Configuring Kafka Producers and Subscribers

Last Modified on 06/12/2025 4:50 am EDT

Starting with V10.0, Cora SeQuence has been renamed to Cora Orchestration.

V10.x

Overview

You configure Kafka activities to set up a messaging mechanism within Cora SeQuence or between Cora SeQuence and other applications. Apache Kafka® is a distributed streaming platform that is designed to be fast, scalable, and durable. Kafka is generally used to move data between systems or applications and to enable applications to consume data required to perform specific actions.

Use cases

Kafka messaging mechanism can be used in different scenarios. Following are a few examples:

- Asynchronous communication with external systems:
  - One-to-one communication scenario:
    1. Receive messages from an ERP system to initiate an invoice approval workflow.
    2. After invoice approval, send message back to the ERP system for further processing.
- Event publishing:
  - One-to-many communication scenario: Send messages to a topic that has multiple subscribers.
    1. Send a message to the topic after a payment process has completed.
    2. Multiple systems that subscribe to the topic perform an action based on the message.
- Multiple tasks per single message:
  - Handle a message sent from an external system to trigger multiple workflows:
    - A solution for welcoming a newly hired employee. One message with information about the new employee triggers multiple workflows, such as starting a workflow that requests the IT department to supply a laptop to the new employee, another workflow that requests the Security department to issue an ID tag, and yet another workflow for the HR department to add the employee to the relevant systems.

Configuration

Configuring a Kafka messaging on Cora SeQuence involves the following steps:

Steps	Performed by
1. Create the Kafka connection	Cora SeQuence Administrator
2. Add a Kafka producer **NOTE:** Required only for the Kafka Producer activity.	Cora SeQuence Administrator

Steps	Performed by
3. Configure the Kafka integration activities in the workflow: a. Configure the Kafka Producer activity b. Configure the Kafka Subscriber activity	Developer

For more details on how Apache Kafka works in Cora SeQuence, see this article.

Create the Kafka connection

Configure the connection to the Kafka service. You can create new connections, edit existing ones, or delete them.

Prerequisites

Before you create a Kafka connection string, make sure that you have:

- A Kafka deployment (either SAAS or IAAS) is up and running.
- The Kafka connection details, including the required credentials.

For more information on the Kafka server requirements, see this article.

Procedure

1. Go to **Administration > Global Settings > Kafka Connections**, and click **Add New Record**.
2. Enter a meaningful name for the Kafka connection string.
3. Enter the Connection URL.
4. If required, enter the relevant credentials to access the Kafka server.
5. By default, Cora SeQuence uses the parameters configured on the Kafka server. To change the default settings, click **Advanced Options**.

To learn more about Kafka parameters, refer to the Kafka documentation.

Add a Kafka Producer

Kafka producer is the mechanism that publishes messages to one or more topics. You need to set up at least one Kafka Producer.

Prerequisite

- Make sure that a Kafka connection exists.

Procedure

1. Go to **Administration > Global Settings > Kafka Connections**, and click **Add New Record**.
2. Enter a meaningful name for the Kafka Producer.
3. Select the Kafka connection.
4. By default, Cora SeQuence uses the parameters configured on the Kafka server. To change the default settings, click **Advanced Options**.
5. Click **Add**.

Delivery guarantee

By default, the Kafka Producer acknowledges message delivery. When a message is sent, the activity waits up to 303 seconds to receive the delivery report. This default timeout is calculated as the sum of `DeliveryReportWaitTimeout`, `LingerMs` and `MessageTimeoutMs` parameters from the Advanced Options.

To set up delivery guarantee, configure the following parameters under **Advanced Options**.

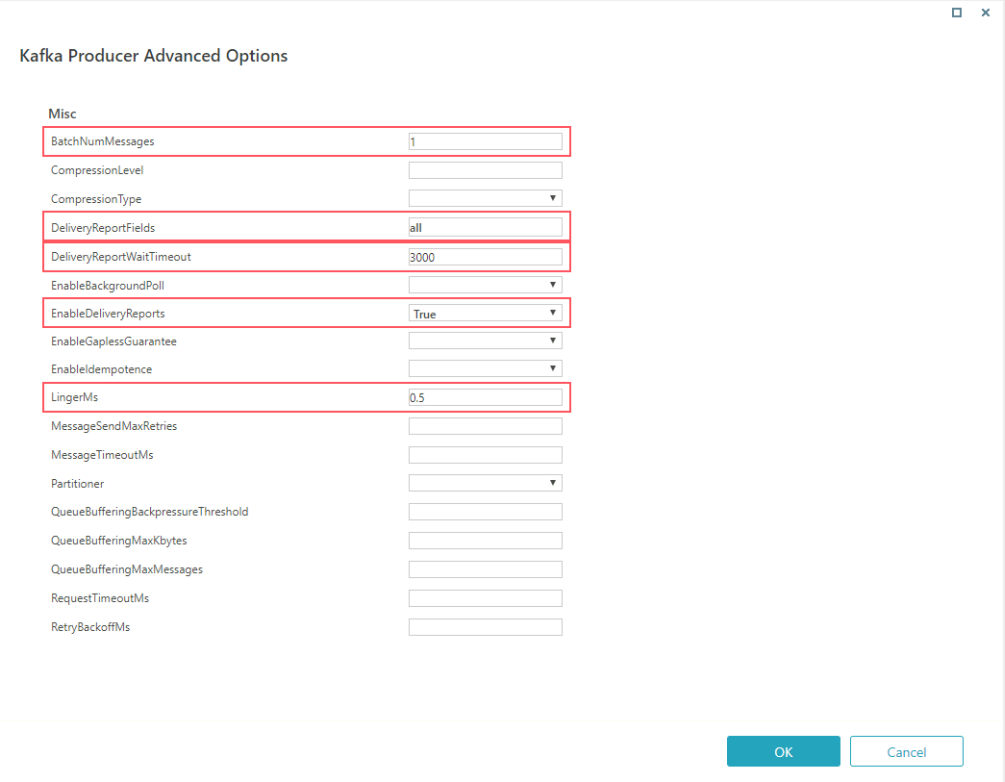
Parameter	Description	Unit	Default	Minimum	Maximum
BatchNumMessages	Determines the maximum number of messages per batch in MessageSet.	int	10000	1	20000
CompressionLevel	Determines the compression level parameter for algorithm selected by the `CompressionType` configuration property. Higher values result in better compression at the cost of more CPU usage. Usable range is algorithm-dependent: • [0-9] for gzip • [0-12] for lz4 • only 0 for snappy • -1 = codec-dependent default compression level	int	-1	-1	12
CompressionType	Determines the compression codec to use for compressing message sets. This is the default value for all topics.	enum	None	-	-
DeliveryReportFields	Determines the fields listed in the delivery reports. Supported values: `key, value, timestamp, headers, all, none`. Disabling unused delivery report fields improves the maximum throughput and reduces memory usage.	string	-	-	-
DeliveryReportWaitTimeout	Determines the time, in milliseconds, to wait for a response from the Kafka server. When the configured period of time elapses, a failure message is logged.	ms	3000	3000	50000

Parameter	Description	Unit	Default	Minimum	Maximum
EnableBackgroundPoll	Specifies whether or not the producer should start a background poll thread to receive delivery reports and event notifications. Generally, it should be set to true. If set to false, you need to call the Poll function manually.	boolean	True	-	-
EnableDeliveryReport	Determines whether or not to enable notification of delivery reports. Typically should be set to true. Set it to false for "fire and forget" semantics and a small boost in performance.	boolean	True	-	-
EnableGaplessGuarantee	When set to `true`, any error that could result in a gap in the produced message series, when a batch of messages fails, will raise a fatal error and stop the producer. Messages failing due to `MessageTimeoutMs` are not covered by this guarantee. Requires `EnableIdempotence=true`.	boolean	False	-	-
EnableIdempotence	When set to `true`, the producer will ensure that messages are successfully produced exactly once and in the original produce order.	boolean	False	-	-
LingerMs	Determines a delay, in milliseconds, before the Producer sends the message to the topic. The messages accumulate before constructing message batches (MessageSets) to transmit to brokers. A higher value allows larger and more effective (less overhead, improved compression) batches of messages to accumulate at the expense of increased message delivery latency.	ms	0.5	-	-

Parameter	Description	Unit	Default	Minimum	Maximum
MessageSendMaxRetries	Determines the number of times the system retries sending a failing Message. **NOTE** Retrying may cause reordering unless `EnableIdempotence` is set to true.	int	2	1	4
MessageTimeoutMs	Determines the local message timeout that is enforced locally and limits the time a produced message waits for successful delivery. A time of 0 is infinite. This is the maximum time kafka may use to deliver a message (including retries). Delivery error occurs when either the retry count or the message timeout are exceeded.	ms	300000	150000	600000
Partitioner	Values: • `Random` - random distribution • `Consistent` - CRC32 hash of key (Empty and NULL keys are mapped to single partition) • `ConsistentRandom` - CRC32 hash of key (Empty and NULL keys are randomly partitioned) • `Murmur2` - Murmur2 hash of key (NULL keys are mapped to single partition) • `Murmur2Random` - Murmur2 hash of key (NULL keys are randomly partitioned. This is functionally equivalent to the default partitioner).	enum	ConsistentRandom	-	-

Parameter	Description	Unit	Default	Minimum	Maximum
QueueBufferingBackpressureThreshold	Determines the threshold of outstanding not yet transmitted broker requests needed to backpressure the producer's message accumulator. If the number of not yet transmitted requests equals or exceeds this number, produce request creation that would have otherwise been triggered (for example, in accordance with `LingerMs`) will be delayed. A lower number yields larger and more effective batches. A higher value improves latency when using compression on slow machines.	int	1	0	2
QueueBufferingMaxKbytes	Determines the maximum total message size sum allowed on the producer queue. The queue is shared by all topics and partitions. This property has higher priority than `QueueBufferingMaxMessages`.	int	1048576	10	2097152
QueueBufferingMaxMessages	Determines the maximum number of messages allowed on the producer queue. The queue is shared by all topics and partitions.	int	100000	1	100000
RequestTimeoutMs	Determines the acknowledge timeout of the producer request.	ms	5000	2500	10000
RetryBackoffMs	Determines the backoff time before retrying a protocol request.	ms	100	50	200

Advanced Options screen



When you set up Kafka to acknowledge delivery, the Producer sends one message at a time. After the message is successfully sent, the workflow continues, and the activity is marked as successful.

- If there are connection issues, a failure response is immediately sent to Cora SeQuence.
- If other issues occur, such as topic overflow, Kafka returns a failure status after a certain delay.

After the failure message is returned, Cora SeQuence sets the activity to failed.

> NOTE
> Carefully consider between the need for delivery guarantee and performance requirements. When you configure delivery guarantee, the system performs additional actions and activities that impact performance.

To learn more about Kafka parameters, refer to the Kafka documentation.

Configure the Kafka Producer activity

The Kafka Producer *activity* is responsible for sending the message object and the topic name to the Kafka *Producer*.
Each activity instance can define a different message based on Cora SeQuence expressions.

Prerequisite

- A Kafka Producer has been configured.

Procedure

1. To add a Kafka Producer activity to your workflow, in the App Studio, select **Integration>Kafka Producer**.
2. On the Kafka Producer Properties screen, enter a significant name and alias, and then click **Next**.
3. Select a Kafka Producer from the list, or create a new one.

4. Click **Next**.
5. Click `ProducerRecord message` , and clear the **IsNull** option.
6. To show the configurable properties, expand the `ProducerRecord message` node.
    - Enter topic name.
      If the topic does not exist, depending on Kafka's configuration, a topic can be automatically created.
    - Set the content and body of the message.
    - Set the key.
7. Click **Finish**.

Configure the Kafka Subscriber activity

The subscriber connects to Kafka and retrieves the message from a topic. Each subscriber connects to a specific topic. The Kafka Subscriber is a JES job.

Prerequisites

- A Kafka producer has been configured.
- Make sure that you have the relevant Group ID.
  To learn more about groups in Kafka, refer to this page.
- Obtain the relevant Kafka topic name from the Project Manager or Business Analyst.

Procedure

1. To add a Kafka Subscriber activity to your workflow, in the App Studio, select **Integration>Kafka Subscriber**.
2. Click the  Kafka Subscriber activity.
3. On the Kafka Subscriber Activity Properties screen, enter a significant name for the activity, and then click **Next**.
4. On the job tab, set the following:
    - **Name**: Enter a name for the actual job performed by the Kafka Subscriber activity.
    - **Scaling**: Set the required scaling settings. (Scaling is available only for **Cora SeQuence V9.3** and later versions.)
      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:
    - **Kafka Connection** (mandatory): Select the relevant connection string.
    - **Group ID** (mandatory): Enter the group ID relevant to your implementation.
    - **Topic Name** (mandatory): Enter the Kafka topic name.
7. By default, Cora SeQuence uses the parameters configured on the Kafka server. To change the default settings, click **Advanced Options**.
8. If your implementation requires that each message is read and processed before moving on to the next message, select **Acknowledgement**.
9. Click **Finish**.

To learn more about Kafka parameters, refer to the Kafka documentation.

V9.2.2 - 9.3

Overview

You configure Kafka activities to set up a messaging mechanism within Cora SeQuence or between Cora SeQuence and other applications. Apache Kafka® is a distributed streaming platform that is designed to be fast, scalable, and durable. Kafka is generally used to move data between systems or applications and to enable applications to consume data required to perform specific actions.

Use cases

Kafka messaging mechanism can be used in different scenarios. Following are a few examples:

- Asynchronous communication with external systems:
  - One-to-one communication scenario:
    1. Receive messages from an ERP system to initiate an invoice approval workflow.
    2. After invoice approval, send message back to the ERP system for further processing.
- Event publishing:
  - One-to-many communication scenario: Send messages to a topic that has multiple subscribers.
    1. Send a message to the topic after a payment process has completed.
    2. Multiple systems that subscribe to the topic perform an action based on the message.
- Multiple tasks per single message:
  - Handle a message sent from an external system to trigger multiple workflows:
    - A solution for welcoming a newly hired employee. One message with information about the new employee triggers multiple workflows, such as starting a workflow that requests the IT department to supply a laptop to the new employee, another workflow that requests the Security department to issue an ID tag, and yet another workflow for the HR department to add the employee to the relevant systems.

Configuration

Configuring a Kafka messaging on Cora SeQuence involves the following steps:

Steps	Performed by
1. Create the Kafka connection	Cora SeQuence Administrator
2. Add a Kafka producer NOTE Required only for the Kafka Producer activity.	Cora SeQuence Administrator
3. Configure the Kafka integration activities in the workflow: a. Configure the Kafka Producer activity b. Configure the Kafka Subscriber activity	Developer

For more details on how Apache Kafka works in Cora SeQuence, see this article.

Create the Kafka connection

Configure the connection to the Kafka service. You can create new connections, edit existing ones, or delete

them.

Prerequisites

Before you create a Kafka connection string, make sure that you have:

- A Kafka deployment (either SAAS or IAAS) is up and running.
- The Kafka connection details, including the required credentials.

For more information on the Kafka server requirements, see this article.

Procedure

1. Go to **Administration > Global Settings > Kafka Connections**, and click **Add New Record**.
2. Enter a meaningful name for the Kafka connection string.
3. Enter the Connection URL.
4. If required, enter the relevant credentials to access the Kafka server.
5. By default, Cora SeQuence uses the parameters configured on the Kafka server. To change the default settings, click **Advanced Options**.

To learn more about Kafka parameters, refer to the Kafka documentation.

Add a Kafka Producer

Kafka producer is the mechanism that publishes messages to one or more topics. You need to set up at least one Kafka Producer.

Prerequisite

- Make sure that a Kafka connection exists.

Procedure

1. Go to **Administration > Global Settings > Kafka Connections**, and click **Add New Record**.
2. Enter a meaningful name for the Kafka Producer.
3. Select the Kafka connection.
4. By default, Cora SeQuence uses the parameters configured on the Kafka server. To change the default settings, click **Advanced Options**.
5. Click **Add**.

Delivery guarantee

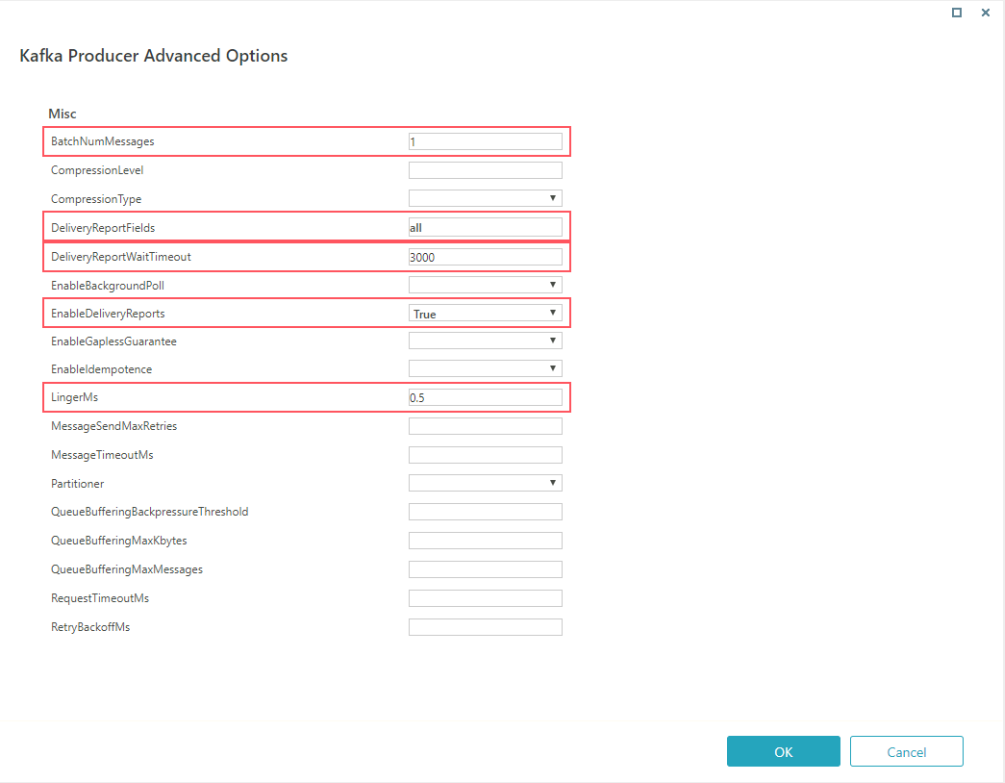
By default, the Kafka Producer does not acknowledge message delivery. When setting up Kafka to handle financial transactions, for example, you need to make sure that every message is delivered.

To set up delivery guarantee, configure the following parameters under **Advanced Options**.

Parameter	Description	Value
BatchNumMessages	Determines the number of messages per batch.	1
DeliveryReportFields	Determines the fields listed in the delivery reports. Supported values: key, value, timestamp, headers, all, none.	all

Parameter	Description	Value
DeliveryReportTimeout	Determines the time, in milliseconds, to wait for a response from the Kafka server. When the configured period of time elapses, a failure message is logged.	3000
EnableDeliveryReport	Determines whether or not to enable notification of delivery reports.	True
LingerMs	Determines a delay, in milliseconds, before the Producer sends the message to the topic.	0.5

Advanced Options screen



When you set up Kafka to acknowledge delivery, the Producer sends one message at a time. After the message is successfully sent, the workflow continues, and the activity is marked as successful.

- If there are connection issues, a failure response is immediately sent to Cora SeQuence.
- If other issues occur, such as topic overflow, Kafka returns a failure status after a certain delay.

After the failure message is returned, Cora SeQuence sets the activity to failed.

NOTE
Carefully consider between the need for delivery guarantee and performance requirements. When you configure delivery guarantee, the system performs additional actions and activities that impact performance.

To learn more about Kafka parameters, refer to the Kafka documentation.

Configure the Kafka Producer activity

The Kafka Producer *activity* is responsible for sending the message object and the topic name to the Kafka *Producer*.
Each activity instance can define a different message based on Cora SeQuence expressions.

Prerequisite

- A Kafka Producer has been configured.

Procedure

1. To add a Kafka Producer activity to your workflow, in the App Studio, select **Integration>Kafka Producer**.
2. On the Kafka Producer Properties screen, enter a significant name and alias, and then click **Next**.
3. Select a Kafka Producer from the list, or create a new one.
4. Click **Next**.
5. Click `ProducerRecord message` , and clear the **IsNull** option.
6. To show the configurable properties, expand the `ProducerRecord message` node.
    - Enter topic name.
      If the topic does not exist, depending on Kafka's configuration, a topic can be automatically created.
    - Set the content and body of the message.
    - Set the key.
7. Click **Finish**.

Configure the Kafka Subscriber activity

The subscriber connects to Kafka and retrieves the message from a topic. Each subscriber connects to a specific topic. The Kafka Subscriber is a JES job.

Prerequisites

- A Kafka producer has been configured.
- Make sure that you have the relevant Group ID.
  To learn more about groups in Kafka, refer to this page.
- Obtain the relevant Kafka topic name from the Project Manager or Business Analyst.

Procedure

1. To add a Kafka Subscriber activity to your workflow, in the App Studio, select **Integration>Kafka Subscriber**.
2. Click the  Kafka Subscriber activity.
3. On the Kafka Subscriber Activity Properties screen, enter a significant name for the activity, and then click **Next**.
4. On the job tab, set the following:
    - **Name**: Enter a name for the actual job performed by the Kafka Subscriber activity.
    - **Scaling**: Set the required scaling settings. (Scaling is available only for **Cora SeQuence V9.3** and later versions.)
      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:
    - **Kafka Connection** (mandatory): Select the relevant connection string.
    - **Group ID** (mandatory): Enter the group ID relevant to your implementation.
    - **Topic Name** (mandatory): Enter the Kafka topic name.
7. By default, Cora SeQuence uses the parameters configured on the Kafka server. To change the default settings, click **Advanced Options**.
8. If your implementation requires that each message is read and processed before moving on to the next message, select **Acknowledgement**.
9. Click **Finish**.

To learn more about Kafka parameters, refer to the Kafka documentation.