Configure Message Bus Activities

Last Modified on 03/30/2026 11:56 am EDT

Starting with V11.1, Cora Orchestration has been renamed to Orchestration AI.

V11.1

Overview

The integrating messaging mechanism integrates your workflow with other workflows or applications or services. This mechanism integrates Cora Orchestration with the default internal queue to connect and send information between applications and services.

The integrating messaging mechanism enables the system to start or resume processes. For example, you can set up a message bus activity for an inter-bank transfer that requires customer's approval based on specific conditions. The message bus sends a message to a queue and Cora Orchestration creates a process for the bank's representative to receive the customer's approval in writing.

NOTE
SQL Server Service Broker is the default service. Starting with V9.8, Cora SeQuence supports the Apache ActiveMQ message broker as well.

Watch this video for a demo of the ApacheMQ integration and required configuration.

With the new integrating messaging mechanism, the system can trigger multiple processes with one message from the queue.

You can configure the integrating messaging mechanism, under the `sequence.engine` node in the web.config file.

```
<messageBus>
    <connections defaultConnectionName="ActiveMQ">
      <!-- <add name="ActiveMQ" type="PNMsoft.Sequence.MessageBus.ActiveMQ.ClientFactory, PNMsoft.Sequence
.MessageBus.ActiveMQ" connectionString="Server=failover:(tcp://40.115.15.69:61616)?initialReconnectDelay=100"
/> -->
      <add name="ActiveMQ" type="PNMsoft.Sequence.MessageBus.ActiveMQ.ClientFactory, PNMsoft.Sequence.Mes
sageBus.ActiveMQ" credentials="dOtg/dWG6Q1TJDAvqHLX6Rvm2dPxGlL+bMx4RBxmA6NFL53cIrvqe5IR7+3VQjTg7B
jnhYQAV+Zup9g4XjjurQ==" connectionString="Server=tcp://localhost:61616?initialReconnectDelay=100" />
      <add name="SqlServiceBroker" type="PNMsoft.Sequence.MessageBus.SqlServiceBroker.ClientFactory, PNMsoft
.Sequence.MessageBus.SqlServiceBroker" />
    </connections>
    <serializers defaultSerializerName="Newtonsoft.Json">
      <add name="Newtonsoft.Json" type="PNMsoft.Sequence.MessageBus.Serializers.NewtonsoftJson.MessageSerial
izerFactory, PNMsoft.Sequence.MessageBus.Serializers.NewtonsoftJson" />
    </serializers>
  </messageBus>
```

Or

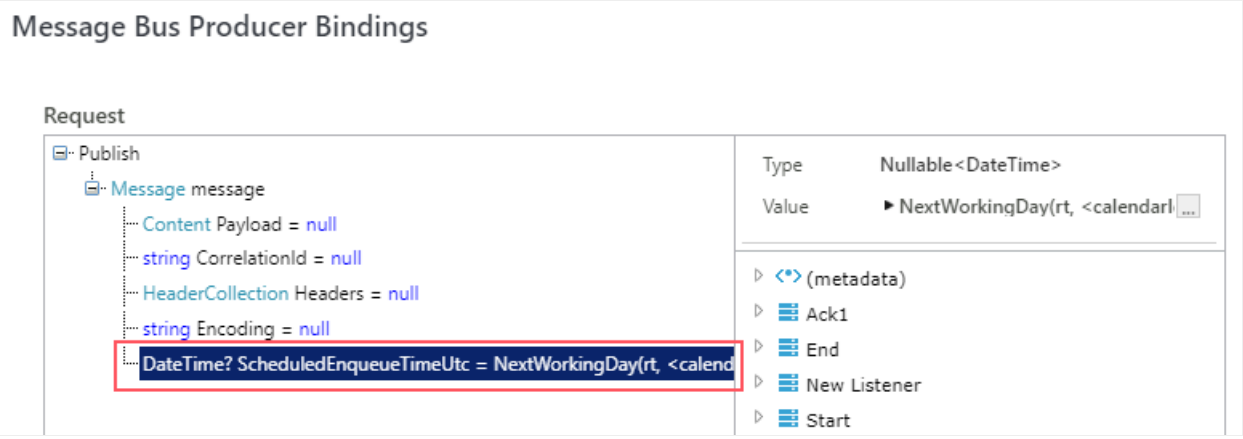
Use the PowerShell Function: Set-CoraSeQuenceApplicationConfiguration with the following parameters and values:

```
Set-CoraSeQuenceApplicationConfiguration -ApplicationType Administration -ConfigurationName MessageBusActiveMQ -TokensValues @{"SEQ_ActiveMQConnectionString" = "Server=failover:(tcp://192.168.40.4:61616);Username=admin;Password=admin"} -Verbose
```

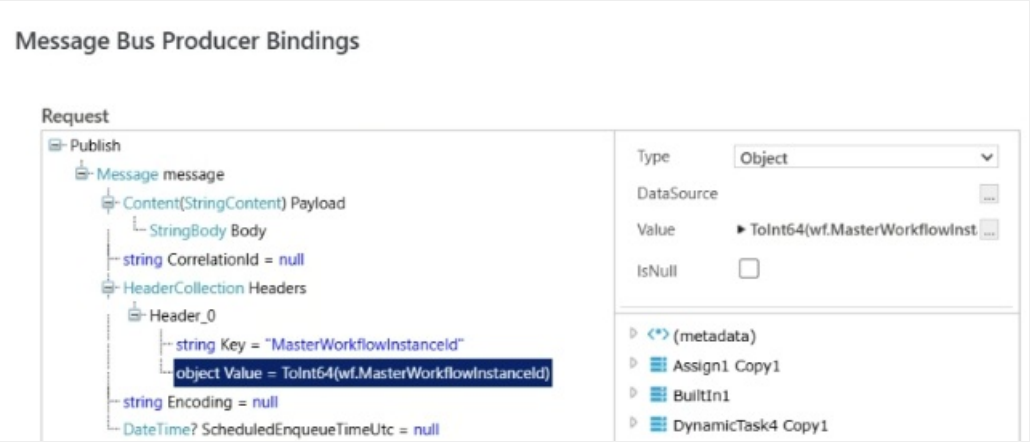
The integrating messaging mechanism uses two activities to listen to messages and start or resume a workflow and to publish the messages to the queue .

- Message Bus Producer: to send messages to the internal queue from within workflows.
- Message Bus Listener: a JES job to get messages from the internal queue from within workflows and acknowledge the message receival.

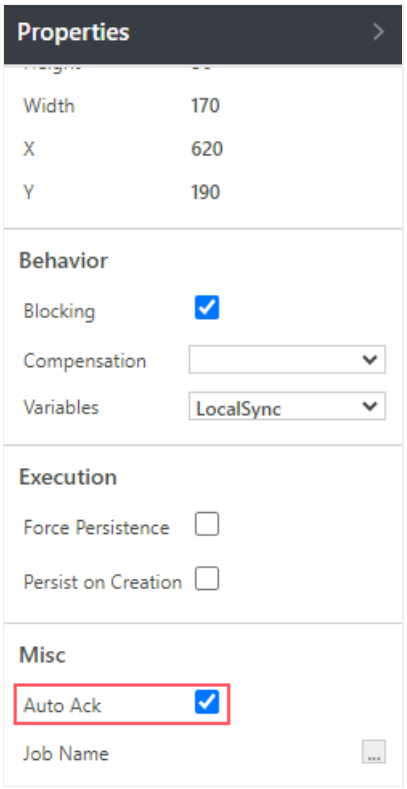
You can control and schedule the availability of messages in the internal queue by setting the `ScheduledEnqueueTimeUtc` parameter in the Message Bus Producer activity bindings.



You can start workflow instances using the Message Bus Listener from a specified master workflow instance, so that the new workflow instance logic can access and utilize data from the master instance's scope.



In the Message Bus Listener activity, when a new **Auto Ack** acknowledgment property is set to True, an acknowledgment is sent from the workflow that the message has been received from the queue. This acknowledgment is sent when the workflow is executed.



However, as a workflow developer you can implement a workflow that acknowledges a message receival at different points in the workflow with the following message bus activity.

- Message Bus Ack: to delay acknowledgment of the message receival from the queue.

The concept of the new Ack activity is similar to the concept of the HTTP Input/Output activities. The Message Bus Ack activity works along with the Message Bus Listener activity that exists in the same workflow template, and is executed before the related Message Bus Ack activity.

To acknowledge a message, you can use either Message Bus Listener activity's Auto Ack property, or the Message Bus Ack activity. For successful results:

- If the Listener activity's "Auto Ack" property is TRUE, then skip the execution of Ack activity.
- If the Listener activity's "Auto Ack" property is FALSE, then mark the message that was received by the listener as candidate for acknowledgement. The actual acknowledgement will be done by the job after the workflow execution.

NOTE
The Message Bus Ack activity is relevant only for the JES engine and not to the BRS, Portal, Admin, and WebAPI.
The SQL service broker *does not* support acknowledge.

In the Message Bus Listener, you can access the message properties of Azure Service Bus to implement appropriate error handling logic based on retry attempts.

Use the following expression to access the properties:

```
JsonValue({Listener}.messagePropertiesJson).<property>
```

Example:

```
JsonValue({Listener1}.Message.Headers["__KnownProperties"].ToString()).DeliveryCount
```

The Integrating messaging can currently work with two types of entities:

- **Queues**: one direction communication — one sender to one receiver. The message delivery follows a first in, first out (FIFO) method. That is, messages are received and processed in the same order in which they enter the queue. The message bus processes only one message at a time. We support the following queue types:
  - Azure Service Bus
  - Active MQ
- **Topics**: one direction communication — one sender to several receivers, through topics. Messages are sent to a topic and delivered to one or more subscribers. The receiver does not communicate with the topic. Use topics if your activity requires sending messages to several receivers.

Messaging profiles

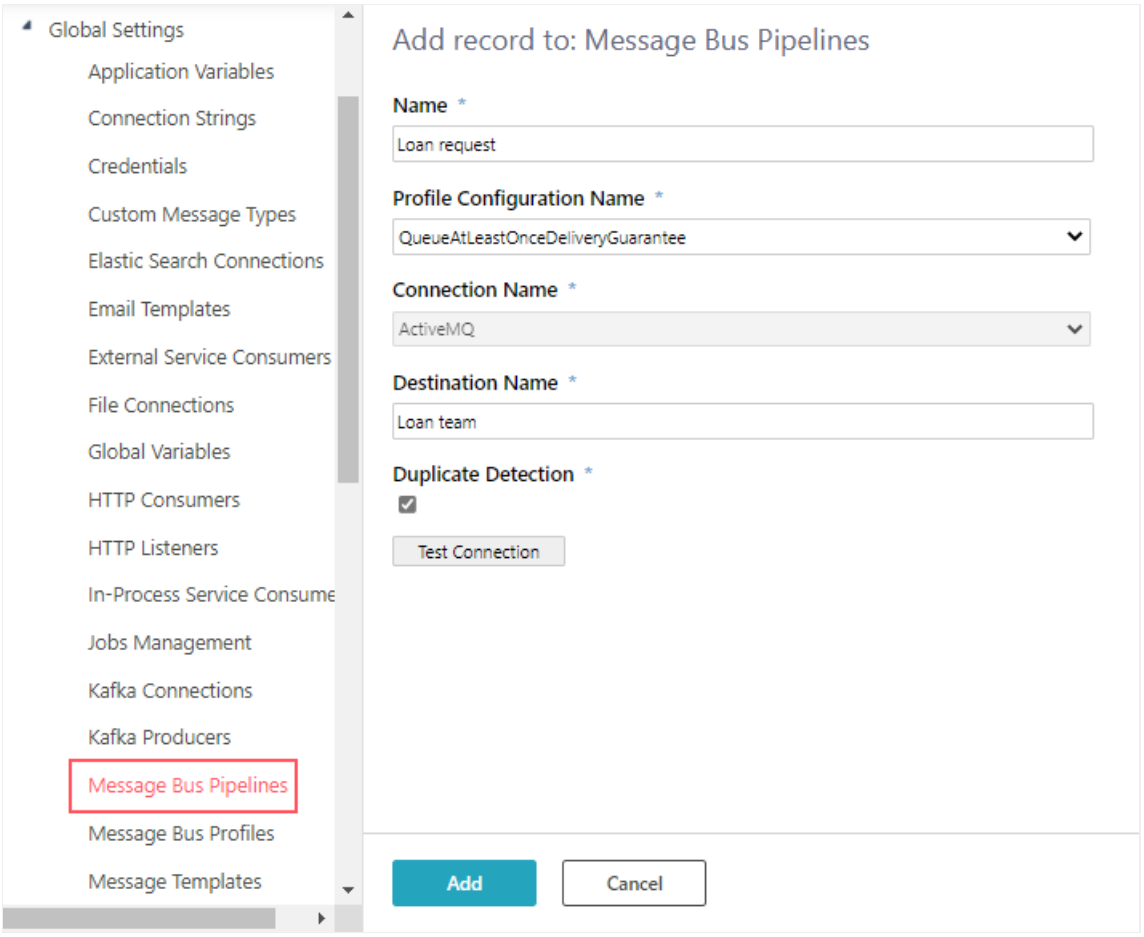
There are four out-of-the-box messaging profiles available to be mapped to the pipelines. These profiles define how messages will be sent in a queue or a topic.

- **Queue - At Least Once Delivery Guarantee**: using a queue mechanism for a one-to-one communication with a guarantee that the message delivers at least once. This option might cause a message to be delivered multiple times in order to guarantee the delivery.
- **Queue - At Most Once Delivery Guarantee**: using a queue mechanism for a one-to-one communication with a guarantee that a message will not be delivered more than once. This option might prevent a message from being delivered at all.
- **Topic - At Least Once Delivery Guarantee**: using a topic mechanism for a one-to-many communication with a guarantee that the message delivers at least once. This option might cause a message to be delivered multiple times in order to guarantee its delivery.
- **Topic - At Most Once Delivery Guarantee**: using a topic mechanism for a one-to-many communication with a guarantee that a message will not be delivered more than one time. This option might cause a message to be not delivered at all.

Define message bus pipelines

As an admin, you can define pipelines for internal queues.

1. In the Administration site, navigate to **Administration**>**Global Settings**>**Message Bus Pipelines**.
2. Click **Add**.
3. Add the following details:



- Name: Pipeline name.
- Profile Configuration Name: type of profile to be mapped to this pipeline.
- Connection Name: ActiveMQ, by default.
- Destination Name: The topic or queue where messages will be delivered.
- Duplicate Detection: Select the check box to enable detecting duplicate message based on the correlationId.
- MaxAutoLockRenewalDuration (seconds): You can configure additional time to extend the time out for document processing tasks.
- Test Connection: To test the connection.

4. Click **Add**.

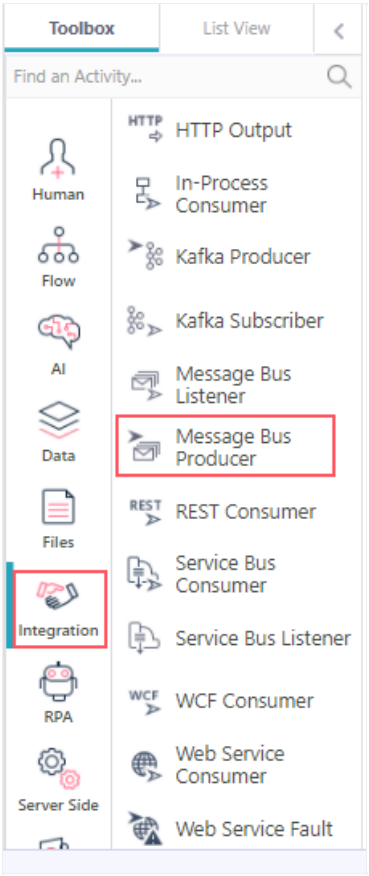
Configure the Message Bus Producer activity

Prerequisites

- Have ActiveMQ installed and configured to set up a connection.
- Set up the Message bus namespace, which is a container for entities, such as queues and topics.
- Create a queue or topic, depending on your implementation requirement.

Procedure

1. In the workflow AppStudio, select **Integration > Message Bus Producer**.
2. Double click **Message Bus Producer.**

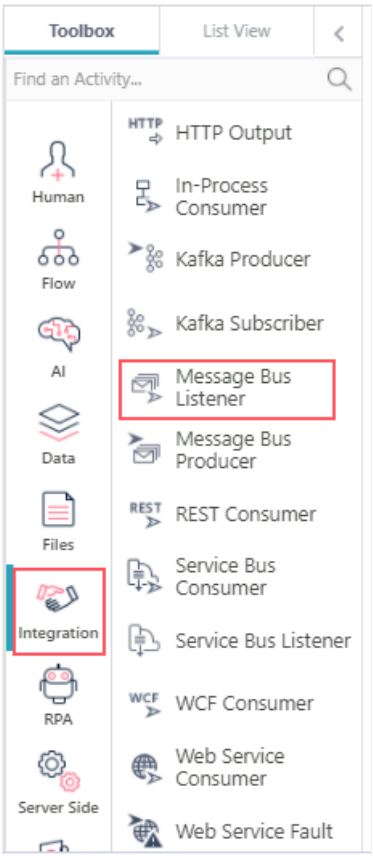


3. On the Message Bus Producer Properties screen, give a significant name and alias to the activity, and then click **Next**.
4. Select the MessageBus pipeline. You can add an expression as the pipeline name.
5. Click **Next**.
6. Expand the Message parameter and add the following details:
    - CorrelationId: string to determine duplicate message. Is mandatory when the pipeline is configured with duplicate detection.
    - MessageBody: body of the message.
    - Headers: the key-value.
7. Click **Finish**.

Configure the Message Bus Listener activity

Procedure

1. In the workflow AppStudio, select **Integration>Message Bus Listener**.
2. Double click **Message Bus Listener**.

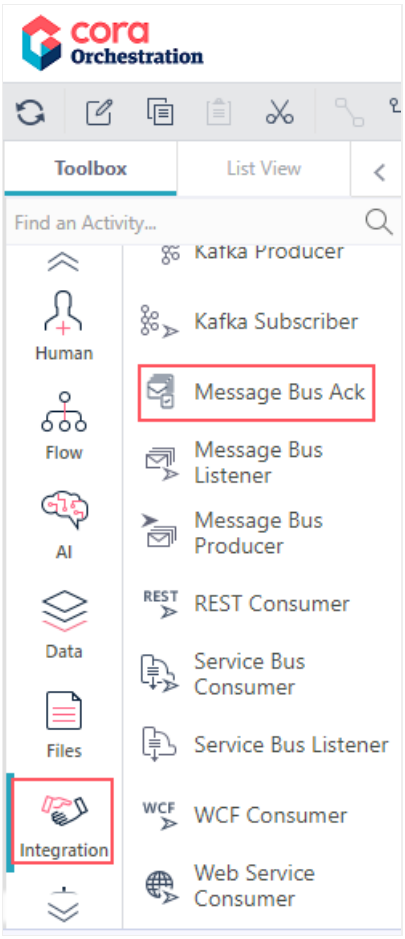


3. On the Message Bus Listener Activity Properties screen, give a significant name and alias to the activity, and then click **Next**.
4. On the Job tab, set the following:
   - **Name**: Enter a name to the actual job performed by the Message Bus Listener activity.
   - **Scaling**: Set the required scaling settings.
     For more details, see this article.
   - **Job is enabled**: Select this option only after you complete the workflow, or if you want to run the job for testing purposes.
5. Click **Next**.
6. On the Command tab, set the following:
   - **Pipeline**: Select from the list the pipeline for your message bus namespace.
7. Click **Finish**.

Configure the Message Bus Ack activity

Procedure

1. In the workflow AppStudio, select **Integration**>**Message Bus Ack**.
2. Add **Message Bus Ack** activity.



3. In the Message Bus Ack properties, give a significant name to the activity.
4. Click **Next**.
5. Select the Message Bus Listener activity for which you want to configure the acknowledgment. You can add an expression as the listener name to support multiple listeners simultaneously.
6. Click **Finish**.

Resume a workflow with the Message Bus Listener activity

Setting up the Message Bus Listener activity to resume a workflow requires additional configuration and integration with an external service.

You should have an active instance of Message Bus Listener for the listener to resume workflow.

After you create the Message Bus Listener activity, it waits for a message to arrive to a queue or topic configured for it. If the `Properties` property of the message contains the `JesActivityInstanceId` key with a value that matches an instance of a Message Bus Listener activity, the Job Execution Service picks up the message and resumes the execution of the workflow.

Procedure

1. In the App Studio:
   a. Place the Message Bus Listener activity anywhere in the workflow, except right after Start.
   b. Get the Message Bus Listener activity instance ID using an expression.
   Example: `{MessageBusListener}.ActivityInstanceId`
2. In the external service:
   a. Configure a Header in the Message Bus Producer activity to include an item with a key named `JesActivityInstanceId` and the value received in 1 b. to the queue or topic configured in the

activity.

NOTE
If the Message Bus Listener activity was already executed, depending on permissions, the activity may be executed again.

Limitations and important notes

- You cannot have two workflows with the same Message Bus Listener activity job definition. If you copy a workflow, you need to redefine the Message Bus Listener job settings.
- Cora SeQuence does not support sessions.
- When you copy or create a new version of a workflow that contains the Message Bus Listener activity, you need to delete the Message Bus Listener activity in the new workflow, and then set it up again.

V10.5.1 and later

Overview

The integrating messaging mechanism integrates your workflow with other workflows or applications or services. This mechanism integrates Cora Orchestration with the default internal queue to connect and send information between applications and services.

The integrating messaging mechanism enables the system to start or resume processes. For example, you can set up a message bus activity for an inter-bank transfer that requires customer's approval based on specific conditions. The message bus sends a message to a queue and Cora Orchestration creates a process for the bank's representative to receive the customer's approval in writing.

NOTE
SQL Server Service Broker is the default service. Starting with V9.8, Cora SeQuence supports the Apache ActiveMQ message broker as well.

 Watch this video for a demo of the ApacheMQ integration and required configuration.

With the new integrating messaging mechanism, the system can trigger multiple processes with one message from the queue.

You can configure the integrating messaging mechanism, under the `sequence.engine` node in the web.config file.

```
<messageBus>
    <connections defaultConnectionName="ActiveMQ">
      <!-- <add name="ActiveMQ" type="PNMsoft.Sequence.MessageBus.ActiveMQ.ClientFactory, PNMsoft.Sequence
.MessageBus.ActiveMQ" connectionString="Server=failover:(tcp://40.115.15.69:61616)?initialReconnectDelay=100"
/> -->
      <add name="ActiveMQ" type="PNMsoft.Sequence.MessageBus.ActiveMQ.ClientFactory, PNMsoft.Sequence.Mes
sageBus.ActiveMQ" credentials="dOtg/dWG6Q1TJDAvqHLX6Rvm2dPxGlL+bMx4RBxmA6NFL53clrvqe5IR7+3VQjTg7B
jnhYQAV+Zup9g4XjjurQ==" connectionString="Server=tcp://localhost:61616?initialReconnectDelay=100" />
      <add name="SqlServiceBroker" type="PNMsoft.Sequence.MessageBus.SqlServiceBroker.ClientFactory, PNMsoft
.Sequence.MessageBus.SqlServiceBroker" />
    </connections>
    <serializers defaultSerializerName="Newtonsoft.Json">
      <add name="Newtonsoft.Json" type="PNMsoft.Sequence.MessageBus.Serializers.NewtonsoftJson.MessageSerial
izerFactory, PNMsoft.Sequence.MessageBus.Serializers.NewtonsoftJson" />
    </serializers>
  </messageBus>
```

Or

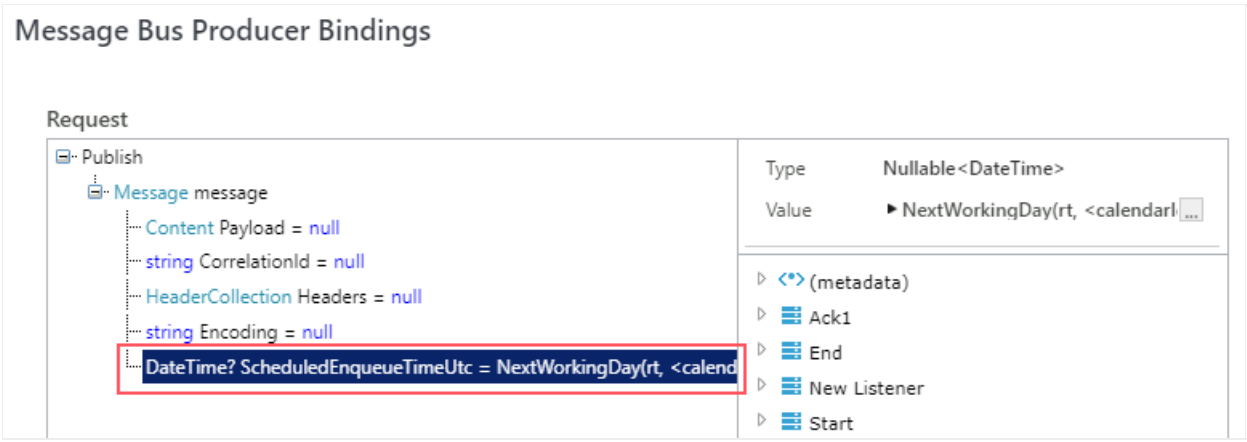
Use the PowerShell Function: Set-CoraSeQuenceApplicationConfiguration with the following parameters and values:

```
Set-CoraSeQuenceApplicationConfiguration -ApplicationType Administration -ConfigurationName MessageBusActive
MQ -TokensValues @{"SEQ_ActiveMQConnectionString" = "Server=failover:(tcp://192.168.40.4:61616);Username=a
dmin;Password=admin"} -Verbose
```

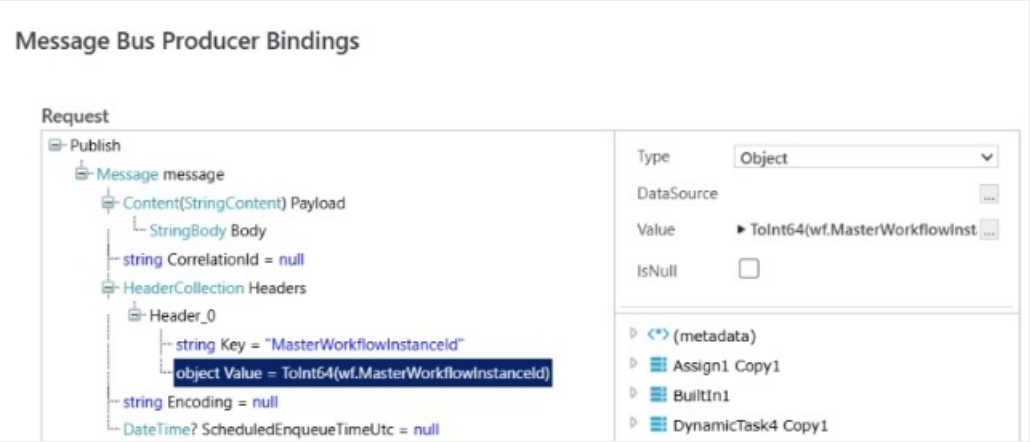
The integrating messaging mechanism uses two activities to listen to messages and start or resume a workflow and to publish the messages to the queue .

- Message Bus Producer: to send messages to the internal queue from within workflows.
- Message Bus Listener: a JES job to get messages from the internal queue from within workflows and acknowledge the message receival.

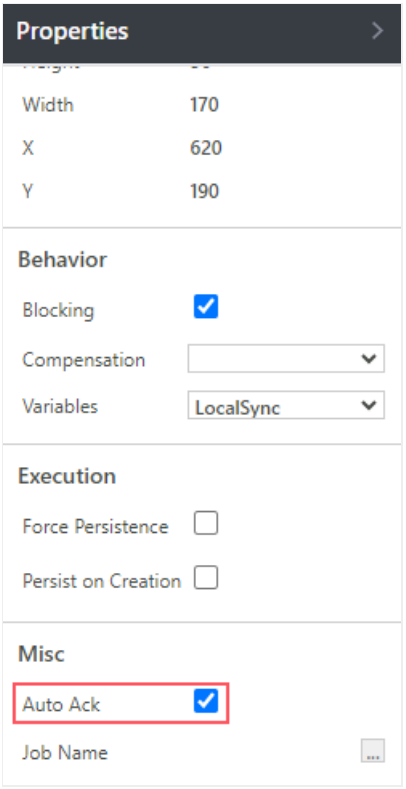
Starting from V10.5.1, you can control and schedule the availability of messages in the internal queue by setting the `ScheduledEnqueueTimeUtc` parameter in the Message Bus Producer activity bindings.



Starting from V10.8, you can start workflow instances using the Message Bus Listener from a specified master workflow instance, so that the new workflow instance logic can access and utilize data from the master instance's scope.



In the Message Bus Listener activity, when a new **Auto Ack** acknowledgment property is set to True, an acknowledgment is sent from the workflow that the message has been received from the queue. This acknowledgment is sent when the workflow is executed.



However, as a workflow developer you can implement a workflow that acknowledges a message receival at different points in the workflow with the following message bus activity.

- Message Bus Ack: to delay acknowledgment of the message receival from the queue.

The concept of the new Ack activity is similar to the concept of the HTTP Input/Output activities. The Message Bus Ack activity works along with the Message Bus Listener activity that exists in the same workflow template, and is executed before the related Message Bus Ack activity.

To acknowledge a message, you can use either Message Bus Listener activity's Auto Ack property, or the Message Bus Ack activity. For successful results:

- If the Listener activity's "Auto Ack" property is TRUE, then skip the execution of Ack activity.

- If the Listener activity's "Auto Ack" property is FALSE, then mark the message that was received by the listener as candidate for acknowledgement. The actual acknowledgement will be done by the job after the workflow execution.

NOTE
The Message Bus Ack activity is relevant only for the JES engine and not to the BRS, Portal, Admin, and WebAPI.
The SQL service broker *does not* support acknowledge.

The Integrating messaging can currently work with two types of entities:

- **Queues**: one direction communication — one sender to one receiver. The message delivery follows a first in, first out (FIFO) method. That is, messages are received and processed in the same order in which they enter the queue. The message bus processes only one message at a time. We support the following queue types:
  - Azure Service Bus
  - Active MQ
- **Topics**: one direction communication — one sender to several receivers, through topics. Messages are sent to a topic and delivered to one or more subscribers. The receiver does not communicate with the topic. Use topics if your activity requires sending messages to several receivers.

Messaging profiles

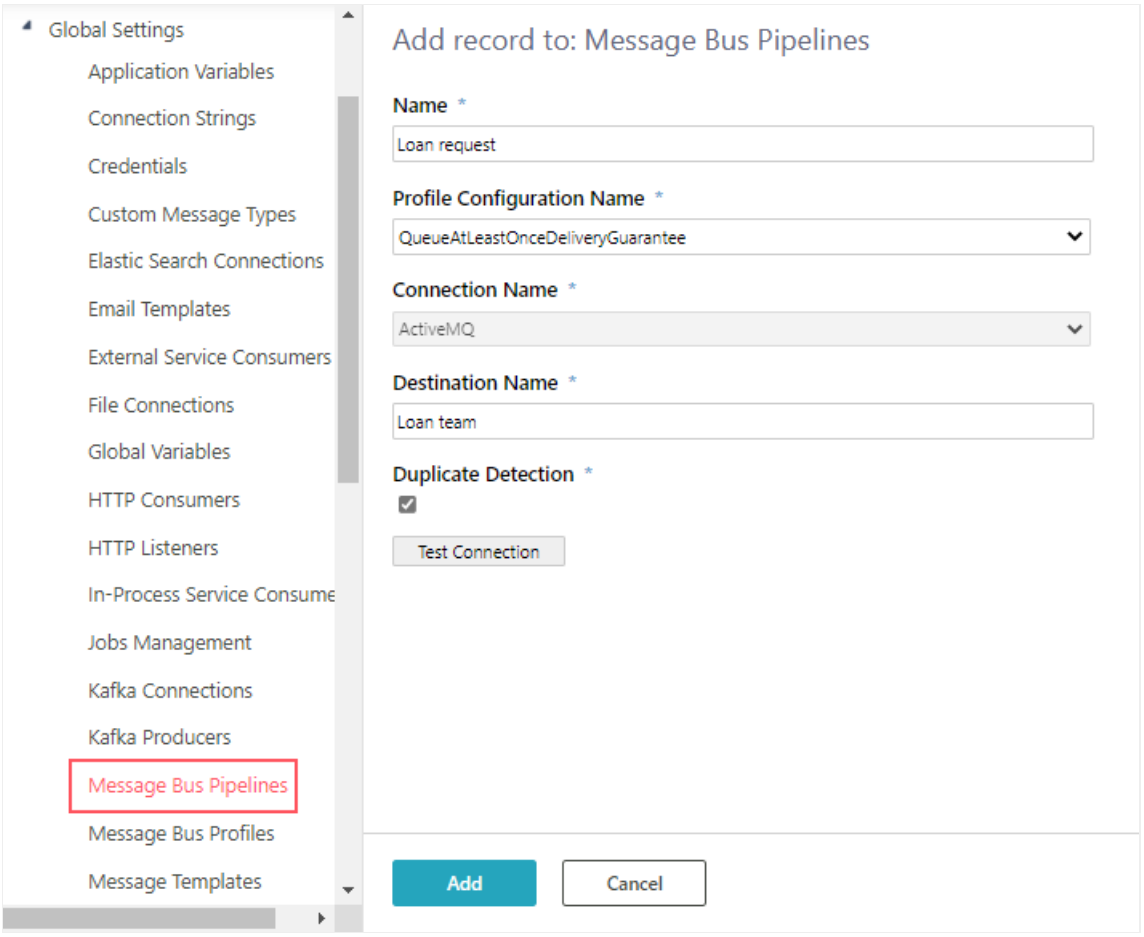
There are four out-of-the-box messaging profiles available to be mapped to the pipelines. These profiles define how messages will be sent in a queue or a topic.

- **Queue - At Least Once Delivery Guarantee**: using a queue mechanism for a one-to-one communication with a guarantee that the message delivers at least once. This option might cause a message to be delivered multiple times in order to guarantee the delivery.
- **Queue - At Most Once Delivery Guarantee**: using a queue mechanism for a one-to-one communication with a guarantee that a message will not be delivered more than once. This option might prevent a message from being delivered at all.
- **Topic - At Least Once Delivery Guarantee**: using a topic mechanism for a one-to-many communication with a guarantee that the message delivers at least once. This option might cause a message to be delivered multiple times in order to guarantee its delivery.
- **Topic - At Most Once Delivery Guarantee**: using a topic mechanism for a one-to-many communication with a guarantee that a message will not be delivered more than one time. This option might cause a message to be not delivered at all.

Define message bus pipelines

As an admin, you can define pipelines for internal queues.

1. In the Administration site, navigate to **Administration** > **Global Settings** > **Message Bus Pipelines**.
2. Click **Add**.
3. Add the following details:



- Name: Pipeline name.
- Profile Configuration Name: type of profile to be mapped to this pipeline.
- Connection Name: ActiveMQ, by default.
- Destination Name: The topic or queue where messages will be delivered.
- Duplicate Detection: Select the check box to enable detecting duplicate message based on the correlationId.
- MaxAutoLockRenewalDuration (seconds): ***Starting from V10.8***, you can configure additional time to extend the time out for document processing tasks.
- Test Connection: To test the connection.

4. Click **Add**.

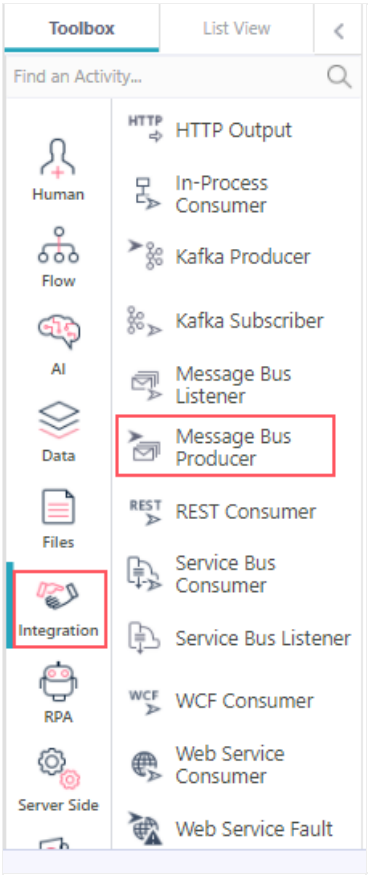
Configure the Message Bus Producer activity

Prerequisites

- Have ActiveMQ installed and configured to set up a connection.
- Set up the Message bus namespace, which is a container for entities, such as queues and topics.
- Create a queue or topic, depending on your implementation requirement.

Procedure

1. In the workflow AppStudio, select **Integration > Message Bus Producer**.
2. Double click **Message Bus Producer.**

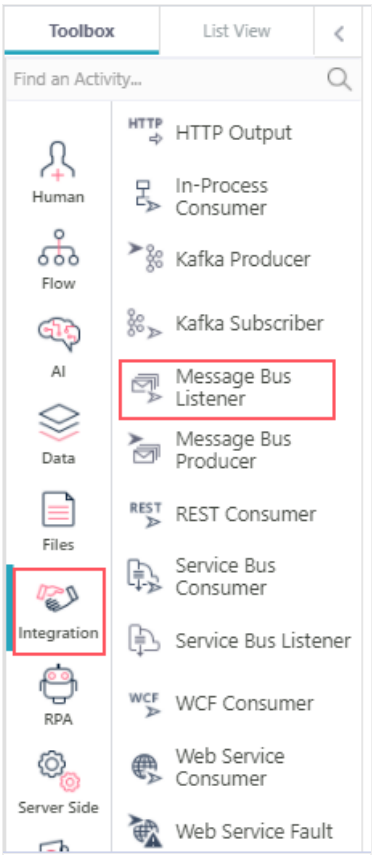


3. On the Message Bus Producer Properties screen, give a significant name and alias to the activity, and then click **Next**.
4. Select the MessageBus pipeline. ***Starting from V10.7***, you can add an expression as the pipeline name.
5. Click **Next**.
6. Expand the Message parameter and add the following details:
    - CorrelationId: string to determine duplicate message. Is mandatory when the pipeline is configured with duplicate detection.
    - MessageBody: body of the message.
    - Headers: the key-value.
7. Click **Finish**.

Configure the Message Bus Listener activity

Procedure

1. In the workflow AppStudio, select **Integration > Message Bus Listener**.
2. Double click **Message Bus Listener**.

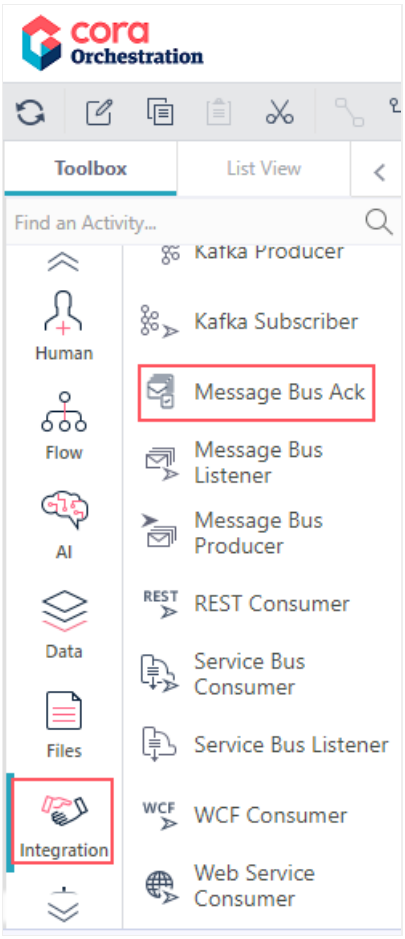


3. On the Message Bus Listener Activity Properties screen, give a significant name and alias to the activity, and then click **Next**.
4. On the Job tab, set the following:
   - **Name**: Enter a name to the actual job performed by the Message Bus Listener activity.
   - **Scaling**: Set the required scaling settings.
     For more details, see this article.
   - **Job is enabled**: Select this option only after you complete the workflow, or if you want to run the job for testing purposes.
5. Click **Next**.
6. On the Command tab, set the following:
   - **Pipeline**: Select from the list the pipeline for your message bus namespace.
7. Click **Finish**.

Configure the Message Bus Ack activity

Procedure

1. In the workflow AppStudio, select **Integration**>**Message Bus Ack**.
2. Add **Message Bus Ack** activity.



3. In the Message Bus Ack properties, give a significant name to the activity.
4. Click **Next**.
5. Select the Message Bus Listener activity for which you want to configure the acknowledgment. ***Starting from V10.7***, you can add an expression as the listener name to support multiple listeners simultaneously.
6. Click **Finish**.

Resume a workflow with the Message Bus Listener activity

Setting up the Message Bus Listener activity to resume a workflow requires additional configuration and integration with an external service.

You should have an active instance of Message Bus Listener for the listener to resume workflow.

After you create the Message Bus Listener activity, it waits for a message to arrive to a queue or topic configured for it. If the `Properties` property of the message contains the `JesActivityInstanceId` key with a value that matches an instance of a Message Bus Listener activity, the Job Execution Service picks up the message and resumes the execution of the workflow.

Procedure

1. In the App Studio:
   a. Place the Message Bus Listener activity anywhere in the workflow, except right after Start.
   b. Get the Message Bus Listener activity instance ID using an expression.
   Example: `{MessageBusListener}.ActivityInstanceId`
2. In the external service:
   a. Configure a Header in the Message Bus Producer activity to include an item with a key named

`JesActivityInstanceId` and the value received in 1 b. to the queue or topic configured in the activity.

NOTE
If the Message Bus Listener activity was already executed, depending on permissions, the activity may be executed again.

Limitations and important notes

- You cannot have two workflows with the same Message Bus Listener activity job definition. If you copy a workflow, you need to redefine the Message Bus Listener job settings.
- Cora SeQuence does not support sessions.
- When you copy or create a new version of a workflow that contains the Message Bus Listener activity, you need to delete the Message Bus Listener activity in the new workflow, and then set it up again.

V9.8-V10.5

Overview

The integrating messaging mechanism integrates your workflow with other workflows or applications or services. This mechanism integrates Cora SeQuence with the default internal queue to connect and send information between applications and services.

The integrating messaging mechanism enables the system to start or resume processes. For example, you can set up a message bus activity for an inter-bank transfer that requires customer's approval based on specific conditions. The message bus sends a message to a queue and Cora SeQuence creates a process for the bank's representative to receive the customer's approval in writing.

NOTE
SQL Server Service Broker is the default service. Starting with V9.8, Cora SeQuence supports the Apache ActiveMQ message broker as well.

 Watch this video for a demo of the ApacheMQ integration and required configuration.

With the new integrating messaging mechanism, the system can trigger multiple processes with one message from the queue.

You can configure the integrating messaging mechanism, under the `sequence.engine` node in the web.config file.

```
<messageBus>
    <connections defaultConnectionName="ActiveMQ">
      <!-- <add name="ActiveMQ" type="PNMsoft.Sequence.MessageBus.ActiveMQ.ClientFactory, PNMsoft.Sequence
.MessageBus.ActiveMQ" connectionString="Server=failover:(tcp://40.115.15.69:61616)?initialReconnectDelay=100"
/> -->
      <add name="ActiveMQ" type="PNMsoft.Sequence.MessageBus.ActiveMQ.ClientFactory, PNMsoft.Sequence.Mes
sageBus.ActiveMQ" credentials="dOtg/dWG6Q1TJDAvqHLX6Rvm2dPxGlL+bMx4RBxmA6NFL53clrvqe5IR7+3VQjTg7B
jnhYQAV+Zup9g4XjjurQ==" connectionString="Server=tcp://localhost:61616?initialReconnectDelay=100" />
      <add name="SqlServiceBroker" type="PNMsoft.Sequence.MessageBus.SqlServiceBroker.ClientFactory, PNMsoft
.Sequence.MessageBus.SqlServiceBroker" />
    </connections>
    <serializers defaultSerializerName="Newtonsoft.Json">
      <add name="Newtonsoft.Json" type="PNMsoft.Sequence.MessageBus.Serializers.NewtonsoftJson.MessageSerial
izerFactory, PNMsoft.Sequence.MessageBus.Serializers.NewtonsoftJson" />
    </serializers>
  </messageBus>
```

Or

Use the PowerShell Function: Set-CoraSeQuenceApplicationConfiguration with the following parameters and values:

```
Set-CoraSeQuenceApplicationConfiguration -ApplicationType Administration -ConfigurationName MessageBusActive
MQ -TokensValues @{"SEQ_ActiveMQConnectionString" = "Server=failover:(tcp://192.168.40.4:61616);Username=a
dmin;Password=admin"} -Verbose
```

The integrating messaging mechanism uses two activities to listen to messages and start or resume a workflow and to publish the messages to the queue .

- Message Bus producer: to send messages to the internal queue from within workflows.
- Message Bus listener: a JES job to get messages from the internal queue from within workflows and acknowledge the message receival.

The Integrating messaging can currently work with two types of entities:

- **Queues**: one direction communication — one sender to one receiver. The message delivery follows a first in, first out (FIFO) method. That is, messages are received and processed in the same order in which they enter the queue. The message bus processes only one message at a time.
- **Topics**: one direction communication — one sender to several receivers, through topics. Messages are sent to a topic and delivered to one or more subscribers. The receiver does not communicate with the topic. Use topics if your activity requires sending messages to several receivers.

Messaging profiles

There are four out-of-the-box messaging profiles available to be mapped to the pipelines. These profiles define how messages will be sent in a queue or a topic.

- **Queue - At Least Once Delivery Guarantee**: using a queue mechanism for a one-to-one communication with a guarantee that the message delivers at least once. This option might cause a message to be delivered multiple times in order to guarantee the delivery.
- **Queue - At Most Once Delivery Guarantee**: using a queue mechanism for a one-to-one communication with a guarantee that a message will not be delivered more than once. This option might prevent a message from being delivered at all.
- **Topic - At Least Once Delivery Guarantee**: using a topic mechanism for a one-to-many

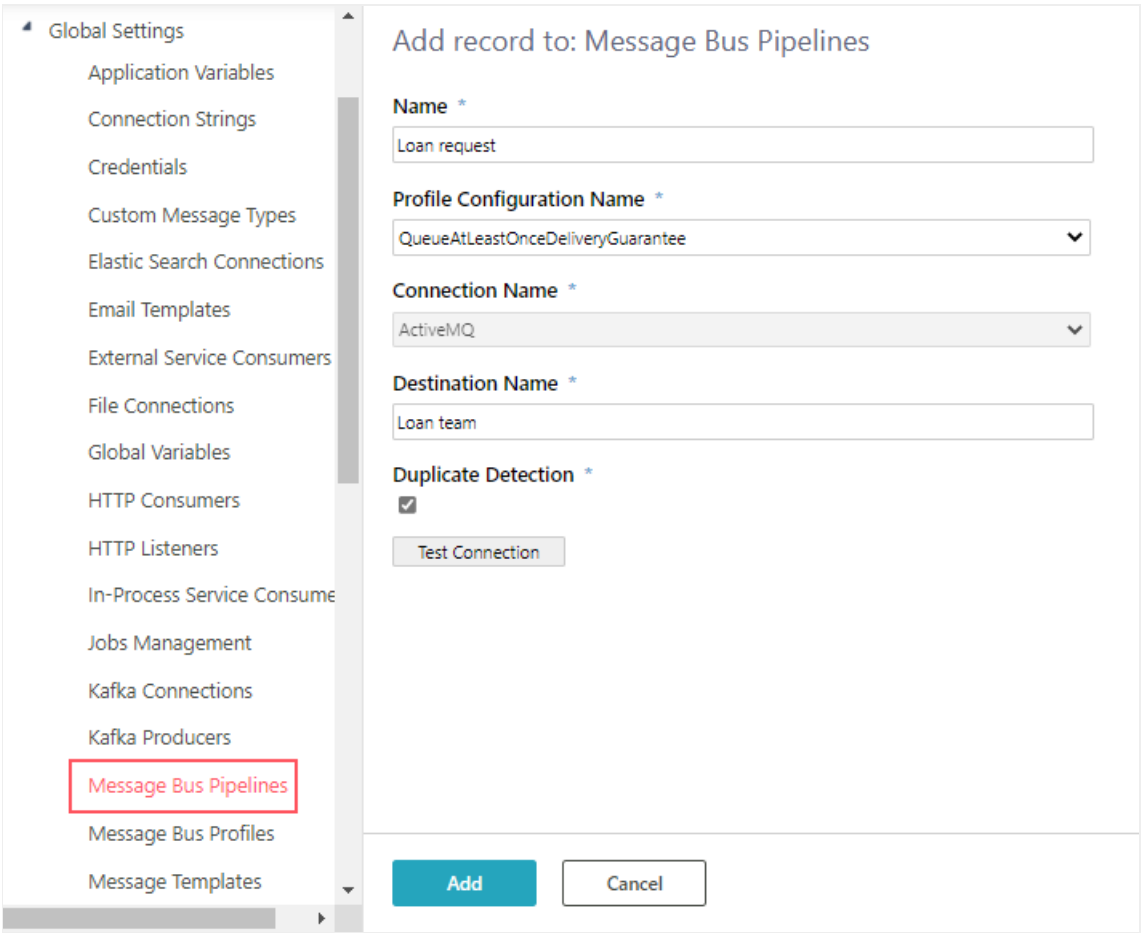
communication with a guarantee that the message delivers at least once. This option might cause a message to be delivered multiple times in order to guarantee its delivery.

- **Topic - At Most Once Delivery Guarantee**: using a topic mechanism for a one-to-many communication with a guarantee that a message will not be delivered more than one time. This option might cause a message to be not delivered at all.

Define message bus pipelines

As an admin, you can define pipelines for internal queues.

1. In the Administration site, navigate to **Administration** > **Global Settings** > **Message Bus Pipelines**.
2. Click **Add**.
3. Add the following details:



   - Name: Pipeline name.
   - Profile Configuration Name: type of profile to be mapped to this pipeline.
   - Connection Name: ActiveMQ, by default.
   - Destination Name: The topic or queue where messages will be delivered.
   - Duplicate Detection: select the check box to enable detecting duplicate message based on the correlationId.
   - Test Connection: To test the connection.
4. Click **Add**.

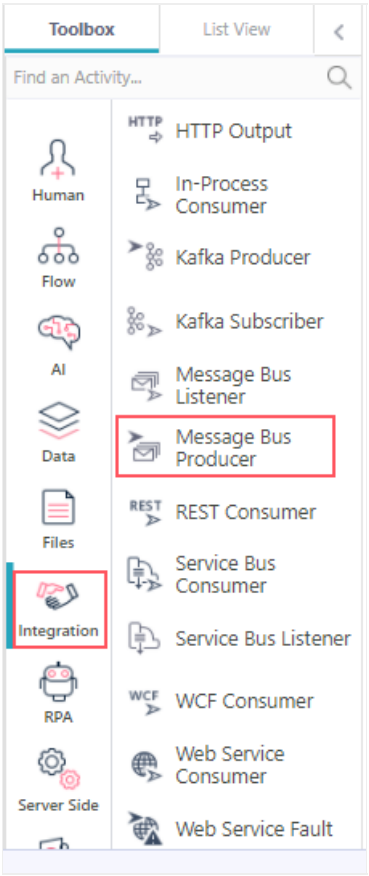
Configure the Message Bus Producer activity

Prerequisites

- Have ActiveMQ installed and configured to set up a connection.
- Set up the Message bus namespace, which is a container for entities, such as queues and topics.
- Create a queue or topic, depending on your implementation requirement.

Procedure

1. In the workflow AppStudio, select **Integration > Message Bus Producer**.
2. Double click **Message Bus Producer.**

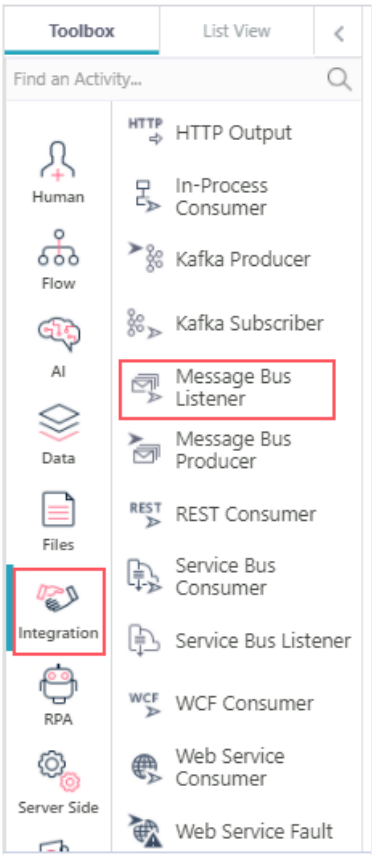


3. On the Message Bus Producer Properties screen, give a significant name and alias to the activity, and then click **Next**.
4. Select the MessageBus pipeline.
5. Click **Next**.
6. Expand the Message parameter and add the following details:
    - CorrelationId: string to determine duplicate message. Is mandatory when the pipeline is configured with duplicate detection.
    - MessageBody: body of the message.
    - Headers: the key-value.
7. Click **Finish**.

Configure the Message Bus Listener activity

Procedure

1. In the workflow AppStudio, select **Integration > Message Bus Listener**.
2. Double click **Message Bus Listener**.



3. On the Message Bus Listener Activity Properties screen, give a significant name and alias to the activity, and then click **Next**.
4. On the Job tab, set the following:
   - **Name**: Enter a name to the actual job performed by the Message Bus Listener activity.
   - **Scaling**: Set the required scaling settings.
     For more details, see this article.
   - **Job is enabled**: Select this option only after you complete the workflow, or if you want to run the job for testing purposes.
5. Click **Next**.
6. On the Command tab, set the following:
   - **Pipeline**: Select from the list the pipeline for your message bus namespace.
7. Click **Finish.**

Resume a workflow with the Message Bus Listener activity

Setting up the Message Bus Listener activity to resume a workflow requires additional configuration and integration with an external service.

You should have an active instance of Message Bus Listener for the listener to resume workflow.

After you create the Message Bus Listener activity, it waits for a message to arrive to a queue or topic configured for it. If the `Properties` property of the message contains the `JesActivityInstanceId` key with a value that matches an instance of a Message Bus Listener activity, the Job Execution Service picks up the message and resumes the execution of the workflow.

Procedure

1. In the App Studio:

a. Place the Message Bus Listener activity anywhere in the workflow, except right after Start.
b. Get the Message Bus Listener activity instance ID using an expression.
Example: `{MessageBusListener}.ActivityInstanceId`

2. In the external service:
a. Configure a Header in the Message Bus Producer activity to include an item with a key named `JesActivityInstanceId` and the value received in 1 b. to the queue or topic configured in the activity.

NOTE
If the Message Bus Listener activity was already executed, depending on permissions, the activity may be executed again.

Limitations and important notes

- You cannot have two workflows with the same Message Bus Listener activity job definition. If you copy a workflow, you need to redefine the Message Bus Listener job settings.
- Cora SeQuence does not support sessions.
- When you copy or create a new version of a workflow that contains the Message Bus Listener activity, you need to delete the Message Bus Listener activity in the new workflow, and then set it up again.