



Using Java Messaging Service (JMS) in Interoperability Productions (Legacy Implementation)

Version 2025.1
2025-06-03

Using Java Messaging Service (JMS) in Interoperability Productions (Legacy Implementation)

PDF generated on 2025-06-03

InterSystems Version 2025.1

Copyright © 2025 InterSystems Corporation

All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™, HealthShare® Health Connect Cloud™, InterSystems® Data Fabric Studio™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Congress Street, Boston, MA 02114, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

Table of Contents

1 JMS Overview (Legacy Implementation)	1
1.1 JMS Messages	1
1.2 Jar Files	1
2 Using JMS Business Services and Operations (Legacy Implementation)	3
3 Creating Custom JMS Services and Operations (Legacy Implementation)	5

1

JMS Overview (Legacy Implementation)

Important: This page discusses a legacy implementation of JMS messaging that should not be used for new implementations. This legacy implementation may be removed in future releases. Instead, use the JMSPEX interoperability adapters ([inbound](#) and [outbound](#)) which InterSystems has implemented using the [PEX](#) framework. Alternatively, use the JMS Messaging API.

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.

Internally, JMS business hosts leverage an InterSystems external server to connect to Java.

1.1 JMS Messages

Within the JMS client's interoperability production, the JMS messages are instances of `EnsLib.JMS.Message`. 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 Jar Files

The jar file for the JMS feature is available at: `install-dir\dev\java\lib\JDK18\intersystems-enslib-jms-3.0.0.jar`

The following client development jar files are also available:

- `install-dir\dev\java\jms\wljmsclient.jar`
- `install-dir\dev\java\jms\wlthint3client.jar`

2

Using JMS Business Services and Operations (Legacy Implementation)

Important: This page discusses a legacy implementation of JMS messaging that should not be used for new implementations. This legacy implementation may be removed in future releases. Instead, use the JMSPEX interoperability adapters ([inbound](#) and [outbound](#)) which InterSystems has implemented using the [PEX](#) framework. Alternatively, use the JMS Messaging API.

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

To enable an InterSystems production to send JMS messages, add a new business operation, specifying the **operation class** of this business operation as `EnsLib.JMS.Operation`. This business operation returns an instance of `EnsLib.JMS.Response` 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. See [Defining Credentials](#).
- **JavaGatewayHost** and **JavaGatewayPort**—The IP address and port of the InterSystems external server that your production is using to enable JMS support. An external server is also known as a Java Gateway. 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.

3

Creating Custom JMS Services and Operations (Legacy Implementation)

Important: This page discusses a legacy implementation of JMS messaging that should not be used for new implementations. This legacy implementation may be removed in future releases. Instead, use the JMSPEX interoperability adapters ([inbound](#) and [outbound](#)) which InterSystems has implemented using the [PEX](#) framework. Alternatively, use the JMS Messaging API.

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:

- Implement a custom business service class using `EnsLib.JMS.InboundAdapter` as its adapter.
- Override the **OnProcessInput()** method with the following signature:

```
Method OnProcessInput(pMessage As %Net.Remote.Object, 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:

- Implement a custom business operation class using `EnsLib.JMS.OutboundAdapter` as its adapter.
- Override the **OnMessage()** method or implement a message map. See [Defining a Message Map](#) for information on message maps.
- Call `..Adapter.GetNewMessage(tMessage)` to get the message that was sent to the business operation by another host in the production.
tMessage is an instance of `%Net.Remote.Object`.
- *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`.
- 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.