Using Java Messaging Service (JMS) in Interoperability Productions

Version 2018.1
2020-02-14
# Table of Contents

About This Book .................................................................................................................................... 1

1 JMS Overview ..................................................................................................................................... 3
   1.1 JMS Messages ................................................................. 3
   1.2 InterSystems Java Gateway .............................................. 3
   1.3 Jar Files ............................................................................. 4

2 Configuring and Using JMS Business Services and Operations .................................................... 5

3 Creating Custom JMS Services and Operations Using the Adapter ............................................ 7
About This Book

This book describes how to configure and use the Java Messaging Service (JMS) to send and receive messages in interoperability productions. It contains the following chapters:

- JMS Overview
- Configuring and Using JMS Business Services and Operations
- Writing Custom JMS Services and Operations Using the Adapter

For a detailed outline, see the table of contents.
1 JMS Overview

The Java Messaging Service (JMS) is a Java messaging framework for providing communication between two or more systems. In this framework, a JMS provider manages a queue of messages sent by JMS clients. A typical JMS message has the following path:

1. A JMS client sends the message to a JMS provider.
2. The JMS provider sends the message to another JMS client.

With interoperability productions, InterSystems products can be a JMS client that both sends and receives JMS messages. InterSystems JMS clients use the EnsLib.JMS.Operation business host to send messages to JMS providers and the EnsLib.JMS.Service business host to receive messages from JMS providers. Advanced users who are familiar with ObjectScript can create their own custom JMS business hosts rather than using these built-in components.

The JMS feature is available in Ensemble 2018.1.2 and later. For the release note related to this feature, see Caché & Ensemble Maintenance Release Changes (2018.1.2).

1.1 JMS Messages

Within the JMS client’s interoperability production, the JMS messages are EnsLib.JMS.Message objects. The text property of message object contains the message content. The type property of the message object specifies the message type such as TextMessage and BytesMessage. The EnsLib.JMS.Message class also provides methods for setting and retrieving properties of the message.

1.2 InterSystems Java Gateway

InterSystems’ support for JMS messaging relies on the InterSystems Java Gateway. The easiest way to set up the Java Gateway is to add the EnsLib.JavaGateway.Service business host to the JMS client’s interoperability production. Once added, the Java Gateway automatically starts when the production is started and stops when the production is stopped. For more information about the Java Gateway, see Using the Java Gateway.
1.3 Jar Files

The jar file for the JMS feature is available at: install-dir\dev\java\lib\JDK18\cache-enslib-jms-2.0.0

The following client development jar files are also available:

- install-dir\dev\java\jms\wljmsclient.jar
- install-dir\dev\java\jms\wlthint3client.jar
InterSystems products can be configured to be a JMS client by adding built-in business hosts to an interoperability production.

To enable an InterSystems product to receive JMS messages, add a new business service to the interoperability production, defining the service class of this business service as EnsLib.JMS.Service. This business service ignores any response.

To enable an InterSystems product to send JMS messages, add a new business operation to the interoperability production, defining the operation class of this business operation as EnsLib.JMS.Operation. This business operation returns a EnsLib.JMS.Response object back to the business host that sent the JMS message to the business operation.

Once you have added these business hosts to the production, configure the following settings on the Settings tab:

- **JMSCredentials** — The credential defined for the username and password of the JMS server. For more information about creating a credential for the JMS username and password, see Defining Credentials.

- **JavaGatewayHost** and **JavaGatewayPort** — The IP address and port of the Java Gateway that your production is using to enable JMS support. If you added the EnsLib.JavaGateway.Service business host to the production, use the IP address and port under its Basic Settings.

- **JMS Server** — URL of the JMS server.

- **JMSFactory** — Name of the QueueConnectionFactory.

- **JMSQueue** — Name of the JMS Queue.

- **JMSClientID** — Name that appears on the JMS Server’s list of active connections.
Creating Custom JMS Services and Operations Using the Adapter

Creating custom JMS business services and business operations requires writing custom ObjectScript code and consequently takes more development resources than using the built-in JMS services and operations, but provides better performance as you can access the Java Gateway proxy object directly.

To develop a custom JMS business service:

1. Implement a custom BusinessService class using EnsLib.JMS.InboundAdapter as its adapter.
2. Override the OnProcessInput() method with the following signature:
   ```
   Method OnProcessInput(pMessage As %Net.Remote.Proxy, Output pOutput As %RegisteredObject) As %Status
   ``
   - `pMessage` is a Gateway proxy object of a Java message object of class `com.intersystems.enslib.jms.Message`. Properties and methods of the Java message object can be accessed using the Gateway proxy interface. The `pMessage` object contains the message received from the JMS provider.

To develop a custom JMS business operation:

1. Implement a custom BusinessOperation class using EnsLib.JMS.OutboundAdapter as its adapter.
2. Override the OnMessage() method or implement a message map. See Defining a Message Map in Developing Productions for information on message maps.
3. Call `.Adapter.GetNewMessage(tMessage)` to get the message that was sent to the business operation by another host in the production.
   - `tMessage` has the class `%Net.Remote.Proxy`.
   - `tMessage` is a Gateway proxy object of a Java message object of class `com.intersystems.enslib.jms.Message`. Properties and methods of the Java message object can be accessed using the Gateway proxy interface. Access `tMessage` with properties and methods that are implemented in Java class `com.intersystems.enslib.jms.Message`.
4. Send the message to the JMS provider by calling `.Adapter.SendMessage(tMessage)`.

Once you have developed your custom JMS business service and JMS business operation, you add them to a production just like you would the built-in JMS business hosts.