Using IBM WebSphere MQ Adapters with Ensemble

Version 2018.1
2018-12-14

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using IBM WebSphere MQ Adapters with Ensemble
Ensemble Version 2018.1 2018-12-14

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

About This Book

This book describes how an Ensemble programmer can add IBM WebSphere MQ adapters to an Ensemble production, so that the production can send messages to and retrieve messages from IBM WebSphere MQ. It does not contain details on the IBM WebSphere MQ product.

This book contains the following sections:

- About the Ensemble IBM WebSphere MQ Adapters
- Using the IBM WebSphere MQ Inbound Adapter
- Using the IBM WebSphere MQ Outbound Adapter
- Troubleshooting
- Reference for Settings

For a detailed outline, see the table of contents.

For more information, try the following sources:

- *Ensemble Best Practices* describes best practices for organizing and developing Ensemble productions.
- *Developing Ensemble Productions* explains how to perform the development tasks related to creating an Ensemble production.
- *Configuring Ensemble Productions* describes how to configure the settings for Ensemble productions, business hosts, and adapters. It provides details on settings not discussed in this book.

For general information, see the *InterSystems Documentation Guide*.

1

About the Ensemble IBM WebSphere MQ Adapters

The Ensemble IBM WebSphere MQ inbound and outbound adapters enable your Ensemble productions to retrieve messages from and send messages to message queues of Ensemble IBM WebSphere MQ. This chapter provides a brief introduction to these adapters.

It is assumed that you are reasonably familiar with the IBM WebSphere MQ product and have access to its formal documentation.

1.1 Prerequisites

In order to use Ensemble IBM WebSphere MQ inbound and outbound adapters, make sure that you have access to IBM WebSphere MQ as described in the chapter “Sending and Receiving IBM WebSphere MQ Messages” in *Using Caché Internet Utilities*.

The adapters use a dynamic-link library that is automatically installed by Caché on all suitable platforms. (This is MQInterface.dll on Windows; the file extension is different for other platforms.) In turn, the Caché dynamic-link library requires IBM WebSphere MQ dynamic-link libraries (the IBM WebSphere MQ client).

1.2 The IBM WebSphere MQ Inbound Adapter

In general, the inbound IBM WebSphere MQ adapter (EnsLib.MQSeries.InboundAdapter) accesses a given queue manager, using a given channel, retrieves messages from a given queue, and sends those messages (as instances of EnsLib.MQSeries.Message) to the associated business service.

The adapter provides runtime settings that you use to specify items like the following:

- The queue manager to use.
- The channel specification to use. This specification includes the name of the channel to use, the transport used by the channel, the server name (or IP address) that is running the IBM WebSphere MQ server, and the port.
- The message queue to check.
- A polling interval, which controls how frequently the adapter checks for new input.

Note: Each retrieved message must be less than 32 KB in length unless long strings are enabled in Ensemble.

1.3 The IBM WebSphere MQ Outbound Adapter

The IBM WebSphere MQ outbound adapter (EnsLib.MQSeries.OutboundAdapter) provides settings that you use to specify the following:

- The queue manager to use.
- The channel specification to use. This specification includes the name of the channel to use, the transport used by the channel, the server name (or IP address) that is running the IBM WebSphere MQ server, and the port.
- The message queue to send messages to.

It provides a method to send a message to the given queue.

Note: The message must be less than 32 KB in length unless long strings are enabled in Ensemble.

1.4 The MQ Adapters and the Event Log

Unlike many other Ensemble adapters, the IBM WebSphere MQ inbound and outbound adapters do not write to the Event Log when they process messages sent to or received from outside Ensemble. To see the inbound and outbound traffic, you can, however, use the message browser as usual.

2

Using the IBM WebSphere MQ Inbound Adapter

This chapter describes the default behavior of the Ensemble IBM WebSphere MQ inbound adapter (EnsLib.MQSeries.InboundAdapter) and describes how to use this adapter in your productions. It discusses the following topics:

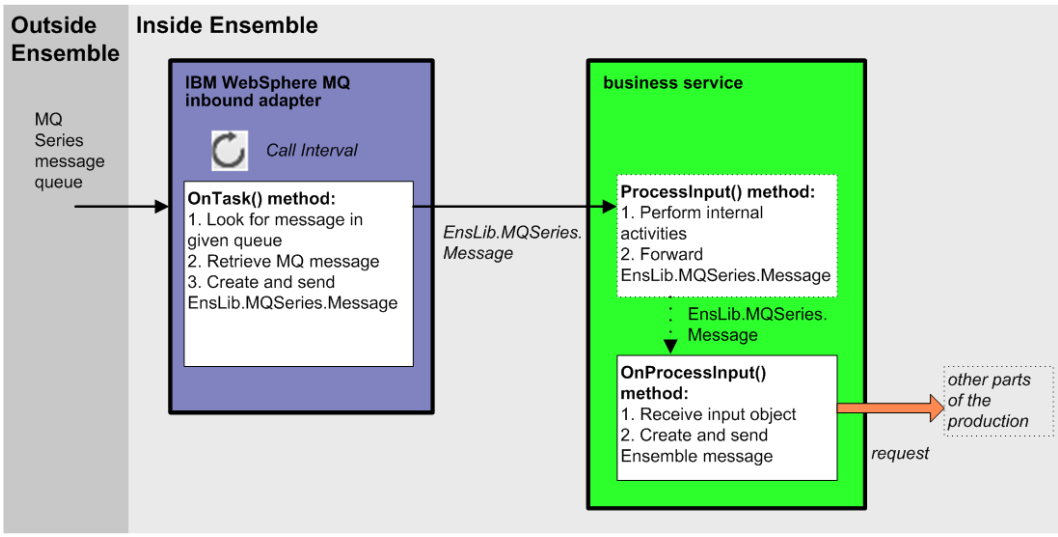
- Overall behavior
- How to create the business service class
- Details on how to implement the OnProcessInput() method
- Details on the helper class that contains the message
- An example.
- How to add and configure the business service

2.1 Overall Behavior

First, it is useful to understand the details that you specify for the adapter. The EnsLib.MQSeries.InboundAdapter class provides runtime settings that you use to specify items like the following:

- The queue manager to use.
- The channel specification to use. This specification includes the name of the channel to use, the transport used by the channel, the server name (or IP address) that is running the IBM WebSphere MQ server, and the port.
- The message queue to check.
- A polling interval, which controls how frequently the adapter checks for new input.

In general, the inbound IBM WebSphere MQ adapter (EnsLib.MQSeries.InboundAdapter) periodically checks the given queue, retrieves any messages, and sends the messages (as instances of EnsLib.MQSeries.Message) to the associated business service. The business service, which you create and configure, uses these messages and communicates with the rest of the production. The following figure shows the overall flow:



More specifically:

1. When the adapter is initialized, it connects to the given queue manager and queue, using the given channel.
2. The adapter regularly executes its **OnTask()** method, which retrieves a message from the queue (if a message is available). The polling interval is determined by the CallInterval setting.
3. When the adapter retrieves a message, it does the following:
   a. The adapter creates an instance of the EnsLib.MQSeries.Message class and puts the message data into it.
   b. The adapter calls the internal ProcessInput method of the associated business service class, passing the EnsLib.MQSeries.Message instance as input.

   Note: The message must be less than 32 KB in length unless long strings are enabled in Ensemble.
4. The internal ProcessInput() method of the business service class executes. This method performs basic Ensemble tasks such as maintaining internal information as needed by all business services. You do not customize or override this method, which your business service class inherits.
5. The ProcessInput() method then calls your custom OnProcessInput() method, passing the EnsLib.MQSeries.Message instance as input. The requirements for this method are described later in "Implementing the OnProcessInput Method."

The response message follows the same path, in reverse.

2.2 Creating a Business Service to Use the Inbound Adapter

To use this adapter in your production, create a new business service class as described here. Later, add it to your production and configure it. You must also create appropriate message classes, if none yet exist. See "Defining Ensemble Messages" in *Developing Ensemble Productions*.

The following list describes the basic requirements of the business service class:

- Your business service class should extend Ens.BusinessService.
- In your class, the *ADAPTER* parameter should equal EnsLib.MQSeries.InboundAdapter.

- Your class should implement the OnProcessInput() method, as described in "Implementing the OnProcessInput Method."
- For other options and general information, see "Defining a Business Service Class" in *Developing Ensemble Productions*.

The following example shows the general structure that you need:

```
Class EMQS.Service Extends Ens.BusinessService
{
Parameter ADAPTER = "EnsLib.MQSeries.InboundAdapter";

Method OnProcessInput(pInput As EnsLib.MQSeries.Message,
                      pOutput As %RegisteredObject) As %Status
{
   set tsc=$$$OK
   //your code here
   Quit tsc
}
}
```

Note: Studio provides a wizard that you can use to create a business service stub similar to the preceding. To access this wizard, click **File —> New** and then click the **Production** tab. Then click **Business Service** and click **OK**. Note that the wizard provides a generic input argument. If you use the wizard, InterSystems recommends that you edit the method signature to use the specific input argument needed with this adapter; the input argument type should be EnsLib.MQSeries.Message.

2.3 Implementing the OnProcessInput() Method

Within your custom business service class, your **OnProcessInput()** method should have the following signature:

```
Method OnProcessInput(pInput As EnsLib.MQSeries.Message,
                      pOutput As %RegisteredObject) As %Status
```

Here *pInput* is the message object that the adapter will send to this business service; this is an instance of EnsLib.MQSeries.Message. Also, *pOutput* is the generic output argument required in the method signature.

The **OnProcessInput()** method should do some or all of the following:

1. Examine the MQ message (EnsLib.MQSeries.Message) and decide how to use it. See "Using the Received Message," later in this chapter.
2. Create an instance of the request message, which will be the message that your business service sends.

   For information on creating message classes, see "Defining Ensemble Messages" in *Developing Ensemble Productions*.
3. For the request message, set its properties as appropriate, using values in the MQ message.
4. Call a suitable method of the business service to send the request to some destination within the production. Specifically, call **SendRequestSync()**, **SendRequestAsync()**, or (less common) **SendDeferredResponse()**. For details, see "Sending Request Messages" in *Developing Ensemble Productions*

   Each of these methods returns a status (specifically, an instance of %Status).
5. Make sure that you set the output argument (`pOutput`). Typically you set this equal to the response message that you have received. This step is required.
6. Return an appropriate status. This step is required.

See the example at the end of this chapter.

2.4 Using the Received Message

The adapter sends an instance of EnsLib.MQSeries.Message to your business service. This object contains the message retrieved from the queue. It has three properties:

- Body — The body of the message.
- MessageID — The message ID.
- BodySize — An integer that indicates the length of the message.

Note: The message retrieved from the queue must be less than 32 KB in length unless long strings are enabled in Ensemble.

2.5 Example

The following business service retrieves messages and forwards them to a business host named `EMQS.MessageProcessor`.

```
Class EMQS.Service Extends Ens.BusinessService
{

Parameter ADAPTER = "EnsLib.MQSeries.InboundAdapter";

Method OnProcessInput(pInput As EnsLib.MQSeries.Message,
pOutput As EMQS.InboundMsg) As %Status
{
    //create Ensemble message to carry the retrieved MQ message
    Set inbound=##class(EMQS.InboundMsg).%New()
    Set inbound.Body=pInput.Body
    Set inbound.MessageId=pInput.MessageId

    //forward this to the message processor
    Set tsc=..SendRequestSync("EMQS.MessageProcessor",inbound,.response)
    Set pOutout=response
    Quit tsc
}

}
```

The message class `EMQS.InboundMsg` is as follows:

```
Class EMQS.InboundMsg Extends Ens.Request
{

Property Body As %String (MAXLEN="");

Property MessageId As %String(MAXLEN = 128);

}
```

2.6 Adding and Configuring the Business Service

To add your business service to an Ensemble production, use the Management Portal to do the following:

1. Add an instance of your business service class to the Ensemble production.
2. Enable the business service.
3. Configure the adapter to access an IBM WebSphere MQ queue and retrieve messages. Specifically:

- Specify the queue manager, channel, and queue to access. The channel specification includes the name of the channel to use, the transport used by the channel, the server name (or IP address) that is running the IBM WebSphere MQ server, and the port.
- Specify the error log to use.
- Specify the character set to convert the messages to.
- Specify how often the adapter looks for messages.

These topics are discussed in the section "Settings for the IBM WebSphere MQ Adapters."

4. Run the production.

3

Using the IBM WebSphere MQ Outbound Adapter

This chapter describes the behavior of the Ensemble IBM WebSphere MQ outbound adapter (EnsLib.MQSeries.OutboundAdapter) and describes how to use this adapter in your productions. It discusses the following topics:

- Overall behavior
- How to create the business operation class
- Details on how to create methods within that business operation
- An example of a business operation class
- How to add and configure the business operation

3.1 Overall Behavior

Within a production, an outbound adapter is associated with a business operation that you create and configure. The business operation receives a message from within the production, looks up the message type, and executes the appropriate method. This method usually executes methods of the associated adapter.

The IBM WebSphere MQ outbound adapter (EnsLib.MQSeries.OutboundAdapter) provides settings that you use to specify the following:

3.2 Creating a Business Operation to Use the Adapter

To create a business operation to use the EnsLib.MQSeries.OutBoundAdapter, you create a new business operation class. Later, add it to your production and configure it.

You must also create appropriate message classes, if none yet exist. See "Defining Ensemble Messages" in *Developing Ensemble Productions*.

The following list describes the basic requirements of the business operation class:

- Your business operation class should extend Ens.BusinessOperation.

- In your class, the *ADAPTER* parameter should equal EnsLib.MQSeries.OutboundAdapter.
- In your class, the *INVOCATION* parameter should specify the invocation style you want to use, which must be one of the following.
  - **Queue** means the message is created within one background job and placed on a queue, at which time the original job is released. Later, when the message is processed, a different background job will be allocated for the task. This is the most common setting.
  - **InProc** means the message will be formulated, sent, and delivered in the same job in which it was created. The job will not be released to the sender's pool until the message is delivered to the target. This is only suitable for special cases.
- Your class should define a *message map* that includes at least one entry. A message map is an XData block entry that has the following structure:

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

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

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

  Here *Sample* is the name of the method, *RequestClass* is the name of an Ensemble request message class, and *ResponseClass* is the name of an Ensemble response message class. In general, these methods will refer to properties and methods of the Adapter property of your business operation.
- For other options and general information, see "Defining a Business Operation Class" in *Developing Ensemble Productions*.

The following example shows the general structure that you need:

```
Class EMQS.NewOperation1 Extends Ens.BusinessOperation
{
Parameter ADAPTER = "EnsLib.MQSeries.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method SampleCall(pRequest As Ens.Request,
                  Output pResponse As Ens.Response) As %Status
{
  Quit $$$ERROR($$$NotImplemented)
}

XData MessageMap
{
<MapItems>
  <MapItem MessageType="Ens.Request">
    <Method>SampleCall</Method>
  </MapItem>
</MapItems>
}
}
```

Note: Studio provides a wizard that you can use to create a business operation stub similar to the preceding. To access this wizard, click **File —> New** and then click the **Production** tab. Then click **Business Operation** and click **OK**.

3.3 Creating Message Handler Methods

When you create a business operation class for use with EnsLib.MQSeries.OutboundAdapter, typically your main task is writing message handlers for use with this adapter, that is, methods that receive Ensemble messages and then send messages to an IBM WebSphere MQ server.

Each message handler method should have the following signature:

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

Here *Sample* is the name of the method, *RequestClass* is the name of an Ensemble request message class, and *ResponseClass* is the name of an Ensemble response message class.

In general, the method should do the following:

1. Examine the inbound request message.
2. Call the **SendMessage()** method of the adapter to send a message to the configured queue of IBM WebSphere MQ.

   Note: The message must be less than 32 KB in length unless long strings are enabled in Ensemble.

3. Examine the response.
4. Use information in the response to create an Ensemble response message (an instance of Ens.Response or a subclass), which the method returns as output.

   For information on defining message classes, see "Defining Ensemble Messages" in *Developing Ensemble Productions*.

5. Make sure that you set the output argument (`pOutput`). Typically you set this equal to the response message. This step is required.
6. Return an appropriate status. This step is required.

3.3.1 Available Methods

The adapter provides the following method:

SendMessage()

```
Method SendMessage(pBody) As %Status
```

Sends a message to the configured queue of IBM WebSphere MQ message. Note that *pBody* can be either a datatype or a character stream, but cannot be binary data or an object.

3.4 Example

The following business operation sends a message to IBM WebSphere MQ:

```
Class EMQS.Operation Extends Ens.BusinessOperation
{

Parameter ADAPTER = "EnsLib.MQSeries.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method Send(pRequest As OutboundMsg, Output pResponse As OutboundMsgResponse) As %Status
{
    Set string=pRequest.Body
```

```
    //Get part of the message so that we can provide
    //some information in the response message
    Set snippet=$Extract(string,1,50)

    //send the message to the configured queue
    Set status=..Adapter.SendMessage(string)
    If $$$ISERR(status) {
        Do $System.Status.DisplayError(status)
        Quit $$$ERROR($$$GeneralError,"Error sending message")
    }

    //create the response message
    Set pResponse=##class(EMQS.OutboundMsgResponse).%New()
    Set pResponse.Body="Message sent: "_snippet

    Quit status
}

XData MessageMap
{
<MapItems>
    <MapItem MessageType="EMQS.OutboundMsg">
        <Method>Send</Method>
    </MapItem>
</MapItems>
}

}
```

3.5 Adding and Configuring the Business Operation

To add your business operation to an Ensemble production, use the Management Portal to do the following:

1. Add an instance of your custom business operation class to the Ensemble production.
2. Enable the business operation.
3. Configure the adapter to access an IBM WebSphere MQ queue and send messages. Specifically:
   - Specify the queue manager, channel, and queue to access. The channel specification includes the name of the channel to use, the transport used by the channel, the server name (or IP address) that is running the IBM WebSphere MQ server, and the port.
   - Specify the error log to use.
   - Indicate the character set that the messages are in.

   These topics are discussed in the section "Settings for the IBM WebSphere MQ Adapters."
4. Run the production.

4

Troubleshooting

If you encounter problems when using the Ensemble adapters to IBM WebSphere MQ, you should first determine whether the client is correctly installed and can communicate with the server. To perform such a test, you can use sample programs that are provided by IBM WebSphere MQ. The executables are in the bin directory of the IBM WebSphere MQ client.

The following steps describe how to use these sample programs on Windows. The details may be different on other operating systems; consult the IBM documentation and check the names of the files present in your client.

1. Create an environment variable called *MQSERVER*. Its value should be of the form *channel_name/transport/server*, where *channel_name* is the name of the channel to use, *transport* is a string that indicates the transport to use, and *server* is the name of the server. For example: `S_antigua/TCP/antigua`
2. At the command line, enter the following command:

   ```
   amqsputc queue_name queue_manager_name
   ```

   where *queue_name* is the name of the queue to use and *queue_manager_name* is the name of the queue manager. For example:

   ```
   amqsputc cachetest QM_antigua
   ```

   If the `amqsputc` command is unrecognized, make sure that the *PATH* environment variable has been updated to include the bin directory of the IBM WebSphere MQ client.

   In case of other errors, consult the IBM documentation.
3. You should see a couple of lines like the following:

   ```
   Sample AMQSPUT0 start
   target queue is cachetest
   ```

4. Now you can send messages. Simply type each message and press Enter after each message. For example:

   ```
   sample message 1
   sample message 2
   ```

5. When you are done sending messages, press Enter twice. You will then see a line like the following:

   ```
   Sample AMQSPUT0 end
   ```

6. To complete this test, we will retrieve the messages you sent to the queue. Type the following command at the command line:

   ```
   amqsgetc queue_name queue_manager_name
   ```

   where *queue_name* is the name of the queue to use and *queue_manager_name* is the name of the queue manager. For example:

7. You should then see a start line, followed by the messages that you sent previously, as follows:

```
Sample AMQSGET0 start
message <sample message 1>
message <sample message 2>
```

8. This sample program waits briefly to receive any other messages and then displays the following:

```
no more messages
Sample AMQSGET0 end
```

If the test fails, consult the IBM documentation. Possible causes of problems include the following:

- Security issues
- Queue is not defined correctly
- Queue manager is not started

Reference for Settings

This section provides the following reference information:

- Settings for the IBM WebSphere MQ Adapters

Also see "Settings in All Productions" in *Managing Ensemble Productions*.

Settings for the IBM WebSphere MQ Adapters

Provides reference information for settings of the IBM WebSphere MQ adapters, EnsLib.MQSeries.InboundAdapter and EnsLib.MQSeries.OutboundAdapter.

Introduction

For both the inbound and outbound adapters to IBM WebSphere MQ, you specify settings that describe the connection to a given queue. The details are nearly identical for both adapters.

Also see "Prerequisites" in the first chapter of this book.

Specifying the Queue Manager, Channel, and Queue

Specify values for the following settings in order to specify the message queue of interest, the queue manager to use, and the channel to use:

QueueName

(Required) Specifies the queue name; this should be a valid queue for the specified queue manager. Also, you must have permission to use this queue.

QueueManager

Specifies the queue manager; this should be a valid queue manager on the IBM WebSphere MQ server and you must have permission to use it.

If you omit this setting, the system uses the default queue manager, as configured in IBM WebSphere MQ. Or, if IBM WebSphere MQ has been the configured so that the queue manager is determined by the queue name, the system uses the queue manager that is appropriate for the given queue name.

Channel

The specification for the channel, in the following form:

```
"channel_name/transport/host_name(port)"
```

Here *channel_name* is the name of the channel to use, *transport* is the transport used by the channel, *host_name* is the server name (or IP address) that is running the IBM WebSphere MQ server, and *port* is the port that this channel should use.

Transport can be one of the following: `TCP`, `LU62`, `NETBIOS`, `SPX`

For example:

```
"CHAN_1/TCP/rodan(1401)"
```

```
"CHAN_1/TCP/127.0.0.1(1401)"
```

If you omit this setting, the system uses the default channel specification, as configured in IBM WebSphere MQ. Or, if the system has been the configured so that the channel is determined by the queue name, the system uses the channel that is appropriate for the given queue name.

Specifying the Log File to Use

Use the following setting to specify the log file to write to.

ErrorFile

Specifies the log file to write error messages to. If you omit this setting, no logging occurs.

Specifying the Character Set of the Messages

You can specify the character set to use for message conversion.

CharSet

Specifies an integer Coded Character Set ID (CCSID) as used in IBM WebSphere MQ.

- For the inbound adapter, this is the character set to convert the messages to. If you do not specify a character set, the MQ system assumes the messages use the default character set specified for the MQ client.
- For the outbound adapter, this setting specifies the character set that the messages are already in; no conversion is done.

For background information on character translation in Caché, see "Localization Support" in the *Caché Programming Orientation Guide*.

Specifying How Often to Check for Messages

This applies only to the inbound adapter. To specify how often to check for messages, use the following setting:

CallInterval

Number of seconds that the adapter will wait before checking again for new messages, before checking for a shutdown signal from the Ensemble framework.

If the adapter finds input, it acquires the data and passes it to the business service. The business service processes the data, and then the adapter immediately begins waiting for new input. This cycle continues whenever the production is running and the business service is enabled and scheduled to be active.

The default CallInterval is 5 seconds. The minimum is 0.1 seconds.

For any settings not listed here, see *Configuring Ensemble Productions*.

Example

The examples in this book used an IBM WebSphere MQ server running on a machine named `antigua`, using the port 1414. The user ID under which the examples were run was authorized to use the queue manager `QM_antigua` and a queue named `cachetest`. Ensemble used a TCP channel named `S_antigua` to communicate with the server. The following table lists the settings:

Setting	Value
Queue Manager	`QM_antigua`
Channel	`S_antigua/TCP/antigua(1414)`
Queue Name	`cachetest`
Error File	`c:\mq-log.txt`