Object fields “in” and “out” as described here.

1. From the Design Home page, select New > Process to create a new process.
2. On the General tab, supply a name and select the agent where the process should run:
3. On the **Fields** tab, define the input and output fields. In this case, both the Input Field and Output Field select the TestPO process object as the Type.

![Process Properties](image)

4. On the **Start** tab, check **Input field is whole payload** so you process the entire contents of the request.

5. Select the Empty step in the canvas and change the **Step Type** to **Assignment** in the properties.

6. Click **Apply** to update the step:

![Workflow Diagram](image)

7. Save the guide and publish it.

**Next:** Define the messaging service in Process Console, so you can run the process on-premises using the Secure Agent’s Process Server.

**Define and Set Debugging for a Messaging Service**

You can now define a messaging service in the Process Console that connects to the queue you created in the JMS provider.

**Define the Messaging Service**

1. From Informatica Cloud, select **Monitor > Systems & Processes** to open the Process Console.

2. Select your agent in the **Console** list. This is the same agent specified in the Process Properties of JMSDequeue.

3. From the **Admin** menu, select **Configure Services > Messaging Service**.
4. To create a new service, enter a **Manager Name** and click **Add Manager**.

![Message Service](image)

5. Select the new Manager Name from the list.

6. On the Messaging Service page, define the queue using the example as a guide.

In some cases, particularly while debugging, you might be using a service that does not set a connection username or password.

For the **Initial Context Properties**, which enable the Secure Agent to get JMS messages, note that:

- Every JMS provider has a unique connection factory. The example uses ActiveMQ so the connection factory is specified as:
  
  `org.apache.activemq.jndi.ActiveMQWASInitialContextFactory`
  
  You can find more information here: [http://activemq.apache.org/maven/apidocs/org/apache/activemq/jndi/ActiveMQWASInitialContextFactory.html](http://activemq.apache.org/maven/apidocs/org/apache/activemq/jndi/ActiveMQWASInitialContextFactory.html)

- Set the provider URL property to the local host where you installed the JMS provider.

- The local-queue specified here (and below as the JNDI Location) is the name of the queue you created in the JMS provider.

![Message Service](image)
Under **Queues & Listeners**, define:

- **JNDI Location**: The queue name you defined in the JMS provider.
- **Listener Class**: Verify that the class name specified is:
  
  ```java
  com.activevos.rt.socrates.web.mom.jms.transport.AeRealTimeJms.Listener
  ```

  **Note**: The **Listener Class** responsible for dispatching messages to Process Server depends on the consumer. For processes implemented with **Process Designer**, use the listener class name:
  
  ```java
  com.activevos.rt.socrates.web.mom.jms.AeRealTimeJmsListener
  ```

  Otherwise, if you are using **Process Developer**, use the listener class name:
  
  ```java
  com.activee.rt.mom.jms.transport.AejmsBpelListener
  ```

- **Listener Count**: You can change this if needed. The default is 15.
- **Default Service**: Name of the process that consumes the messages, in this case, JMSDequeue.

7. Click **Update** to save the Messaging Service.

**Enable Debugging**

Next, change some of the default properties to allow for easier debugging:

1. From the **Catalog** menu, select **Contributions**.
2. From the **Contributions** list, select the service.
3. Click the name of the service in the **Deployed Processes** section.
4. Set the **Logging Level** and **Persistence Type** to **Full**.
Using these options, you retain data handled by the queue and get feedback for debugging. Otherwise, for monitoring purposes, you can see only that the process ran:

**Create the Enqueue Process**

You can now create a process to hold the enqueue message. The completed process looks like this on the canvas:

To create the process, follow these steps:

1. From the Design Home page, select **New > Process** to create a new process named “JMSEnqueue” in this example.
2. On the **General** tab, select the agent where the process should run.
3. On the **Fields** tab, create a temporary field to store the data being enqueued. Add the process object to the temporary field. Recall that the TestPO object simply contains a Name and a Description:

![Image of Fields tab with TestPO object added]

4. Create an Assignment step that adds data to both elements:

![Image of Assignment step with TestPO object attributes assigned]
5. Add a Service Call step and select the JMS Enqueue Service defined above:

6. On the Input tab, define the following properties:
   - **Messaging Manager Name**: The name of the queue you defined within the Process Console.
   - **Destination Name**: The JNDI name of the queue you assigned within the Process Console.
   - **Message**: The name of the field that contains the message to be enqueued.
   - **Format**: The format of the message. In this case, select XML.
   - **Unwrap Root Element**: Leave this checked to access elements within the XML structure.

7. Click **Apply**. Then save and publish the process.
**Create a Cloud Process to Invoke the JMS Service**

The JMSEnqueue process uses the Secure Agent’s Process Server you selected in the properties to execute on-premises.

To access this process from the cloud, create a new process and add a subprocess step to invoke JMSEnqueue. This process runs on the Cloud Server.

1. Select **New > Process** to create a new process.

   In the example, this process is named RunJMSExample. Note that the example passes no inputs into this process. As shown above, the sample hardcodes a name and description in the Assignment step.

2. On the **General** tab, choose to run the process on the **Cloud Server**.

3. Click the **Empty** step to display the properties.

4. For **Step Type**, choose Subprocess.

5. In the **Process** list, select the JMSEnqueue process you created above.

6. Save and publish the process.

The completed process looks like this on the canvas:
Test the New Service

To test the new service, you need access to a REST client. This example uses the Advanced Rest Client application, available as an extension to Google Chrome.

Execute the Process Using a REST Client

To execute the process from a REST client, get the Service URL for the RunJMSExample, which is visible on the Design Home page.

1. Choose RunJMSExample in the list of processes to view the Service URL in its properties, as shown here (note that the public context used for this example implies that the process was configured to be invoked anonymously):

2. Copy the Service URL and paste it into the REST client.

3. Select POST.

4. Click Send.

View the Process Execution in the Process Console

After you have executed the RunJMSExample in the REST client, you can see the process execution path in the Process Console.

1. Log in to Informatica Cloud.

2. From the Monitor menu, choose Service & Process Console.

4. From the list of processes, click the RunJMSExample. This entry was created when you invoked the process from the REST Client above.

5. You can see in the log that each step in the process executed successfully:

6. Click the Detail icon to display detailed information about each step of process execution:

Author

Kellie Delaney
Principal Technical Writer