



Using External Messaging Platforms in Productions

Version 2025.1
2025-06-03

Using External Messaging Platforms in Productions

PDF generated on 2025-06-03

InterSystems IRIS® 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 Sending Messages to Amazon SNS from a Production	1
1.1 Using the Business Operation	1
1.2 Using the Adapter	1
1.3 See Also	2
2 Retrieving Amazon SQS Messages from a Production	3
2.1 Using the Business Service	3
2.2 Using the Adapter	3
2.3 See Also	4
3 Sending Messages to Amazon SQS from a Production	5
3.1 Using the Business Operation	5
3.2 Amazon SQS Message Class	5
3.3 Using the Adapter	6
3.4 See Also	6
4 Retrieving JMS Messages from a Production	7
4.1 Using the Business Service	7
4.2 Using the Adapter	7
4.3 See Also	8
5 Sending Messages to JMS from a Production	9
5.1 Using the Business Operation	9
5.2 JMS Message Class	9
5.3 Using the Adapter	10
5.4 See Also	11
6 Retrieving Kafka Messages from a Production	13
6.1 Using the Business Service	13
6.2 Using the Adapter	13
6.3 See Also	14
7 Sending Messages to Kafka from a Production	15
7.1 Using the Business Operation	15
7.2 Kafka Message Class	15
7.3 Using the Adapter	16
7.4 See Also	16
8 Retrieving RabbitMQ Messages from a Production	17
8.1 Using the Business Service	17
8.2 Using the Adapter	17
8.3 See Also	18
9 Sending Messages to RabbitMQ from a Production	19
9.1 Using the Business Operation	19
9.2 RabbitMQ Message Class	19
9.3 Using the Adapter	20
9.4 See Also	20
Messaging Platforms Settings Reference	21
Amazon AWS Settings	22
Amazon SNS Settings	23

Amazon SQS Settings	24
JMS Settings	25
Kafka Settings	27
RabbitMQ Settings	29

1

Sending Messages to Amazon SNS from a Production

[Amazon SNS](#) is a cloud service that delivers messages from a publisher to a subscriber. You can configure your interoperability production to be an SNS publisher by using the [built-in SNS business operation](#) or by creating a custom business operation using the [SNS outbound adapter](#). There is also an API for use outside of a production.

There are three components to a message sent to SNS: a topic, a subject, and the content of the message. Each message sent by a publisher to SNS must be associated with a specific *topic*. SNS pushes messages to subscribers who have subscribed to a particular topic. A message sent to SNS can also include a *subject*, which SNS uses as the Subject line when the message is distributed to subscribers as an email.

1.1 Using the Business Operation

InterSystems provides a built-in business operation that can be used to publish messages to SNS without needing to write custom code. To use this business operation:

1. Add `EnsLib.AmazonSNS.BusinessOperation` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [Amazon SNS Settings](#). The settings include the topic and optional subject.

If you need to be able to specify the topic and subject on a message-by-message basis, use the [adapter](#) instead.

3. Configure other business hosts to send `EnsLib.AmazonSNS.PublishRequest` requests to this business operation.

A message of type `EnsLib.AmazonSNS.PublishRequest` has one string property: `Message`.

When this business operation receives a message of type `EnsLib.AmazonSNS.PublishRequest`, it publishes that message to SNS, using the topic and subject configured for the business operation.

1.2 Using the Adapter

If the [built-in SNS business operation](#) does not meet your needs, you can directly use the SNS adapter as follows:

1. Create a [custom business operation](#) class:

- The class should extend `Ens.BusinessOperation`.
- The *ADAPTER* parameter should equal `EnsLib.AmazonSNS.OutboundAdapter`.
- The class should define a message map:

```
XData MessageMap
{
<MapItems>
  <MapItem MessageType="messageclass">
    <Method>methodname</Method>
  </MapItem>
  . . .
</MapItems>
}
```

- The class should define all the methods in the message map. These methods are known as message handlers. Each message handler should have the following signature:

Class Member

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status {
}
```

The message handlers can call the instance methods of the adapter, which is accessible as the `Adapter` property of the business operation. The general syntax for calling these methods is as follows:

ObjectScript

```
do ..Adapter.MethodName(arguments)
```

The SNS outbound adapter has one instance method, named **Publish()**, with the following signature:

Class Member

```
Method Publish(arnTopic As %String,
              message As %String,
              subject As %String) As %Status{
}
```

2. Add your business operation to your production. See [Adding Business Hosts](#).
3. Configure settings of this business host as described in [Amazon SNS Settings](#).
4. Configure other business hosts to send requests to this business operation.

Note: The SNS outbound adapter was developed using the InterSystems PEX framework, so the source code for the adapter looks different from most other adapters. For example, the adapter methods are actually wrappers for methods written in a Java PEX component.

1.3 See Also

- [Amazon SNS Settings](#)
- Using the Amazon SNS Messaging API (for use without a production)

2

Retrieving Amazon SQS Messages from a Production

An InterSystems IRIS interoperability production can be an SQS consumer. You have two options: use a [built-in business service](#) or build your own business service that uses the [SQS inbound adapter](#). There is also an API for use outside of a production.

2.1 Using the Business Service

InterSystems provides a built-in business service that can be used to retrieve messages from Amazon SQS without needing to write custom code. To use this business service:

1. Add `EnsLib.AmazonSQS.Service` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [Amazon SQS Settings](#).

The business service periodically polls for new messages. For each new message, the business service creates a message of type `EnsLib.AmazonSQS.Message` and populates that with the message contents, and then forwards the message to the configured targets within the production using asynchronous requests.

For details on this message class, see [Amazon SQS Message Class](#).

2.2 Using the Adapter

If the [built-in business service](#) does not meet your needs, you can directly use the SQS inbound adapter as follows:

1. Create a [custom business service class](#):
 - Your business service class should extend `Ens.BusinessService`.
 - In your class, the *ADAPTER* parameter should equal `EnsLib.AmazonSQS.InboundAdapter`.
 - Your class should implement the `OnProcessInput()` method:

Class Member

```
Method OnProcessInput(pInput As %Net.Remote.Object,  
                      Output pOutput As %RegisteredObject) As %Status {  
}
```

2. Add your business service to your production. See [Adding Business Hosts](#).
3. Configure settings of this business host as described in [Amazon SQS Settings](#).

2.3 See Also

- [Amazon SQS Settings](#)
- [Sending Messages to Amazon SQS from a Production](#)
- Using the Amazon SQS Messaging API (for use without a production)

3

Sending Messages to Amazon SQS from a Production

An InterSystems IRIS interoperability production can be an Amazon SQS producer. You have two options: use a [built-in business operation](#) or build your own business operation that uses the [SQS outbound adapter](#). There is also an API for use outside of a production.

3.1 Using the Business Operation

InterSystems provides a built-in business operation that can be used to publish messages to an Amazon SQS message queue without needing to write custom code. To use this business operation:

1. Add `EnsLib.AmazonSQS.BusinessOperation` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [Amazon SQS Settings](#).
3. Configure other business hosts to send `EnsLib.AmazonSQS.Message` requests to this business operation. See [Amazon SQS Message Class](#) for details.

When this business operation receives a message of type `EnsLib.AmazonSQS.Message`, it publishes that message to SQS.

3.2 Amazon SQS Message Class

The `EnsLib.AmazonSQS.Message` class has properties for defining the message, including the following:

- `queue` defines the Amazon SQS queue where the producer is sending messages.
- `body` defines the content of the message

For full descriptions of the message identifiers this class makes available through message object properties, refer to the [Amazon SQS documentation](#).

You can also use the `messageAttributes` property to specify custom metadata for your message. The `messageAttributes` property accepts a `%ListOfObjects` collection of `%External.Messaging.SQSMessageAttribute` objects. See the documentation for the Amazon SQS API for further guidance on creating SQS message attribute objects. For general information about the use of message attributes to attach custom metadata, refer to the [Amazon SQS documentation](#).

3.3 Using the Adapter

If the [SQS business operation](#) does not meet your needs, you can directly use the SQS outbound adapter as follows:

1. Create a [custom business operation](#) class:
 - The class should extend `Ens.BusinessOperation`.
 - The *ADAPTER* parameter should equal `EnsLib.AmazonSQS.OutboundAdapter`.
 - The class should define a message map:

```
XData MessageMap
{
  <MapItems>
    <MapItem MessageType="messageclass">
      <Method>methodname</Method>
    </MapItem>
    . . .
  </MapItems>
}
```

- The class should define all the methods in the message map. These methods are known as message handlers. Each message handler should have the following signature:

Class Member

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status {
}
```

The message handlers can call the instance methods of the adapter, which is accessible as the `Adapter` property of the business operation. The general syntax for calling these methods is as follows:

ObjectScript

```
do ..Adapter.MethodName(arguments)
```

The SQS outbound adapter has one instance method, named **SendMessage()**, with the following signature:

Class Member

```
Method SendMessage(msg As EnsLib.AmazonSQS.Message) As %Status{
}
```

3.4 See Also

- [Amazon SQS Settings](#)
- [Retrieving Amazon SQS Messages from a Production](#)
- [Using the Amazon SQS Messaging API \(for use without a production\)](#)

4

Retrieving JMS Messages from a Production

An InterSystems IRIS interoperability production can be a JMS consumer. You have two options: use a [built-in business service](#) or build your own business service that uses the [JMS inbound adapter](#). There is also an API for use outside of a production.

Important: This page refers to classes in the `EnsLib.JMSPEX` package, which is implemented using the [PEX](#) framework. *Do not* use classes from the legacy `EnsLib.JMS` package, which may be removed in future releases.

4.1 Using the Business Service

InterSystems provides a built-in business service that can be used to retrieve JMS messages without needing to write custom code. To use this business service:

1. Add `EnsLib.JMSPEX.Service` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [JMS Settings](#).

The business service periodically polls the configured JMS server for new messages. For each new message, the business service creates a message of type `EnsLib.JMSPEX.Message` and populates that with the message contents, and then forwards the message to the configured targets within the production using asynchronous requests.

For details on this message class, see [JMS Message Class](#).

4.2 Using the Adapter

If the [built-in business service](#) does not meet your needs, you can directly use the JMS inbound adapter as follows:

1. Create a [custom business service class](#):
 - Your business service class should extend `Ens.BusinessService`.
 - In your class, the `ADAPTER` parameter should equal `EnsLib.JMSPEX.InboundAdapter`.
 - Your class should implement the `OnProcessInput()` method:

Class Member

```
Method OnProcessInput(pInput As %Net.Remote.Object,  
                    Output pOutput As %RegisteredObject) As %Status {  
}
```

2. Add your business service to your production. See [Adding Business Hosts](#).
3. Configure settings of this business host as described in [JMS Settings](#).

4.3 See Also

- [JMS Settings](#)
- [Sending Messages to JMS from a Production](#)
- Using the JMS Messaging API (for use without a production)

5

Sending Messages to JMS from a Production

An InterSystems IRIS interoperability production can be a JMS producer. You have two options: use a [built-in business operation](#) that leverages the outbound adapter or build your own business operation that uses the [JMS outbound adapter](#). There is also an API for use outside of a production.

Important: This page refers to classes in the `EnsLib.JMSPEX` package, which is implemented using the [PEX](#) framework. *Do not* use classes from the legacy `EnsLib.JMS` package, which may be removed in future releases.

5.1 Using the Business Operation

InterSystems provides a built-in business operation that can be used to messages to a JMS queue or topic without needing to write custom code. To use this business operation:

1. Add `EnsLib.JMSPEX.BusinessOperation` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [JMS Settings](#).
3. Configure other business hosts to send `EnsLib.JMSPEX.Message` requests to this business operation. See [JMS Message Class](#) for details.

When this business operation receives a message of type `EnsLib.JMSPEX.Message`, it sends that message to JMS.

5.2 JMS Message Class

The `EnsLib.JMSPEX.Message` contains the following properties for defining the message:

- `destination` defines the JMS queue or topic where the producer is sending messages.
- `type` defines the message type ("Text" or "Bytes").
- `textBody` or `bytesBody` defines the content of the message body. Of the two, set the property corresponding to the message type.

You can also use the `properties` property to attach metadata to your message. The `properties` property accepts a `%ListOfObjects` collection of `%External.Messaging.JMSMessageProperty` objects. See the documentation about the JMS API for further guidance on creating JMS message property objects. For general information about JMS message properties, refer to the [JMS documentation](#).

5.3 Using the Adapter

If the [built-in business operation](#) does not meet your needs, you can directly use the JMS outbound adapter as follows:

1. Create a [custom business operation](#) class:

- The class should extend `Ens.BusinessOperation`.
- The `ADAPTER` parameter should equal `EnsLib.JMSPEX.OutboundAdapter`.
- The class should define a message map:

```
XData MessageMap
{
  <MapItems>
    <MapItem MessageType="messageclass">
      <Method>methodname</Method>
    </MapItem>
    . . .
  </MapItems>
}
```

- The class should define all the methods in the message map. These methods are known as message handlers. Each message handler should have the following signature:

Class Member

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status {
}
```

The message handlers can call the instance methods of the adapter, which is accessible as the `Adapter` property of the business operation. The general syntax for calling these methods is as follows:

ObjectScript

```
do ..Adapter.MethodName(arguments)
```

The JMS outbound adapter has one instance method, named **SendMessage()**, with the following signature:

Class Member

```
Method SendMessage(msg As EnsLib.JMSPEX.Message) As %Status{
}
```

2. Add your business operation to your production. See [Adding Business Hosts](#).
3. Configure settings of this business host as described in [JMS Settings](#).
4. Configure other business hosts to send requests to this business operation.

5.4 See Also

- [JMS Settings](#)
- [Retrieving JMS Messages from a Production](#)
- [Using the JMS Messaging API \(for use without a production\)](#)

6

Retrieving Kafka Messages from a Production

An InterSystems IRIS interoperability production can be a Kafka Consumer. You have two options: use a [built-in business service](#) or build your own business service that uses the [Kafka inbound adapter](#). There is also an API for use outside of a production.

6.1 Using the Business Service

InterSystems provides a built-in business service that can be used to retrieve messages from a Kafka topic without needing to write custom code. To use this business service:

1. Add `EnsLib.Kafka.Service` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [Kafka Settings](#).

The business service periodically polls the configured Kafka topic for new messages. For each new message, the business service creates a message of type `EnsLib.Kafka.Message` and populates that with the message contents, and then forwards the message to the configured targets within the production using asynchronous requests.

For details on this message class, see [Kafka Message Class](#).

6.2 Using the Adapter

If the [built-in Kafka business service](#) does not meet your needs, you can directly use the Kafka inbound adapter as follows:

1. Create a [custom business service class](#):
 - Your business service class should extend `Ens.BusinessService`.
 - In your class, the `ADAPTER` parameter should equal `EnsLib.Kafka.InboundAdapter`.
 - Your class should implement the `OnProcessInput()` method:

Class Member

```
Method OnProcessInput(pInput As %Net.Remote.Object,  
                    Output pOutput As %RegisteredObject) As %Status {  
}
```

- Add your business service to your production. See [Adding Business Hosts](#).
- Configure settings of this business host as described in [Kafka Settings](#).

6.3 See Also

- [Kafka Settings](#)
- [Sending Messages to Kafka from a Production](#)
- Using the Kafka Messaging API (for use without a production)

7

Sending Messages to Kafka from a Production

An InterSystems IRIS interoperability production can be a Kafka Producer, which sends messages to a Kafka topic. You have two options: use a [built-in business operation](#) or create a [custom business operation](#) that uses the Kafka outbound adapter. There is also an API for use outside of a production.

7.1 Using the Business Operation

InterSystems provides a built-in business operation that can be used to send messages to a Kafka topic without needing to write custom code. To use this business operation:

1. Add `EnsLib.Kafka.BusinessOperation` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [Kafka Settings](#).
3. Configure other business hosts to send `EnsLib.Kafka.Message` requests to this business operation. See [Kafka Message Class](#) for details.

When this business operation receives a message of type `EnsLib.Kafka.Message`, it sends the contents of that message to the Kafka topic specified in the message.

7.2 Kafka Message Class

The `EnsLib.Kafka.Message` message class has the following properties:

- `topic` defines the Kafka topic where the Producer is sending messages.
- `value` (a string) defines the content of the Kafka message. If `value` is set, you should not set `binaryValue`.
- `binaryValue` (a binary stream of arbitrary length) defines the content of the Kafka message when the length of the message exceeds the maximum length of a `%String`. If `binaryValue` is set, you should not set `value`.
- `key` defines an optional tag for the Kafka message.

7.3 Using the Adapter

If the [built-in Kafka business operation](#) does not meet your needs, you can directly use the Kafka outbound adapter as follows:

1. Create a [custom business operation](#) class:

- The class should extend `Ens.BusinessOperation`.
- The `ADAPTER` parameter should equal `EnsLib.Kafka.OutboundAdapter`.
- The class should define a message map:

```
XData MessageMap
{
  <MapItems>
    <MapItem MessageType="messageclass">
      <Method>methodname</Method>
    </MapItem>
    . . .
  </MapItems>
}
```

- The class should define all the methods in the message map. These methods are known as message handlers. Each message handler should have the following signature:

Class Member

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status {
}
```

The message handlers can call the instance methods of the adapter, which is accessible as the `Adapter` property of the business operation. The general syntax for calling these methods is as follows:

ObjectScript

```
do ..Adapter.MethodName(arguments)
```

The Kafka outbound adapter has one instance method, named **SendMessage()**, with the following signature:

Class Member

```
Method SendMessage(message As EnsLib.Kafka.Message) As %Status{
}
```

2. Add your business operation to your production. See [Adding Business Hosts](#).
3. Configure settings of this business host as described in [Kafka Settings](#).
4. Configure other business hosts to send requests to this business operation.

7.4 See Also

- [Kafka Settings](#)
- [Retrieving Kafka Messages from a Production](#)
- [Using the Kafka Messaging API \(for use without a production\)](#)

8

Retrieving RabbitMQ Messages from a Production

An InterSystems IRIS interoperability production can be a RabbitMQ consumer. You have two options: use a [built-in business service](#) or build your own business service that uses the [RabbitMQ inbound adapter](#). There is also an API for use outside of a production.

8.1 Using the Business Service

InterSystems provides a built-in business service that can be used to retrieve RabbitMQ messages without needing to write custom code. To use this business service:

1. Add `EnsLib.RabbitMQ.Service` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host as described in [RabbitMQ Settings](#).

The business service periodically polls the configured RabbitMQ queue for new messages. For each new message, the business service creates a message of type `EnsLib.RabbitMQ.Message` and populates that with the message contents, and then forwards the message to the configured targets within the production using asynchronous requests.

For details on this message class, see [RabbitMQ Message Class](#).

8.2 Using the Adapter

If the [built-in business service](#) does not meet your needs, you can directly use the RabbitMQ inbound adapter as follows:

1. Create a [custom business service class](#):
 - Your business service class should extend `Ens.BusinessService`.
 - In your class, the `ADAPTER` parameter should equal `EnsLib.RabbitMQ.InboundAdapter`.
 - Your class should implement the `OnProcessInput()` method:

Class Member

```
Method OnProcessInput(pInput As %Net.Remote.Object,  
                      Output pOutput As %RegisteredObject) As %Status {  
}
```

2. Add your business service to your production. See [Adding Business Hosts](#).
3. Configure settings of this business host as described in [RabbitMQ Settings](#).

8.3 See Also

- [RabbitMQ Settings](#)
- [Sending Messages to RabbitMQ from a Production](#)
- Using the RabbitMQ Messaging API (for use without a production)

9

Sending Messages to RabbitMQ from a Production

An InterSystems IRIS interoperability production can be a RabbitMQ publisher. You have two options: use a [built-in business operation](#) or build your own business operation that uses the [RabbitMQ outbound adapter](#). There is also an API for use outside of a production.

9.1 Using the Business Operation

InterSystems provides a built-in business operation that can be used to publish messages to RabbitMQ without needing to write custom code. To use this business operation:

1. Add `EnsLib.RabbitMQ.Operation` to your production. See [Adding Business Hosts](#).
2. Configure settings of this business host. See [RabbitMQ Settings](#).
3. Configure other business hosts to send `EnsLib.RabbitMQ.Message` requests to this business operation.

When this business operation receives a message of type `EnsLib.RabbitMQ.Message`, it publishes that message to RabbitMQ.

9.2 RabbitMQ Message Class

The `EnsLib.RabbitMQ.Message` class contains several properties for defining the message, including the following:

- `exchange` defines the RabbitMQ exchange where the publisher is sending messages.
- `routingKey` defines the routing key which the exchange will use to route the message.
- `deliveryMode` defines whether the message will be treated as persistent (if the value is 2) or transient (if the value is 1).
- `contentEncoding` defines the encoding of the message content (such as UTF-8).
- `encodedContent` defines the content of the message, encoded as specified by `contentEncoding`.

For full descriptions of the message properties in this class, refer to the [RabbitMQ documentation](#).

9.3 Using the Adapter

If the [built-in business operation](#) does not meet your needs, you can directly use the RabbitMQ outbound adapter as follows:

1. Create a [custom business operation](#) class:
 - The class should extend `Ens.BusinessOperation`.
 - The `ADAPTER` parameter should equal `EnsLib.RabbitMQ.OutboundAdapter`.
 - The class should define a message map:

```
XData MessageMap
{
  <MapItems>
    <MapItem MessageType="messageclass">
      <Method>methodname</Method>
    </MapItem>
    . . .
  </MapItems>
}
```

- The class should define all the methods in the message map. These methods are known as message handlers. Each message handler should have the following signature:

Class Member

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status {
}
```

The message handlers can call the instance methods of the adapter, which is accessible as the `Adapter` property of the business operation. The general syntax for calling these methods is as follows:

ObjectScript

```
do ..Adapter.MethodName(arguments)
```

The RabbitMQ outbound adapter has one instance method, named **SendMessage()**, with the following signature:

Class Member

```
Method SendMessage(msg As EnsLib.RabbitMQ.Message) As %Status{
}
```

2. Add your business operation to your production. See [Adding Business Hosts](#).
3. Configure settings of this business host as described in [RabbitMQ Settings](#).
4. Configure other business hosts to send requests to this business operation.

9.4 See Also

- [RabbitMQ Settings](#)
- [Retrieving RabbitMQ Messages from a Production](#)
- Using the RabbitMQ Messaging API (for use without a production)

Messaging Platforms Settings Reference

This section provides reference information for business hosts and adapters that provide connections to external message platforms.

Amazon AWS Settings

Provides reference information for the common Amazon AWS settings inherited by all business hosts and adapters that communicate with Amazon messaging platforms.

Summary

All the business hosts and adapters that communicate with Amazon messaging platforms have the following settings:

Group	Settings
AWS	CredentialsFile , Region

CredentialsFile

If blank, Amazon uses the [default credential provider chain](#) to obtain the credentials needed to access SNS. If you prefer to use an AWS credential file, enter its pathname.

Region

Identifies the AWS region that you want to access. For a list of regions, see [Amazon Regions, Availability Zones, and Local Zones](#).

See Also

- [Amazon SNS Settings](#)
- [Amazon SQS Settings](#)

Amazon SNS Settings

Provides reference information for the settings of `EnsLib.AmazonSNS.OutboundAdapter` and `EnsLib.AmazonSNS.BusinessOperation`.

Summary

`EnsLib.AmazonSNS.OutboundAdapter` and `EnsLib.AmazonSNS.BusinessOperation` have the following settings:

Group	Settings
AWS	CredentialsFile , Region
SNS	(Only in the business operation) ARNTopic , Subject

The remaining settings are common to all business operations. For information, see [Settings for All Business Operations](#).

ARNTopic

(Only for the business operation) Specifies the SNS topic to associate the messages with.

Subject

(Only for the business operation) Specifies an optional subject to associate the messages with.

See Also

- [Sending Messages to Amazon SNS from a Production](#)
- [Using the Amazon SNS Messaging API \(for use without a production\)](#)

Amazon SQS Settings

Provides reference information for the settings of `EnsLib.AmazonSQS.InboundAdapter`, `EnsLib.AmazonSQS.BusinessService`, `EnsLib.AmazonSQS.OutboundAdapter`, and `EnsLib.AmazonSQS.BusinessOperation`.

Summary

The Amazon SQS adapters and business hosts have the following settings:

Group	Settings
Amazon SQS	(Inbound adapter and business service only) Queue , DeleteAfterReceive , ReceiveSettings
AWS	CredentialsFile , Region

The remaining settings are common to all business hosts. For information, see [Settings for All Business Services](#) and [Settings for All Business Operations](#).

DeleteAfterReceive

Determines whether the message is deleted from the queue after the production receives it.

Queue

Defines the Amazon SQS queue from which the consumer is receiving messages.

ReceiveSettings

An optional JSON string defining settings for message retrieval. The list of available settings is the same as the list of properties of the `%External.Messaging.SQSReceiveSettings`, with each property name serving as the key; see [Using the Amazon SQS Messaging API](#).

See Also

- [Retrieving Amazon SQS Messages from a Production](#)
- [Sending Messages to Amazon SQS from a Production](#)
- [Using the Amazon SQS Messaging API \(for use without a production\)](#)

JMS Settings

Provides reference information for the settings of `EnsLib.JMSPEX.InboundAdapter`, `EnsLib.JMSPEX.Service`, `EnsLib.JMSPEX.OutboundAdapter`, and `EnsLib.JMSPEX.Operation`.

Summary

The JMS adapters and business hosts have the following settings:

Group	Settings
JMS Settings	(For the inbound adapter and business service) QueueOrTopicName , ReceiveSettings , URL , InitialContextFactoryName , ConnectionFactoryName , ClientID , Credentials (For the outbound adapter and business operation) URL , InitialContextFactoryName , ConnectionFactoryName , ClientID , Credentials
Gateway Settings	ExtraClassPaths

The remaining settings are common to all business hosts. For information, see [Settings for All Business Services](#) and [Settings for All Business Operations](#).

ClientID

Specifies a string to identify the production as a JMS client.

ConnectionFactoryName

Specifies the JMS connection factory to use when creating the connection.

Credentials

Specifies the InterSystems credentials that correspond to the username and password of a JMS client. For details on creating credentials, see [Defining Reusable Items for Use in Settings](#).

ExtraClassPaths

Specifies additional Gateway class paths delimited by "|".

InitialContextFactoryName

Specifies the Java class name that provides the initial JMS context factory.

QueueOrTopicName

(Only for the inbound adapter and business service) Specifies the JMS queue or topic from which the consumer is receiving messages.

ReceiveSettings

(Only for the inbound adapter and business service) Specifies a JSON string containing settings for message retrieval. The list of available settings is the same as the list of properties of the `%External.Messaging.JMSReceiveSettings` class, with each property name serving as the key; see [Using the JMS Messaging API](#). At a minimum, this JSON string needs to specify the subscriber property.

URL

Specifies the URL for the JMS server.

See Also

- [Retrieving JMS Messages from a Production](#)
- [Sending Messages to JMS from a Production](#)
- [Using the JMS Messaging API \(for use without a production\)](#)

Kafka Settings

Provides reference information for the settings of `EnsLib.Kafka.InboundAdapter`, `EnsLib.Kafka.Service`, `EnsLib.Kafka.OutboundAdapter`, and `EnsLib.Kafka.Operation`.

Summary

The Kafka adapters and business hosts have the following settings:

Group	Settings
Kafka Settings	(For the inbound adapter and business service) Topic , GroupID , ReceiveSettings , Servers , Credentials , SecurityProtocol , SASLMechanism , TrustStoreLocation , TrustStoreCredentials , KeyStoreLocation , KeyStoreCredentials , KeyCredentials (For the outbound adapter and business operation) ClientID , Servers , Credentials , SecurityProtocol , SASLMechanism , TrustStoreLocation , TrustStoreCredentials , KeyStoreLocation , KeyStoreCredentials , KeyCredentials
Gateway Settings	ExtraClassPaths

The remaining settings are common to all business hosts. For information, see [Settings for All Business Services](#) and [Settings for All Business Operations](#).

ClientID

(For the outbound adapter and business operation) Defines the Kafka client ID of the Producer.

Credentials

Defines the InterSystems credentials that correspond to the username and password of a Kafka client. For details on creating credentials, see [Defining Reusable Items for Use in Settings](#).

ExtraClassPaths

Additional Gateway class paths delimited by "|".

GroupID

(For the inbound adapter and business service) Defines the ID of the Consumer's consumer group.

KeyCredentials

Optionally) defines the InterSystems credentials which can be used to gain password-protected access to a private key within the keystore at the location specified by `KeyStoreLocation`

KeyStoreCredentials

Optionally defines the InterSystems credentials which can be used to gain password-protected access to the keystore at the location specified by `KeyStoreLocation`.

KeyStoreLocation

Optionally specifies the file system path to the keystore which contains the keys necessary to establish an SSL/TLS connection with your Kafka broker cluster.

ReceiveSettings

An optional JSON string defining settings for message retrieval. The list of available settings is the same as the list of properties of the `%External.Messaging.KafkaReceiveSettings`, with each property name serving as the key; see [Using the Kafka Messaging API](#).

Servers

Defines a comma-separated list of IP address:port entries that identify servers in the Kafka cluster.

SASLMechanism

Specifies the SASL authentication mechanism used to authenticate the Consumer using the credentials specified by `Credentials`. Choose one of the following:

- PLAIN
- SCRAM-SHA-256
- SCRAM-SHA-512

SecurityProtocol

Specifies the security protocol which secures connections to your Kafka broker cluster. Currently, this property supports two values:

- SASL_PLAINTEXT, which performs SASL authentication of the client over an unencrypted channel.
- SASL_SSL, which uses the truststore and keystore information you provide to establish an SSL/TLS connection over which SASL authentication takes place.

Topic

(For the inbound adapter and business service) Defines the Kafka topic from which to retrieve messages.

TrustStoreCredentials

Optionally defines the InterSystems credentials which can be used to gain password-protected access to the truststore at the location specified by `TrustStoreLocation`.

TrustStoreLocation

Optionally specifies the file system path to the truststore which contains the certificate authority certificates necessary to validate a certificate from your Kafka broker cluster and establish an SSL/TLS connection.

See Also

- [Retrieving Kafka Messages from a Production](#)
- [Sending Messages to Kafka from a Production](#)
- [Using the Kafka Messaging API \(for use without a production\)](#)

RabbitMQ Settings

Provides reference information for the settings of `EnsLib.RabbitMQ.InboundAdapter`, `EnsLib.RabbitMQ.Service`, `EnsLib.RabbitMQ.OutboundAdapter`, and `EnsLib.RabbitMQ.Operation`.

Summary

The RabbitMQ adapters and business hosts have the following settings:

Group	Settings
RabbitMQ Settings	(For the inbound adapter and business service) Queue Name , ExchangeName , BindingKeys , ReceiveSettings , MQHost , MQPort , MQVirtualHost , Credentials , EnableSSL , TLSVersion , PrivateKeyCredentials , ClientKeyFile , KeyStoreCredentials , KeyStoreFile , EnableHostnameVerification (For the outbound adapter and business operation) MQHost , MQPort , MQVirtualHost , Credentials , EnableSSL , TLSVersion , PrivateKeyCredentials , ClientKeyFile , KeyStoreCredentials , KeyStoreFile , EnableHostnameVerification
Gateway Settings	ExtraClassPaths

The remaining settings are common to all business hosts. For information, see [Settings for All Business Services](#) and [Settings for All Business Operations](#).

BindingKeys

(Only for the inbound adapter and business service) Optionally defines the keys that bind the queue your production is receiving messages from to the exchange you named.

ClientKeyFile

A string specifying the path to the client's private key file (if the server is configured to perform peer verification). See [Connecting to RabbitMQ](#).

Credentials

Defines the InterSystems credentials that correspond to the username and password of a RabbitMQ client. For details on creating credentials, see [Defining Reusable Items for Use in Settings](#).

EnableHostnameVerification

Specifies whether the peer verification process includes a verification that the hostname of the server matches the name on the server certificate. See [Connecting to RabbitMQ](#).

EnableSSL

Select this to enable TLS/SSL communication with the RabbitMQ server. See [Connecting to RabbitMQ](#).

ExchangeName

(Only for the inbound adapter and business service) Optionally defines the RabbitMQ exchange which routes messages to the queue.

Note: For information about how RabbitMQ routes messages, refer to the [RabbitMQ documentation](#).

ExtraClassPaths

Additional Gateway class paths delimited by "|".

KeyStoreCredentials

Defines the InterSystems credentials that containing a key store password for RabbitMQ. For details on creating credentials, see [Defining Reusable Items for Use in Settings](#). See [Connecting to RabbitMQ](#).

KeyStoreFile

Key store filename. See [Connecting to RabbitMQ](#).

MQHost

Defines the hostname or IP address for the RabbitMQ server.

MQPort

Defines the port number for communicating with RabbitMQ.

MQVirtualHost

Optionally defines the virtual hostname for RabbitMQ.

PrivateKeyCredentials

Credentials containing the private key password. See [Connecting to RabbitMQ](#).

Queue Name

(Only for the inbound adapter and business service) Defines the RabbitMQ queue from which the consumer is receiving messages.

ReceiveSettings

(Only for the inbound adapter and business service) An optional JSON string defining settings for message retrieval. The list of available settings is the same as the list of properties of the `%External.Messaging.RabbitMQReceiveSettings` class, with each property name serving as the key; see [Using the RabbitMQ Messaging API](#).

TLSVersion

Specify the TLS version to use. See [Connecting to RabbitMQ](#).

See Also

- [Retrieving RabbitMQ Messages from a Production](#)
- [Sending Messages to RabbitMQ from a Production](#)
- [Using the RabbitMQ Messaging API \(for use without a production\)](#)