Informatica® Cloud Application Integration
December 2023

Kafka Connector Guide

Informatica Cloud Application Integration Kafka Connector Guide
December 2023

Publication Date: 2024-04-04

Table of Contents

Preface

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

This guide assumes that you have an understanding of the Kafka message streaming platform and how to create connections and processes in Application Integration.

CHAPTER 1

Introduction to Kafka Connector

This chapter includes the following topics:

Kafka Overview

Kafka is a distributed publish-subscribe messaging system that is fast, scalable, and durable.

Kafka stores messages in the form of topics. Consumers read data from a topic. Producers write data to a topic. Kafka topics are partitioned and replicated across multiple nodes thereby allowing distributed processing.

Kafka consumers and producers do not exchange data directly between themselves. A Kafka broker works as an intermediary between consumers and producers to disseminate data. A Kafka cluster can consist of multiple brokers. Kafka uses ZooKeeper to manage the Kafka cluster.

Kafka Connector Overview and Implementation

You can use Kafka Connector to perform the following tasks:

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

Kafka Connector is implemented by using a Camel event listener. Processes that connect to Kafka can be triggered upon an event such as arrival of a Kafka message in a Kafka stream.

CHAPTER 2

Kafka Connections and Processes

This chapter includes the following topics:

Kafka Connections Overview

Use a Kafka connection to connect to a Kafka stream and read data from or write data to a topic.

You can also add additional Kafka brokers to form a Kafka cluster and create a Kafka connection. Use the Kafka connection to access an Apache Kafka broker, and read and write data.

After you create a Kafka connection, you can validate and save the connection.

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

To view a video about creating a Kafka connection, and download a sample Kafka connection, see the following community article:

https://knowledge.informatica.com/s/article/DOC-18183

High availability with Kafka brokers

You can specify a list of Kafka brokers to form a Kafka cluster and ensure high availability.

You can specify the Kafka broker connection string in the `<host_name>:<port_number>` or `<IP_address>:<port_number>` format where host, IP address, and port belong to the Kafka server where the broker is running. To connect through other Kafka brokers when one broker is down, you can specify multiple hosts in the following format:

```
hostname1:port1,hostname2:port2,hostname3:port3
```

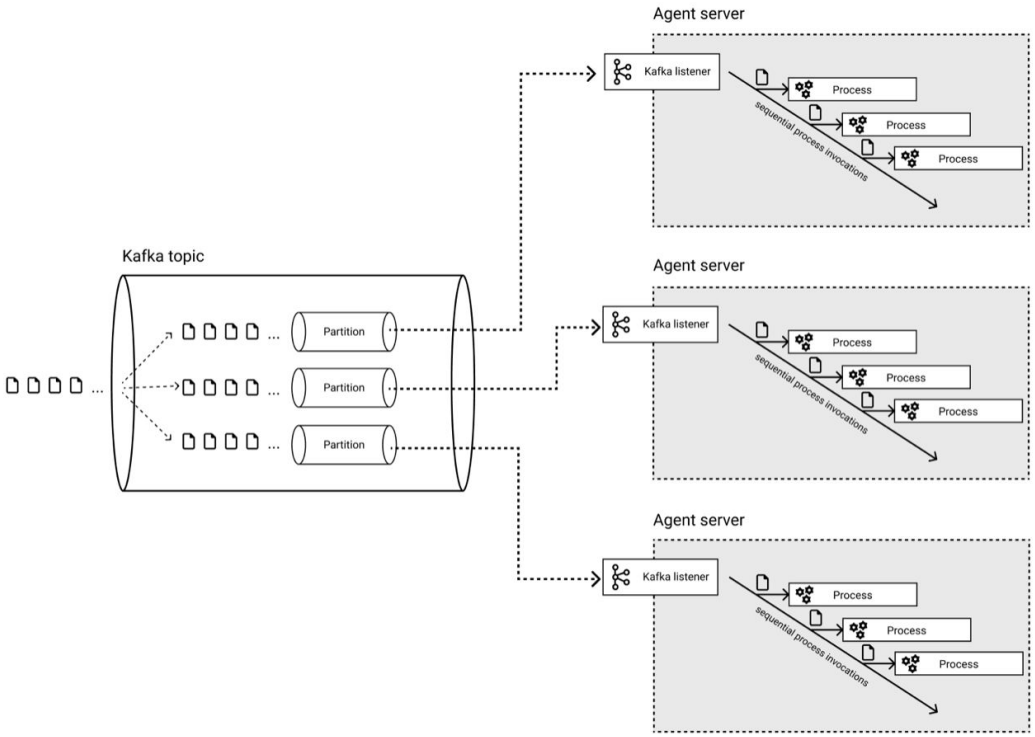
If any broker in a Kafka cluster goes down, the other active brokers in the cluster continue to listen to the topic.

Kafka partitions

You can configure partitions in Kafka to process messages in parallel and enable concurrent message consumption.

The number of partitions must be equal to the number of concurrent consumers of the client. If the number of partitions is less than the number of consumers, some consumer instances will not receive messages to process. If the number of partitions is more than the number of consumers, some consumer instances will process messages from several partitions.

The following image shows how a Kafka listener works with partitions in a Secure Agent group:



In a Secure Agent cluster, the Kafka consumer is active only on one node. However, if the same Kafka consumer runs on a Secure Agent group, it processes messages in parallel across all the nodes in the group. Kafka distributes messages among client instances without duplication.

The Kafka partitions and the Kafka consumers that run on a Secure Agent group distribute the load between the Secure Agent nodes. This configuration also scales to process more data when you add more partitions and Secure Agents to the group. This increases the Kafka message throughput.

You can configure multiple concurrent consumers by enabling the load balancing option in the **Event Sources** tab of the Kafka connection. You can use this option if your Process Server uses a Secure Agent cluster configuration so that it creates one consumer per Secure Agent.

For more information about the event source properties, see "Kafka Event Source Properties" on page 11.

Ensure that the messages are evenly distributed between partitions so that all the Secure Agents in a group are equally loaded. This will provide maximum throughput and evenly share the load on the Secure Agents.

For more information about Kafka partitioning, see the Kafka documentation.

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. Unique name for the Kafka connection that identifies it in the Process Designer. The name must start with an alphabet and can contain only alphabets, numbers, or 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 you want to use for the connector or service connector. Select **Kafka**.
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 before changing the runtime environment from the Cloud Server to the Secure Agent, or from the Secure Agent to the Cloud Server. **Note:** You can publish Kafka connections only on the AWS PODs on the Cloud Server.
Connection Test	Indicates whether the connection test was successful or not. By default, the property displays the results of the connection test.
OData-Enabled	Not supported for Kafka Connector.

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

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

Kafka Connection Properties

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

Connection Settings

You must specify the Kafka host and Kafka port that you want to connect to. You can also enable SSL authentication and SASL authentication.

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

Property	Description
Kafka Host	Required. The primary Kafka host that the Secure Agent or the Cloud Server must connect to with a TCP connection.
Kafka Port	Required. The primary Kafka port number that the Secure Agent or the Cloud Server must connect to with a TCP connection.
Additional Kafka Brokers	Comma-separated list of the Kafka brokers that the Secure Agent must connect to with a TCP connection. Enter the host name or IP address of the system where the Kafka broker is running and a port number without any space in the following format: `<host_name>:<port_number>` or `<IP_address>:<port_number>` For example, `CAI1A:9090,CAI2A:9091,10.75.167.66:9092,10.75.167.67:9093` **Note:** - When you connect to a Kafka broker over SSL, you must specify the fully qualified domain name for the host name. Otherwise, the connection test fails with an SSL handshake error. - When you connect to a Kafka broker over SSL and SASL, you must ensure that all the brokers have the same SSL and SASL settings. - When you test a connection, you must ensure that a Kafka server is up and running on all the brokers. Otherwise, the connection test fails. - You must have a unique host name for each IP address. Otherwise, an error occurs.

Other Attributes	Optional. A list of common Apache Camel configuration properties to connect to the Kafka brokers.
	Specify the properties in key-value pairs. Separate multiple properties with an ampersand character (&) without any space.
	For example, enter the following phrase to specify the client ID of the application making request and to use batching for processing or streaming:
	`clientId=client123&batching=false`
	If you want to set additional properties for a Kafka consumer or Kafka producer that you can't set directly in the Camel configuration, prefix the properties with `additionalProperties`.
	For example,
	`additionalProperties.transactional.id=12345&additionalProperties.schema.registry.url=\http://localhost:8811/avro`
	For more information about the properties that you can specify, see the Apache Camel documentation.

Property	Description
Use SSL	Optional. Determines whether SSL authentication must be used. Default is **No**.
Use SASL	Optional. Determines whether SASL authentication must be used. To configure Kerberos authentication for the Kafka client, select **Yes**. Default is **No**.

SSL Settings

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

Informatica recommends that if you are connecting to a Kafka broker that is published on the Cloud Server, you must configure SSL authentication.

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

Property	Description
SSL Truststore Location	The absolute path and file name of the truststore file on the Secure Agent machine that contains the SSL certificate to establish two-way secure communication with the Kafka broker. For example: `C:/SSL/Certs_208/icrt-truststore.jks` If you choose to run the Kafka connection on the Cloud Server, you must save the truststore file of the JKS (Java KeyStore) format on your local machine. Use the **Choose File** option to select the truststore file. The maximum allowed file size is 50 KB.
SSL Truststore Password	The password for the truststore file that contains the SSL certificate.
SSL 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 a two-way secure communication with the Kafka broker. For example: `C:/SSL/Keys/icrt-keystore.jks` If you choose to run the Kafka connection on the Cloud Server, you must save the keystore file of the JKS (Java KeyStore) format on your local machine. Use the **Choose File** option to select the keystore file. The maximum allowed file size is 50 KB.
SSL Keystore Password	The password for the keystore file required for secure communication.
SSL Key Password	The SSL key password of the client.

SASL Settings

You can configure SASL authentication so that the Kafka broker can authenticate the Kafka producer and the Kafka consumer. Kafka uses the Java Authentication and Authorization Service (JAAS) for SASL authentication. To enable SASL authentication, you must specify the SASL mechanism that the Kafka broker must use for authentication. Based on the SASL mechanism value you specify, you must also provide the formatted JAAS configuration that the Kafka broker must use for authentication.

You can configure Kerberos authentication for a Kafka client by placing the required Kerberos configuration files on the Secure Agent machine and specifying the required JAAS configuration in the Kafka connection. The JAAS configuration defines the keytab and principal details that the Kafka broker must use to authenticate the Kafka client.

Note: The Kerberos server where the Kafka cluster is running and the Process Server must use the same time zone. Otherwise, an error occurs when producing a message.

If you enable SASL authentication, you must configure the following authentication properties in the **SASL Settings** section:

Property	Description
SASL Mechanism	The SASL mechanism that the Kafka broker must use to authenticate the Kafka producer and the Kafka consumer. Enter one of the following values: - **PLAIN**. Use this value to configure plain SASL authentication for the Kafka client. - **GSSAPI**. Use this value to configure Kerberos authentication for the Kafka client. Default is **PLAIN**.

Jaas Config	The formatted JAAS configuration that the Kafka broker must use to authenticate the Kafka producer and the Kafka consumer.
	To use the PLAIN SASL mechanism, you must specify the user name and password to connect to Kafka in the JAAS configuration as shown in the following example:
	`org.apache.kafka.common.security.plain.PlainLoginModule required username="<value>" password="<value>";`
	To configure Kerberos authentication for the Kafka client, you must specify the keytab and principal details in the JAAS configuration as shown in the following example:
	`com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true storeKey=true keyTab="<path and file name of the keytab file on the Secure Agent machine>" serviceName="kafka" principal="kafka/<host_name>";`

Kafka Event Source Properties

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

For each Kafka 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 Kafka topic for new messages. After you define an event source for a Kafka connection, you can publish the connection. You can then access the event source in a process and deploy the process to consume the process objects generated by the event source downstream.

When you run a Kafka connection on the Cloud Server, the Kafka consumer can read a maximum of 5 MB Kafka messages.

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

Note: If the content format specified in the event source of the Kafka connection does not match what the producer sends, the consumer process won't get triggered. You can see the error details in the Catalina logs.

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.
Description	Optional. A description for the Kafka event source that appears in the Process Designer.
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. **Note:** You must not update this value when the Kafka consumer process is running. Otherwise, you might see duplicate messages. Default is **Yes**.

The following table describes the Kafka 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. When you enable this option, the Process Server distributes the routes across different Secure Agent machines in a Secure Agent cluster to ensure load balancing. 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 for load balancing.
Topic	Required. The name of the Kafka topic from which you want to read messages.
Group Id	Required. The ID of the group that the Kafka consumer belongs to.

Event Source Property	Description
Content Format	Required. The format in which you want to read the messages. Select one of the following options: - TEXT - XML - JSON Default is **TEXT**.

Other Attributes	Optional. A list of additional Apache Camel configuration properties for the Kafka consumer.
	Specify the properties in key-value pairs. Separate multiple properties with an ampersand character (&) without any space.
	For example, enter the following phrase to specify the maximum amount of data the server should return for a fetch request and the maximum amount of time the client will wait for the response of a request:
	`fetchMaxBytes=50000000&consumerRequestTimeoutMs=30000`
	When you configure consumer-specific connection properties on both the **Event Sources** tab and the **Properties** tab, the connection properties on the **Event Sources** tab take precedence.
	To increase the throughput and invoke more processes simultaneously, you can configure the following consumer-specific properties:
	`maxPollRecords`. Specifies the maximum number of records to return in a single call to poll messages from broker per consumer. Default is 500.
	`maxPollIntervalMs`. Specifies the maximum delay between the invocations of poll() when using consumer group management. Default is 300000 milliseconds or 5 minutes.
	`partitionAssignor`. Specifies the class name of the partition assignment strategy that the client uses to distribute partition ownership amongst consumer instances when group management is used. Default is `org.apache.kafka.clients.consumer.RoundRobinAssignor`.
	For example, enter the following phrase to increase the throughput:
	`maxPollRecords=100&maxPollIntervalMs=480000&partitionAssignor=org.apache.kafka.clients.consumer.RoundRobinAssignor`
	For more information about the properties that you can specify, see the Apache Camel documentation.

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*.

Kafka Event Target Properties

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

For each Kafka connection that you create, you can include one or more event targets that specify operations for writing 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 a Kafka topic.

When you run a Kafka connection on the Cloud Server, the Kafka producer can write a maximum of 5 MB Kafka messages.

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

Note: If the content format specified in the event source of the Kafka connection does not match what the producer sends, the consumer process won't get triggered. You can see the error details in the Catalina logs.

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.
Description	Optional. A description for the Kafka event target that appears in the Process Designer.

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

Event Target Property	Description
Topic	Required. The name of the Kafka topic to which you want to write messages.
Group Id	Required. This field is reserved for future use. However, you must enter a value in this field because it is a required field. Kafka Connector ignores the value that you enter.
Content Format	Required. The format in which you want to write the messages. Select one of the following options: - TEXT - XML - JSON Default is **TEXT**.
Other Attributes	Optional. A list of additional Apache Camel configuration properties for the Kafka producer. Specify the properties in key-value pairs. Separate multiple properties with an ampersand character (&) without any space. For example, enter the following phrase to group records into fewer requests whenever multiple records are being sent to the same partition and the size of the TCP receive buffer to use when reading data: `batchSize=10000&receiveBufferBytes=60000` When you configure producer-specific connection properties in both the **Event Targets** tab and the **Properties** tab, the connection properties in the **Event Targets** tab take precedence. For more information about the properties that you can specify, see the Apache Camel documentation.

Configuring Kerberos Authentication for a Kafka Client

You can configure Kerberos authentication for a Kafka client by placing the required Kerberos configuration files on the Secure Agent machine and specifying the required JAAS configuration in the Kafka connection.

The JAAS configuration defines the keytab and principal details that the Kafka broker must use to authenticate the Kafka client.

1. Get the following files from your Kafka Kerberos administrator.
   - `kafka_server.keytab`
   - `krb5.ini` or `krb5.conf`
   - `kafka_jaas.conf`
2. Copy the `kafka_server.keytab` file to the Secure Agent machine.
3. On the Secure Agent machine, based on the operating system that you use, copy the `krb5.ini` file or the `krb5.conf` file to the specified location:

Operating System	File Location
Windows	`C:\\Windows\krb5.ini`
Linux	`/etc/krb5.conf`

4. Copy all the broker and KDC server entries from the Kafka Kerberos environment to the `hosts` file of the Secure Agent machine.
5. Perform the following steps in the Kafka connection:
   a. Enter the host name and port number of the Kafka broker that is in a Kerberized domain.
   b. In the **Use SASL** field, select the value as **Yes**.
   c. In the **SASL Mechanism** field, enter the value as **GSSAPI**.
   d. In the **Jaas Config** field, provide details about the Kerberos keytab file and the principal from the `kafka_jaas.conf` file as shown in the following example:

```
com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true storeKey=true
keyTab="<path and file name of the keytab file on the Secure Agent machine>"
serviceName="kafka" principal="<principal value from the kafka_jaas.conf file>";
```

6. Save and publish the Kafka connection.

Kafka Process Creation

After you create a Kafka connection, a Kafka consumer, and a Kafka producer, you can use them in a process. The Kafka connection, Kafka consumer process, and the Kafka producer process must run on the same Secure Agent.

Kafka consumer process

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

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

Kafka consumers can consume headers sent with Kafka messages. The message headers are displayed in the **event** input field of the **Process View Detail** page in Application Integration Console and can be used in a process.

Note: The headers values are converted to string by default. You must ensure that the header uses a basic primitive type such as integer, floating point, or string. Otherwise, you might see random characters.

Kafka producer process

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

- Add an Assignment step to prepare the Kafka message that you want to write to the topic.
- Add a Service step to select the Kafka producer. In the input field, select the Kafka message that you prepared in the Assignment step.
- If you want to receive a response ID after you run a process, you can configure an output field to receive the Kafka response.

To view a video about reading and writing Kafka messages, and download a sample process, see the following community article:

https://knowledge.informatica.com/s/article/DOC-18233

Rules and guidelines for Kafka connection

After you publish a Kafka connection on the Cloud Server, if the Kafka broker goes down after processing certain messages, and you restart the Kafka broker within 24 hours, you do not see duplicate messages. This guarantees data accuracy and integrity, and ensures that there is no data loss.

For example, after you publish a Kafka connection, consider that the Kafka broker goes down after reading 500 messages and processing 100 messages. If the Kafka broker is restarted and a connection is established within the next 24 hours, the remaining 400 messages will get processed.

In case you want to process all the messages again, you must unpublish and publish the connection.

INDEX

K