Enabling Real-Time Data Flow between IBM MQ and ActiveMQ with JMS

Integrating IBM MQ and ActiveMQ for Message Delivery using JMS

-Votarikari Shravan (912)

Table Of Contents

Introduction

This project aims to establish a communication bridge between IBM MQ and ActiveMQ messaging systems, enabling seamless message exchange through the Java Message Service (JMS) protocol.

Objective

- Integrate IBM MQ and ActiveMQ to facilitate reliable and efficient message delivery.
- Utilize JMS as the standardized messaging protocol for message exchange.
- Develop a solution that ensures compatibility and interoperability between the two messaging platforms.

Pre-requisites

- Apache-activemq-5.x.x
- IBM MQ 9.x.x
- IBM/ACE Toolkit

Download and install all required software sets as mentioned above.

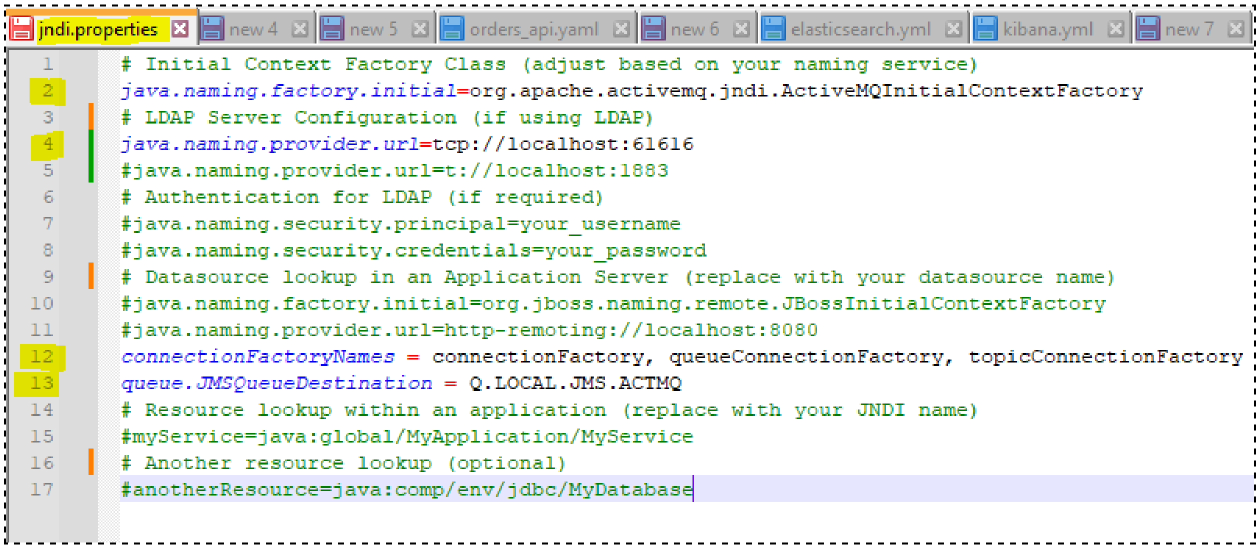
Preparation

To establish a communication between IBM MQ and Active MQ, we need to define IBM\ACE Integration solutions as required following

- ACE Application (**App_ConnIBMMQtoActiveMQ**)
- JMS Provider Policy Project (**JMSProvider**)
- Jar Files to define a property file and required classes for Active MQ.

JMS Property File Setup:

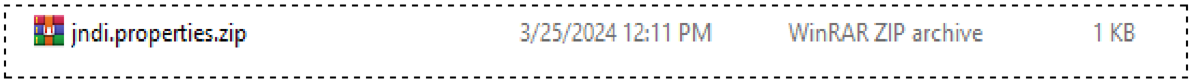
To establish a communication between IBM MQ and ActiveMQ we need to configure a JMS Administered Objects that is configured by creating a jms.properties file and convert it into a jar file.
Create a text file with a name jndi in C:\activemq folder with .properties extension.



```
# Initial Context Factory Class (adjust based on your naming service)
java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory
# LDAP Server Configuration (if using LDAP)
java.naming.provider.url=tcp://localhost:61616
#java.naming.provider.url=t://localhost:1883
# Authentication for LDAP (if required)
#java.naming.security.principal=your_username
#java.naming.security.credentials=your_password
# Datasource lookup in an Application Server (replace with your datasource name)
#java.naming.factory.initial=org.jboss.naming.remote.JBossInitialContextFactory
#java.naming.provider.url=http-remoting://localhost:8080
connectionFactoryNames = connectionFactory, queueConnectionFactory, topicConnectionFactory
queue.JMSQueueDestination = Q.LOCAL.JMS.ACTMQ
# Resource lookup within an application (replace with your JNDI name)
#myService=java:global/MyApplication/MyService
# Another resource lookup (optional)
#anotherResource=java:comp/env/jdbc/MyDatabase
```

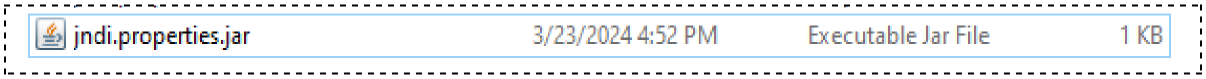
Convert this file as a jar file by following steps:

1. Add to zip file



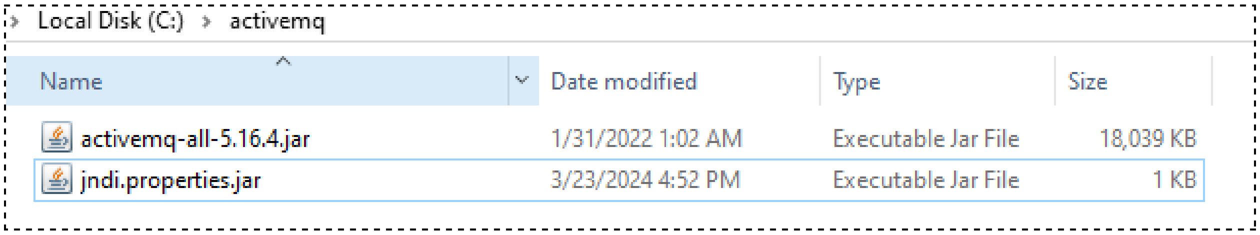
jndi.properties.zip	3/25/2024 12:11 PM	WinRAR ZIP archive	1 KB

2. Rename zipped file as jar file



jndi.properties.jar	3/23/2024 4:52 PM	Executable Jar File	1 KB

NOTE: The Value given for ***Type 4 Driver Class JARs URL*** is a Folder Path, where we store Jars to associate in connecting IBM MQ with Active MQ.



Local Disk (C:) › activemq

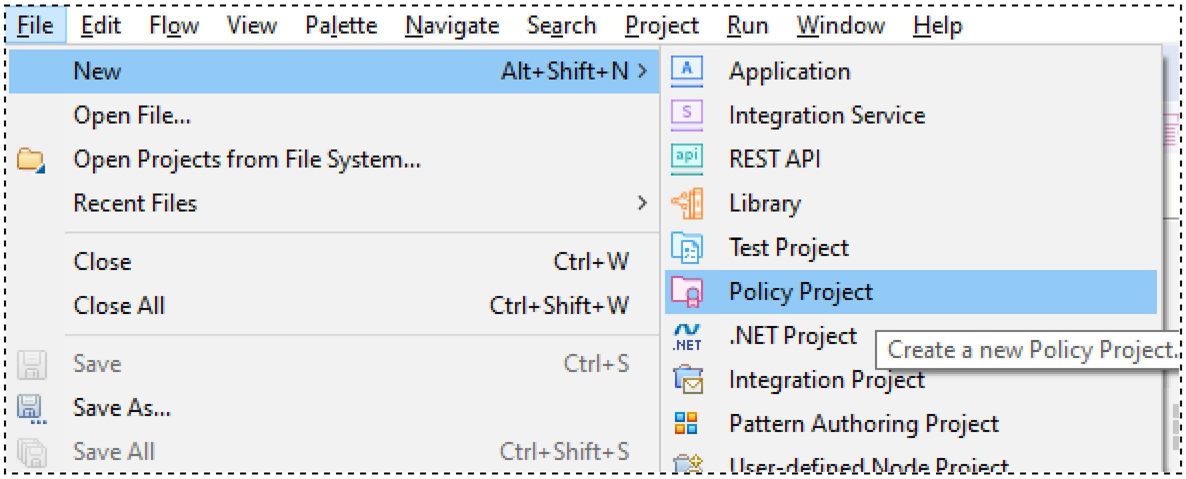
Name	Date modified	Type	Size
activemq-all-5.16.4.jar	1/31/2022 1:02 AM	Executable Jar File	18,039 KB
jndi.properties.jar	3/23/2024 4:52 PM	Executable Jar File	1 KB

In ACE Toolkit :

Step 1:

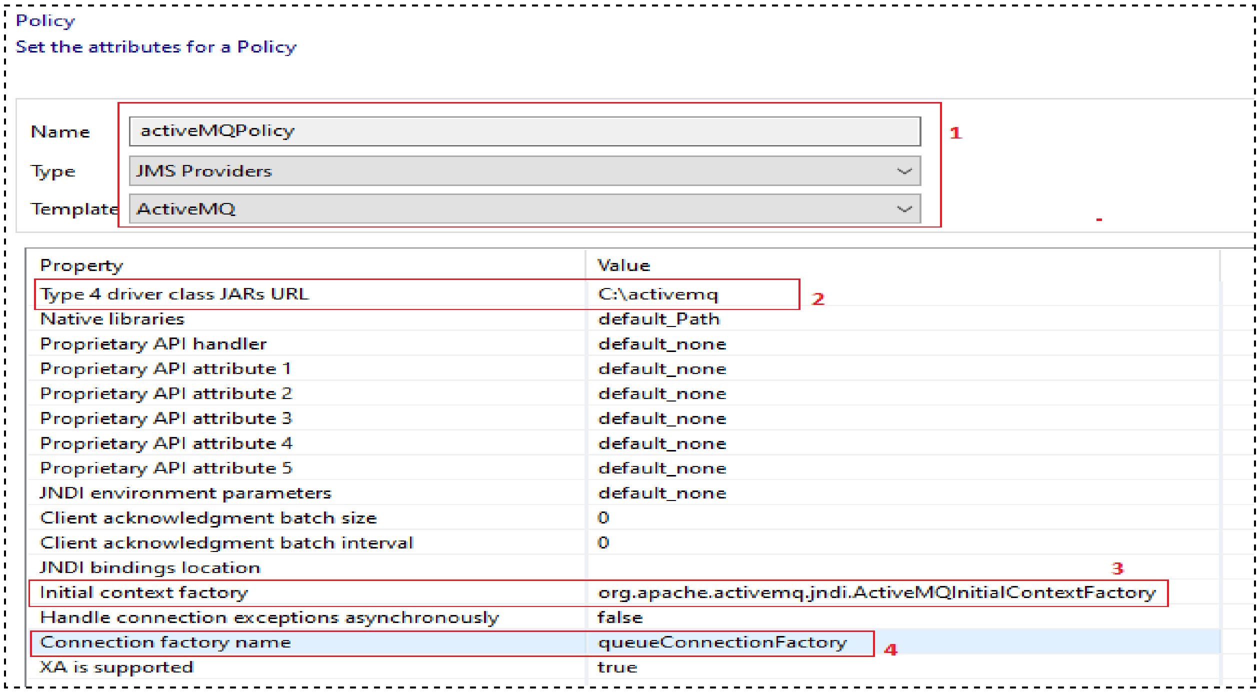
Create a Policy Project as following with a name **JMSProvider Policy**

File -> New -> Policy Project



Step 2:

Create a Policy in this Policy Project. And Give the property values as given below



Policy

Set the attributes for a Policy

Name	activeMQPolicy
Type	JMS Providers
Template	ActiveMQ

1

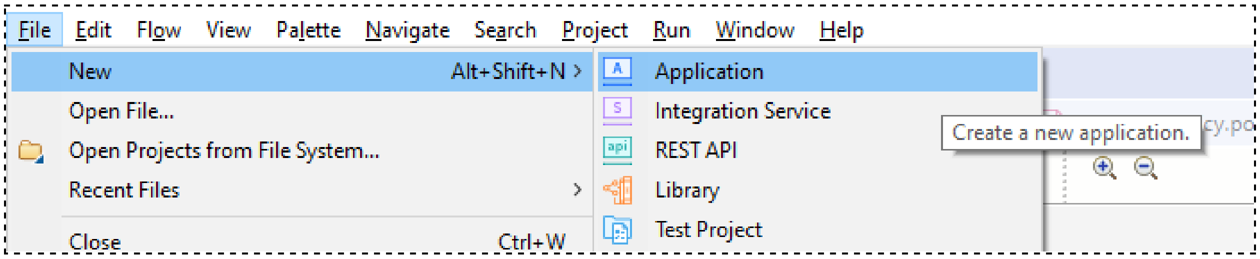
Property	Value
Type 4 driver class JARs URL	C:\activemq
Native libraries	default_Path
Proprietary API handler	default_none
Proprietary API attribute 1	default_none
Proprietary API attribute 2	default_none
Proprietary API attribute 3	default_none
Proprietary API attribute 4	default_none
Proprietary API attribute 5	default_none
JNDI environment parameters	default_none
Client acknowledgment batch size	0
Client acknowledgment batch interval	0
JNDI bindings location	
Initial context factory	org.apache.activemq.jndi.ActiveMQInitialContextFactory
Handle connection exceptions asynchronously	false
Connection factory name	queueConnectionFactory
XA is supported	true

2 3 4

Step 3:

Open ACE Toolkit and Create an application as follows with the name **App_ConnIBMMQtoActiveMQ.**

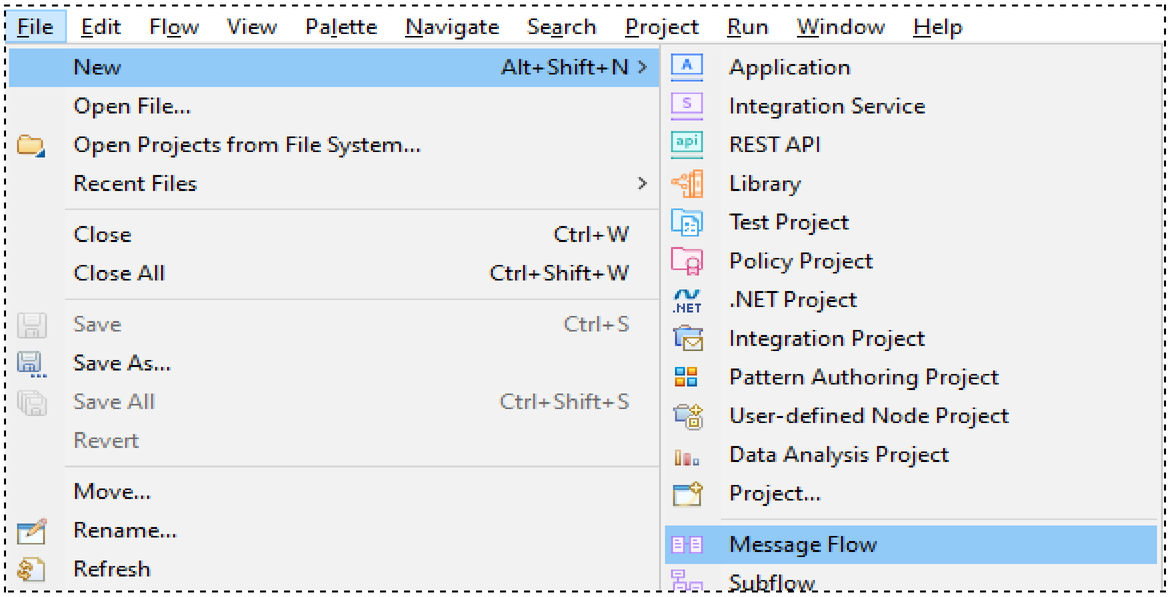
File -> New -> Application



Step 4:

Create a New Message Flow

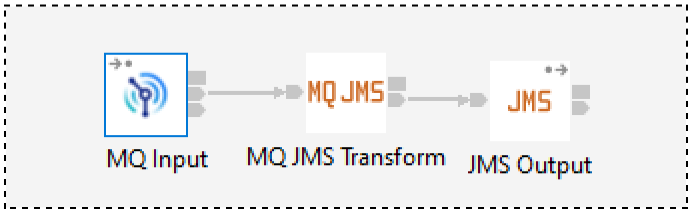
File -> New -> Message Flow



Step 4:

On the Message flow Editor in ACE Toolkit drag and drop following nodes from the toolkit.

- MQ Input Node
- MQ JMS Transform
- JMS Output



Step 5:

Connect all the nodes as above mentioned and configure their properties

MQ Input Node:

- Basic
  - Queue Name : <Queue Name on your machine>
- MQ Connection
  - Connection : Local Queue Manager
  - Destination Queue Manager Name : <Queue Manager name on you local machine>
- Input Message Parsing
  - Message Domain : XMLNSC

JMS Output Node:

- Basic
  - Destination queue : <JMS Destination Queue Name as given in jndi.properties file>
- JMS Connection
  - JMS Provider Name :<give your policy name as defined in Step 1 and 2 in a *{PolicyProjectName}:Policy Name* format>

Run The Setup

We prepared all the aspects required, now it is time to run the servers to deploy and test the above application then we can achieve a communication between IBM MQ and ActiveMQ.

ActiveMQ:

Step 1:
Unzip downloaded ActiveMQ server , copy the folder and paste in C:\Program Files folder

Step 2:
Set Environment Variable for the bin folder in an activemq folder to run these commands from anywhere on a command prompt.

Step 3:
Run activemq server by opening a command prompt as follows

```
> acitvemq.bat start
```

Step 4:
Open web console for activemq through following link
http://127.0.0.1:8161/

Step 5:
Create a Queue on Activemq with a name as given in jndi.properties file.

IBM MQ:

Start IBM MQ and create a Queue Manager and a Queue name as given on MQ Input node in the preparation for creating application as above.

IBM APP Connect Toolkit:

Deploy the Policy Project and Application.

Test the Application

Now put the message into IBM MQ Queue in a XMLNSC format
Check the Current Queue Depth on Active MQ.