Informatica

Informatica® Cloud Application Integration
August 2024

RabbitMQ Connector Guide

Informatica Cloud Application Integration RabbitMQ Connector Guide
August 2024

Publication Date: 2024-08-07

Table of Contents

Preface

Read the *RabbitMQ Connector Guide* to learn how to set up and use RabbitMQ connections. Learn how to configure and use the RabbitMQ connection to read data from or write data to RabbitMQ queues.

CHAPTER 1

Introduction to RabbitMQ Connector

This chapter includes the following topics:

RabbitMQ Connector Overview

You can use RabbitMQ Connector to perform the following tasks:

- Configure a RabbitMQ connection, a RabbitMQ producer, and a RabbitMQ consumer.
- Listen for and read messages on a particular queue.
- Write messages to a particular queue.

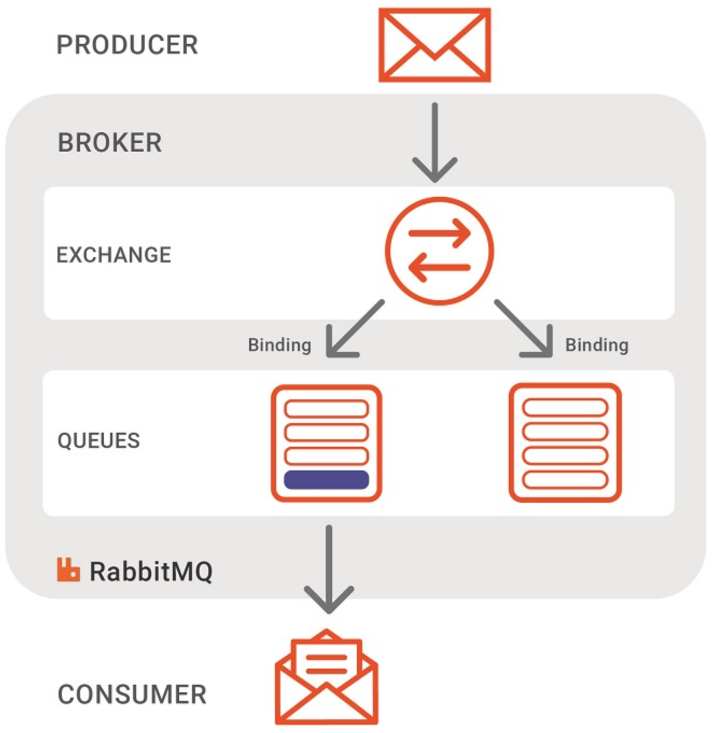
RabbitMQ Overview

RabbitMQ is a distributed message broker system that is fast, scalable, and durable.

RabbitMQ uses Advanced Messaging Queuing Protocol (AMQP) for secure transfer of messages.

A producer publishes messages to an exchange. After the exchange receives the messages, it routes the messages to different queues. Based on the message routing key, the exchange creates a binding with the queue and routes the messages to the queue. The message stays in the queue until a consumer connects to the queue and subscribes to the message.

The following image shows the message flow in RabbitMQ:



RabbitMQ Concepts

To successfully use the RabbitMQ connection, you must understand the following RabbitMQ messaging concepts:

- Virtual Hosts
- Exchanges
- Consumer Acknowledgements
- Publisher Acknowledgements
- Dead Letter Routing
- Heartbeats
- Durable Queues
- Headers

Virtual Hosts

A virtual host provides a way to segregate applications using the same RabbitMQ instance. Different users can have different access privileges to different virtual hosts and queues and exchanges can be created, so they only exist in one virtual host.

Exchanges

When a producer publishes a message to a RabbitMQ broker, the messages are sent to an exchange. Exchanges are responsible for routing the messages to different queues. An exchange accepts messages from the producer and routes the messages to queues based on header attributes, bindings, and routing keys.

A binding is a connection between an exchange and a queue. Routing key is an attribute in the message, which helps the exchange to route the messages to queues.

The following types of exchanges are available in RabbitMQ:

- **Direct Exchange**

  A direct exchange delivers messages to queues based on a message routing key. In a direct exchange, the message is routed to the queues whose binding key matches the routing key of the message.

- **Fanout Exchange**

  A fanout exchange routes messages to all of the queues that are bound to it.

- **Topic Exchange**

  The topic exchange does a wildcard match between the routing key and the routing pattern specified in the binding.

- **Headers Exchange**

  Headers exchanges use the message header attributes for routing and routes messages to queues based on header values.

Consumer Acknowledgements

After a message is delivered to the consumer, RabbitMQ marks the message for deletion. To ensure that RabbitMQ delivers the messages to the consumer and the consumer handles the message successfully, RabbitMQ uses acknowledgements. The consumer sends an acknowledgement to RabbitMQ that the message has been successfully received, processed, and that RabbitMQ can delete the message.

If a consumer connection is lost and the consumer is unable to send an acknowledgement to RabbitMQ, RabbitMQ considers that the message was not processed completely and re-queues the message. If there are any other consumers online at the same time, RabbitMQ redelivers the message.

Automatic Acknowledgements

In automatic acknowledgement mode, RabbitMQ considers the message to be successfully delivered immediately after the message is sent. Automatic message acknowledgements increases the performance and throughput, but reduces the safety of message delivery.

If a consumer connection is lost, the message sent by RabbitMQ is lost.

Negative Acknowledgements

In negative acknowledgement mode, a consumer is unable to process a delivery immediately but other consumers might be able to process the message. RabbitMQ re-queues message for another consumer to receive and handle the message.

Publisher Acknowledgements

To ensure that the message published by the producer has reached the RabbitMQ server and the RabbitMQ server handles the message successfully, RabbitMQ sends an acknowledgement to the publisher using the same mechanism used for consumer acknowledgements.

Dead Letter Routing

RabbitMQ uses Dead Letter Routing to republish a message to an exchange when any one of the following event occurs:

- The message is negatively acknowledged by a consumer and is not re-queued by the RabbitMQ broker.
- The message expires after it exceeds the time to live period.
- The message was dropped because its queue exceeded a length limit.

The dead lettered messages are routed to the dead letter exchange specified in the same virtual host. Dead lettered messages are routed to the dead letter exchanges based on the routing key specified for the queue or the routing keys that the message were originally published with.

Heartbeats

RabbitMQ uses heartbeats to detect any disrupted connections or defends idle connections that might be terminated. The heartbeat timeout determines the time period after which the RabbitMQ server and client considers the connection as unreachable.

Durable Queues

Queues in RabbitMQ that are defined as durable are persisted to disk and can survive a broker restart. A durable queue does not ensure that the messages routed to that queue can survive a broker restart. In a durable queue, only persistent messages can survive a broker restart.

Headers

RabbitMQ messages consist of headers and a body. Headers are attributes set on a RabbitMQ message to control message delivery in RabbitMQ queues. Headers are set by the producer when it publishes a message and the consumer after it receives the message.

For example, you can use the following headers:

- rabbitmq.ROUTING_KEY. The routing key used to deliver a message to the queue. The routing key specified in the header overrides the value specified in the RabbitMQ connection.
- rabbitmq.DELIVERY_MODE. If the value is set to 2, RabbitMQ ensures that messages are persistent in a durable queue and can survive a broker restart.
- rabbitmq.EXPIRATION. Specifies the time period after which a message will expire in a queue.
- rabbitmq.EXCHANGE_OVERRIDE_NAME. Specifies the target exchange name that overrides the exchange name specified in the RabbitMQ connection.
- rabbitmq.EXCHANGE_NAME. Name of the exchange that the consumer received the message from.

CHAPTER 2

RabbitMQ Connections

This chapter includes the following topics:

RabbitMQ Connections Overview

Use a RabbitMQ connection to connect to a RabbitMQ broker and read data from or write data to a queue.

After you create a RabbitMQ connection, validate and save the connection.

You can then publish the RabbitMQ connection and click the **Metadata** tab to view the generated process objects for the connection.

Basic Connection Properties

The following table describes the basic properties that you can configure on the **Properties** tab of the connection creation page:

Property	Description
Name	Required. The name of the RabbitMQ connection that identifies it in the Process Designer. The name must start with an alphabet and can contain only alphabets, numbers, or hyphens (-). You cannot specify the same name for a connection in the same project.
Location	Optional. The location of the project or folder where you want to save the connection. Click **Browse** to select a location. If the **Explore** page is currently active and a project or folder is selected, the default location for the connection is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.

Property	Description
Description	Optional. Description of the connection.
Type	Required. The type of connection that you want to use for the connector or service connector. Select **RabbitMQ**.
Run On	Required. The runtime environment for the connection. You can run the connection on a Secure Agent group, a Secure Agent machine, or the Cloud Server. You must unpublish the connection to change the runtime environment.
Connection Test	Not supported for RabbitMQ Connector.
OData-Enabled	Not supported for RabbitMQ Connector.

After you configure the basic properties, you must also define the following properties:

- The properties applicable to the RabbitMQ connection type
- The event source properties and event target properties for the RabbitMQ connection

RabbitMQ Connection Properties

When you select **RabbitMQ** as the connection type, you can configure RabbitMQ-specific connection properties on the **Properties** tab of the connection creation page.

Connection Settings

You must specify the RabbitMQ host and RabbitMQ port that you want to connect to. You can also enable TLS or SSL authentication.

The following table defines the RabbitMQ connection properties that you must configure in the **Connection Settings** section:

Property	Description
Host Name	Required. Host name of the machine where the RabbitMQ instance or cluster runs that the Secure Agent must connect to with a TCP connection.
Port	Required. Port number for the host where the RabbitMQ instance or cluster runs. Default is **5762**.

Property	Description
Virtual Host	Optional. Specify the virtual host that the RabbitMQ channel connects to. Default is **/**.
TLS Protocol	Optional. Specify whether you want to use TLS or SSL protocol to connect to RabbitMQ. Select one of the following values: - TLS - SSLv3 - Disabled Default is **Disabled**.

Access Settings

You can configure the user name and password to connect to a RabbitMQ instance with different privileges and permissions. Users can also be assigned permissions to specific virtual hosts.

The following table defines the RabbitMQ connection properties that you must configure in the **Acesss Settings** section:

Property	Description
User Name	Required. User name to connect to RabbitMQ.
Password	Required. Password to connect to RabbitMQ.

Security Protocol Settings

You can configure TLS or SSL authentication to encrypt and securely transfer data between a RabbitMQ producer, RabbitMQ consumer, and a RabbitMQ instance. When you configure TLS or SSL authentication, a Certificate Authority signs and issues a certificate to the RabbitMQ client. The RabbitMQ broker uses the certificate to verify the identity of the client.

If you enable SSL authentication, you must configure the following authentication properties in the **Security Protocol Settings** section:

Property	Description
Trust All	Select **Yes** to enable the connection to trust any server certificate. Default is **No**.
Truststore Location	The absolute path and file name of the truststore file on the Secure Agent machine that contains the TLS or SSL certificate to establish two-way secure communication with the RabbitMQ broker. For example: `C:/SSL/Certs_208/icrt-truststore.jks` If left blank, the connection uses the default truststore that is set through the `javax.net.ssl.trustStore` JVM argument.
Truststore Password	The password for the truststore file that contains the TLS or SSL certificate. If left blank, the connection uses the default password that is set through the `javax.net.ssl.trustStorePassword` JVM argument.

Property	Description
Keystore Location	The absolute path and file name of the keystore file on the Secure Agent machine that contains the keys and certificates required to establish two-way secure communication with the RabbitMQ broker. For example: `C:/SSL/Keys/icrt-keystore.jks` If left blank, the connection uses the default keystore that is set through the `javax.net.ssl.keyStore` JVM argument. The connection trusts any server certificate regardless of whether the **Trust All** property is set to **Yes** or **No**.
Keystore Password	The password to access the keystore file that contains the TLS or SSL certificate for secure communication. If left blank, the connection uses the default password that is set through the `javax.net.ssl.keyStorePassword` JVM argument.

Advanced Settings

You configure advanced RabbitMQ attributes to ensure that the message is delivered to a queue successfully.

The following table defines the RabbitMQ connection properties that you must configure in the **Advanced Settings** section:

Property	Description
Connection Timeout	Required. Number of seconds to wait when attempting to connect to RabbitMQ. Default is **60** seconds.
Heartbeat Timeout	Required. Heartbeat timeout in seconds requested for the RabbitMQ connection. Default is **60** seconds.
Enable Mandatory Message Routing	Optional. Select **Yes** to throw an exception when a message cannot be routed to a queue. If you select **No**, the RabbitMQ server silently drops the message. Default is **Yes**.
Enable Guaranteed Message Delivery	Optional. Select **Yes** to throw an exception when the message cannot be delivered and the mandatory message routing is enabled. Default is **Yes**.

RabbitMQ Event Source Properties

When you create a RabbitMQ event source, you create a RabbitMQ consumer to read RabbitMQ messages. You can use each RabbitMQ event source in a process that reads RabbitMQ messages.

For each RabbitMQ connection that you create, you can add one or more event sources. An event source serves as a start event that listens or monitors a specified RabbitMQ queue for new messages. After you define an event source for a RabbitMQ connection, you can publish the connection only on a Secure Agent. You can use the event source in a process to consume the messages from a queue as process objects. You can then deploy the process on the same Secure Agent where you published the RabbitMQ connection.

To create event sources for a RabbitMQ connection, click **Add Event Source** on the **Event Sources** tab. Select the event source type as **RabbitMQ Monitor**.

The following table describes the basic event source properties that you can configure:

Property	Description
Name	Required. The event source name that appears in the Process Designer. The name must be unique for the connection. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
Description	Optional. A description for the RabbitMQ event source that appears in the Process Designer. The description cannot exceed 4,000 characters.
Enabled	Select **Yes** to make the event source available immediately after it is published. Select **No** to disable the event source until you are ready to use it. Default is **Yes**.

The following table describes the RabbitMQ event source properties that you can configure:

Event Source Property	Description
Enable Load Balancing	Required. Determines whether the connection must be deployed on all the Secure Agents in a group or on the selected Secure Agent for load balancing. You must enable this option only if the Process Server uses a Secure Agent Cluster configuration. Default is **No**. **Note:** After you publish a connection and run a process, if you toggle the load balancing option, you might see duplicate messages. To avoid this issue, Informatica recommends that you create a new connection if you want to change the load balancing option.
Exchange Name	Optional. The exchange name where the produced messages are sent to. If left blank, the default exchange is used.
Routing Key	Optional. The routing key to route the message to a queue.
Queue	Required. The name of the RabbitMQ queue from where the messages can be consumed.
Payload Format	Required. Specifies the type of message payload format you expect on the destination. Select one of the following options: - TEXT - XML - JSON - BINARY Default is **TEXT**.
Object List Field Name	Optional. If the payload format is JSON, the queue accepts a JSON message that might contain an array of objects. If it does, enter the value here to be used as the field name of the object list in the RabbitMQ message body.
Enable Dead Letter Routing	Optional. Select **Yes** to route faulty messages to a target dead letter queue. **Note:** If you have configured dead letter routing for the RabbitMQ broker, you must ensure that the **Enable Dead Letter Routing** field is disabled. Default is **No**.

Event Source Property	Description
Dead Letter Exchange	Name of the dead letter exchange. If left blank, the default exchange is used.
Dead Letter Routing Key	Routing key for the dead letter exchange to route the message to a dead letter queue.
Other Attributes	Optional. If you need to set RabbitMQ advanced attributes for the event source, you can supply them here. Contact Informatica Global Customer Support for assistance.
Enable Automatic Message Acknowledgement	Optional. Select **Yes** if you want messages to be automatically acknowledged. **Note:** If you have configured dead letter routing for the RabbitMQ broker, you must ensure that the **Enable Automatic Message Acknowledgement** field is disabled. Otherwise, messages will not be routed to the dead letter queue. Default is **No**.
Prefetch Count	Optional. The maximum number of unacknowledged messages delivered on a RabbitMQ channel. Specify **0** to deliver unlimited unacknowledged messages. Default is **100**.

You can view the status of each event source in the published connection. If the status of the event source is stopped, you can republish the connection and restart the event source. When you republish the connection, all the event sources in the connection start by default.

For more information about starting and stopping event sources in a listener-based connection, see *Connectors for Cloud Application Integration and Monitor*.

RabbitMQ Event Target Properties

When you create a RabbitMQ event target, you create a RabbitMQ producer to write RabbitMQ messages. You can use each RabbitMQ event target in a process that writes RabbitMQ messages.

For each RabbitMQ connection that you create, you can include one or more event targets that specify operations and call the event target from a process to send messages to a RabbitMQ exchange.

To create event targets for a RabbitMQ connection, click **Add Event Target** on the **Event Targets** tab. Select the event target type as **RabbitMQ Writer**.

The following table describes the basic event target properties that you can configure:

Property	Description
Name	Required. The event target name that appears in the Process Designer. The name must be unique for the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
Description	Optional. A description for the RabbitMQ event target that appears in the Process Designer. The description cannot exceed 4,000 characters.

The following table describes the RabbitMQ event target properties that you can configure:

Event Target Property	**Description**
Exchange Name	Optional. The exchange name where the produced messages are sent to. If left blank, the default exchange is used.
Routing Key	Optional. The routing key to route the message to a target queue.
Payload Format	Required. Specifies the type of message payload format you expect on the destination. Select one of the following options: - TEXT - XML - JSON - BINARY Default is **TEXT**.
Other Attributes	Optional. If you need to set RabbitMQ advanced attributes for the event target, you can supply them here. Contact Informatica Global Customer Support for assistance.

RabbitMQ Connections Metadata

After you publish a RabbitMQ connection, you can view the actions and objects associated with the RabbitMQ connection on the **Metadata** tab of the **Connection** page.

Under the **Actions** section, you can view the event sources and event targets configured for the RabbitMQ connection.

Under the **Objects**, you can view the objects associated with the RabbitMQ connection. The published metadata for a RabbitMQ connection consists of the following objects:

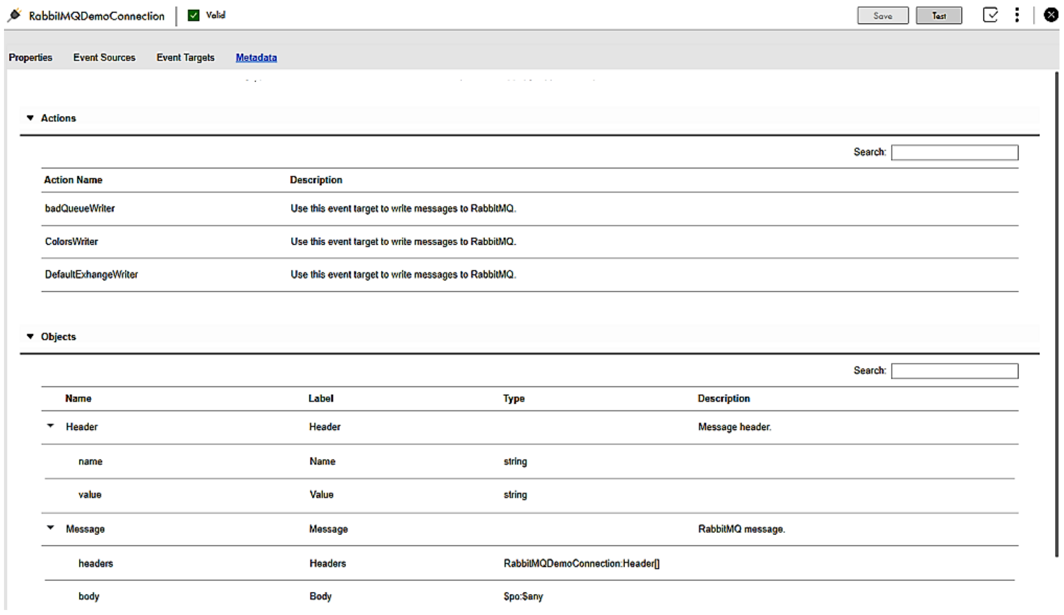
- **Headers**

  The **Header** object contains name and value fields that represents RabbitMQ header name and the value for that header.

- **Messages**

  RabbitMQ messages consist of headers and body. The headers in a RabbitMQ message are a list of name-value pairs that contain multiple attributes for a RabbitMQ message. The RabbitMQ message body contains the message payload accessible to the process as a process object of type `$po:$any`.

The following image shows the published metadata of a RabbitMQ connection:



▼ Actions

Search:

Action Name	Description
badQueueWriter	Use this event target to write messages to RabbitMQ.
ColorsWriter	Use this event target to write messages to RabbitMQ.
DefaultExhangeWriter	Use this event target to write messages to RabbitMQ.

▼ Objects

Search:

Name	Label	Type	Description
▼ Header	Header		Message header.
name	Name	string	
value	Value	string	
▼ Message	Message		RabbitMQ message.
headers	Headers	RabbitMQDemoConnection:Header[]	
body	Body	$po:$any	

CHAPTER 3

RabbitMQ Processes

This chapter includes the following topics:

RabbitMQ Process Creation

After you create a RabbitMQ connection, a RabbitMQ consumer, and a RabbitMQ producer, you can use them in a process.

RabbitMQ consumer process

Use the following guidelines when you create a RabbitMQ consumer process:

- Select the process binding as event.
- Select the RabbitMQ consumer that you created in the RabbitMQ connection.

  Note: The process automatically creates an **event** field in the **Input Fields** section.

RabbitMQ producer process

Use the following guidelines when you create a RabbitMQ producer process:

- Add a Service step to select the RabbitMQ producer that you created in the RabbitMQ connection.
- In the input field, select the RabbitMQ message that you want to write to the target exchange.

Message Payload

The event used in the process contains an process object with a payload that represents the RabbitMQ message. You can process the RabbitMQ messages in one of the following format:

- JSON
- XML
- BINARY
- TEXT

When the RabbitMQ message is in JSON or XML format, the message payload for RabbitMQ is made accessible to the process as a process object of type `$po:$any`.

For example, the following RabbitMQ message is represented in JSON format:

```
[
{"firstName":"cindy", "lastName": "louis", "department": "qa", "id":1},
{"firstName":"billy", "lastName": "joe", "department": "dev", "id":12}
]
```

If you select JSON as the content format for the event, the Process Designer receives the JSON message payload as a process object list:

```
<message>
        <headers>
         <name>rabbitmq.ROUTINGKEY</name>
         <value>defaultQueue</value>
        </headers>
        <body>
   <employee>
       <firstName>cindy</firstName>
       <lastName>louis</lastName>
       <department>qa</department>
       <id>1</id>
   </employee>
   <employee>
       <firstName>billy</firstName>
       <lastName>joe</lastName>
       <department>dev</department>
       <id>12</id>
   </employee>
  <body>
</message>
```

Note: If the payload is JSON, the array does not have a field name as defined in the RabbitMQ message. You can specify an in the **Object List Field Name** property and set the implicit object field name. In the above example, *employee* is specified as the object field name for the event source in the **Object List Field Name** property.

Rules and Guidelines for RabbitMQ Processes

Consider the following rules and guidelines for RabbitMQ processes:

- When the message in a Rabbit event source or target is in JSON or XML format, the Process Designer receives the message as a process object.
- When the message in a Rabbit event source or target is in Binary format, the Process Designer receives the message in Base64 encoded String format in the process object.
- When the message in a RabbitMQ event source or target is in text format, the Process Designer receives the message as a String in the process object.
- When you use a RabbitMQ event source in a process, you must deploy the process on the same Secure Agent where you published the RabbitMQ connection.
- When you use a RabbitMQ event target in a process, you can deploy the process on the Cloud Server, a Secure Agent group, or any Secure Agent machine.

INDEX