Informatica®

Informatica® Application Integration
April 2024

Amazon SQS Connector Guide

Informatica Application Integration Amazon SQS Connector Guide
April 2024

Publication Date: 2024-04-04

Table of Contents

Preface

Read the *Amazon SQS Connector Guide* to learn how to set up and use SQS connections.

This guide assumes that you have an understanding of message queue systems such as SQS.

CHAPTER 1

Introduction to Amazon SQS Connector

This chapter includes the following topics:

Amazon SQS Connector Overview

Amazon SQS is a distributed queue system. With Amazon SQS, web services can queue messages generated by one application component so they are available to be consumed by another component. Similar to other messaging systems like JMS, the queue is a temporary repository for messages that await processing. Each message can contain up to 256 KB of text, in any format.

Amazon SQS provides an API that enables applications to retrieve the messages. You might also be able to use the Amazon SQS Extended Client Library for Java, which enables you to store larger payloads with Amazon S3.

With Amazon SQS connections and related events, you can enable processes that are triggered by receipt of a message in a monitored SQS queue. For example, you might have an SQS queue that triggers an order processing workflow on a service bus in Amazon Web Services communicating over an API.

Refer to the Amazon SQS documentation for more information.

Amazon SQS Implementation

Support for Amazon SQS was implemented using a Camel event listener. Processes that connect to Amazon SQS can be triggered upon the arrival of an Amazon SQS message.

When you publish a process that uses the SQS connector, the binding details are stored in a catalog (`entrypoints.xml`) that determines how to route the message.

The message payload is defined by the connector.

Using the Amazon SQS Connector, a single connection can be available both as a source and a target. When you trigger a process, a fan out is supported, so one message can trigger multiple processes.

CHAPTER 2

Amazon SQS Connections and Processes

This chapter includes the following topics:

Amazon SQS Connections

Use an Amazon SQS connection to connect to an Amazon SQS stream to send messages to a queue, receive messages from a queue, or delete messages from a queue.

You can add one or more event sources and event targets. You must have read, write, and delete permissions on an Amazon SQS queue to use it as an event source or event target.

After you create an Amazon SQS connection, you can validate and save the connection.

You can then publish the Amazon SQS 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 page:

Property	Description
Name	Required. Unique name for the Amazon SQS connection that identifies it in the Process Designer. The name must start with an alphabet and can contain only alphabets, numbers, and hyphens (-).
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.
Description	Optional. Description of the connection.
Type	Required. The type of connection that you want to use for the connector or service connector. Select **Amazon SQS.**
Run On	Required. The name of the Secure Agent group or the Secure Agent machine where the connection must run.
Connection Test	Not supported for Amazon SQS Connector.
OData-Enabled	Not supported for Amazon SQS Connector.

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

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

After you publish the connection, the **Metadata** tab displays the generated process objects.

Amazon SQS Connection Properties

IAM authentication provides secured access to Amazon SQS resources. You can use the AWS IAM system to map policies to IAM roles or external resources and to determine the list of permissions that can be assigned to the IAM roles.

Note: The IAM role that you configure in the Amazon SQS connection properties will have access to all the queues that you include in the Amazon SQS event source and event target properties. You must create a separate app connection to configure an IAM role to access a different set of Amazon SQS queues.

When you select **Amazon SQS** as the connection type, you can configure Amazon SQS specific connection properties. The following table defines the Amazon SQS connection properties that you must configure in the **Connection Properties** section:

Property	Description
Access Key	AWS access key ID of the requester. Alternatively, you can pass the following access key in the AWS SDK in the Java system properties: `aws.accessKeyId` You can pass the key using the `AWS_ACCESS_KEY_ID` environment variable.
Secret Key	AWS secret key of the requester. Alternatively, you can pass the following secret key in the AWS SDK in the Java system properties: `aws.secretKey` You can pass the secret key using the `AWS_SECRET_ACCESS_KEY` environment variable.
Use EC2 Role to Assume Role	Enables the EC2 role to assume another IAM role specified in the IAM Role ARN option.
IAM Role ARN	The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role assumed by the user to use the dynamically generated temporary security credentials.
External Id	Provides a more secure access to the Amazon SQS bucket when the Amazon SQS bucket is in a different AWS account. You can use external ID to grant access to your AWS resources to a third party.
SQS Endpoint	Required. The region that applies to all event sources and event targets of the connection. For example: `sqs.eu-west-1.amazonaws.com`

Basic Event Source Properties

You can add one or more event sources for a connection. An event source serves as a start event that listens or monitors a specified location for new files or messages. After you define an event source for a connection, you can publish the connection only on a Secure Agent and not on the Cloud Server. You can then access the event source in a process and deploy the process only on a Secure Agent to consume the process objects generated by the event source downstream.

To create event sources for a connection, from the **Event Sources** tab, click **Add Event Source**, and select the event source type from the available list.

The following table describes the event source properties that are available for all event source types:

Property	Description
Name	Required. Unique name for the event source.
Description	Optional. Description for the event source.
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**.

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*.

Amazon SQS Event Source Properties

You create an Amazon SQS event source to dequeue messages or subscribe to a service. Each event source is available for use in a process that handles SQS messages. You can publish and unpublish the connection to determine when each event source is available for use.

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

Property	Description
Queue Name	Required. Name of the Amazon SQS queue. If the named queue does not exist, the connection automatically creates the queue. **Note:** The question mark (?) character is prohibited in this field. It is not supported by Apache Camel and you cannot publish a connection that contains this value.
Query Parameters	Optional. You can specify other parameters that define how to connect to servers, if required by the SQS client. For example, you can use the standard URI query syntax to provide additional information for the client to successfully connect.
Queue Owner AWS Account ID	Optional. The AWS account ID of the account owner that created the queue if the queue already exists.
Region	Optional. The AWS region for the SQS queue. For example: `us-west-1`

Property	Description
Content Format	Required. Identifies the format of the content to be processed. Select one of the following values: - **TEXT**. The content is a string and handled like plain text. - **XML**. The content is parsed and converted to an object or a list of process objects. - **JSON**. The content is parsed and converted to an object or a list of process objects.
Objectlist Field Name	Optional. If the payload format is JSON and the queue receives an array of objects as a JSON message, this value determines the field name of the object list in the message body.

The following table describes the advanced properties for an event source:

Property	Description
Visibility Timeout	Optional. Duration in seconds for which a message is hidden from subsequent retrieve requests after the message is retrieved by a RetrieveMessage request. Immediately after the component receives the message, the message is still in the queue. Amazon SQS does not automatically delete a message after it is processed because of the probability that the message might not have been received. This field determines the duration of the timeout during which SQS prevents other components from receiving and processing the same message. If the **Extend Message Visibility** option is enabled, a scheduled background task extends the message visibility in the queue. You can extend the timeout up to a maximum of 12 hours. Enter a value from 0 to 43200.
Policy	Optional. The JSON-based access policy language required by Amazon to be applied to any new queues created by the process.
Maximum Message Size	Optional. The maximum message size in bytes permitted in this queue. Enter a value between 1000 and 256000.
Message Retention Period	Optional. The retention period, in seconds, for messages in this queue. Range: 60 seconds to 14 days.
Max Messages Per Poll	Optional. The maximum number of messages that the event source can receive in one poll.
Extend Message Visibility	Optional. Use this option to extend the message visibility timeout. Enabling this option extends the visibility timeout in the queue for a particular message when the visibility timeout is insufficient to fully process and delete the message. Default is **No**.
Dead queue	Optional. The name of the queue where you want to route messages that cannot be processed. **Note:** The question mark (?) character is prohibited in this field. It is not supported by Apache Camel and you cannot publish a connection that contains this value.
Wait Time	Duration in seconds for which the ReceiveMessage action waits until a message is in the queue to include in the response. Enter a value between 0 and 20.

Basic Event Target Properties

For each connection that you define, you can include one or more event targets that specify operations for writing files or messages, or when the event target is called from a process. For example, you might define an event target that reads from a process object and writes to comma-delimited files.

To set event target properties for the connection, from the **Event Targets** tab, click **Add Event Target**, and select the event target type from the available list.

The following table describes the basic properties:

Property	Description
Name	Required. Unique name for the event target.
Description	Optional. Description of the event target.

Amazon SQS Event Target Properties

For each connection, you can include one or more event targets that specify options for writing SQS messages when the event target is called from a process.

The following table describes the Amazon SQS event target properties:

Property	Description.
Queue Name	Required. Name of the Amazon SQS queue. If the named queue does not exist, the connection automatically creates the queue. **Note:** The question mark (?) character is prohibited in this field. It is not supported by Apache Camel and you cannot publish a connection that contains this value.
Query Parameters	Optional. You can specify other parameters that define how to connect to servers, if required by the SQS client. For example, you can use the standard URI query syntax to provide additional information for the client to successfully connect.
Queue Owner AWS Account ID	Optional. The AWS account ID of the account owner that created the queue (if the queue already exists).
Region	Optional. The AWS region for the SQS queue. For example: `us-west-1`
Content Format	Required. Identifies the format of the content to be processed. Select one of the following values: - **TEXT**. The content is a string and handled like plain text. - **XML**. The content is parsed and converted to an object or a list of process objects. - **JSON**. The content is parsed and converted to an object or a list of process objects.

The following table describes the advanced properties for an event target:

Property	Description
Visibility Timeout	Optional. Duration in seconds for which messages are hidden from subsequent retrieve requests after the messages are retrieved by a ReceiveMessage request.
Policy	Optional. The JSON-based access policy language required by Amazon to be applied to any new queues created by the process.
Maximum Message Size	Optional. The maximum message size in bytes permitted in this queue. Enter a value between 1000 and 256000.
Message Retention Period	Optional. The retention period, in seconds, for messages in this queue. Range: 60 seconds to 14 days.
Delay	Optional. The period, in seconds, for which the action for sending messages must be delayed.
Message Group Id Strategy	Required. Use this option for FIFO queues. The option defines the strategy for setting the messageGroupId attribute for the message. Select one of the following values: - useConstant. The **CamelSingleMessageGroup** string is used as the message group ID. - useExchangeId. The value of the custom Camel message header **AeMessageGroupId** is used as the message group ID. - usePropertyValue. The value of the custom Camel message header **AeMessageGroupId** is used as the message group ID. Default is **useConstant**.
Message Deduplication Id Strategy	Required. Use this option for FIFO queues. The option defines the deduplication strategy for the message. Select one of the following values: - useExchangeId. The messages are deduplicated by the Camel exchangeId. - useContentBasedDeduplication. The messages are deduplicated by the message content. No messageDeduplicationId is set. Default is **useExchangeId**.

SQS Message Payload

The event received by the process is an object with a payload that represents the SQS message data.

Similar to other connectors, the message payload for SQS might be made accessible to the process as a process object of type $any.

For example, if the event target payload has plain text content with attributes, the payload looks similar to the following example:

```
{
   "msg":{
      "headers":[
         {
            "name":"attr1",
            "value":"attr1_val"
         },
         {
            "name":"attr2",
            "value":"attr2_val"
```

```
            }
         ],
         "body":"some plain content"
      }
   }
```

The event response would be similar to the following example:

```
<field name="sqsResponse">
         <root xmlns="">
            <MD5OfBody>fd6fb6b12324123fc8473fc7391cfdb3</MD5OfBody>
            <messageId>910aac08-e530-4afd-8f08-3868bd9d321f</messageId>
            <delaySeconds>0</delaySeconds>
         </root>
      </field>
```

If the target type is XML or JSON and you send an object in the body of the message payload, it appears similar to the following example:

```
{
   "msg":{
      "headers":[
         {
            "name":"attr1",
            "value":"attr1_val"
         },
         {
            "name":"attr2",
            "value":"attr2_val"
         }
      ],
      "body":{
         "key":"value"
      }
   }
}
```

If the payload is XML, the field name is the element name of the list.

If the payload is JSON, the array does not have a field name as defined by the payload and you can specify an Objectlist Name for the Event Source so that the implicit object field name can be set.

SQS Processes

To create a process that handles SQS messages, perform the following steps:

1. Create a process and select a Secure Agent where the SQS broker must run.
2. Configure the other process properties.
3. On the **Start** tab, select **Event** in the **Binding** field, as SQS consumes events.
   You can see the available event sources (grouped by *Connection:event name*).
4. In the **Event Source Name** field, select an event source.

Process Example

To queue messages on the SQS queue, you might define a process that includes:

1. An Assignment step to assign the message to the orders queue.
2. A Wait step to include a timer.
3. A Service step that uses a connection and event target.

The event target can be handled like other service calls. The input argument is a process object that defines the data required for the transport.