Cloudera Runtime 7.3.2

Kafka Connect

Date published: 2019-08-22
Date modified: 2026-03-31



https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Kafka Connect Overview....6

Setting up Kafka Connect in Cloudera Base on premises....6

Using Kafka Connect....7
Configuring the Kafka Connect Role....8
Managing, Deploying and Monitoring Connectors....8
Writing Kafka data to Ozone with Kafka Connect....9
Writing data to Ozone in an unsecured cluster with Kafka Connect....9
Writing data to Ozone in a Kerberos and TLS/SSL enabled cluster with Kafka Connect....11
Using the AvroConverter....13
Enabling and configuring the AvroConverter....15
Configuring AvroConverter for Debezium connectors with Schema Registry integration....17
Reading Debezium Avro records from Kafka....18
Configuring EOS for source connectors....18

Securing Kafka Connect....20
Kafka Connect to Kafka broker security....20
Configuring TLS/SSL encryption for the Kafka Connect role....21
Configuring Kerberos authentication for the Kafka Connect role....22
Kafka Connect REST API security....22
Kafka Connect REST API authentication....22
Kafka Connect Rest API authorization....24
Kafka Connect connector configuration security....25
Kafka Connect Secrets Storage....26
Configuring connector JAAS configuration and Kerberos principal overrides....34
Configuring a Nexus repository allow list....35

Single Message Transforms....35
Configuring an SMT chain....36
ConvertFromBytes....38
ConvertToBytes....40

Connectors....42
Installing Kafka Connect connectors....45
Installing custom developed Kafka Connect connectors....46
Installing FileStream connectors....47
Setting up the Debezium Db2 Source connector....48
Setting up the Debezium MySQL Source connector....49
Setting up the Debezium Oracle Source connector....50
Setting up the Debezium PostgreSQL Source connector....51
Setting up the Debezium SQL Server Source connector....52
History topic producer and consumer properties reference....53
Setting up the Apache Iceberg Sink connector....54

Example configurations ... 57
HTTP Source connector ... 60
Stateless NiFi Source properties reference ... 62
HTTP Source properties reference ... 66
JDBC Source connector ... 70
Stateless NiFi Source properties reference ... 72
JDBC Source properties reference ... 76
JMS Source connector ... 82
Stateless NiFi Source properties reference ... 84
JMS Source properties reference ... 89
MQTT Source connector ... 92
Stateless NiFi Source properties reference ... 94
MQTT Source properties reference ... 98
SFTP Source connector ... 102
Stateless NiFi Source properties reference ... 104
SFTP Source properties reference ... 108
Stateless NiFi Source and Sink ... 119
Dataflow development best practices for Stateless NiFi ... 121
Kafka Connect worker assignment and Stateless NiFi ... 128
Kafka Connect log files and Stateless NiFi ... 130
Kafka Connect tasks and Stateless NiFi ... 130
Developing a dataflow for Stateless NiFi ... 133
Deploying a dataflow using Stateless NiFi ... 137
Downloading and viewing the predefined Stateless NiFi dataflows shipped in Cloudera ... 140
Configuring flow.snapshot for Stateless NiFi connectors ... 141
Tutorial: developing and deploying a JDBC Source dataflow in Kafka Connect using Stateless NiFi ... 143
Syslog TCP Source connector ... 153
Stateless NiFi Source properties reference ... 156
Syslog TCP Source properties reference ... 160
Syslog UDP Source connector ... 168
Stateless NiFi Source properties reference ... 170
Syslog UDP Source properties reference ... 174
ADLS Sink connector ... 180
Stateless NiFi Sink properties reference ... 183
ADLS Sink properties reference ... 187
Amazon S3 Sink ... 196
Configuration example ... 196
Amazon S3 Sink properties reference ... 197
HDFS Sink ... 198
Configuration example for writing data to HDFS ... 198
Configuration example for writing data to Ozone FS ... 200
HDFS Sink properties reference ... 201
HDFS Stateless Sink connector ... 203
Stateless NiFi Sink properties reference ... 209
HDFS Stateless Sink properties reference ... 212
HTTP Sink connector ... 220
Stateless NiFi Sink properties reference ... 222
HTTP Sink properties reference ... 226
InfluxDB Sink connector ... 233
Stateless NiFi Sink properties reference ... 235
InfluxDB Sink properties reference ... 239
JDBC Sink connector ... 244
Stateless NiFi Sink properties reference ... 246
JDBC Sink properties reference ... 250
Kudu Sink connector ... 256

Stateless NiFi Sink properties reference 258
Kudu Sink properties reference 262
S3 Sink connector 268
Stateless NiFi Sink properties reference 272
S3 Sink properties reference 275

Kafka Connect Overview

Get started with Kafka Connect.

Kafka Connect is a tool for streaming data between Apache Kafka and other systems in a reliable and scalable fashion. Kafka Connect makes it simple to quickly define connectors that move large collections of data into and out of Kafka. Source connectors can ingest entire databases or collect metrics from all your application servers into Kafka topics, making the data available for stream processing with low latency. Sink connectors can deliver data from Kafka topics into secondary storage and query systems or into batch systems for offline analysis.

Kafka Connect in Cloudera is shipped with many differentCloudera-developed as well as publicly available sink and source connectors. Each of which cover a specific use case for streaming data. In addition to the connectors available by default, installing custom developed or third-party connectors is also possible. All connectors can be deployed, managed, and monitored using the Streams Messaging Manager UI (recommended), Streams Messaging Manager REST API, or Kafka Connect REST API.

Related Information

Setting up Kafka Connect in Cloudera Base on premises
Using Kafka Connect
Securing Kafka Connect
Single Message Transforms
Connectors
Streams Messaging Reference

Setting up Kafka Connect in Cloudera Base on premises

Learn how to set up Kafka Connect in an existing Cloudera Base on premises cluster.

About this task

Kafka Connect in Cloudera is provided in the form of a Kafka service role. The role is called Kafka Connect. To set up Kafka Connect in your cluster, you must add one or more Kafka Connect role instances to the Kafka service in your cluster.



Note: If you want to set up Kafka Connect when deploying a new instance of Kafka, start by using the **Add Service** wizard to add and configure a new Kafka service. Follow the prompts given by the wizard. On the **Assign Roles** page, assign one or more hosts for the Kafka Connect role. On the **Review Changes** page configure Kafka Connect properties based on your cluster and requirements.

Procedure

1. Select the Kafka service.
2. Go to Instances.
3. Click Add Role Instances.
4. Select hosts for the Kafka Connect role by clicking the Select Hosts box under Kafka Connect.
5. Select one or more host and click Ok.
6. Click Continue.
7. Review and configure the properties available on the Review Changes page based on your cluster and requirements.
8. Start the Kafka Connect role.

   By default Kafka Connect roles do not start when first added.

a) Select Kafka Connect roles by checking the checkbox next to each role.
b) Click Actions for SelectedStart .
c) Click Start when prompted.

Results

One or more Kafka Connect roles are deployed and running on your cluster.

What to do next

Integrate Kafka Connect with Streams Messaging Manager. Integration enables the **Connect** section in Streams Messaging Manager, which allows you to interact with Kafka Connect through a UI interface. Integration is enabled by default in most scenarios. Manual configuration is only required if the Kafka Connect role instances and Streams Messaging Manager are located on different Cloudera clusters.

Related Information

Integrating Kafka Connect with Streams Messaging Manager

Using Kafka Connect

Learn how you can manage, monitor and configure Kafka Connect.

There are multiple ways that you can manage, monitor and configure Kafka Connect in Cloudera. It is important to make a distinction between the Kafka Connect roles and Kafka Connect connectors as they are managed through different interfaces.

The Kafka Connect Role

Kafka Connect roles are roles that you can deploy within a Kafka service. They represent Kafka Connect workers. Because these roles are deployed with and managed by Cloudera Manager, their management and configuration is done through Cloudera Manager. In other words the Kafka Connect workers you deploy are managed and configured in Cloudera Manager. For more information see *Configuring Kafka Connect* and the Cloudera Manager documentation.

Kafka Connect Connectors

Connectors are not managed by Cloudera Manager, instead multiple other interfaces can be used to interact with them. These are the following:

- Streams Messaging Manager UI

  The Streams Messaging Manager UI is a recommended interface in Cloudera to manage connectors. Comprehensive documentation is provided on the usage of the UI in *Monitoring Kafka Connect*.
- The Streams Messaging Manager REST API

  The Streams Messaging Manager REST API is a recommended interface in Cloudera to manage connectors. For more information, see the Streams Messaging Manager REST API Reference.

  Note: Both the Streams Messaging Manager UI and Streams Messaging Manager REST API support the same management actions.
- The Kafka Connect REST API

  The Kafka connect REST API can be used to manage connectors. However, its usage is not recommended by Cloudera, nor is separate documentation provided. For more information, see the upstream Apache Kafka documentation.

Related Information

Managing and monitoring Kafka Connect using Streams Messaging Manager

Streams Messaging Manager REST API Reference
Apache Kafka Documentation

Configuring the Kafka Connect Role

Learn more about how you can configure the Kafka Connect role as well as its notable properties.

The Kafka Connect role is deployed with and managed by Cloudera Manager. You can view a list of configuration properties as well as configure them by going to Kafka service Configuration and selecting the Kafka Connect filter in the Filters pane. Most of the properties available are the standard worker properties defined in upstream Apache Kafka. However, there are a number of notable properties that you should be familiar with when using Kafka Connect in Cloudera. The notable properties are the following:

Broker List for Kafka Connect

These are the brokers Kafka Connect should connect to. You specify the brokers by adding IP:port or hostname:port pairs. The brokers you specify here should be the brokers that are running in the same cluster that Kafka Connect is deployed on. While you can specify a single broker, Cloudera recommends that you specify multiple for high availability.

group.id

The group ID is a unique string that Kafka Connect roles use to form a cluster of connect workers. This property should always be set to the same string for all Kafka Connect roles that are deployed in the same cluster. Kafka Connect will not function properly If there is a mismatch between the group IDs. Therefore, Cloudera recommends that you use the default values for all roles that you deploy.

plugin.path

The directory where the Kafka Connect connector plugins are stored. This is the directory where JAR files for custom connectors can be placed. Cloudera recommends that you use the default path which is /var/lib/kafka.

kafka.connect.rest.port and kafka.connect.secure.rest.port

These are the ports that Kafka Connect API will accept requests on. If the Kafka Connect role is TLS/SSL enabled, it will use the port specified in kafka.connect.secure.rest.port, if TLS/SSL is not enabled, it will use the port specified in kafka.connect.rest.port. Take note of the ports as they are needed when you set up Streams Messaging Manager to manage and monitor Kafka Connect.

A comprehensive list of all properties available for the Kafka Connect role is available in Kafka Properties in Cloudera Runtime.

Related Information

Kafka Properties in Cloudera Runtime

Managing, Deploying and Monitoring Connectors

Learn more about managing connectors.

You can manage, monitor, deploy and interact with Kafka Connect and Kafka Connect connectors either though the Streams Messaging Manager UI or Streams Messaging Manager REST API. For more information see Monitoring Kafka Connect, as well as the Streams Messaging Manager REST API Reference.

Related Information

Managing and monitoring Kafka Connect using Streams Messaging Manager
Streams Messaging Manager REST API Reference

Writing Kafka data to Ozone with Kafka Connect

You can use the HDFS Sink Connector developed by Cloudera and shipped with Cloudera Runtime to write Kafka data to the Ozone filesystem. This can be done on both secure and unsecure clusters by deploying and configuring a new connector with the Streams Messaging Manager UI.

Writing data to Ozone in an unsecured cluster with Kafka Connect

You can use the HDFS Sink Connector developed by Cloudera, i to write Kafka topic data to Ozone in an unsecure cluster. Connector deployment and configuration is done using the Streams Messaging Manager UI.

About this task

The following steps walk you through how the Cloudera-developed HDFS Sink Connector can be set up and deployed using the Streams Messaging Manager UI.

In addition to the connector setup, these steps also describe how you can create a test topic and populate it with data using Kafka command line tools. If you already have a topic that is ready for use and do not want to create a test topic, you can skip steps 1 through 3. These steps deal with topic creation, message consumption, and message production. These steps are optional.

Before you begin

An unsecure Cloudera Base on premises cluster with Kafka, Streams Messaging Manager, and Ozone is set up and configured.

Procedure

1. Create a Kafka topic:
   a) SSH into one of the hosts in your cluster.

   ```
   ssh [***USER***]@[***MY-CLUSTER-HOST.COM***]
   ```

   b) Create a topic with the kafka-topics tool.

   ```
   kafka-topics --create --bootstrap-server [***MY-CLUSTER-
   HOST.COM:9092***] --replication-factor 1 --partitions 1 --top
   ic [***TOPIC***]
   ```

   If an out of memory exception is displayed while running this command, increase the JVM heap with the following command and try again:

   ```
   export KAFKA_OPTS="-Xmx1g -Xms1g"
   ```

   c) Verify that the topic is created.

   ```
   kafka-topics --list --bootstrap-server [***MY-CLUSTER-HOST.COM:9092***]
   ```

2. Produce messages to your topic with the kafka-console-producer.

   ```
   kafka-console-producer --broker-list [***MY-CLUSTER-HOST.COM:9092***] --t
   opic [***TOPIC***]
   ```

   Enter test messages once the tool is running.

   ```
   >MY FIRST MESSAGE
   >MY SECOND MESSAGE
   ```

3. Consume messages:

a) Open a new terminal session and log in to one of the hosts in your cluster.
b) Consume messages with the kafka-console-consumer.

```
kafka-console-consumer --from-beginning --bootstrap-server [***MY-CLUSTER-HOST.COM:9092***] --topic [***TOPIC***]
```

The messages you produced with kafka-console-producer appear. In addition, you can switch to the terminal session that is running kafka-console-producer and produce additional messages. These new messages appear in real time in the session running kafka-console-consumer.

4. Deploy and configure an HDFS Sink Connector:
a) In Cloudera Manager, select the Streams Messaging Manager service.
b) Click Streams Messaging Manager Web UI.
c) Click the Connect option in the left-side menu.
d) Click + New Connector to add a new connector.
e) Go to the Sink Connectors tab and select the HDFS Sink Connector.

On the UI the HDFS Sink Connector is represented by its class name, which is com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnector.

f) Enter a name for the connector.
g) Configure the connector.

Use the following example as a template:

```
{
 "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnector",
 "hdfs.uri": "ofs://ozone1/vol1/bucket1/",
 "hdfs.output": "/topics_output/",
 "tasks.max": "1",
 "topics": "testTopic",
 "hadoop.conf.path": "file:///etc/hadoop/conf",
 "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.txt.TxtPartitionWriter",
 "value.converter": "org.apache.kafka.connect.storage.StringConverter",
 "output.storage": "com.cloudera.dim.kafka.connect.hdfs.HdfsPartitionStorage",
 "hdfs.kerberos.authentication": "false"
}
```

Ensure that you replace the values of hdfs.uri and hdfs.output with valid Ozone paths. The template gives an example of how these paths should look like. Replace any other values depending on your cluster configuration and requirements.

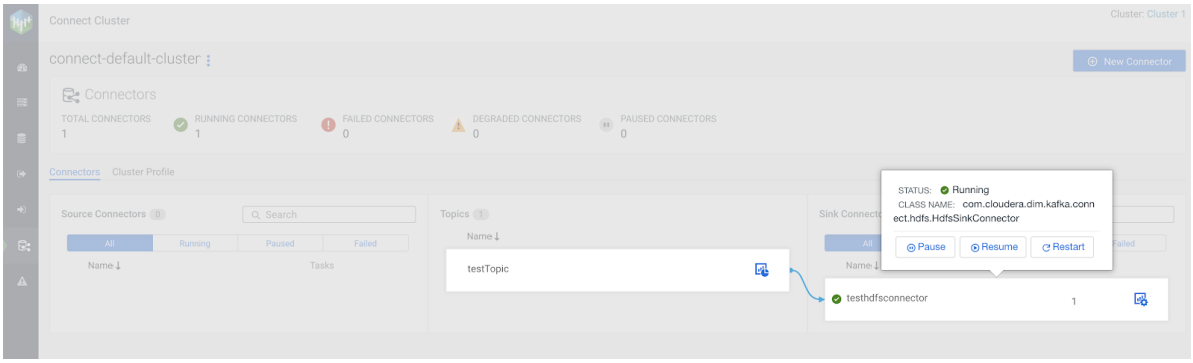
h) Click Validate.

The validator displays any JSON errors in your configuration. Fix any errors that are displayed. If your JSON is valid, the JSON is valid message is displayed in the validator.

i) Click Next.
j) Review your connector configuration.
k) Click Deploy .

5. Verify that connector deployment is successful:
a) In the Streams Messaging Manager UI, click the Connect option in the left-side menu.
b) Click on either the topic or the connector you created.

If connector deployment is successful, a flow is displayed between the topic you specified and the connector you created.



6. Verify that topic data is written to Ozone.

   You can do this by listing the files under the ofs:// location you specified in the connector configuration.

Writing data to Ozone in a Kerberos and TLS/SSL enabled cluster with Kafka Connect

You can use the HDFS Sink Connector developed by Cloudera to write Kafka topic data to Ozone in a secure cluster. Connector deployment and configuration is done using the Streams Messaging Manager UI.

About this task

The following steps walk you through how the Cloudera-developed HDFS Sink Connector can be set up and deployed using the Streams Messaging Manager UI.

In addition to the connector setup, these steps also describe how you can create a test topic and populate it with data using Kafka command line tools. If you already have a topic that is ready for use and do not want to create a test topic, you can skip steps 1 through 3. These steps deal with topic creation, message consumption, and message production. They are not necessary to carry out.

Before you begin

- Ensue that a Kerberos and TLS/SSL enabled Cloudera Base on premises cluster with Kafka, Streams Messaging Manager, and Ozone is set up and configured.
- To create a test topic with the Kafka console tools, you must ensure that a .properties client configuration file is available for use.

  You can create one using the following example as a template:

```
sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required u
seKeyTab=true keyTab="[***PATH TO KEYTAB FILE***]" principal="[***KERBEROS
 PRINCIPAL***]";
security.protocol=SASL_SSL
sasl.mechanism=GSSAPI
sasl.kerberos.service.name=kafka
ssl.truststore.location=[***TRUSTSTORE LOCATION***]
```

Procedure

1. Create a Kafka topic:
   a) SSH into one of the hosts in your cluster.

```
ssh [***USER***]@[***MY-CLUSTER-HOST.COM***]
```

   b) Create a topic with the kafka-topics tool.

```
kafka-topics --create --bootstrap-server [***MY-CLUSTER-
HOST.COM:9093***] --replication-factor 1 --partitions 1 --topic [***T
OPIC***] --command-config [***CLIENT CONFIG FILE***]
```

If an out of memory exception is thrown, increase the JVM heap with the following command and try again:

```
export KAFKA_OPTS="-Xmx1g -Xms1g"
```

c) Verify that the topic is created.

```
kafka-topics --list --bootstrap-server [***MY-CLUSTER-HOST.COM:9093***]
 --command-config [***CLIENT CONFIG FILE***]
```

2. Produce messages to your topic with the kafka-console-producer.

```
kafka-console-producer --broker-list [***MY-CLUSTER-HOST.COM:9093***] --t
opic [***TOPIC***] --producer.config [***CLIENT CONFIG FILE***]
```

Enter test messages once the tool is running.

```
>MY FIRST MESSAGE
>MY SECOND MESSAGE
```

3. Consume messages:

a) Open a new terminal session and log in to one of the hosts in your cluster.

b) Consume messages with the kafka-console-consumer.

```
kafka-console-consumer --from-beginning --bootstrap-server [***MY-
CLUSTER-HOST.COM:9093***] --topic [***TOPIC***] --consumer.conf
ig [***CLIENT CONFIG FILE***]
```

The messages you produced with kafka-console-producer appear. You can switch to the terminal session that is running kafka-console-producer and produce additional messages. These new messages appear in real time in the session running kafka-console-consumer.

4. Deploy and configure an HDFS Sink Connector:

a) In Cloudera Manager, select the Streams Messaging Manager service.

b) Click Streams Messaging Manager Web UI to log in to the UI.

If prompted, enter a valid username as well as a password and click SIGN IN.

c) Click the Connect option in the left-side menu.

d) Click + New Connector to add a new connector.

e) Go to the Sink Connectors tab and select the HDFS Sink Connector.

On the UI the HDFS Sink Connector is represented by its class name, which is com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnector.

f) Enter a name for the connector.

g) Configure the connector.

Use the following example as a template:

```
{
 "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConne
ctor",
 "hdfs.uri": "ofs://ozone1/vol1/bucket1/",
 "tasks.max": "1",
 "topics": "testTopic",
 "hdfs.kerberos.authentication": "true",
 "hdfs.kerberos.user.principal": "${cm-agent:ENV:kafka_connect_service
_principal}",
 "hdfs.kerberos.keytab.path": "${cm-agent:keytab}",
 "hdfs.kerberos.namenode.principal": "hdfs/_HOST@REALM",
 "hdfs.output": "/topics_output/",
 "hadoop.conf.path": "file:///etc/hadoop/conf",
```

```
 "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.tx
t.TxtPartitionWriter",
 "value.converter": "org.apache.kafka.connect.storage.StringConverter",
 "output.storage": "com.cloudera.dim.kafka.connect.hdfs.HdfsPartition
Storage"
}
```

Ensure that you replace the values of hdfs.uri and hdfs.output with valid Ozone paths. The template gives an example of how these paths should look like. Replace other values depending on your cluster and requirements.

h) Click Validate.

The validator displays any JSON errors in your configuration. Fix any errors that are displayed. If your JSON is valid, the JSON is valid message is displayed.

i) Click Next.

j) Review your connector configuration.

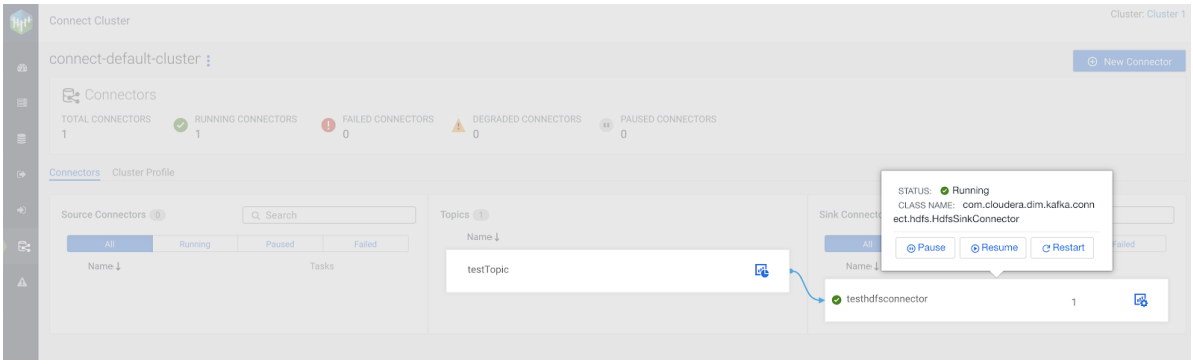
k) Click Deploy to deploy the connector.

5. Verify that connector deployment is successful:

a) In the Streams Messaging Manager UI, click the Connect option in the left-side menu.

b) Click on either the topic or the connector you created.

If connector deployment is successful, a flow is displayed between the topic you specified and the connector you created.



6. Verify that topic data is written to Ozone.

You can do this by listing the files under the ofs:// location you specified in the connector configuration.

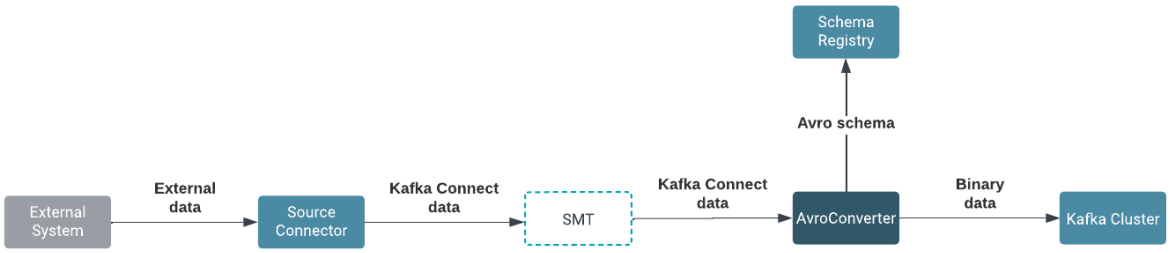
Using the AvroConverter

The AvroConverter is a Kafka Connect converter shipped with Cloudera Runtime that enables Kafka Connect connectors to serialize or deserialize Kafka messages, consisting of key and value pairs, from or to Avro. Using the Avro format can make your messages more compact compared to using for example JSON format. AvroConverter is able to convert from Avro format to Kafka Connect internal data format, and from Kafka Connect internal data to Avro format, and is able to handle Avro schemas integrated with Schema Registry automatically.

Using AvroConverter with a source connector

When using the AvroConverter with a Kafka Connect source connector, the source connector fetches the data from an external system. The fetched data is converted into Kafka Connect internal data format. If Single Message Transforms (SMT) are configured, the messages are converted based on the SMT conditions as well. If the AvroConverter is enabled, the messages are converted from the internal data format to Avro. At this step, the AvroConverter automatically also creates a new schema version in Schema Registry for the Avro data, if the schema has not been already created. In case the schema already exists in Schema Registry, schema creation is skipped and the existing ID will be assigned to the payload during the serialization. The converted data in Avro is written to the Kafka cluster with its corresponding schema ID from Schema Registry.

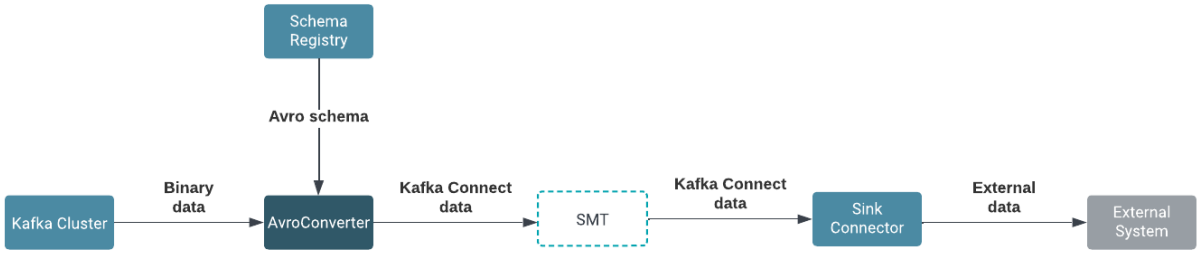
The following illustration shows the steps how AvroConverter works with a Kafka Connect source connector:



Using AvroConverter with a sink connector

When using the AvroConverter with a Kafka Connect sink connector, the data is fetched from the Kafka cluster, which includes its schema ID from Schema Registry. The AvroConverver extracts the schema ID and gets the corresponding schema from Schema Registry. The data is converted from Avro format to the Kafka Connect internal data format. If SMT are configured, the messages are converted based on the SMT conditions as well. The converted data is written by the Kafka Connect sink connector to the external system.

The following illustration shows the steps how AvroConverter works with a Kafka Connect sink connector:



Note: If the SMTs contain steps to change the schema, the schema will not be updated in Schema Registry. The schema is handled automatically by the AvroConverter before any SMTs are applied.

Mapping logical data types

AvroConverter is capable of mapping the data from the Kafka Connect internal data format to Avro. The mapping happens automatically during the conversion, which means that no property needs be configured. The AvroConverter also handles Kafka Connect specific logical types and maps them to Avro ones and vice versa.

The corresponding logical types for Kafka Connect and Avro with their direction is shown in the following table:

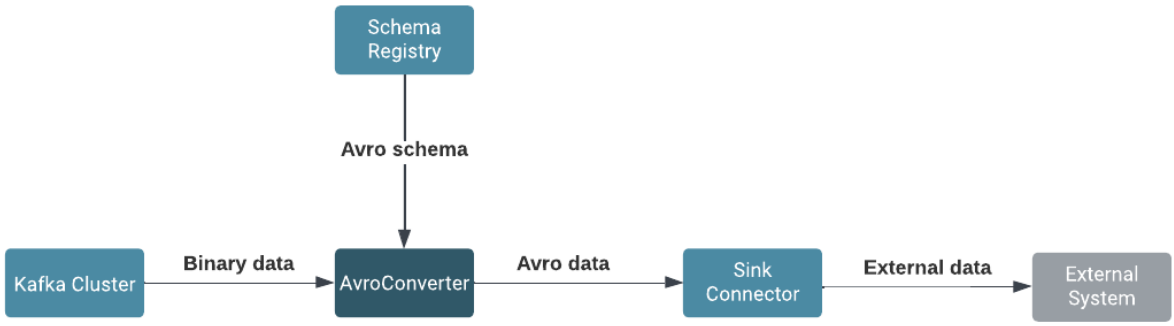
Table 1: Logical Type Mapping

Kafka Connect Logical Type	Direction	Avro Logical Type
Decimal	##	Decimal
Date	##	Date
Time	##	TimeMillis
Time	#	TimeMicros
Timestamp	##	TimestampMillis
Timestamp	#	TimestampMicros

Handling passthrough

When passthrough is enabled, messages are not converted between the Avro format and the Kafka Connect internal data format. In this case, the data is passed through unchanged to the Kafka Connect sink connector. If the passthrough is disabled, messages are converted from Avro to the Kafka Connect internal data format, which can make your input and output formats to be different.

The following illustration shows what happens when passthrough is enabled in the AvroConverter for a sink connector:



Enabling passthrough is recommended when both the input and output formats of your messages are Avro, and it can save any performance overhead generated by the conversion process. You need to ensure that the passthrough is only enabled if the connector can handle Avro objects. Cloudera provides the following connectors that are capable of handling passthrough:

- HDFS Sink connector
- S3 Sink connector

Any other third party connectors can make use of the passthrough if the connectors are implemented to handle the passthrough of the message.

Warning: Passthrough is enabled by default in AvroConverter. You need to make sure that the connector is able to handle passthrough before using AvroConverter or you must disable the passthrough. In case the connector cannot handle the passthrough, the connector will fail.

You can enable and disable passthrough by setting the passthrough.enabled property to true or false when configuring the connector. For more information, see the *KafkaAvroSerializer properties reference* documentation under the *Related information*

Enabling and configuring the AvroConverter

When the AvroConverter is enabled as a key or value converter, you also need to provide further configuration properties that affect the behavior of the AvroConverter. All configuration properties must be prefixed with either the value.converter. or key.converter. prefixes. These prefixes control whether a specific property is applied to the key or value converter.

Enabling AvroConverter

You can enable AvroConverter for any connector using the key.converter and value.converter properties to the connector configuration. The value of these properties must be set to the class name of the AvroConverter.

For example, if you want to use the AvroConverter for both key and value conversion, you must add the following properties to the connector configuration:

```
"key.converter": "com.cloudera.dim.kafka.connect.converts.AvroConverter",
"value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConverter",
```

Conversion is enabled separately for key or value conversion, which enables you to use different converters for the key and value. For example, if you want to use the AvroConverter to convert values, but use a different converter, such as the StringConverter, to convert keys, you must add the following properties to the connector configuration:

```
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConverter",
```

Configuring passthrough

The passthrough configuration in AvroConverter controls whether or not the data is converted into the Kafka Connect internal data format before it is written into an output file or if data is passed through the connector without conversion. Passthrough is enabled by the passthrough.enabled property, which is set to true by default.

The passthrough can be set with different prefixes for key and value converters as shown in the following example configuration:

```
"key.converter.passthrough.enabled":  "false",
"value.converter.passthrough.enabled":  "false",
```



Warning:

Passthrough is enabled by default in AvroConverter. You need to make sure that the connector is able to handle passthrough before using AvroConverter or you must disable the passthrough. In case the connector cannot handle the passthrough, the connector will fail. For more information, see the *Handling passthrough* section.

Enabling SerDes

To transform messages, the AvroConverter uses an embedded KafkaAvroSerializer and KafkaAvroDeserializer, which need to be configured with the AvroConverter to enable further control over the behavior.

You must provide the URL of the Schema Registry server to be able to use the KafkaAvroSerializer and KafkaAvroDeserializer, because both the serializer and the deserializer contain an embedded Schema Registry client. You can specify the Schema Registry server URL with the schema.registry.url property as shown in the following example:

```
"key.converter.schema.registry.url":  "http://registryserver.example.com:9090/api/v1"
"value.converter.schema.registry.url":  "http://registryserver.example.com:9090/api/v1"
```

You can configure additional properties related to serialization, deserialization and the Schema Registry client using the property reference documentations under the *Related information*.

Configuration example

The following example configuration shows a connector configuration with AvroConverter set as key and value converter with connection to a remote Schema Registry server.

```
{
"value.converter":  "com.cloudera.dim.kafka.connect.converts.AvroConverter", // use AvroConverter for the conversion of the Kafka message value
"value.converter.passthrough.enabled":  "false", // disable passthrough
"value.converter.schema.registry.url": "http://registryserver.example.com:9090/api/v1", // URL of the schema registry server that the Schema Registry client inside the value converter will connect to
"value.converter.schema.registry.auth.username = [***USERNAME***],  // username to be used for authentication by the value converter's embedded Schema Registry client
```

```
"value.converter.schema.registry.auth.password = [***PASSWORD***], //  pa
ssword to be used for authentication by the value converter's embedded Schem
a Registry client
"key.converter":  "com.cloudera.dim.kafka.connect.converts.AvroConverter",
"key.converter.schema.registry.auth.username = [***USERNAME***],
"key.converter.schema.registry.auth.password = [***PASSWORD***],
"key.converter.passthrough.enabled":  "false",
"key.converter.schema.registry.url":  "HTTP://
REGISTRYSERVER.EXAMPLE.COM:9090/API/V1"
}
```

Configuring AvroConverter for Debezium connectors with Schema Registry integration

Learn about how to configure the AvroConverter for Debezium connector with Schema Registry integration.

When you want to use the AvroConverter for the Debezium connectors, you need to configure the passthrough, schema compatibility, schema name value and key prefix properties.

passthrough.enabled

You must disable the passthrough to have automatically created schemas for CDC records in Kafka as shown in the following example:

```
"value.converter.passthrough.enabled": "false"
"key.converter.passthrough.enabled": "false"
```

This will ensure that the CDC event fields are loaded into the internal schema of Kafka Connect, from which the field will be converted into Avro format by the AvroConverter before writing it to Kafka. When doing this conversion, the schemas are automatically created in Schema Registry as well.

schema.compatibility

Whenever a schema is created in Schema Registry, compatibility is also set to it by default. Based on the compatibility, if the schema evolves, the new version will be validated against old ones. For more information about the compatibility in Schema Registry, see the *Compatibility policies* documentation under the *Related information*.

The compatibility must be set before the first CDC event is processed by the connector, because the first schema will be created based on the compatibility policies. If backward incompatible changes could happen in the database, consider setting the parameters to NONE, because in this case the Avro schemas will also be incompatible.

The default compatibility can be set based on the following example:

```
“value.converter.schema.compatibility”: “NONE”
“key.converter.schema.compatibility”: “NONE”
```

Note:

The connector task will stop and fail if compatibility issues happen during schema evolution. In this case manual intervention is needed to continue the task.

schema.name.key.suffix and schema.name.value.suffix

When a new schema is created, the name of the schema consists of the name of the topic and an appended value, if set. However, it is possible to set additional suffixes for schema names. The default suffix is empty for the values, but :k is added as a prefix for keys. Key and value suffixes can be set on converters according the following example:

```
"value.converter.schema.name.value.suffix", "value_suffix"
"key.converter.schema.name.key.suffix", "key_suffix"
```

Additional configuration properties for Debezium connectors and Schema Registry integration with AvroConverter

The bigint.unsigned.handling.mode in a Debezium connector handles how bigint data types are mapped, which have the following values that can affect the behavior of the logical data type conversion:

precise

The bigint unsigned values will be stored as byte arrays in Avro with decimal logical type. You can get a byte array or a BigDecimal automatically when you would like to read this data. The object type you get depends on if logical type conversion is enabled or not.

long

Whatever is stored in the database, it will be stored in Avro as long, and you get a Java long object in Java Virtual Machine (JVM). In this case, precision of the mapping is smaller, but no need for logical type handling.

Check schema.name.adjustment.mode and field.name.adjustment.mode to ensure that there are no conflicts between your database related names and the Avro specific naming convention.

Check decimal.handling.mode to decide how precise numbers you need in different cases, and if you need to use logical type for higher precision

Check time.precision.mode to decide how precise dates and times you need in different cases, and if you need to use logical type for higher precision.

Reading Debezium Avro records from Kafka

Learn how to read Debezium CDC Avro records from Kafka using the AvroConverter functions.

Avro records can be consumed from Kafka with the Schema Registry integration as described in the *Integrating Kafka with Schema Registry* documentation.

If any of your connectors produce Avro records with logical types, you do not need to write the conversion from raw Avro data to proper Java objects as you can enable automatic conversion. To enable automatic logical type conversion, you need to set the logical.type.conversion.enabled property in the consumer configuration as shown with the following in the following example:

```
“logical.type.conversion.enabled”: “true”
```

This conversion is handled by the internal SerDes classes in KafkaAvroSerializer and KafkaAvroDeserializer.

Logical type conversion automatically happens in Debezium connectors when they produce the Avro events to Kafka. Ensure that the logical.type.conversion.enabled property is not set as true for the key/value SerDes classes of the Avro Converter. The logical.type.conversion.enabled property only needs to be enabled on the consumer/producer side when the consumer/producer is not part of a Kafka Connect connector.

Related Information

Single Message Transforms

KafkaAvroDeserializer properties reference

Schema Registry Client properties reference

Compatibility policies

Integrating Kafka with Schema Registry

Configuring EOS for source connectors

Learn how to enable and configure exactly-once semantics (EOS) for Kafka Connect source connectors.

About this task

EOS is a framework that enables Kafka and Kafka applications to guarantee that each message is delivered precisely once without it being duplicated or lost. EOS is supported by some Kafka Connect source connectors. To enable EOS, you must enable EOS for the Kafka Connect service roles (Kafka Connect workers) and configure your connectors to use EOS.

EOS is enabled for the Kafka Connect service roles in Cloudera Manager with the Exactly-once support for source connectors property. Connectors can be configured using the Streams Messaging Manager UI (recommended), Streams Messaging Manager API, or Kafka Connect API.

Before you begin

- Enabling EOS for Kafka Connect service roles only enables the use of EOS on the level of the service. For individual connectors to make use of EOS, the connector must be specifically configured to use EOS.
- Not all connectors shipped in Cloudera support EOS. The following list collects the source connectors shipped in Cloudera that support EOS.
  - JDBC Source
  - SFTP Source
  - NiFi Stateless Source
  - FileStreamSource

  Important: The NiFi Stateless Source connector is used to deploy your own, custom developed dataflows as Kafka Connect connectors. In order for your connector to support EOS, the dataflow that you deploy must include a source processor that supports state management on the scope of the cluster. For more information see the official NiFi documentation.
- If you are using a third-party connector, review the documentation of the connector to learn whether EOS is supported.

Procedure

1. Configure Kafka Connect service roles:
   a) In Cloudera Manager, select the Kafka service.
   b) Go to Configuration.
   c) Set Exactly-once support for source connectors in the cluster to preparing.
   d) Restart all Kafka Connect service roles.
   e) Set the Exactly-once support for source connectors in the cluster to enabled.
   f) Restart all Kafka Connect service roles.
2. Configure source connectors.

   The following table collects the common source connector properties that you can use to enable and configure EOS. Configure the following properties based on your requirements. Cloudera recommends that you use the Streams Messaging Manager UI to configure and deploy connectors. For more information on connector configuration and deployment see *Deploying and managing Kafka Connect connectors in Streams Messaging Manager*

Table 2: EOS specific Kafka Connect source connector properties

Name	Default value	Description
exactly.once.support	requested	Permitted values are requested and required. If set to required, forces a preflight check for the connector to ensure that it can provide exactly-once delivery with the given configuration. Some connectors may be capable of providing exactly-once delivery but not signal to Kafka Connect that they support this. In this case, review the documentation for the connector before connector deployment and set this property to requested. Additionally, if the value is set to required but the worker that performs preflight validation does not have exactly-once support enabled for source connectors, requests to create or validate the connector will fail.
transaction.boundary	poll	Permitted values are poll, connector, and interval. If set to poll, a new producer transaction is started and committed for every batch of records that each task from this connector provides to Kafka Connect. If set to connector, relies on connector-defined transaction boundaries; note that not all connectors are capable of defining their own transaction boundaries, and in that case, attempts to create them with this property set to connector will fail. If set to interval, commits transactions only after a user-defined time interval has passed.
offsets.storage.topic	null	The name of a separate offsets topic to use for this connector. If left empty or not specified, the worker's global offsets topic name is used. If specified, the offsets topic is created if it does not already exist on the Kafka cluster targeted by this connector (which may be different from the one used for the worker's global offsets topic if the bootstrap.servers property of the connector's producer has been overridden from the worker's).
transaction.boundary.interval.ms	null	If transaction.boundary is set to interval, determines the interval for producer transaction commits by connector tasks. If unset, defaults to the value of the worker-level offset.flush.interval.ms property.

Related Information

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Securing Kafka Connect

Learn about the different options and features available for securing your Kafka Connect deployment.

Kafka Connect to Kafka broker security

Learn about the options you have for securing the connection between Kafka Connect and Kafka brokers.

Configuring TLS/SSL encryption for the Kafka Connect role

Kafka Connect roles inherit the TLS/SSL configuration of the parent Kafka service. If you are deploying Kafka Connect roles under a Kafka service that already has TLS/SSL enabled, Cloudera Manager will automatically enable TLS/SSL for Connect as well. If required however, you can manually enable or disable TLS/SSL.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the following properties based on your cluster and requirements.

Table 3:

Cloudera Manager Property	Description
Enable TLS/SSL for Kafka Connect ssl_enabled	Encrypt communication between clients and Kafka Connect using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
Kafka Connect TLS/SSL Server JKS Keystore File Location ssl_server_keystore_location	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Kafka Connect is acting as a TLS/SSL server.
Kafka Connect TLS/SSL Server JKS Keystore File Password ssl_server_keystore_password	The password for the Kafka Connect keystore file.
Kafka Connect TLS/SSL Server JKS Keystore Key Password ssl_server_keystore_keypassword	The password that protects the private key contained in the JKS keystore used when Kafka Connect is acting as a TLS/SSL server.
Kafka Connect TLS/SSL Trust Store File ssl_client_truststore_location	The location on disk of the trust store, used to confirm the authenticity of TLS/SSL servers that Kafka Connect might connect to. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
Kafka Connect TLS/SSL Trust Store Password ssl_client_truststore_password	The password for the Kafka Connect TLS/SSL Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
ssl.client.auth	Client authentication mode for SSL connections. This configuration has three valid values, required, requested, and none. If set to required, client authentication is required. If set to requested, client authentication is requested and clients without certificates can still connect. If set to none, which is the default value, no client authentication is required
Supported SSL/TLS versions for Kafka Connect	Lists the TLS protocol versions enabled on the Kafka Connect REST server, metrics, and JMX ports. Defaults to the centralized Cloudera Manager TLS settings. Visible when the TLS_ADVANCED_CONTROL feature flag is enabled. Set this to match your security policy, for example, TLSv1.3 or TLSv1.2,TLSv1.3. Do not use TLS 1.0 or TLS 1.1. For information about enabling the feature flag, see *Transport Layer Security Support and Custom Protocol Configuration*.
TLS Cipher List for Kafka Connect	Lists the cipher suites Kafka Connect accepts. Defaults to the centralized Cloudera Manager cipher list. Visible when the TLS_ADVANCED_CONTROL feature flag is enabled.

4. Click Save Changes.
5. Restart the service.

Results
TLS/SSL encryption is configured for the Kafka Connect role.

Configuring Kerberos authentication for the Kafka Connect role
Learn how Kerberos authentication is configured for the Kafka Connect role.

Kafka Connect roles inherit the Kerberos configuration of the parent Kafka service. If you are deploying Kafka Connect roles under a Kafka service that already has Kerberos enabled, Cloudera Manager will automatically enable Kerberos for Kafka Connect as well. Other than making sure that the Kafka service's Kerberos configuration is correctly set, no additional user action is required.

Kafka Connect REST API security
Learn about the options you have for securing the Kafka Connect REST API.

Kafka Connect REST API authentication
Learn about the authentication options that are available for the Kafka Connect REST API.

Configuring TLS/SSL client authentication for the Kafka Connect REST API
Learn how to configure TLS/SSL client authentication for the Kafka Connect REST API.

About this task

You can secure the Kafka Connect API by configuring the Kafka Connect roles to require SSL Client authentication. This can be done by setting the SSL Client Authentication property to required. When set to required, only clients that pass SSL client authentication will be able to access the Kafka Connect API. As a result, any client that you would like to give access to should have its certificate added to the Kafka Connect truststore. This includes Streams Messaging Manager as well. Cloudera recommends that in secure environments only Streams Messaging Manager is given access to the Kafka Connect API.

In addition to setting client authentication to required, you may also want to consider setting up a firewall using third party tools to further secure access to the Kafka Connect API. Note however, that even with a firewall in place and SSL authentication set to required, if Streams Messaging Manager is given access to the Kafka Connect API, then any user that has access to Streams Messaging Manager will be able to interact with the Kafka Connect API. This is due to Streams Messaging Manager not enforcing authorization checks when users are accessing Kafka Connect functionality within Streams Messaging Manager. This is true for both the Streams Messaging Manager UI and Streams Messaging Manager REST API. As a result, caution is advised even if the Kafka Connect API itself is secured.

Complete the following steps to set SSL Client Authentication to required.

Procedure

1. Select the Kafka Service.
2. Go to Configuration.
3. Find the SSL Client Authentication property.
4. Set the property to required.

   Important:

   The SSL Client Authentication property is available for both Broker and Connect roles. Make sure that you are configuring the property for the Kafka Connect role or for the role group that includes your Kafka Connect roles.
5. Click Save Changes.
6. Restart the service.

Results
Only authenticated clients are allowed to connect to the Kafka Connect API.

What to do next
If you are using Streams Messaging Manager to manage and monitor Kafka Connect, and you are not using auto TLS, add the certificate of Streams Messaging Manager to the Kafka Connect truststore.

Configuring SPNEGO Authentication and trusted proxies for the Kafka Connect REST API
Learn how you enable SPNEGO authentication and configure trusted proxies for the Kafka Connect REST API.

About this task



Note: SPNEGO authentication is automatically enabled if Kerberos is enabled for the Kafka service. As a result, SPNEGO authentication might already be enabled. However, If you upgraded your cluster from a version that did not have SPNEGO authentication available to a version that includes SPNEGO authentication, SPNEGO is not automatically enabled.

You can secure the Kafka Connect REST API by enabling SPNEGO authentication. This can be done with the Enable SPNEGO Authentication For Kafka Connect property in Cloudera Manager. If SPNEGO authentication is enabled, only users authenticated with Kerberos are able to access and use the REST API. Additionally, if Ranger authorization is enabled for the Kafka service, authenticated users are only able perform the operations that they are authorized for. If Ranger is not enabled, by default all authenticated users are able to perform all operations. Because users authenticate using Kerberos, securing the REST API using SPNEGO requires that Kerberos is enabled for the Kafka service.

In addition, the authentication framework supports trusted proxies. If the authenticated principal is recognized as a trusted proxy principal, then the authenticator accepts the doAs query parameter in the REST URL to specify the acting user. For example:

```
https://[***KAFKA CONNECT HOST***]:28085/connector-permissions?doAs=systest
```

In this example, systest is specified as the acting user. The doAs parameter is only accepted if the authenticated principal is recognized as a trusted proxy. By default, the Knox and Streams Messaging Manager service principals (specifically, the short names of their Kerberos service principals) are recognized as trusted proxies. Trusted proxies can be configured with the List Of Trusted Proxy Services Cloudera Manager property.

Before you begin

Ensure that Kerberos is enabled for the Kafka service. For more information, see Enable Kerberos authentication.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Go to Configuration.
3. Find and enable the Enable SPNEGO Authentication For Kafka Connect property.
4. Find and configure the List Of Trusted Proxy Services property.

   Add the Kerberos principal short names of the services that you want to act as trusted proxies.
5. Click Save Changes.
6. Restart the Kafka service.

Configuring allowed LDAP and OAuth URLs
You can restrict which LDAP and OAuth URLs are allowed for internal clients used by Kafka Connect connectors. Configuring a trusted list of URLs can harden the security of your deployment and can prevent an attacker from setting malicious values in connector clients. Configuration is done in Cloudera Manager by setting Java options using an advanced configuration snippet.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find the Kafka Connect Environment Advanced Configuration Snippet (Safety Valve) property.
4. Add or update the EXTRA_ARGS environment variable with the appropriate Java options.
   - To set allowed LDAP URLs, use the com.cloudera.kafka.ldap.allowed.urls Java option. For example:

     ```
     EXTRA_ARGS=-Dcom.cloudera.kafka.ldap.allowed.urls="HTTP://WWW.LDAP-EXAMPLE-1.COM,HTTP://WWW.LDAP-EXAMPLE-2.COM"
     ```

   - To set allowed OAuth URLs, use the org.apache.kafka.sasl.oauthbearer.allowed.urls Java option. For example:

     ```
     EXTRA_ARGS=-Dorg.apache.kafka.sasl.oauthbearer.allowed.urls="HTTP://WWW.OAUTH-EXAMPLE-1.COM,HTTP://WWW.OAUTH-EXAMPLE-2.COM"
     ```

5. Click the Save Changes button.
6. Restart the Kafka service.

Kafka Connect Rest API authorization

Learn about the authorization model shipped with Kafka Connect as well as how the model is integrated in Ranger.

Kafka Connect authorization model

Learn about the authorization model shipped with Kafka Connect.

Kafka Connect comes with an authorization model. Implementations are pluggable and it is up to the implementation how the capabilities of the model are utilized. The authorization model is implemented by default in Ranger. As a result, Cloudera recommends that you use Ranger.

Resource types

Cluster

Cluster is a singleton resource representing the cluster. This resource has the following operations:

- View: view connector plugins and the root path (basic cluster information).
- Validate: validate connector plugin configuration.
- Manage: manage cluster level resources such as loggers.

Connector

Connector is the resource representing a connector and its tasks. This resource has the following operations:

- View: view connector and task configuration as well as status.
- Manage: pause, resume, and restart connectors.
- Create: create connectors.
- Edit: edit connector configuration.
- Delete: delete connectors.

Error codes

When enabled, the Kafka Connect authentication and authorization framework can return the following HTTP errors in case of an authentication or authorization failure:

401 Unauthorized

The user attempts to connect the Kafka Connect REST server without authentication.

403 Forbidden

The user attempts an operation on a resource that is not permitted, but the user is entitled to view the resource. For example, the user tries to restart a connector that they only have view access to.

404 Not Found

The user attempts an operation on a resource that is not permitted and the resource is not visible by the user.

API endpoint to report user rights on a connector

The Kafka Connect version shipped with Cloudera Runtime includes the /connector-permissions endpoint. This endpoint is unique to Kafka Connect in Cloudera Runtime. It returns all connectors the user has permission to view along with access rights for each connector. The purpose of the endpoint is to allow the Streams Messaging Manager UI to enable and disable UI controls based on what the user can do with each connector. This endpoint is not meant to check access rights in the client, these are always checked on the backend. The endpoint responds in the following format:

```
{"createAllowed":true,
  "permissions": {
      "connectorName-1":{"editable":true, "deletable":true, "manageable":t
rue},
      "connectorName-2":{"editable":true, "deletable":true, "manageable":t
rue}
  }
}
```

Semantics for listing type endpoints

Listing endpoints (/connectors and /connector-permissions) only return the connectors that the user is permitted to view. In case the authenticated user has no rights to view any connectors it returns an empty list. Similarly to other endpoints, it responds with 401 Unauthorized if there is no authenticated user.

Kafka Connect Ranger integration

Learn about how the Kafka Connect authorization model is integrated with Ranger.

If Ranger authorization is enabled for the Kafka service and you have Kerberos and SPNEGO authentication enabled for Kafka Connect, you have the ability to set up fine grained access to the Kafka Connect REST API using the Ranger console.

This means that you can create policies and limit what operations an authenticated user is permitted to carry out. Policies that are set up are valid for any user that accesses the Kafka Connect REST API. This includes direct REST API access as well as access through Streams Messaging Manager.

If a user tries to carry out an operation on a resource that they do not have permission for, the REST API returns an error. For more information on the specific error messages, see *Kafka Connect authorization model*. In Streams Messaging Manager, the policies control what UI operations are available for the user. For example, if a user does not have access rights to view select connectors, those connectors are not shown when logged in to the Streams Messaging Manager UI. Similarly, if a user for example does not have the rights to create connectors, the New Connector option is not be available for the user.

Related Information

Kafka Connect authorization model

Configuring SPNEGO Authentication and trusted proxies for the Kafka Connect REST API

Kafka Connect connector configuration security

Learn about the options you have for hardening the security of connector configurations.

Kafka Connect Secrets Storage

Learn about the Kafka Connect Secrets Storage feature which can be used to hide and securely store sensitive data found in connector configurations.

In most production environments, the Kafka Connect connectors that you deploy connect to and move data either from or into secured services. Because of this, connectors that you deploy might require various credentials to access services. Credentials like passwords, access keys, or any other sensitive information must be provided to the connectors in their configuration. By default, the connector configurations can be easily read, their contents are unencrypted. Additionally, if you are using Streams Messaging Manager to manage and deploy connectors, all sensitive information might be visible to anyone who has access to Streams Messaging Manager with appropriate authorization rights.

Kafka Connect Secrets Storage is a feature within Kafka Connect that makes it possible for you to hide any sensitive values found in connector configurations and to store these values in a secure manner. For example, assume that you want to deploy a connector that connects to a secured service that requires a username and password for access. By default the password could be easily read by anyone with access to the connector configuration as the configuration is stored in plaintext. This poses a security risk. To circumvent this, you can mark the password as a secret. Once you mark the password as a secret, the Kafka Connect Secret Storage feature ensures that the password is stored and handled in a secure manner by doing the following:

- The property is replaced by a reference in the configuration. The reference resolves to the actual password at runtime. That is, the value is no longer readable if the configuration is accessed.
- The actual value of the password is encrypted and stored in an internal Kafka topic used by the feature.
- The value gets hidden (locked) on the Streams Messaging Manager UI.

You configure Kafka Connect Secrets Storage properties in Cloudera Manager at the Kafka service level. Available properties include the global password, PBE salt, secrets topic name, and the algorithm used to wrap the encryption key. On FIPS-enabled clusters where the Bouncy Castle FIPS jars are configured, Cloudera recommends setting the Kafka Connect Secrets Storage Secret Key Wrap Algorithm (kafka.connect.secret.key.wrap.algorithm) property to AESKWP. Other FIPS-approved algorithms supported by the configured security provider may also be used; AESK WP is the algorithm validated by Cloudera.

Kafka Connect Secrets Storage terms and concepts

Review the list of terms and concepts related to the Kafka Connect Secrets Storage feature.

Secure Connector

A secure connector is any connector intended to be used in Cloudera Kafka Connect whose configuration has a sensitive property.

Secret

Part of a secure connector configuration that must be protected. A secret can be any value used in the connector configuration that you mark as confidential data. For example, a secret can be a password, a JAAS credential, an SSL certificate, and so on. When you mark a property as a secret it becomes protected. Specifically, the value of the property is no longer stored in plaintext within the configuration. Rather, it is replaced by a secret reference. The actual value is encrypted and stored in an internal Kafka topic (secrets topic).

Understand however, that this protection covers the configuration only. If you mark a property a secret that is otherwise public information, while the value of that property will be hidden in the configuration, the value could be looked up by other means. For example, sink connectors have a topics property. If auto topic creation is enabled, topics will be created by the connector automatically. This means that even if the topics property is marked as a secret, the value of the property could be retrieved by other means. For example, by listing the topics of the Kafka service.

Secrets topic

The secrets topic is an internal Kafka topic. The Kafka Connect Secrets Storage feature stores its internal state, configuration, secrets, and so on in this internal topic.

Secret reference

Part of a secure connector configuration that is already protected. Instead of the configuration's original value, a placeholder (reference) is used, which refers to the original value using a connector name and a bundle ID. The Kafka Connect Secrets Storage feature resolves these references and substitutes them with the original values stored in the secrets topic when connectors are used.

Encryption key

The encryption key is a randomly generated key. Its established value is unique at a certain point in time. However, it can be changed. It is used to encrypt and decrypt hashed secrets protecting them against tampering and retrieval.

Global key

Global key is a unique cryptographic key that is derived from a user supplied password. The global key protects the encryption key when the encryption key is persisted to the disk or exchanged on the network. The encryption key is hashed and encrypted with the global key. As a result, it is impossible to tamper with it, or retrieve the encryption key from wiretapping the network traffic in a man-in-the-middle style attack.

Bundle and Bundle ID

The set of secrets of a secure connector constitute a bundle. The bundle is identified by its bundle ID, which is maintained by the Kafka Connect Secrets Storage feature. If a secret is added, edited, or deleted, a new bundle is created, with a new bundle ID.

Managing secrets using the REST API

Learn how you can use the Kafka Connect Secrets Storage feature to manage secrets in connector configurations using the Kafka Connect REST API.

You can use the Kafka Connect Secrets Storage feature with the Kafka Connect REST API. The following sections walk you through how you can mark properties as secrets and how you can manage already existing secrets.



Important: The Kafka Connect Secrets Storage feature can also be used when managing connectors using the Streams Messaging Manager UI. For more information, see Deploying and managing Kafka Connect connectors in Streams Messaging Manager.

Mark a configuration value as a secret in a connector configuration

To mark a property value in a connector configuration as a secret, you must:

- Add the property to the configuration.
- Add the secret.properties property. The value of this property is a comma separated list of the property keys that you want to mark as secrets.

For example, assume that you have two sensitive properties in your configuration that you want to mark as secrets, password.for.connector and some.confidential.config.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config”,
“topics” = “...”,
“password.for.connector” = “mypassword”,
“some.confidential.config” = “sensitive information”
}
```

If the secret.properties is present in the configuration, the Kafka Connect Secrets Storage feature gets enabled. Once the connector configuration is submitted, the feature processes the configuration and makes a number of changes.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config”,
“secret.bundle.id” = “18a1b3a6-1e2f”,
```

```
“topics” = “...”,
“password.for.connector” = “${secret:Custom/18a1b3a6-1e2f:password.for.conne
ctor}”,
“some.confidential.config” = “${secret:Custom/18a1b3a6-1e2f:some.confidentia
l.config}”
}
```

Notice the following changes:

- A secret.bundle.id property is added. The bundle is the set of secrets of a secure connector.

  The bundle is identified by its bundle ID, which is maintained by the Kafka Connect Secrets Storage feature. If a secret is added, deleted, or edited, a new bundle is created with a new bundle ID.
- The password.for.connector and some.confidential.config properties had their values replaced with secret references.

  These references are placeholder values consisting of the connector name, bundle ID, and property key. If a reference like this is present in the configuration, it means that the value for that property is securely stored in an internal Kafka topic (secrets topic). The Kafka Connect Secrets Storage feature resolves these references and substitutes them with the original values stored in the secrets topic when the connectors are used.

Update a connector configuration with a new secret

If you want to update an already existing configuration with a new secret property, you must add the new property to the configuration as well as the secret.properties property. For example, assume that you have a configuration where password.for.connector and some.confidential.config are already marked as secrets. Now, you want to add the new. secret.config property to the configuration and mark it as a secret as well.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “18a1b3a6-1e2f”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/18a1b3a6-1e2f:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/18a1b3a6-1e2f:some.confident
ial.config}”,
“new.secret.config” = “[***A NEW SECRET VALUE***]”
}
```

Once submitted and processed, the configuration will look like the following example:

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/c8a753d6-8242:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/c8a753d6-8242:some.confident
ial.config}”,
“new.secret.config” = “${secret:Custom/c8a753d6-8242:new.secret.config}”
}
```

Notice the following changes:

- The secret.bundle.id value has changed.

This is done because adding a new secret creates a new set of secrets resulting in a new bundle being created.

- The values (secret references) of password.for.connector and some.confidential.config are updated with the new bundle ID.
- The value of new.secret.config is replaced by a secret reference.

  This means that the value is now securely stored in the secrets topic.

Update a connector configuration by changing a secret

Updating already existing secret values can be done by updating the appropriate property values. For example, assume that you have three properties marked as secrets, password.for.connector, some.confidential.config, and new.secret.config. You want to update the value of new.secret.config.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/c8a753d6-8242:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/c8a753d6-8242:some.confident
ial.config}”,
“new.secret.config” = “[***CHANGED VALUE***]”
}
```

Once submitted and processed, the configuration will look like the following example:

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “14b2f85f-1dcb”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/14b2f85f-1dcb:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/14b2f85f-1dcb:some.confident
ial.config}”,
“new.secret.config” = “${secret:Custom/14b2f85f-1dcb:new.secret.config}”
}
```

Notice the following changes:

- The secret.bundle.id value has changed.

  This is done because deleting a secret creates a new set of secrets resulting in a new bundle being created. In this case the deleted secret is removed from the bundle.
- The values (secret references) of password.for.connector and some.confidential.config are updated with the new bundle ID.

  This is done because updating a secret creates a new set of secrets resulting in a new bundle being created.
- The value of new.secret.config is replaced by a secret reference.

  This means that the new value is now securely stored in the secrets topic.

Remove an existing secret from an existing configuration

Secrets can be removed from a configuration by updating the configuration and deleting the unnecessary properties from both the configuration and secret.properties. For example, assume you have three properties marked as secrets, password.for.connector, some.confidential.config, and new.secret.config.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,some.confidential.config,ne
w.secret.config”,
“secret.bundle.id” = “14b2f85f-1dcb”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/14b2f85f-1dcb:password.for.con
nector}”,
“some.confidential.config” = “${secret:Custom/14b2f85f-1dcb:some.confident
ial.config}”,
“new.secret.config” = “${secret:Custom/14b2f85f-1dcb:new.secret.config}”
}
```

Assume you want to delete some.confidential.config. To do this, you need to delete the property from the configuration and remove the property key from secret.properties.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,new.secret.config”,
“secret.bundle.id” = “14b2f85f-1dcb”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/14b2f85f-1dcb:password.for.conn
ector}”,
“new.secret.config” = “${secret:Custom/14b2f85f-1dcb:new.secret.config}”
}
```

Once submitted and processed, the configuration will look like the following example:

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,new.secret.config”,
“secret.bundle.id” = “15641fca-483b”,
“topics” = “...”,
“password.for.connector” = “${secret:Custom/15641fca-483b:password.for.conn
ector}”,
“new.secret.config” = “${secret:Custom/15641fca-483b:new.secret.config}”
}
```

Notice the following changes:

- The secret.bundle.id value has changed.

  This is done because deleting a secret creates a new set of secrets resulting in a new bundle being created. In this case the deleted secret is removed from the bundle.
- The values (secret references) of password.for.connector and some.confidential.config are updated with the new bundle ID.

Remove all secrets

To remove all secrets from a configuration, you must remove all properties that were marked as secrets and also remove the secret.properties property. However, you must leave secret.bundle.id intact. For example, assume you had a single property marked as a secret, password.for.connector.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.properties” = “password.for.connector,new.secret.config”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”,
“password.for.connector”= “${secret:Custom/c8a753d6-8242:password.for.conne
ctor}
}
```

In a case like this, you need to remove password.for.connector and secret.properties from the configuration. However, you must preserve secret.bundle.id. The reason why you need to leave this property intact is because the encrypted values for the secret properties must be deleted from the internal secrets topic. The Kafka Connect Secrets Storage feature can only do this if it has a bundle ID for reference.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“secret.bundle.id” = “c8a753d6-8242”,
“topics” = “...”
}
```

Once the configuration is submitted and processed, secret.bundle.id is removed. The connector configuration at this point is considered unsecured.

```
{
“name” = “Custom”,
“connector.class” = “my.custom.secure.connector”,
“topics” = “...”
}
```

Re-encrypting secrets

Using the Kafka Connect Secrets Storage feature places encrypted confidential information in an internal Kafka topic (secrets topic) used by the feature. If you want to change the encryption key used to encrypt the information placed in the topic, all secrets must be re-encrypted and must be migrated to a new topic. This is done using the connect-secret-storage-migration tool that is shipped with Cloudera Runtime.

About this task



Important:

Do not follow this procedure if you only need to change the algorithm used to wrap the encryption key. To change the wrap algorithm, update the Kafka Connect Secrets Storage Secret Key Wrap Algorithm (kafka.connect.secret.key.wrap.algorithm) property in Cloudera Manager and perform a rolling restart of all Kafka Connect roles.

The connect-secret-storage-migration tool migrates existing secrets from the currently used secrets topic (source) to a newly created one (target). It achieves this by reading through the source topic, decrypting the secrets stored within it using the existing global key, and re-encrypts them with a new, randomly generated encryption key and a new global key that you configure. After encryption is finished, newly encrypted secrets are produced into the target topic. The new encryption key will have a different security context (different global password and salt) than the currently used one.

The tool requires three configuration files to function. These configuration files provide the tool with information regarding the Kafka service as well as the source and target topics. The configuration files must be created manually.

Before you begin

- In the configuration files you prepare, you must specify the current global password and PBE salt used by the feature. These values are specified in Cloudera Manager with the Kafka Connect Secrets Storage PBE Salt and Kafka Connect Secrets Storage Global Password properties. The default values for these properties are randomly generated and are hidden, but can be changed at any time. As a result, if you do not know the global password or PBE salt value, you must change both values, note down the values you configured, and restart all Kafka Connect roles before you complete the following steps.
- In FIPS clusters that use JDK 11, ensure that the KAFKA_OPTS environment variable is configured in your shell environment and includes the necessary JVM arguments. The KAFKA_OPTS configuration required for this tool is the same as for all other Kafka CLI tools. See Configuring Kafka command line tools in FIPS clusters for configuration steps.

Procedure

1. Stop all Kafka Connect roles.
2. Prepare the configuration files required for the tool.

   You need to prepare three different configuration files. A Kafka client configuration file that the tool can use to access the Kafka service. Additionally, two configuration files that contain information regarding the source and target topics. All three files must be deployed on the cluster host where you run the tool.

   a) Create a Kafka client configuration file.

   This configuration file must contain all client properties that the tool requires to access the Kafka service. The properties you need to specify in this file depend on the security configuration of the Kafka service. The following tabs contain examples for some of the most commonly used security configurations.

   For SASL_SSL Kerberos

   ```
   bootstrap.servers=[***HOST***]:[***PORT***]
   security.protocol=SASL_SSL
   sasl.mechanism=GSSAPI
   sasl.kerberos.service.name=kafka
   sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule require
   d useKeyTab=true storeKey=true keyTab="[***PATH TO KEYTAB***]" princi
   pal="[***KERBEROS PRINCIPAL***]";
   ssl.truststore.location=[***PATH TO TRUSTSTORE.JKS***]
   ssl.truststore.password=[***TRUSTSTORE PASSWORD***]
   ```

   For SASL_SSL PLAIN

   ```
   bootstrap.servers=[***HOST***]:[***PORT***]
   security.protocol=SASL_SSL
   sasl.mechanism=PLAIN
   sasl.jaas.config=org.apache.kafka.common.security.plain.Pla
   inLoginModule required username="[***USERNAME***]" passwo
   rd="[***PASSWORD***]";
   ssl.truststore.location=[***PATH TO TRUSTSTORE.JKS***]
   ssl.truststore.password=[***TRUSTSTORE PASSWORD***]
   ```

   b) Create the configuration file identifying the source topic and its security context. For example:

   ```
   kafka.connect.secret.storage.topic : [***SOURCE TOPIC NAME***]
   kafka.connect.secret.global.password : [***CURRENT GLOBAL PASSWORD***]
   kafka.connect.secret.pbe.salt : [***CURRENT PBE SALT***]
   ```

Replace *[***SOURCE TOPIC NAME***]* with the name of the currently used secrets topic. The name of the current topic is specified in Cloudera Manager in the Kafka Connect Secrets Storage Topic Name property.

Replace *[***GLOBAL PASSWORD***]* with the currently used global password.

Replace *[***CURRENT PBE SALT***]* with the currently used PBE salt value.

You can choose to not specify kafka.connect.secret.global.password in the configuration. If the property is not added to the configuration, you are prompted to enter the password when you run the tool.

c) Create the configuration file identifying the target topic and its security context. For example:

```
kafka.connect.secret.storage.topic : [***TARGET TOPIC NAME***]
kafka.connect.secret.global.password : [***NEW GLOBAL PASSWORD***]
kafka.connect.secret.pbe.salt : [***NEW PBE SALT***]
kafka.connect.secret.storage.topic.replication.factor : [***REPLICATION
 FACTOR***]
kafka.connect.secret.storage.topic.configs.min.insync.replicas : [***IN-
SYNC REPLICA NUMBER***]
```

You can choose to not specify kafka.connect.secret.global.password in the configuration. If the property is not added to the configuration, you are prompted to enter the password when you run the tool.

3. Run the tool.



Important: If your cluster is running in FIPS mode and uses JDK 11, ensure that the KAFKA_OPTS environment variable is configured in your shell environment and includes the necessary JVM arguments. Otherwise the tool will not run.

You must specify the three configuration files you created as parameters. The order in which you specify the files is fixed. The first file must be the Kafka client configuration, the second must be the file for the source topic, and the third file must be the file for the target topic. For example:

```
connect-secret-storage-migration [***KAFKA CLIENT
 CONFIGURATION***] [***SOURCE TOPIC CONFIGURATION***] [***TARGET TOPIC
 CONFIGURAITON***]
```

4. If prompted, enter the current and new global password.

   The tool only prompts you to enter the passwords if they were not specified in the configuration files. The tool does not echo the passwords that you type in the console.

5. The tool prompts you to ensure that the Kafka Connect cluster (all Connect roles) are stopped. Press Enter to continue.
6. Wait until the tool is finished with re-encryption.

   The tool reports its progress. Upon successful completion, the tool exits. Upon failure, an error message is printed. In such a case, fix any issues that the tool reports and run the tool again.

7. In Cloudera Manager, select the Kafka service.
8. Go to Configuration.
9. Find and configure the following properties:

   - Kafka Connect Secrets Storage Topic Name
   - Kafka Connect Secrets Storage Topic Replication Factor
   - Kafka Connect Secrets Storage Topic Minimum In-Sync Replicas
   - Kafka Connect Secrets Storage Global Password
   - Kafka Connect Secrets Storage PBE Salt

   Add the values you specified in the configuration file identifying the target topic.

10. Restart all Kafka Connect roles.
11. Verify that Kafka Connect is working as expected.

This can be done by verifying that all connectors that have properties marked as secrets are running and are in a healthy state. That is, the connectors are resolving all referenced secrets correctly. If you are experiencing issues, you can revert the configuration changes in Cloudera Manager and restart the re-encryption process.

12. Delete the configuration files you created.
13. Delete the source secrets topic.



Important: Ensure that you verified that Kafka Connect and your connectors are working as expected before deleting the topic. When you delete the topic, the secrets stored within it are lost. As a result, if there are issues with Kafka Connect, you might not be able to revert the cluster to a working state.

Configuring connector JAAS configuration and Kerberos principal overrides

Learn how to configure Kafka Connect connectors to override the default JAAS configuration and Kerberos principal that they use to connect to the Kafka service. Configuring these overrides enables you to set up fine grained access rules for the connectors that you deploy.

About this task

By default Kafka Connect connectors, specifically the clients managed by Connect, use the JAAS configuration and Kerberos principal of the Connect worker (Kafka Connect role) to establish a connection with the Kafka service. As a result, all connectors that you deploy authenticate with the same principal. Additionally, if authorization is enabled, authorization also happens using that same principal. Because the connectors use the same principal as the worker and because the worker has all permission on the Kafka service, all connectors have full access to Kafka by default.

This behavior might not be desirable and you might want to configure fine grained access rules for individual connectors that you deploy. For example, you might want to have your connectors to be able to manage specific Kafka resources only, but not have access to other resources. To achieve this, you must configure connectors to be required to specify unique connection credentials for the Kafka service connection. Configuration is done in Cloudera Manager using Connector Kafka Client Configuration Override Policy and Require Connectors To Override Kafka Client JAAS Configuration Kafka service properties.

Procedure

1. In Cloudera Manager select the Kafka service.
2. Go to Configuration.
3. Find and configure the following properties.

Table 4:

Property Name	Description
Connector Kafka Client Configuration Override Policy	Specifies what client configuration can be overridden by the connector. Its value is the class name or alias of the ConnectorClientConfigOverridePolicy implementation. The framework includes three implementations (policies). These are None, All, and Principal. • If set to None, configurations cannot be overridden. This is the default setting. • If set to Principal, SASL configurations can be overridden • If set to All, all configurations can be overridden
Require Connectors To Override Kafka Client JAAS Configuration	Specifies whether Connector configurations should override the sasl .jaas.config property of the Kafka clients used by the connector. If set to true, connector configurations must override the client's sasl .jaas.config property. The override must also not match the Kerberos credentials of the Connect worker. Additionally, if set to true, a Unio nComposite'policy is installed under the connector.client.config.over ride.policy configuration. If Connector Kafka Client Configuration Override Policy is set to None, it is upgraded to Principal. Otherwise, the existing policy is also applied by the composite policy.

4. Click Save Changes.
5. Restart the Kafka service.

What to do next

- Update the configuration of all existing connectors to include the sasl.jaas.config property with a valid JAAS configuration.
- Ensure that the principal specified for the connector has appropriate permissions assigned to it in your authorization service.

Configuring a Nexus repository allow list

To harden the security of your Kafka Connect deployment, you can configure a Nexus repository allow list. If an allow list is configured, Stateless NiFi Source and Sink connectors are only able to fetch NiFi extensions from the specified list of Nexus repositories.

About this task

NiFi dataflows within Kafka Connect are deployed using the Stateless NiFi Source and Sink connectors. When you deploy your custom developed dataflows with the Stateless NiFi connectors, you can configure the nexus.url property. This property specifies the URL of a Nexus repository that hosts the NiFi extensions that are not bundled with the connectors but are required by the dataflow that is run within the connector.

By default connectors you deploy are allowed to connect to any Nexus repository that you specify. However, this may pose a security risk. Because of this, you can configure Kafka Connect to only allow its connectors to connect to specific Nexus repositories. That is, you can configure a Nexus server allow list. If an allow list is configured, connectors are only allowed to connect and fetch NiFi extensions from the servers that are included in the list.

The allow list is configured in Cloudera Manager using the List Of Allowed Nexus Repository Urls property.

Procedure

1. In Cloudera Manager, select the Kafka service.
2. Go to Configuration.
3. Find and configure the List Of Allowed Nexus Repository Urls property.

   Add the URLs of the Nexus repositories that you want to allow your connectors to fetch extensions from.
4. Click Save Changes.
5. Restart the Kafka service.

Results

The Nexus URLs specified in the nexus.url property of the Stateless NiFi Source and Sink connectors are validated against the list of configured URLs. Connections are only allowed to the repositories configured in the list.

Single Message Transforms

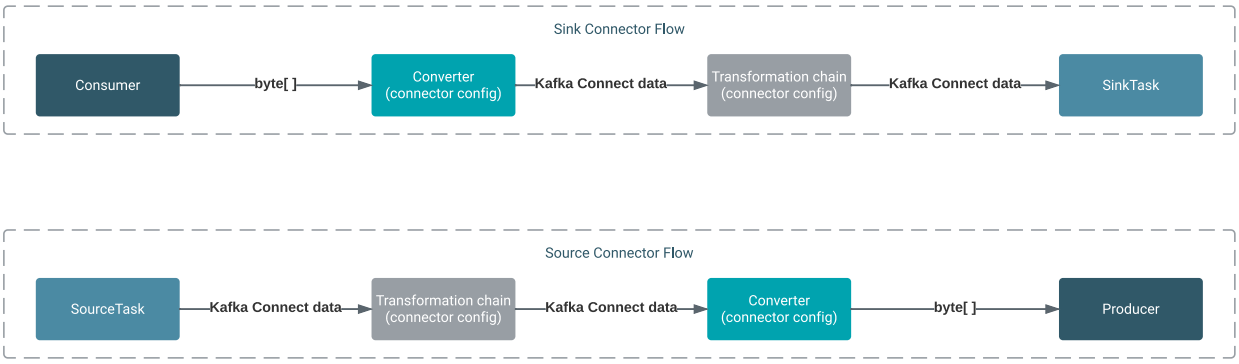
Single Message Transforms (SMT) is a message transformation framework that you can deploy on top of Kafka Connect connectors to apply message transformations and filtering. Learn about the SMT framework as well as the transformation plugins available in Cloudera.

Kafka Connect connectors provide ready-to-use tools to integrate between Kafka and external data stores. Still, in many use cases, the data moved by the connectors require some sanitization and transformation. To provide extra flexibility built on top of connectors, Kafka Connect also supports an SMT framework.

The SMT framework installs a transformation chain on top of connectors that modifies and filters the data on a single message basis. An SMT chain consists of transform and predicate plugins. Transform plugins are used to modify the data. For example, you can insert, replace, mask as well as perform various other modifications on the messages

moved by connectors. Predicate plugins are used to add additional logic to your chain so that the transformation chain is only applied to messages which satisfy specified conditions.

The SMT framework requires that data is converted to the Kafka Connect internal data format. This data format is specific to Kafka Connect and consists of a structure and schema descriptor (SchemaAndValue) specific to Connect.



Supported SMT plugins

Cloudera Kafka ships with and supports the following SMT plugins:

- All Apache Kafka plugins. For more information, see Transformations in the Apache Kafka documentation.
- All Debezium plugins. For more information, see Transformations in the Debezium documentation.
- The following Cloudera specific plugins:
  - ConvertFromBytes on page 38
  - ConvertToBytes on page 40

Note: The ConverFromBytes and ConvertToBytes transformation plugins transform binary data to and from the Kafka Connect internal data format. These plugins are specifically developed to enable the use of the SMT framework with connectors that only support binary data.

Configuring an SMT chain

Learn how to configure a Single Message Transformation (SMT) chain for Kafka Connect connectors.

SMT chains can be configured within the configuration of a Kafka Connect connector using SMT specific configuration properties. To set up a chain, you first define your transformation chain with the transforms property and optionally define your predicates using the predicates property. Afterward, you use transforms.* and predicates.* to configure the plugins in the chain. For example, the following configuration snippet sets up a transformation chain that filters messages based on their header and removes a specified field from messages.

```
transforms=FilterAudit,MaskField

transforms.MaskField.type=org.apache.kafka.connect.transforms.MaskField$Va
lue
transforms.MaskField.fields=CreditCardNumber

transforms.FilterAudit.type=org.apache.kafka.connect.transforms.Filter
transforms.FilterAudit.predicate=IsAudit
transforms.FilterAudit.negate=false

predicates=IsAudit
predicates.IsAudit.type=org.apache.kafka.connect.transforms.predicates.Has
HeaderKey
predicates.IsAudit.name=Audit
```

The following sections go through the properties in this example and give an overview on how to set up a transformation chain.

Configuring transforms

The transforms property contains a comma-separated list of transformation aliases. Each alias represents one step in the transformation chain. The aliases you add to the property are arbitrary names, they are used in other properties to configure that particular transformation step. For example, the following defines a two step transformation chain.

```
transforms=FilterAudit,MaskField
```

The transforms.*[***ALIAS***]*.type property specifies which transformation plugin should be used in a transformation step. *[***ALIAS***]* is one of the aliases that you specified in transforms. The value of the property is the fully qualified name of the transformation plugin that should be used in the step. For example, the following line specifies org.apache.kafka.connect.transforms.MaskField$Value as the plugin for the MaskField step.

```
transforms.MaskField.type=org.apache.kafka.connect.transforms.MaskField$
Value
```

Many transformation plugins support changing both the key and the value of a record. For these plugins, typically, a nested value or key class can be referenced as the type.

The transforms.*[***ALIAS***].[***KEY***]* property is used to configure the transformation plugins in your chain. This property is passed to the transformation plugin itself with transforms.*[***ALIAS***]* stripped from the property key. *[***ALIAS***]* is the alias of a plugin you specified in transforms. *[***KEY***]* is a property key that the plugin accepts. For example, the MaskField plugin has a fields property that specifies which fields should be removed from the structure.

```
transforms.MaskField.fields=CreditCardNumber
```

Configuring predicates

Predicates are a separate set of plugins. You use them to conditionally enable certain steps in the transformation chain. Predicates are configured in a similar way to transforms. You must specify the predicate aliases, associate the aliases with a plugin, and set plugin specific properties.

```
predicates=IsAudit
predicates.IsAudit.type=org.apache.kafka.connect.transforms.predicates.Ha
sHeaderKey
predicates.IsAudit.name=Audit
```

In this example the IsAudit predicate is an instance of the HasHeaderKey predicate plugin. This predicate returns true for records where a specific header key is present. predicates.IsAudit.name=Audit configures the predicate to look for the Audit header in the records.

After a predicate is set up, you can associate the predicate with any number of transformation steps using the pred icate property. If a predicate is associated with a transformation, that transformation step is only applied to the messages that match the condition specified in the predicate.

A good example for using a predicate is the Filter transformation plugin. This is because Filter filters (drops) all messages by default. Therefore, it must be used together with predicates to specify filtering logic. For example, the following configuration instructs the SMT framework that the FilterAudit step should only be invoked for messages where the IsAudit predicate returns true. That is, all messages with the Audit header will be filtered and will not be transformed by any subsequent steps in the transformation chain.

```
transforms.FilterAudit.predicate=IsAudit
transforms.FilterAudit.negate=false
```

The condition of a predicate can be inverted using negate. If negate is set to true, the SMT framework applies the transformation to any record that does **not** match the condition. For example, the following configuration instructs the SMT framework that the FilterAudit step should only be invoked for messages where the IsAudit predicate returns false.

```
transforms.FilterAudit.predicate=IsAudit
transforms.FilterAudit.negate=true
```

ConvertFromBytes

ConvertFromBytes is a Cloudera specific transformation plugin that converts binary data to the Kafka Connect internal data format. You can use this plugin to make connectors that only support binary data compatible with the Single Message Transforms (SMT) framework.

Fully qualified names

- com.cloudera.dim.kafka.connect.transformations.convert.ConvertFromBytes$Key
- com.cloudera.dim.kafka.connect.transformations.convert.ConvertFromBytes$Value

Description

Important: Ensure that you have an in-depth understanding about the following aspects of the connector that you plan on using with ConvertFromBytes.

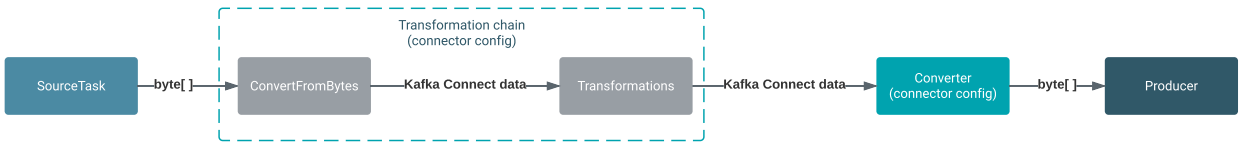
- The type of the connector
- The converter type used by the connector
- The header converter type used by the connector

These aspects of a connector heavily influence how the plugin and how the transformation chain that includes this plugin must be configured.

The ConvertFromBytes transformation plugin accepts binary data and converts it into the Kafka Connect internal data format with a nested converter that transforms binary data. To support header based converter logic, for example when using the AvroConverter with the schema ID encoded in the record header, the plugin requires a header converter to correctly transform record headers when interacting with the converter. This plugin supports both key and value conversion.

Using this plugin with connectors that only support binary data makes the connector fully compatible with the SMT framework. On their own, connectors that only support binary data have limited compatibility with transformations even if the binary data is structured. This is because transformations are only fully supported on data that is in the Kafka Connect internal data format. Binary only connectors like the Mirror Source (MirrorSourceConnector) or the Stateless NiFi Source connectors emit data that has the BYTES schema and do not provide conversion to the Kafka Connect internal data format by default. When you use a binary only connector with the ConvertFromBytes plugin, the binary data is parsed into a compatible structure, which can then be further processed with the transformation chain.

Figure 1: Source connector example flow with ConvertFromBytes



Example

The following configuration example adds a ConvertFromBytes transformation as a first step of the transformation chain. The conversion uses a schemaless JSON transformation to parse the binary data. The transformation steps, the

connector, or the converter, whichever comes directly after FromBytes, receives a properly structured record instead of binary data.

```
{
     "transforms" : "FromBytes,...",
     "transforms.FromBytes.type" : "com.cloudera.dim.kafka.connect.transformations.convert.ConvertFromBytes$Value",
     "transforms.FromBytes.converter" : "org.apache.kafka.connect.json.JsonConverter",
     "transforms.FromBytes.converter.schemas.enable" : "false"
}
```

Configuration properties

Table 5: ConvertFromBytes properties reference

Property	Default Value	Required	Description
converter		True	The fully qualified name of the converter implementation to use. For example: com.cloudera.dim .kafka.connect.converts.Avro Converter
header.converter	org.apache.kafka.connect.storage .SimpleHeaderConverter	True	The fully qualified name of the header converter implementation to use. This converter must match the header converter of the connector.
converter.		False	A configuration prefix. Use this prefix to configure the properties of the converter specified in conv erter. Property keys and values specified with the prefix are passed directly to the converter with the prefix stripped. For example: `transformers.[***TRANSFORM ALIAS***].converter.[***CONVERTER PROPERTY KEY***]:[***CONVERTER PROPERTY VALUE***]`

Property	Default Value	Required	Description
header.converter.		False	A configuration prefix. Use this prefix to configure the properties of the header converter specified in header.converter. Property keys and values specified with the prefix are passed directly to the header converter with the prefix stripped. For example: `transforms.[***TRANSFORM ALIAS***].converter.[***HEADER CONVERTER PROPERTY KEY***]:[***HEADER CONVERTER PROPERTY VALUE***]`

ConvertToBytes

ConvertToBytes is a Cloudera specific transformation plugin that converts Kafka Connect internal data to binary data. You can use this plugin to make connectors that only support binary data compatible with the Single Message Transforms (SMT) framework.

Fully qualified names

- com.cloudera.dim.kafka.connect.transformations.convert.ConvertToBytes$Key
- com.cloudera.dim.kafka.connect.transformations.convert.ConvertToBytes$Value

Description

Important: Ensure that you have an in-depth understanding about the following aspects of the connector that you plan on using with ConvertToBytes.

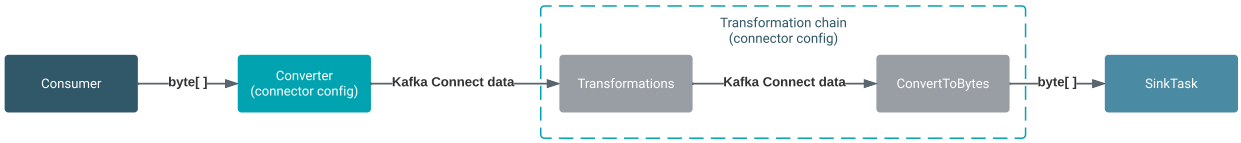
- The type of the connector
- The converter type used by the connector
- The header converter type used by the connector

These aspects of a connector heavily influence how the plugin and how the transformation chain that includes this plugin must be configured.

The ConvertToBytes transformation plugin accepts data in the Kafka Connect internal data format and converts it to binary data with a nested converter. To support header based converter logic, for example when using the AvroConverter with the schema ID encoded in the record header, the plugin requires a header converter to correctly transform record headers when interacting with the converter. This plugin supports both key and value conversion.

Using this plugin with connectors that only support binary data makes the connector fully compatible with the SMT framework. On their own, connectors that only support binary data, for example the Stateless NiFi Sink connector, have limited compatibility with transformations even if the binary data is structured. This is because the format of the data after transformations are carried out is normally the Kafka Connect internal data format. Binary only connectors, however, expect data that has the BYTES schema and do not provide conversion from the Kafka Connect internal data format by default. When you use the ConvertToBytes plugin with a binary only connector, the structured data is converted to binary format, which can then be picked up by the connector.

Figure 2: Sink connector example flow with ConvertToBytes



Example

The following configuration example adds a `ConvertToBytes` transformation as the last step of the transformation chain. The conversion uses a schemaless JSON transformation to serialize the structured data. The transformation steps, the connector, or the converter, whichever comes directly after ToBytes, receives a properly structured record instead of binary data.

```
{
     "transforms" : "...,ToBytes",
     "transforms.ToBytes.type" : "com.cloudera.dim.kafka.connect.transfor
mations.convert.ConvertToBytes$Value",
     "transforms.ToBytes.converter" : "org.apache.kafka.connect.json.Json
Converter",
     "transforms.ToBytes.converter.schemas.enable" : "false"
}
```

Configuration properties

Table 6: ConvertToBytes properties reference

Property	Default Value	Required	Description
converter		True	The fully qualified name of the converter implementation to use. For example: com.cloudera.dim .kafka.connect.converts.Avro Converter
header.converter	org.apache.kafka.connect.storage .SimpleHeaderConverter	True	The fully qualified name of the header converter implementation to use. This converter must match the header converter of the connector.
converter.		False	A configuration prefix. Use this prefix to configure the properties of the converter specified in conv erter. Property keys and values specified with the prefix are passed directly to the converter with the prefix stripped. For example: `transfor ms.[***TRANSFORM ALIAS***].con vert er.[***CONVERTER PROPERTY KEY***]:[***CONVERTER PROPERTY VALUE***]`

Property	Default Value	Required	Description
header.converter.		False	A configuration prefix. Use this prefix to configure the properties of the header converter specified in header.converter. Property keys and values specified with the prefix are passed directly to the header converter with the prefix stripped. For example: `transforms.[***TRANSFORM ALIAS***].converter.[***HEADER CONVERTER PROPERTY KEY***]:[***HEADER CONVERTER PROPERTY VALUE***]`

Connectors

Learn what Kafka Connect connectors are shipped with Cloudera Runtime.

Cloudera Runtime comes prepackaged with a number of Cloudera developed Kafka Connect connectors. In addition, connectors that come packaged with the version of Apache Kafka that is included in Cloudera Runtime are also available for use. Manually installing and using your own custom connectors is also possible. The following collects the connectors shipped in Cloudera Runtime.



Note: Cloudera supports installing and using third party plugins, including connectors, but provides no professional support for their use.

Debezium connectors

Debezium connectors capture changes from a wide variety of databases and produce the captured changes into Kafka. Using Debezium connectors makes it possible for your applications to consume and respond to change events regardless of where the changes originated from.

Table 7: Debezium connectors

Connector	Description
Debezium Db2 Source	The Debezium Db2 Source connector captures row-level changes in Db2 database tables and transfers the changes to Kafka topics.
Debezium MySQL Source	The Debezium MySQL Source connector reads the binary log (binlog) of a MySQL server, produces change events for row-level INSERT, UPDATE, and DELETE operations, and transfers the changes to Kafka topics.
Debezium Oracle Source	The Debezium Oracle Source connector captures and records row-level changes that occur in databases on an Oracle server, including tables that are added while the connector is running, and transfers the changes to Kafka topics.
Debezium PostgreSQL Source	The Debezium PostgreSQL Source connector captures row-level INSERT, UPDATE, and DELETE operations, produces change evens for each change, and transfers the changes to Kafka topics.

Connector	Description
Debezium SQL Server Source	The Debezium SQL Server Source connector captures row-level INSERT, UPDATE, and DELETE operations that occur in the schemas of a SQL Server database, produces change events for each change, and transfer the changes to Kafka topics.

Note: Cloudera does not provide documentation on how to use or configure Debezium connectors. Documentation only covers the setup steps that are required before you can deploy the connectors in a Cloudera cluster. To learn more about these connectors, see the appropriate version of the Debezium Documentation.

Third-party connectors

Third-party connectors are developed and maintained outside Cloudera. Cloudera certifies selected third-party connectors for use in Cloudera Runtime.

Table 8: Third-party connectors

Connector	Description
Apache Iceberg Sink	The Apache Iceberg Sink connector reads records from Kafka topics and writes them to Iceberg tables.

Note: Cloudera does not provide documentation on how to use or configure the Apache Iceberg Sink connector. Documentation only covers the setup steps that are required before you can deploy the connector in a Cloudera cluster. To learn more about this connector, see the Apache Iceberg Sink Connector documentation.

Stateless NiFi connectors

The Stateless NiFi Source and Sink connectors allow you to run NiFi dataflows within Kafka Connect. Using these connectors can grant you access to a number of NiFi features without having the need to deploy or maintain NiFi on your cluster.

Stateless NiFi connectors fall in to two categories. You have the base Stateless NiFi Source (`StatelessNiFiSource`) and Stateless NiFi Sink (`StatelessNiFiSink`) connectors. In addition, there are a number of ready-to-use connectors based on Stateless NiFi Source and Sink. These Stateless NiFi-based connectors run predefined dataflows developed by Cloudera and cover common data movement use cases.

Table 9: Base connectors

Connector	Description
Stateless NiFi Source	The Stateless NiFi Source and Sink connectors allow you to run NiFi dataflows within Kafka Connect. Using these connectors can grant you access to a number of NiFi features without having the need to deploy or maintain NiFi on your cluster.
Stateless NiFi Sink	

Table 10: Predefined flows/Stateless NiFi-based connectors

Connector	Description
HTTP Source	The HTTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The HTTP Source connector listens on a port for HTTP POST requests and transfers the request body to a Kafka topic.
JDBC Source	The JDBC Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The JDBC Source connector reads records from a database table and transfers each record to Kafka in Avro or JSON format.

Connector	Description
JMS Source	The JMS Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The JMS Source Connector consumes messages from a JMS broker and transfers the message body to Kafka.
MQTT Source	The MQTT Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The MQTT Source connector consumes messages from an MQTT broker and transfers them to Kafka.
SFTP Source	The SFTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The SFTP Source connector obtains files from an SFTP server and transfers them to Kafka.
Syslog TCP Source	The Syslog TCP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The Syslog TCP Source connector listens on a port for syslog messages over TCP and transfers them to Kafka.
Syslog UDP Source	The Syslog UDP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The Syslog UDP Source connector listens on a port for syslog messages over UDP and transfers them to Kafka.
ADLS Sink	The ADLS Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka connect framework. The ADLS Sink connector fetches messages from Kafka and uploads them to ADLS.
HDFS Stateless Sink	The HDFS Stateless Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. The HDFS Stateless Sink Connector fetches messages from Kafka and stores them in HDFS.
HTTP Sink	The HTTP Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The HTTP Sink connector obtains messages from a Kafka topic and transfers their content in a HTTP POST requests to a specified endpoint.
InfluxDB Sink	The InfluxDB Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. The InfluxDB Sink Connector fetches messages from Kafka and loads them to InfluxDB.
JDBC Sink	The JDBC Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The JDBC Sink connector fetches messages from Kafka and loads them into a database table.
Kudu Sink	The Kudu Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The Kudu Sink connector fetches messages from Kafka and loads them into a table in Kudu.
S3 Sink	The S3 Sink Connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. The S3 Sink connector fetches messages from Kafka and uploads them to AWS S3.

Standard connectors

The following are either Cloudera developed connectors or connectors that come packaged with Apache Kafka.

Table 11: Standard connectors

Connector	Description
Amazon S3 Sink	The Amazon S3 Sink connector is Cloudera developed connector that consumes data from Kafka topics and streams the data to an S3 bucket.

Connector	Description
HDFS Sink	The HDFS Sink connector is a Cloudera developed connector that transfer data from Kafka topics to files on HDFS clusters.
`MirrorSourceConnector`	The `MirrorSourceConnector` is a connector used internally by Streams Replication Manager. Within Streams Replication Manager, this connector is responsible for replicating topics between the source and target cluster. Standalone use of this connector is not recommended by Cloudera.
`MirrorHeartbeatConnector`	The `MirrorHeartbeatConnector` is a connector used internally by Streams Replication Manager. Within Streams Replication Manager, this connector is responsible for creating the heartbeats topic in the target cluster. It also periodically produces heartbeats into the heartbeats topic. Standalone use of this connector is not recommended by Cloudera.
`MirrorCheckpointConnector`	The `MirrorCheckpointConnector` is a connector used internally by Streams Replication Manager. Within Streams Replication Manager, this connector is responsible for replicating the committed group offsets between the source and target cluster. In addition, the connector is also capable of periodically applying the offsets to the consumer groups in the target cluster. Standalone use of this connector is not recommended by Cloudera.

Note: Standalone use of the `MirrorCheckpointConnector`, `MirrorHeartbeatConnector`, and `MirrorSourceConnector` is not recommended by Cloudera. Additionally Cloudera does not provide documentation on how to use or configure these connectors. However, you can review Streams Replication Manager driver architecture to learn more about how Streams Replication Manager uses these connectors internally.

Example connectors (non-production)

The FileStream Source and Sink connectors are example connectors packaged with Apache Kafka. Both FileStream connectors are meant to be used to demonstrate the capabilities of Kafka Connect and are not production ready. As a result, even though these connectors are shipped with Cloudera Runtime, they must be installed before they can be deployed. Prior to connector installation, the connectors are also not selectable for deployment on the Streams Messaging Manager UI. See Installing FileStream connectors on page 47 for installation steps.

Table 12: Non-production example connectors

Connector	Description
FileStream Sink	The FileStream Sink connector reads data from Kafka and transfers that data to a local file.
FileStream Source	The FileStream Source connector reads data from a file and transfers the data to Kafka.

Note: Cloudera does not provide documentation regarding how to use or configure FileStream connectors. For more information on connector use, see the Apache Kafka Documentation.

Note:

Information presented on this page contains content that is modified and adapted from the Debezium Documentation by the Debezium Community, which is licensed under CC BY 3.0.

Installing Kafka Connect connectors

Learn how to install custom developed (third party) connectors as well as the FileStream connectors in Cloudera.

Kafka Connect connectors are distributed as:

- A directory of JAR files:

  The directory includes the JAR for the connector itself, as well as all its dependencies.
- An uber JAR/FAT JAR/JAR with dependencies file:

  This is a single JAR file that contains the connector, as well as its dependencies.

In Cloudera, all connectors that do not come prepackaged with Cloudera Runtime, as well as the FileStream connectors (FileStreamSourceConnector and FileStreamsSinkConnector), must be installed manually. This is done by making the connector JAR files available on all cluster hosts in a specific location. After installation is complete, you will be able to deploy the connector using the Streams Messaging Manager UI (recommended), Streams Messaging Manager API, or Kafka Connect API.

The location of the JAR files is determined by the Kafka Connect role's plugin.path property. Kafka Connect discovers connectors by looking at this directory path on the host machines. By default, the plugin.path property is set to /var/lib/kafka. This means that, by default, any connector placed in this directory will be discovered by Kafka Connect. Cloudera recommends that you use the default path.

The installation steps differ for custom developed connectors and the FileStream connectors. This is because the JAR file for the FileStream connectors is by default available on Cloudera cluster hosts. Additionally, FileStream connectors can be installed with an alternate installation method that involves the usage of an advanced configuration snippet.



Note: Cloudera supports installing and using third party plugins, including connectors, but provides no professional support for their use.

Installing custom developed Kafka Connect connectors

Learn how to install custom developed (third party) connectors in Cloudera.

Before you begin

The following steps assume that plugin.path is set to /var/lib/kafka, which is the default path.



Important: Ensure that the JDK version of your platform is compatible with the Java version that the connector was compiled with. Mismatched versions prevent connectors from loading properly.



Important: Steps 1 and 2 must be carried out on all hosts in the cluster that have Kafka Connect roles deployed on them.

Procedure

1. Log in to a host that is running a Kafka Connect role.
2. Make the connector files available in or readable from /var/lib/kafka.

   How you choose to complete this step will largely depend on your cluster environment. For example:

   - You can download or copy the files directly to /var/lib/kafka.
   - You can choose to place connector files in a location different from /var/lib/kafka and create symlinks that point to the location where the connector files are available.

   Regardless of what method you choose, this step is considered complete once the JAR files are readable from /var/lib/kafka.
3. Ensure that the Kafka system user has read, write, and execute permissions for the plugin directory.

   This step is only required if the default kafka system user was changed.
4. Restart all Kafka Connect roles:
   a) In Cloudera Manager, select the Kafka service.
   b) Go to Instances.
   c) Select all Kafka Connect instances by checking the checkbox next to each instance.
   d) Click Actions for selectedRestart.

e) Click Restart to confirm.

The roles are restarted once a **Finished** status is displayed.

Results

The connector is installed and is available for deployment. You are now able to deploy and manage the new connector from the Streams Messaging Manager UI.

What to do next

Deploy an instance of the connector using Streams Messaging Manager. For more information, see *Deploying and managing Kafka Connect connectors in Streams Messaging Manager*.

Related Information

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Installing FileStream connectors

Learn how to install the FileStream example connectors (FileStreamSourceConnector and FileStreamSinkConnector) that are shipped with Cloudera Runtime but are not installed by default. You can choose between two installation methods.

About this task

The JAR file for the FileStream connectors is shipped with Cloudera Runtime and is readily available on the cluster hosts. However, the file is not added to the Kafka Connect plugin.path directory by default. This is because the connectors are meant to be used for demonstrating the capabilities of Kafka Connect and are not production ready. As a result, even though these connectors do come packaged with Cloudera Runtime, they must be installed before they can be deployed. The JAR file is located at /opt/cloudera/parcels/CDH/jars/connect-file-*[***KAFKA COMPONENT VERSION***]*.jar.

You have two options when installing the FileStream connectors. You can install the connectors by copying or symlinking the JAR files to the plugin.path directory. Alternatively, you can add the location of the JAR file to the Kafka Connect role's CLASSPATH environment variable using an advanced configuration snippet.

The main difference between the two installation methods is that copying or symlinking the file requires that you log in to each Kafka Connect host in your cluster. Using an advanced configuration snippet, on the other hand, enables you to install the connector on all hosts by changing a single property in Cloudera Manager.

Although using an advanced configuration snippet is more convenient than copying or symlinking, be aware that setting an advanced configuration snippet is considered an advanced configuration practice. Therefore, Cloudera advises caution if you choose to install the FileStream connectors using an advanced configuration snippet.

Before you begin

- Go to Cloudera Runtime component versions and note down the component version of Apache Kafka. You need to specify the version during installation.

  The version is made up of three parts. It contains the upstream Apache Kafka version (first three digits), the Cloudera Runtime version (digits four to six), and the Cloudera Runtime build number (last three digits denominated with a dash). For example: 3.1.1.7.1.8.0-801.
- The following steps assume that plugin.path is set to /var/lib/kafka, which is the default path.

Procedure

1. Install the FileStream connectors:

   For Copy or symlink

   Complete the following actions on all Kafka Connect hosts.

   a. Log in to a host that is running a Kafka Connect role.

b. Copy or symlink the JAR file of the connectors to /var/lib/kafka.

- To copy the file directly to /var/lib/kafka use the following command:

```
cp /opt/cloudera/parcels/CDH/jars/connect-file-[***KAFKA COMPONENT
 VERSION***].jar /var/lib/kafka/
```

- To create a symlink in /var/lib/kafka that points to /opt/cloudera/parcels/CDH/jars/connect-file-*[***KAFKA COMPONENT VERSION***]*.jar use the following command:

```
ln -sf /opt/cloudera/parcels/CDH/jars/connect-file-[***KAFKA
 COMPONENT VERSION***].jar /var/lib/kafka/connect-file-[***KAFKA
 COMPONENT VERSION***].jar
```

Replace *[***KAFKA COMPONENT VERSION***]* with the component version of Apache Kafka. Regardless of what method you choose, this step is considered complete once the JAR file is readable from /var/lib/kafka.

c. Ensure that the Kafka system user has read, write, and execute permissions for the plugin directory.

This step is only required if the default kafka system user was changed.

For Using an advanced configuration snippet

a. In Cloudera Manager, select the Kafka service.

b. Go to Configuration.

c. Find the Kafka Connect Environment Advanced Configuration Snippet (Safety Valve) property.

d. Click ✚ to add a property and enter the following key and value pair:

- Key: CLASSPATH
- Value: /opt/cloudera/parcels/CDH/jars/connect-file-*[***KAFKA COMPONENT VERSION***]*.jar

Replace *[***KAFKA COMPONENT VERSION***]* with the component version of Apache Kafka.

e. Click Save Changes.

2. Restart all Kafka Connect roles:
   a) In Cloudera Manager, select the Kafka service.
   b) Go to Instances.
   c) Select all Kafka Connect instances by checking the checkbox next to each instance.
   d) Click Actions for selectedRestart.
   e) Click Restart to confirm.

   The roles are restarted once a **Finished** status is displayed.

Results

The FileStream connectors are installed and available for deployment. You are now able to deploy and manage the FileStream connectors from the Streams Messaging Manager UI.

What to do next

Deploy an instance of the FileStream Source or FileStream Sink connector using Streams Messaging Manager. For more information, see *Deploying and managing Kafka Connect connectors in Streams Messaging Manager*.

Related Information

Setting an Advanced Configuration Snippet for a Cloudera Runtime Service

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium Db2 Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium Db2 Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium Db2 Source connector, you must download and deploy the Db2 JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium DB2 connector works as well as its configuration and properties, see *Debezium connector for Db2* in the Debezium documentation.



Important: In addition to the following configuration steps, appropriately configuring your Db2 tables is also required. Ensure that you review *Setting up Db2* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the Db2 JDBC driver.

   The JDBC driver can be downloaded from *Maven Central*. Cloudera recommends that you use version 11.5.0.0.
2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

   The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to db2-connector-java.jar. For example:

   ```
   cp ./jcc-[***VERSION***].jar /var/lib/kafka_connect_jdbc/db2-connector-j
   ava.jar
   ```
3. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/db2-connector-java.jar
   ```
4. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The Db2 JDBC driver is deployed on all Kafka Connect hosts. Deploying an instance of the Debezium Db2 Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium Db2 Source connector. Additionally, review *Debezium connector for Db2* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for Db2

Setting up Db2

Maven central

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium MySQL Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium MySQL Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium MySQL Source connector, you must download and deploy the MySQL JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium MySQL connector works as well as its configuration and properties, see the *Debezium connector for MySQL* in the Debezium documentation.

Important: In addition to the following configuration steps, appropriately configuring your MySQL server is also required. Ensure that you review *Setting up MySQL* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the MySQL JDBC driver.

   The JDBC driver can be downloaded from the *MySQL Product Archives* page. Download the platform independent version. Cloudera recommends that you use version 9.1.0.
2. Extract the files from the downloaded archive.

   You can use the tar command or any other archive manager to extract the archive. For example:

   ```
   tar -xf mysql-connector-java-[***VERSION***].tar.gz
   ```
3. Navigate to the directory where the driver was extracted.

   ```
   cd mysql-connector-java-[***VERSION***]
   ```
4. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

   The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to mysql-connector-java .jar. For example:

   ```
   cp ./mysql-connector-java-[***VERSION***].jar /var/lib/kafka_connect_jdbc/
   mysql-connector-java.jar
   ```
5. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/mysql-connector-java.jar
   ```
6. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The MySQL JDBC driver is deployed on all Kafka Connect hosts. Deploying an instance of the Debezium MySQL Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium MySQL Source connector. Additionally, review *Debezium connector for MySQL* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for MySQL

Setting up MySQL

MySQL Product Archives

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium Oracle Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium Oracle Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium Oracle Source connector, you must download and deploy the Oracle JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium Oracle Source connector works as well as its configuration and properties, see the *Debezium Connector for Oracle* in the Debezium documentation.

Note: In addition to the following configuration steps, appropriately configuring your Oracle server is also required. Ensure that you review the *Setting up Oracle* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the Oracle JDBC driver.

   The JDBC driver can be downloaded from *Maven Central*. Cloudera recommends that you use version 19.x, 21.x, or 23.x (ojdbc11.jar).
2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.
3. The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to oracle-connector-jav a.jar. For example:

   ```
   cp ./ojdbc8-[***VERSION***].jar /var/lib/kafka_connect_jdbc/oracle-connect
   or-java.jar
   ```
4. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/oracle-connector-java.jar
   ```
5. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The Oracle JDBC driver is deployed on all Kafka Connect hosts. Deploying an instance of the Debezium Oracle Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium Oracle Source connector. Additionally, review *Debezium Connector for Oracle* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium Connector for Oracle

Setting up Oracle

Maven Central

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium PostgreSQL Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium PostgreSQL Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium PostgreSQL Source connector, you must download and deploy the PostgreSQL JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium PostgreSQL connector works as well as its configuration and properties, see the *Debezium connector for PostgreSQL* in the Debezium documentation.

Important: In addition to the following configuration steps, appropriately configuring your PostgreSQL server is also required. Ensure that you review the *Set up* section in the Debezium documentation for the PostgreSQL connector and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the PostgreSQL JDBC driver.

The JDBC driver can be downloaded from *Maven Central*. Cloudera recommends that you use version 42.7.7.

2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

   The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to postgresql-connector-java.jar. For example:

   ```
   cp ./postgresql-[***VERSION***].jar /var/lib/kafka_connect_jdbc/postgre
   sql-connector-java.jar
   ```

3. Set correct file permissions. For example:

   ```
   chmod 644 /var/lib/kafka_connect_jdbc/postgresql-connector-java.jar
   ```

4. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The PostgreSQL JDBC driver is deployed on all Kafka Connect hosts. Deploying instance of the Debezium PostgreSQL Source connector in your cluster is now possible

What to do next

Deploy an instance of the Debezium PostgreSQL Source connector. Additionally, review *Debezium connector for PostgreSQL* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for PostgreSQL

Set up (PostgreSQL connector)

Maven Central

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Setting up the Debezium SQL Server Source connector

Learn about the Cloudera specific setup steps required before you can deploy the Debezium SQL Server Source connector.

About this task

In Cloudera, before deploying an instance of the Debezium SQL Server Source connector, you must download and deploy the SQL Server JDBC driver on all Kafka Connect hosts. Otherwise, you will not be able to deploy the connector. The following list of steps walks you through this process.

For more information regarding how the Debezium PostgreSQL connector works as well as its configuration and properties, see the *Debezium connector for SQL Server* in the Debezium documentation.

Important: In addition to the following configuration steps, appropriately configuring your SQL Server database is also required. Ensure that you review the *Setting up SQL Server* in the Debezium documentation and complete all setup tasks either before or after completing the following steps.

Procedure

1. Download the SQL Server JDBC driver.

   The JDBC driver can be downloaded from the *mssql-jdbc Github repository*. Cloudera recommends that you use version 12.4.2.jre8.

2. Copy and rename the driver to the location specified in the JDBC Driver Jars Path For Debezium Kafka property.

The default location is /var/lib/kafka_connect_jdbc. The JDBC driver must be renamed to mssql-connector-java .jar. For example:

```
cp ./mssql-jdbc-[***VERSION***].jre8.jar /var/lib/kafka_connect_jdbc/ms
sql-connector-java.jar
```

3. Set correct file permissions. For example:

```
chmod 644 /var/lib/kafka_connect_jdbc/mssql-connector-java.jar
```

4. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The SQL Server JDBC JDBC driver is deployed on all Kafka Connect hosts. Deploying instance of the Debezium SQL Server Source connector in your cluster is now possible.

What to do next

Deploy an instance of the Debezium SQL Server Source connector. Additionally, review *Debezium connector for SQL Server* in the Debezium documentation to learn more about the connector and its features.

Related Information

Debezium connector for SQL Server

Setting up SQL Server

mssql-jdbc GitHub repository

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

History topic producer and consumer properties reference

This section collects the required pass-through properties for the history topic. The list includes both producer and consumer properties. When you configure the Debezium SQL Server connector in a secure environment, you must configure both the producer and consumer properties.

- database.history.kafka.bootstrap.servers=${cm-agent:ENV:KAFKA_BOOTSTRAP_SERVERS}
- database.history.kafka.topic=sqlserver.schema-changes.history
- database.history.producer.security.protocol=SASL_SSL
- database.history.producer.ssl.keystore.location=${cm-agent:ENV:CONNECT_SSL_SERVER_KEYSTORE_LOCATION}
- database.history.producer.ssl.keystore.password=${cm-agent:ENV:CONNECT_SSL_SERVER_KEYSTORE_PASSWORD}
- database.history.producer.ssl.truststore.location=${cm-agent:ENV:CONNECT_SSL_SERVER_TRUSTSTORE_LOCATION}
- database.history.producer.ssl.truststore.password=${cm-agent:ENV:CONNECT_SSL_SERVER_TRUSTSTORE_PASSWORD}
- database.history.producer.ssl.key.password=${cm-agent:ENV:CONNECT_SSL_SERVER_KEYSTORE_KEYPASSWORD}
- database.history.producer.sasl.kerberos.service.name=${cm-agent:ENV:kafka_service_user_name}
- database.history.producer.sasl.mechanism=GSSAPI
- database.history.producer.sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true storeKey=true serviceName="${cm-agent:ENV:kafka_service_user_name}" keyTab="${cm-agent:keytab}" principal="${cm-agent:ENV:kafka_connect_service_principal}"
- database.history.consumer.security.protocol=SASL_SSL
- database.history.consumer.ssl.keystore.location=${cm-agent:ENV:CONNECT_SSL_SERVER_KEYSTORE_LOCATION}
- database.history.consumer.ssl.keystore.password=${cm-agent:ENV:CONNECT_SSL_SERVER_KEYSTORE_PASSWORD}

- database.history.consumer.ssl.truststore.location=${cm-agent:ENV:CONNECT_SSL_SERVER_TRUSTSTORE_LOCATION}
- database.history.consumer.ssl.truststore.password=${cm-agent:ENV:CONNECT_SSL_SERVER_TRUSTSTORE_PASSWORD}
- database.history.consumer.ssl.key.password=${cm-agent:ENV:CONNECT_SSL_SERVER_KEYSTORE_KEYPASSWORD}
- database.history.consumer.sasl.mechanism=GSSAPI
- database.history.consumer.sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true storeKey=true serviceName="${cm-agent:ENV:kafka_service_user_name}" keyTab="${cm-agent:keytab}" principal="${cm-agent:ENV:kafka_connect_service_principal}"

cloudera.offset.flush.interval.ms

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting. Type: Long. Default: inherits role-level offset.flush.interval.ms when not set.

Related Information

Pass-through database history properties for configuring producer and consumer clients | Debezium

Setting up the Apache Iceberg Sink connector

Install the Apache Iceberg Sink connector plugin on each Kafka Connect worker and configure Ranger policies so you can deploy a connector instance in Streams Messaging Manager.

About this task

Before you deploy an instance of the Apache Iceberg Sink connector, you must download and copy the connector plugin ZIP to every Kafka Connect worker. Cloudera does not automate this installation. As a standalone plugin, the connector may require additional JARs for your storage and catalog configuration. Collect the required JARs and place them in the extracted connector directory on each worker.

The connector supports Amazon S3, Azure Data Lake Storage (ADLS), and Ozone as storage backends. For catalogs, it supports Hive Metastore and JDBC catalogs.

Configure the connector entirely through the connector JSON in Streams Messaging Manager. Use iceberg.catalog.* properties for catalog and storage settings, and iceberg.hadoop.* properties for Hive Metastore client and Kerberos settings.

For connector properties, single message transformations (SMTs), and routing, see the *Apache Iceberg Sink Connector documentation* in Related information.

Before you begin

Import CA certificates into the Cloudera Manager Auto-TLS truststore. The required certificates depend on your storage backend:

- Amazon S3: Import all four Amazon Root CA certificates (amazonrootca1 through amazonrootca4) into the Auto-TLS truststore. In Cloudera Manager, go to Administration Security Update Auto-TLS Truststore . Restart Hive and Kafka Connect on the affected hosts.
- Azure ADLS: Import the DigiCert Global Root G2 or the Microsoft RSA Root Certificate Authority 2017 certificate into the Auto-TLS truststore. In Cloudera Manager, go to Administration Security Update Auto-TLS Truststore .
- Ozone: No additional certificate import is required.

If your cluster mounts /tmp with noexec, set the JAVA_TOOL_OPTIONS Kafka Connect worker environment variable to -Djava.io.tmpdir=/var/lib/kafka before you restart the service.

On Kerberos clusters, the upstream Apache Iceberg connector's bundled Hive Metastore client (iceberg-kafka-connect-runtime-hive, Hive 2.3.10) connects to HMS over SASL/Kerberos on plain Thrift, not TLS on the HMS Thrift port. This connector does not support TLS on the HMS Thrift interface combined with Kerberos SASL, and the upstream Apache Iceberg Kafka Connect guide does not document this combination. To disable HMS Thrift SSL in Cloudera Manager:

1. On the Hive service Configuration page, add hive.metastore.use.SSL=false to Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml.
2. Restart Hive Metastore.

Procedure

1. Download the certified Apache Iceberg Sink connector ZIP from Cloudera Archive.
2. Copy the connector ZIP to the plugin path on each Kafka Connect worker.

   For example:

   ```
   scp iceberg-kafka-connect-runtime-hive-[***VERSION***].zip roo
   t@[***HOST***]:/var/lib/kafka
   ```

3. Extract the ZIP on each worker.

   For example:

   ```
   ssh root@[***HOST***] "unzip -d /var/lib/kafka/ /var/lib/kafka/iceberg-ka
   fka-connect-runtime-hive-[***VERSION***].zip"
   ```

4. Set file permissions so the kafka user can read the plugin files.
5. Configure storage-specific settings for Hive Metastore and Kafka Connect.

   The required configuration depends on your storage backend.

   Amazon S3

   On the Hive service Configuration page, add the following bucket-scoped S3A settings to Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml. Replace the placeholders with your values.

   ```
   fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem
   fs.s3a.bucket.[***BUCKET***].access.key=[***ACCESS_KEY***]
   fs.s3a.bucket.[***BUCKET***].secret.key=[***SECRET_KEY***]
   fs.s3a.bucket.[***BUCKET***].aws.credentials.provider=org.apache
   .hadoop.fs.s3a.SimpleAWSCredentialsProvider
   fs.s3a.bucket.[***BUCKET***].endpoint=s3.[***REGION***].amazona
   ws.com
   fs.s3a.bucket.[***BUCKET***].endpoint.region=[***REGION***]
   ```

   Azure ADLS

   The connector requires two sets of ADLS credentials:

   - An App Registration secret: AZURE_CLIENT_ID, AZURE_TENANT_ID, and AZURE_CLIENT_SECRET.
   - Storage account credentials: azureStorageAccountName, azureStorageAccountKey, and azureStorageContainer.

   On the Hive service Configuration page, add the storage account key to Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml. Hive Metastore derives the remaining settings from the warehouse path.

   ```
   fs.azure.account.key.[***AZURE_STORAGE_ACCOUNT_NAME***].dfs.cor
   e.windows.net=[***AZURE_STORAGE_ACCOUNT_KEY***]
   ```

Add the App Registration secret values as environment variables in the Kafka Connect environment.

Ozone

a. Inject the Ozone filesystem JAR into the connector plugin. The iceberg-kafka-connect-runtime-hive package does not include ozone-filesystem-hadoop3-*.jar. Without it, any ofs:// access fails with ClassNotFoundException. On every Kafka Connect host, find the Iceberg runtime artifact under /var/lib/kafka/ and copy the JAR into its lib/ directory.

b. Provision an iceberg.hadoop-conf-dir directory on each Kafka Connect host. Create a directory that the kafka user can read (for example, /tmp/kc-iceberg-hadoop-conf/) containing:

- core-site.xml, hdfs-site.xml, and ozone-site.xml, copied from the Kafka Connect process directory.
- A hand-authored hive-site.xml containing the HMS URI, sasl.enabled=true, the Kerberos principal, use.SSL=false, and the Ozone filesystem bindings fs.ofs.impl=org.apache.hadoop.fs.ozone.RootedOzoneFileSystem and fs.AbstractFileSystem.ofs.impl=org.apache.hadoop.fs.ozone.RootedOzFs.

c. Create the Ozone volume and bucket, and grant ACLs. Access to Ozone is native (ACL and Ranger). As an Ozone Manager admin, run the following commands on an Ozone Manager host:

```
kinit -kt [***OM_KEYTAB***] [***OM_PRINCIPAL***]
ozone sh volume create /[***VOLUME***]
ozone sh bucket create /[***VOLUME***]/[***BUCKET***]
```

Grant all ACLs on both the volume and the bucket to each user involved, including kafka (the Kafka Connect worker principal) and hive (HMS reads and writes at commit time):

```
for U in hive kafka [***YOUR_PRINCIPAL***]; do
  ozone sh volume addacl -a user:$U:a /[***VOLUME***]
  ozone sh bucket addacl -a user:$U
:a /[***VOLUME***]/[***BUCKET***]
done
```

6. Configure the connector control topic for exactly-once semantics (EOS).

   The connector enables exactly-once semantics (EOS) by default. It uses a compacted Kafka control topic. If your cluster has auto.create.topics.enable set to true, the control topic is created automatically when the connector starts. If not, create a compacted topic first (for example, control-iceberg), then set the topic name with the iceberg.control.topic property. For details, see Control topic in the Apache Iceberg Sink Connector documentation.

7. Configure Ranger policies for Kafka, Schema Registry, and Hadoop SQL.

   Grant these permissions:

   - Kafka: Grant consume, describe, and create on the source topics and consumer group. Also grant consume and describe on the connect-*[***CONNECTOR_NAME***]* and control-topic consumer groups.
   - Schema Registry: Grant read on the Schema Registry service and on the schema group, branch, and version you use.
   - Hadoop SQL: Grant create and write on the target Hive database, table, and storage URL. The storage URL scheme depends on your storage backend:
     - Amazon S3: s3a://*[***BUCKET***]*/*[***WAREHOUSE_PREFIX***]*/*
     - Azure ADLS: abfs s://*[***AZURE_STORAGE_CONTAINER***]*@*[***AZURE_STORAGE_ACCOUNT_NAME***]*.dfs.core.windows.net/*[***WAREHOUSE_PREFIX***]*/*
     - Ozone: ofs://*[***OZONE_SERVICE_ID***]*/*[***VOLUME***]*/*[***BUCKET***]*/*

   ADLS and Ozone require additional Ranger service policies for the kafka principal:

   Azure ADLS

In the CM_ADLS service, grant read and write access on the storage account, container, and warehouse prefix path that the connector uses.

Ozone

In the OZONE service, grant read, list, and read_acl on the volume. Grant read, write, create, delete, list, and all on the bucket and on keys within it.

8. Pre-create the target database.

   The connector automatically creates the target table, but not the database. Create the database through HiveServer2 before you deploy the connector. Use the URI scheme for your storage backend: s3a:// for Amazon S3, abfss:// for ADLS, or ofs:// for Ozone. For example:

```
CREATE DATABASE IF NOT EXISTS [***HIVE_DB***]
  LOCATION 's3a
://[***BUCKET***]/[***WAREHOUSE_PREFIX***]/[***HIVE_DB***].db';
```

9. In Cloudera Manager, restart all Kafka Connect service roles.

Results

The Apache Iceberg Sink connector plugin is on all Kafka Connect hosts. You can now deploy a connector instance in your cluster.

What to do next

Deploy a connector instance in the Streams Messaging Manager UI. For deployment steps, see *Deploying and managing Kafka Connect connectors in Streams Messaging Manager* in Related information.

To automatically add new Schema Registry fields as columns in the Iceberg table, include the following property in the connector configuration:

```
"iceberg.tables.evolve-schema-enabled": "true"
```

Related Information

Apache Iceberg Sink Connector for Kafka Connect

Deploying and managing Kafka Connect connectors in Streams Messaging Manager

Example configurations

Configuration examples for the Apache Iceberg Sink connector that you can import into the Streams Messaging Manager UI.

The following configurations can be imported into the Streams Messaging Manager UI.

Kerberized cluster with Auto-TLS, Hive catalog, and Amazon S3 (single table)

Replace the bracketed placeholders with your values:

```
{
  "name": "[***CONNECTOR_NAME***]",
  "config": {
    "connector.class": "org.apache.iceberg.connect.IcebergSinkConnector",
    "tasks.max": "1",
    "topics": "[***KAFKA_TOPIC***]",

    "iceberg.catalog.type": "hive",
    "iceberg.catalog.uri": "thrift://[***HIVE_METASTORE_HOST***]:9083",
    "iceberg.catalog.warehouse": "s3a://[***BUCKET_NAME***]/[***PATH***]/",
    "iceberg.catalog.client.region": "[***AWS_REGION***]",
    "iceberg.catalog.io-impl": "org.apache.iceberg.aws.s3.S3FileIO",
    "iceberg.catalog.s3.access-key-id": "[***AWS_ACCESS_KEY***]",
    "iceberg.catalog.s3.secret-access-key": "[***AWS_SECRET_KEY***]",
```

```
    "iceberg.hadoop.hive.metastore.use.SSL": "false",
    "iceberg.hadoop.hive.metastore.sasl.enabled": "true",
    "iceberg.hadoop.hive.metastore.kerberos.principal": "hive/_H
OST@[***REALM***]",
    "iceberg.hadoop.hive.metastore.schema.verification": "false",

    "iceberg.tables": "[***HIVE_DB***].[***HIVE_TABLE***]",
    "iceberg.tables.auto-create-enabled": "true",
    "iceberg.control.topic": "[***CONTROL_TOPIC_NAME***]",
    "iceberg.control.commit.interval-ms": "60000",
    "iceberg.control.commit.timeout-ms": "30000",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConver
ter",
    "value.converter.passthrough.enabled": "false",
    "value.converter.schema.registry.url": "https://[***SR_HOST***]:7790/ap
i/v1",
    "value.converter.schema.registry.client.ssl.protocol": "SSL",
    "value.converter.schema.registry.client.ssl.trustStorePath":
 "[***TRUSTSTORE_PATH***]",
    "value.converter.schema.registry.client.ssl.trustStorePassword":
 "[***TRUSTSTORE_PASSWORD***]",

    "consumer.override.sasl.mechanism": "GSSAPI",
    "consumer.override.sasl.jaas.config": "com.sun.security.auth.m
odule.Krb5LoginModule required useKeyTab=true storeKey=true keyTab=
\"[***KEYTAB_PATH***]\" principal=\"[***PRINCIPAL***]@[***REALM***]\" servi
ceName=\"kafka\";",
    "value.converter.sasl.jaas.config": "com.sun.security.auth.m
odule.Krb5LoginModule required useKeyTab=true storeKey=true keyTab=
\"[***KEYTAB_PATH***]\" principal=\"[***PRINCIPAL***]@[***REALM***]\" servi
ceName=\"schemaregistry\";"
  }
}
```

Kerberized cluster with Auto-TLS, Hive catalog, and Azure ADLS

Replace the bracketed placeholders with your values:

```
{
  "name": "[***CONNECTOR_NAME***]",
  "config": {
    "connector.class": "org.apache.iceberg.connect.IcebergSinkConnector",
    "tasks.max": "1",
    "topics": "[***KAFKA_TOPIC***]",

    "iceberg.catalog.type": "hive",
    "iceberg.catalog.uri": "thrift://[***HIVE_METASTORE_HOST***]:9083",
    "iceberg.catalog.warehouse": "abfss:
//[***AZURE_STORAGE_CONTAINER***]@[***AZURE_STORAGE_ACCOUNT_NAME***].dfs.cor
e.windows.net/[***WAREHOUSE_PREFIX***]/",
    "iceberg.catalog.io-impl": "org.apache.iceberg.azure.adlsv2.ADLSFileIO",
    "iceberg.hadoop-conf-dir": "/var/lib/kafka/iceberg-hadoop-conf",

    "iceberg.hadoop.hive.metastore.use.SSL": "false",
    "iceberg.hadoop.hive.metastore.sasl.enabled": "true",
    "iceberg.hadoop.hive.metastore.kerberos.principal": "hive/_HO
ST@[***REALM***]",
    "iceberg.hadoop.hadoop.security.authentication": "kerberos",
    "iceberg.tables": "[***HIVE_DB***].[***HIVE_TABLE***]",
    "iceberg.tables.auto-create-enabled": "true",
    "iceberg.control.topic": "[***CONTROL_TOPIC_NAME***]",
```

```
    "iceberg.control.group.id": "ctrlGroup_[***CONNECTOR_NAME***]",

    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConvert
er",
    "value.converter.passthrough.enabled": "false",
    "value.converter.schema.registry.url": "https://[***SR_HOST***]:7790/ap
i/v1",
    "value.converter.schema.registry.client.ssl.protocol": "SSL",
    "value.converter.schema.registry.client.ssl.trustStorePath":
 "[***TRUSTSTORE_PATH***]",
    "value.converter.schema.registry.client.ssl.trustStorePassword":
 "[***TRUSTSTORE_PASSWORD***]",

    "consumer.override.sasl.mechanism": "GSSAPI",
    "consumer.override.sasl.jaas.config": "com.sun.security.auth.m
odule.Krb5LoginModule required useKeyTab=true storeKey=true keyTab=
\"[***KEYTAB_PATH***]\" principal=\"[***PRINCIPAL***]@[***REALM***]\" servi
ceName=\"kafka\";",
    "value.converter.sasl.jaas.config": "com.sun.security.auth.m
odule.Krb5LoginModule required useKeyTab=true storeKey=true keyTab=
\"[***KEYTAB_PATH***]\" principal=\"[***PRINCIPAL***]@[***REALM***]\" servi
ceName=\"schemaregistry\";"
  }
}
```

Kerberized cluster with Auto-TLS, Hive catalog, and Ozone

Replace the bracketed placeholders with your values:

```
{
  "name": "[***CONNECTOR_NAME***]",
  "config": {
    "connector.class": "org.apache.iceberg.connect.IcebergSinkConnector",
    "tasks.max": "1",
    "topics": "[***KAFKA_TOPIC***]",

    "iceberg.catalog.type": "hive",
    "iceberg.catalog.uri": "thrift://[***HIVE_METASTORE_HOST***]:9083",
    "iceberg.catalog.warehouse": "of
s://[***OZONE_SERVICE_ID***]/[***VOLUME***]/[***BUCKET***]/",
    "iceberg.catalog.io-impl": "org.apache.iceberg.hadoop.HadoopFileIO",
    "iceberg.hadoop-conf-dir": "/var/lib/kafka/iceberg-hadoop-conf",

    "iceberg.hadoop.hive.metastore.use.SSL": "false",
    "iceberg.hadoop.hive.metastore.sasl.enabled": "true",
    "iceberg.hadoop.hive.metastore.kerberos.principal": "hive/_HO
ST@[***REALM***]",
    "iceberg.hadoop.hadoop.security.authentication": "kerberos",
    "iceberg.tables": "[***HIVE_DB***].[***HIVE_TABLE***]",
    "iceberg.tables.auto-create-enabled": "true",
    "iceberg.control.topic": "[***CONTROL_TOPIC_NAME***]",
    "iceberg.control.group.id": "ctrlGroup_[***CONNECTOR_NAME***]",

    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConvert
er",
    "value.converter.passthrough.enabled": "false",
    "value.converter.schema.registry.url": "https://[***SR_HOST***]:7790/ap
i/v1",
    "value.converter.schema.registry.client.ssl.protocol": "SSL",
    "value.converter.schema.registry.client.ssl.trustStorePath":
 "[***TRUSTSTORE_PATH***]",
```

```
    "value.converter.schema.registry.client.ssl.trustStorePassword":
 "[***TRUSTSTORE_PASSWORD***]",

    "consumer.override.sasl.mechanism": "GSSAPI",
    "consumer.override.sasl.jaas.config": "com.sun.security.auth.m
odule.Krb5LoginModule required useKeyTab=true storeKey=true keyTab=
\"[***KEYTAB_PATH***]\" principal=\"[***PRINCIPAL***]@[***REALM***]\" servi
ceName=\"kafka\";",
    "value.converter.sasl.jaas.config": "com.sun.security.auth.m
odule.Krb5LoginModule required useKeyTab=true storeKey=true keyTab=
\"[***KEYTAB_PATH***]\" principal=\"[***PRINCIPAL***]@[***REALM***]\" servi
ceName=\"schemaregistry\";"
  }
}
```

Dynamic table routing (fan-out)

You can enable dynamic table routing to write records to multiple Iceberg tables based on a field value in the record. Enable dynamic routing when each record carries the fully qualified destination table name (database.table) in a field.

Do not set the iceberg.tables property when iceberg.tables.dynamic-enabled is set to true. With iceberg.tables.auto-create-enabled set to true, missing tables, and namespaces if Ranger allows, are created on the first write. Pre-create the Hive database and grant Ranger access on [***HIVE_DB***].*.

The following Avro schema shows a field, target_table, that holds the target table. Records must set this field to a fully qualified table name:

```
{
  "type": "record",
  "name": "UserRouted",
  "namespace": "example",
  "fields": [
    {"name": "id", "type": "long"},
    {"name": "name", "type": "string"},
    {"name": "amount", "type": "double"},
    {"name": "target_table", "type": "string"}
  ]
}
```

In addition to the properties described in *Kerberized cluster with Auto-TLS, Hive catalog, and Amazon S3 (single table)*, set the following properties on the connector:

```
"iceberg.tables.dynamic-enabled": "true"
"iceberg.tables.route-field": "target_table"
"iceberg.tables.auto-create-enabled": "true"
"iceberg.control.topic": "[***CONTROL_TOPIC_NAME***]"
```

HTTP Source connector

The HTTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The HTTP Source connector listens on a port for HTTP POST requests and transfers the request body to a Kafka topic. The Kafka topic this connector transfers messages to is determined by the value of the topics property. The connector does not perform record processing, the messages are transferred to Kafka as they are received in the HTTP request. The connector uses HTTPS for secure communication. Optionally, client certificate authentication (mutual TLS) can also be configured.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Source property reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the HTTP Source properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- The HTTP Source connector must use at least one-way SSL, it cannot be used without SSL.
- The ListenHTTP processor in the NiFi flow keeps running when the connector is paused. Due to this, the connector accepts requests, but ultimately times out and sends an error response to the client.
- The tasks.max property must be 1 to avoid port conflict due to executing multiple tasks on the same Kafka Connect node. Cloudera recommends that you use a load balancer to keep track of the actual node which runs the connector.

Configuration example

In this example, the connector accepts client requests sent to the connector’s URL. The connector's URL is based on the configuration of multiple properties. For example assume the following:

- The connector is running on kc-host-1.
- Listening Port is set to port 8080.
- Base Path is set to data-ingest.

In a case like this, the connector's URL will be the following:

```
https://kc-host-1:8080/data-ingest
```



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case and therefore can be removed from the configuration.

```
{
```

```
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "HTTP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.HTTP Source Parameters:Listening Port": "[***PORT***]",
 "parameter.HTTP Source Parameters:Base Path": "[***BASE PATH***]",
 "parameter.HTTP Source Parameters:Keystore Filename": "[***PATH TO
 KEYSTORE***]",
 "parameter.HTTP Source Parameters:Keystore Key Password": "[***KEYSTORE KEY
 PASSWORD***]",
 "parameter.HTTP Source Parameters:Keystore Password": "[***KEYSTORE
 PASSWORD***]",
 "parameter.HTTP Source Parameters:Keystore Type": "[***KEYSTORE TYPE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Listening Port

The port that the client needs to use in the URL.

Base Path

The path that the client needs to use in the URL.

Keystore *

The properties for accessing the keystore used for HTTPS.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose

name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not

specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

HTTP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the HTTP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the

accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source property reference.

Authorized Issuer DN Pattern

Description

A regular expression to apply against the Issuer's Distinguished Name of incoming connections. You can use this property when the Client Authentication property is set to REQUIRED (mutual TLS).

Default Value

.*

Accepted Values

Required

false

Authorized Subject DN Patter

Description

A regular expression to apply against the Subject's Distinguished Name of incoming connections. You can use this property when the Client Authentication property is set to REQUIRED (mutual TLS).

Default Value

.*

Accepted Values

Required

false

Base Path

Description

The base path or context path of the URL.

Default Value

contentListener

Accepted Values

Required

true

Client Authentication

Description

The client authentication policy used for HTTPS.

Default Value

NONE

Accepted Values

NONE, REQUIRED

Required

true

Keystore Filename

Description

The fully-qualified filename of a keystore.

This keystore is used to establish a secure connection between this connector and its clients using HTTPS.

Default Value

Accepted Values

Required

true

Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename property.

Default Value

Accepted Values

Required

true

Keystore Password

Description

The password used to access the contents of the keystore configured in the Keystore Filename property.

Default Value

Accepted Values

Required

true

Keystore Type

Description

The type of the keystore configured in the Keystore Filename property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Listening Port

Description

The port to listen on for communication.

Default Value

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore.

When using one-way SSL (Client Authentication is set to NONE), this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

When using one-way SSL (Client Authentication is set to NONE), this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

When using one-way SSL (Client Authentication is set to NONE), this property must be completely removed from the configuration.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interva l.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

JDBC Source connector

The JDBC Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The JDBC Source connector reads records from a database table and transfers each record to Kafka in Avro or JSON format. The Kafka topic this connector transfers messages to is determined by the value of the topics parameter in the configuration. The schema of the records can be inherited from the schema of the database table, or it can be a predefined schema retrieved from Schema Registry.

The connector supports incremental loading based on a table column containing increasing values (typically an ID sequence or timestamp). When the connector is started for the first time with incremental loading turned on, it can load all the existing data from the source table, or it can start from the current maximum value of the incrementing column.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Source property reference.

Connector/dataflow-specific properties

These are the properties that are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the JDBC Source properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- The tasks.max property must be set to 1. Setting the property to any other value will not have an effect. This is because this connector performs incremental loading and must run as a single task.
- If Initial Load Strategy is set to Start at Beginning and “Maximum-value Columns is not set, the connector tries to load the entire content of the source database table to the target Kafka topic with every execution of the connector. Set Maximum-value Columns to avoid this.

Configuration example

In this example, the connector connects to a PostgreSQL database using username/password authentication, queries the records from a table, converts each record to JSON format, and sends them to a Kafka topic. The schema of the records is determined from the schema of the database table. The data loading is incremental, only new records are loaded based on a column containing incrementing values in the table.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "JDBC Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "key.attribute": "kafka.message.key",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.JDBC Source Parameters:Database Connection URL": "[***JDBC
 URL***]",
 "parameter.JDBC Source Parameters:Database Driver Location": "[***PATH TO
 JDBC DRIVER***]",
 "parameter.JDBC Source Parameters:Database Driver Class Name": "org.post
gresql.Driver",
 "parameter.JDBC Source Parameters:Database Type": "PostgreSQL",
 "parameter.JDBC Source Parameters:Database User Name": "[***USERNAME***]",
 "parameter.JDBC Source Parameters:Database User Password":
 "[***PASSWORD***]",
 "parameter.JDBC Source Parameters:Database Table Name": "[***TABLE
 NAME***]",
 "parameter.JDBC Source Parameters:Kafka Message Key Column": "[***COLUMN
 NAME***]",
 "parameter.JDBC Source Parameters:Maximum-value Columns": "[***COLUMN
 NAME***]",
 "parameter.JDBC Source Parameters:Kafka Message Data Format": "JSON",
 "parameter.JDBC Source Parameters:Schema Access Strategy": "Inherit Schema"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Database Connection URL

The JDBC URL of the PostgreSQL database. For example, jdbc:postgresql://myhost:5432/mydb.

Database Driver Location

A comma-separated list of files or folders containing the JDBC client libraries.

Database Driver Class Name

The Java class name of the PostgreSQL Driver implementation.

Database Type

The type of the database. Because this is a PostgreSQL example, the property is set to PostgreSQL.

Database User Name

The username used for authenticating to the database.

Database User Password

The password of the user.

Database Table Name

The name of the database table to query data from.

Kafka Message Key Column

Specifies a database table column. The value of the column specified is used as the key of the Kafka message.

Maximum-value Columns

The name of the column used for incremental loading. The column value must be increasing, for example, coming from a sequence or system timestamp.

Kafka Message Data Format

The format of the messages the connector sends to Kafka.

Schema Access Strategy

Specifies the strategy used for determining the schema of the database record. In this example, this property is set to Inherit Schema, meaning that the schema is determined (inherited) from the database table.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

JDBC Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the JDBC Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source property reference.

Additional WHERE clause

Description

Specifies a custom condition to add in the WHERE clause of the SQL query.

Default Value

Accepted Values

Required

false

Columns to Return

Description

A comma-separated list of column names to be returned by the query. All columns are returned if it is not specified.

Default Value

Accepted Values

Required

false

Custom Query

Description

Specifies a custom SQL query to use instead of generating the query from the table and column names.

Default Value

Accepted Values

Required

false

Database Connection URL

Description

The database specific connection URL used for connecting to the database. For example, jdbc:postgresql://localhost:5432/postgres.

Default Value

Accepted Values

Required

true

Database Driver Class Name

Description

The database driver class name For example, org.postgresql.Driver.

Default Value

Accepted Values

Required

true

Database Driver Location

Description

A comma-separated list of files or folders containing the JDBC client libraries.

Default Value

Accepted Values

Required

true

Database Table Name

Description

The name of the database table to query.

Default Value

Accepted Values

Required

true

Database Type

Description

The database type used for generating database specific SQL queries.

Default Value

Generic

Accepted Values

Generic, Oracle, Oracle 12+, MS SQL 2008, MS SQL 2012+, MySQL, PostgreSQL

Required

true

Database User Name

Description

The database user name. If username/password authentication is not required by the database server, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Database User Password

Description

The database user password. If username/password authentication is not required by the database server, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Date Format

Description

Specifies the format to use when writing Date fields to JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Initial Load Strategy

Description

Specifies how existing rows in the database table are handled when the connector is started for the first time.

Default Value

Start at Beginning

Accepted Values

Start at Beginning, Start at Current Maximum Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector sends to Kafka. The database row is converted to this format. If the output format is Avro and Schema Access Strategy is set to Inherit Schema, the schema is embedded in the output message. If the output format is Avro and Schema Access Strategy is set to Schema Registry, the schema is not embedded in the output message.

Default Value

Avro

Accepted Values

Avro, JSON

Required

true

Kafka Message Key Column

Description

Specifies a database table column. The value of the column specified is used as the key of the Kafka message.

Default Value

Accepted Values

Required

false

Kerberos Keytab

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum-value Columns

Description

A comma-separated list of column names used for incremental loading.

Default Value

Accepted Values

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the database record.

- If set to Inherit Schema, the schema is determined from the database schema.
- If set to Schema Registry, the schema is read from Schema Registry.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Inherit Schema

Accepted Values

Inherit Schema, Schema Registry, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when writing Time fields to JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when writing Timestamp fields to JSON.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

tue

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interva l.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

JMS Source connector

The JMS Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The JMS Source Connector consumes messages from a JMS broker and transfers the message body to Kafka. The Kafka topic this connector transfers messages to is determined by the value of the topics parameter in the configuration. The JMS Source connector does not perform record processing, the messages are transferred to Kafka as they are consumed from the JMS broker.

TLS can be used to establish a secure connection between the connector and the JMS broker. The keystore and truststore files necessary for securing the connection must be present on the cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Source property reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the JMS Source properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.

Configuration example

In this example, the connector connects to an ActiveMQ Message Broker using a secured connection and username/password authentication.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "JMS Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter"
,,
 "tasks.max": "1"
```

```
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",,
 "topics": "[***TOPIC NAME***]"
 "parameter.JMS Source Parameters:JMS Broker URI": "[***BROKER URI***]",
 "parameter.JMS Source Parameters:JMS Client Libraries": "[***PATH TO CLIENT
 LIBRARIES***]",
 "parameter.JMS Source Parameters:JMS Connection Factory Class Name": "or
g.apache.activemq.ActiveMQSslConnectionFactory",
 "parameter.JMS Source Parameters:JMS Destination Name": "[***QUEUE
 NAME***]",
 "parameter.JMS Source Parameters:JMS Destination Type": "QUEUE",
 "parameter.JMS Source Parameters:JMS User Name": "[***USERNAME***]",
 "parameter.JMS Source Parameters:JMS User Password": "[***PASSWORD***]",
 "parameter.JMS Source Parameters:Truststore Filename": "[***PATH TO
 TRUSTSTORE***]",
 "parameter.JMS Source Parameters:Truststore Password": "[***TRUSTSTORE
 PASSWORD***]",
 "parameter.JMS Source Parameters:Truststore Type": "[***TRUSTSTORE
 TYPE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

JMS Broker URI

The URI of the ActiveMQ Message Broker. For example, ssl://activemq-hostname:61617.

JMS Client Libraries

The path to the directory containing ActiveMQ client libraries.

JMS Connection Factory Class Name

The Java class name of the ActiveMQ ConnectionFactory implementation.

JMS Destination Name

The name of the destination (in this case a queue) on the ActiveMQ Message Broker that the messages are received from.

JMS Destination Type

The type of the destination. In this example, the type is QUEUE.

JMS User Name

The username used for authenticating to the ActiveMQ Message Broker.

JMS User Password

The password of the user.

Truststore *

These are the properties for accessing the truststore containing the ActiveMQ Message Broker's certificate used for secure communication.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging

Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to / tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interva l.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

JMS Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the JMS Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source property reference.

JMS Broker URI

Description

The URI to use to connect to the JMS Message Broker. For example, ssl://myhost:61617

Default Value

Accepted Values

Required

true

JMS Client Libraries

Description

The path to the directory containing the JMS client libraries.

Default Value

Accepted Values

Required

true

JMS Connection Factory Class Name

Description

The fully qualified name of the JMS ConnectionFactory implementation class. For example, org.apache.activemq.ActiveMQSslConnectionFactory.

Default Value

Accepted Values

Required

true

JMS Destination Name

Description

The name of the JMS Destination.

Default Value

Accepted Values

Required

true

JMS Destination Type

Description

The type of the JMS Destination.

Default Value

QUEUE

Accepted Values

QUEUE, TOPIC

Required

true

JMS User Name

Description

The username used for authentication.

If username/password authentication is not required by the JMS Message Broker, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

JMS User Password

Description

The password used for authentication.

If username/password authentication is not required by the JMS Message Broker, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Filename

Description

The fully-qualified filename of a keystore. This keystore is used for mutual TLS towards the JMS Message Broker.

If the JMS Message Broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename property.

If the JMS Message Broker broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Password

Description

The password used to access the contents keystore configured in the Keystore Filename property.

If the JMS Message Broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Type

Description

The type of the keystore configured in the Keystore Filename property.

If the JMS Message Broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with the JMS server using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

MQTT Source connector

The MQTT Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The MQTT Source connector consumes messages from an MQTT broker and transfers them to Kafka. The Kafka topic this connector transfers messages to is determined by the value of the topics property in the configuration. This connector does not perform record processing. The messages are transferred to Kafka as they are consumed from the MQTT broker. TLS can be used to establish a secure connection between the connector and the MQTT broker. The keystore and truststore files necessary for securing the connection must be present on the cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Source property reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the MQTT Source properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- If the keystore-related properties are removed and an empty truststore is provided, the connector does not use TLS for connecting to the MQTT broker. TLS is used if a truststore is provided that has a key in it.
- When the connector is deployed in a cluster environment, the system might move it from one cluster node to another. During this process, the connector is stopped, deployed to the target node, and started there. The time between the connector stopping and starting again might be several minutes. To avoid losing the messages that arrived to the MQTT broker during this time, set the following properties:
  - Set MQTT Clean Session to false.
  - Specify the MQTT Client ID.
  - Set MQTT Quality of Service to 1 or 2.

Configuration example

In this example, the connector fetches messages from an MQTT broker and transfers them to a Kafka topic.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "MQTT Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.MQTT Source Parameters:MQTT Broker URI": "tcp
://[***HOST***]:[***PORT***]",
 "parameter.MQTT Source Parameters:MQTT Quality of Service": "0",
```

```
 "parameter.MQTT Source Parameters:MQTT Topics": "[***MQTT TOPIC NAME***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case:

topics

The name of the Kafka topic that the connector sends messages to.

MQTT Broker URI

The URI of the MQTT broker. In this example, the connection is not secure. As a result, the URI starts with tcp://. The port number you typically use in a case like this is 1883. If TLS is used, the URI should start with ssl://, and typically you use port number 88883.

MQTT Topics

The comma-separated list of MQTT topics that the connector fetches messages from.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to / tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

MQTT Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the MQTT Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source property reference.

Keystore Filename

Description

The fully-qualified filename of a keystore. This keystore is used for mutual TLS towards the MQTT server.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename property.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Password

Description

The password used to access the contents keystore configured in the Keystore Filename property.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Type

Description

The type of the keystore configured in the Keystore Filename property.

If the MQTT broker does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

MQTT Broker URI

Description

The URI to use to connect to the MQTT broker.

Example URI if SSL is not used (keystore-related parameters are removed and and the default truststore is used): tcp://localhost:1883

Example URI if SSL is used: ssl://localhost:8883

Default Value

tcp://localhost:1883

Accepted Values

Required

true

MQTT Password

Description

Password to use when connecting to the MQTT broker.

If username-password authentication is not required by the MQTT broker, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

MQTT Quality of Service

Description

The Quality of Service (QoS) to receive the message with.

0 - at most once

1 - at least once

2 - exactly once

Default Value

0

Accepted Values

0,1 or 2

Required

true

MQTT Topics

Description

Specifies the MQTT topic to subscribe to. Use an MQTT wildcard to subscribe to multiple topics simultaneously.

Default Value

Accepted Values

Required

true

MQTT Username

Description

Username to use when connecting to the MQTT broker.

If username-password authentication is not required by the MQTT broker, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with the MQTT server using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

MQTT Client ID

Description

The MQTT client ID to use.

If it is not set, a UUID is generated. In such a case, this property must be removed from the configuration.

Default Value

Accepted Values

Required

false

MQTT Clean Session

Description

Specifies whether to start a new session.

If it is set to true, the client and server discard any previous sessions and start a new one. The new session lasts as long as the network connection. State data associated with a session is not reused in any subsequent sessions.

If it is set to false, the server resumes communications with the client based on the state from the current session (identified by the Client ID). The client and server store the session after being disconnected. When a session that is not a clean session is disconnected, the server stores

further QoS 1 and QoS 2 messages that match any subscriptions that the client has at the time of disconnection. These messages are stored as part of the session state.

Default Value

true

Accepted Values

true, false

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

SFTP Source connector

The SFTP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The SFTP Source connector obtains files from an SFTP server and transfers them to Kafka. The SFTP Source connector supports record processing. In case record processing is enabled, input file is split into records and the records are transferred to Kafka. If record processing is not enabled, the connector forwards files to Kafka as they are fetched from the SFTP server.

If record processing is enabled, the input file can contain records in either JSON, CSV or Grok format. If the input records are in JSON or CSV format, the connector can either infer the schema of the records based on the data or read the schema from Schema Registry. If the input records are in Grok format, the connector can either derive the schema using the field names from the value of the Grok Expression property or read the schema from Schema Registry.

In addition, if record processing is enabled, the connector writes the records to Kafka in Avro format. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on. The Kerberos keytab used to access Schema Registry must also be on the same cluster node that the connector runs on. The connection to Schema Registry can be secured by TLS. The truststore file necessary for securing the connection must also be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Source property reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the SFTP Source properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- The Schema Registry URL property is mandatory even if Schema Registry is not used. If Schema Registry is not used, use the default value, or completely remove the property from the configuration.
- Schema Branch and Schema Version can not be specified at the same time.
- The value of the Schema Access Strategy property is not independent of the value of the Input Data Format property. As a result, you must exercise caution when configuring Schema Access Strategy.
  - If the value of Input Data Format is JSON, the possible values for Schema Access Strategy are Schema Registry or Infer Schema.
  - If the value of Input Data Format is CSV, the possible values for Schema Access Strategy are Schema Registry or Infer Schema.
  - If the value of Input Data Format is GROK, the possible values for Schema Access Strategy are Schema Registry or Field Names From Grok Expression.
- If the Enable Record Processing property is set to false (record processing disabled), the entire input file is transferred to Kafka as one message.
- If the Enable Record Processing property is set to true (record processing enabled), the output is always in Avro format. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

Configuration example

In this example, the connector fetches files from an SFTP server that only requires Basic Authentication (username and password) and transfers these files to a Kafka topic.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "SFTP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
```

```
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[*** KAFKA TOPIC NAME***]",
 "parameter.SFTP Source Parameters:Hostname": "[***SFTP SERVER
 HOSTNAME***]",
 "parameter.SFTP Source Parameters:Port": "22",
 "parameter.SFTP Source Parameters:Username": "[***SFTP USERNAME***]",
 "parameter.SFTP Source Parameters:Password": "[***SFTP PASSWORD***]",
 "parameter.SFTP Source Parameters:Remote Path": "[***FOLDER PATH***]",
 "parameter.SFTP Source Parameters:File Filter Regex": ".*",
 "parameter.SFTP Source Parameters:Enable Record Processing": "false"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Hostname

The hostname of the remote system where the SFTP server runs. For example, my.sftp-server.com.

Port

The port that the remote system is listening on for file transfers.

Username

The username for connecting to the SFTP server.

Password

The password used to authenticate the user towards the SFTP server. In this example, the SFTP server requires Basic Authentication.

Remote Path

The path on the remote system that points to the directory from which to pull files. For example, / uploads.

File Filter Regex

The Java regular expression to use for filtering filenames. Only files whose names match the regular expression are fetched.

Enable Record Processing

Determines whether the contents of a file get parsed as records before sending them to Kafka. In this example, the property is set to false, meaning that the entire file gets forwarded to Kafka as one message.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi

c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

SFTP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the SFTP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source property reference.

Avro Schema Write Strategy

Description

Specifies whether the record schema is written to the output Avro file.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

CSV Character Set

Description

The character set used to read the input CSV files.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

UTF-8

Accepted Values

Required

true

CSV Escape Character

Description

The escape character used in the input CSV files to escape other special characters.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

\

Accepted Values

Required

true

CSV Quote Character

Description

The quote character used in the input CSV files.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

"

Accepted Values

Required

true

CSV Record Separator

Description

The record separator used in the input CSV files.

This property is ignored if the input is not a CSV file or if record processing is not enabled.

Default Value

\n

Accepted Values

Required

true

CSV Treat First Line as Header

Description

Specifies whether the first line in the input file is handled as a header.

Ignored if the input is not a CSV file or record processing is not enabled.

Default Value

false

Accepted Values

true, false

Required

true

CSV Trim Fields

Description

Specifies whether whitespace characters are removed from the beginning and the end of fields.

Ignored if the input is not a CSV file or record processing is not enabled.

Default Value

true

Accepted Values

true, false

Required

true

CSV Value Separator

Description

The value separator used in the input CSV files.

Ignored if the input is not a CSV file or record processing is not enabled.

Default Value

,

Accepted Values

Required

true

Completion Strategy

Description

Specifies what to do with the original file on the server once it has been fetched.

If the Completion Strategy fails, a warning is logged but the data is still transferred.

Default Value

None

Accepted Values

None, Move File, Delete File

Required

true

Date Format

Description

Specifies the format used for parsing date fields in the input data.

This property is only used if Input Data Format is set to CSV or JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Enable Record Processing

Description

Enables or disables record processing.

If set to true, the Input Data Format is considered and the file gets parsed into records. In this case the Records Per Kafka Message property defines how many records are written into one Kafka message.

If set to false, the entire file gets forwarded to Kafka as one message.

Default Value

true

Accepted Values

true, false

Required

true

File Filter Regex

Description

The Java regular expression to use for filtering filenames. Only files whose names match the regular expression are fetched.

Default Value

.*

Accepted Values

Required

true

Follow Symlink

Description

If set to true, both symbolic files and nested symbolic subdirectories are pulled. Otherwise, symbolic files are not read and symbolic link subdirectories are not traversed.

Default Value

false

Accepted Values

true, false

Required

true

Grok Expression

Description

Specifies the format of a line in Grok format. This allows the connector to understand how to parse each line in the input file. If a line in the file does not match this pattern, the line is handled according to what is set in the Grok No Match Behavior property.

A valid Grok expression must be specified using this property even if Grok format is not used.

Default Value

%{GREEDYDATA:message}

Accepted Values

Required

true

Grok No Match Behavior

Description

Specifies how to handle lines that do not match the pattern set in the Grok Expression property.

If set to append-to-previous-message, non-matching lines are appended to the last field of the previous message.

If set to skip-line, non-matching lines are skipped.

If set to raw-line, non-matching lines are only added to the _raw field.

Default Value

append-to-previous-message

Accepted Values

append-to-previous-message, skip-line, raw-line

Required

true

Host Key File

Description

The fully-qualified filename of the host key file.

If supplied, this file is used as the host key.

If a host key is not supplied, but Strict Host Key Checking is set to true, the known_hosts and known_hosts2 files from the ~/.ssh directory are used.

If a host key is not supplied and Strict Host Key Checking is set to false, no host key file is used.

This parameter must either contain the fully-qualified name of a file, or be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Hostname

Description

The hostname or IP address of the remote system.

Default Value

localhost

Accepted Values

Required

true

Ignored Dotted Files

Description

Specifies whether to ignore files whose names begin with a dot (".").

Default Value

true

Accepted Values

true, false

Required

true

Input Data Format

Description

The format in which the input file contains record-oriented data.

If Enable Record Processing is set to false, this setting is ignored.

Default Value

JSON

Accepted Values

JSON, CSV, GROK

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Move Destination Directory

Description

The fully-qualified name of the directory on the remote server to move the original file to once it is ingested. This property is ignored unless the Completion Strategy property is set to Move File. The specified directory must already exist on the remote system.

This parameter must either contain the fully-qualified name of a directory, or be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Password

Description

The password to use when connecting to the SFTP server.

If the server does not require a password, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Path Filter Regex

Description

The Java Regular Expression to use for filtering paths.

If Search Recursively is set to true, only subdirectories whose path matches the given regular expression are scanned.

If Search Recursively is set to false, this property is ignored.

Default Value

.*

Accepted Values

Required

true

Port

Description

The port that the remote system is listening on for file transfers.

Default Value

22

Accepted Values

Required

true

Private Key File

Description

The fully-qualified filename of a private key file.

If no private key is used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Private Key Password

Description

The password used to access the private key.

If no private key is used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Record Per Kafka Message

Description

Specifies how many records to write into each Kafka message.

If Enable Record Processing is set to false, this setting is ignored.

Default Value

1

Accepted Values

Required

true

Remote Path

Description

The path on the remote system from which to pull files.

Default Value

.

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the input records if the Enable Record Pr ocessing property is set to true .

The value you set here depends on the input data format.

If set to Schema Registry, the schema is read from Schema Registry.

This setting works with all input data formats.

If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if your input data format is either JSON or CSV.

If set to Field Names From Grok Expression, the schema is determined using the field names in the Grok Expression property. This setting can only be used if your input data format is GROK.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Field Names From Grok Expression

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.

If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server.

If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

true

Search Recursively

Description

Specifies whether to pull files from arbitrarily nested subdirectories. Subdirectories are not traversed if set to false.

Default Value

false

Accepted Values

true, false

Required

true

Strict Host Key Checking

Description

Specifies whether strict enforcement of host keys is applied.

Default Value

false

Accepted Values

true, false

Required

true

Time Format

Description

Specifies the format used for parsing time fields in the input data. This property is only used if Input Data Format is set to CSV or JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format used for parsing timestamp fields in the input data. This property is only used if Input Data Format is set to CSV or JSON.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Username

Description

Username for connecting to the SFTP server.

Default Value

Accepted Values

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

Stateless NiFi Source and Sink

The Stateless NiFi Source and Sink connectors allow you to run NiFi dataflows within Kafka Connect. Using these connectors can grant you access to a number of NiFi features without having the need to deploy or maintain NiFi on your cluster. Learn more about what Stateless NiFi is and how it shipped in Cloudera.

Apache NiFi is a powerful tool for authoring and running dataflows. It provides many capabilities that are necessary for large-scale enterprise deployments, such as data persistence and resilience, data lineage and traceability, as well as multi-tenancy. This, however, requires an administrator who ensures that this process is running and operational. Additionally, in most cases, adding more capabilities to your deployments results in more complexity.

There are times, however, when users do not need all the power of NiFi, and running it in a much simpler form factor is sufficient. A common use case is to use NiFi to pull data from many different sources, perform manipulations (for example, convert JSON to Avro), filter some records, and then publish the data to Apache Kafka. Another common use case is to pull data from Apache Kafka, perform manipulations and filtering, and then publish the data elsewhere.

For deployments where NiFi acts only as a bridge into and out of Kafka, it can be simpler to operationalize such a deployment by running the dataflow within Kafka Connect. The Stateless NiFi Source and Sink connectors allow users to do just that.

Stateless NiFi

Dataflows within Kafka Connect run using the Stateless NiFi dataflow engine. Stateless NiFi differs from the traditional NiFi engine in the following ways:

- Stateless NiFi is an engine that is designed to be embedded. This makes it convenient to run it within the Kafka Connect framework.
- Stateless NiFi does not provide a user interface (UI) or a REST API. Additionally, it does not support modifying the dataflow while it is running.
- Stateless NiFi does not persist flowfile content to disk. Instead, it holds the content in memory.
- Stateless NiFi does not use data prioritizers. Instead, it operates on data in a first-in-first-out order. Dataflows built for Stateless NiFi must have a single source and a single destination. The only exception to this is when the data is routed to exactly one of multiple destinations, such as a failure destination or a success destination.
- Stateless NiFi does not currently provide access to data lineage/provenance.
- Stateless NiFi does not support cyclic graphs. While it is common and desirable in traditional NiFi to have a failure relationship from a processor route back to the same processor, this can result in a `StackOverflowException` in Stateless NiFi. The preferred approach in Stateless NiFi is to create an output port for failures and route the data to that output port.

Stateless NiFi in Cloudera

Stateless NiFi in Cloudera comes in two flavors. You have the base Stateless NiFi Source (`StatelessNiFiSource`) and Stateless NiFi Sink (`StatelessNiFiSink`) connectors. In addition, there are a number of ready-to-use connectors based on Stateless NiFi Source and Sink.

Stateless NiFi Source and Sink connectors

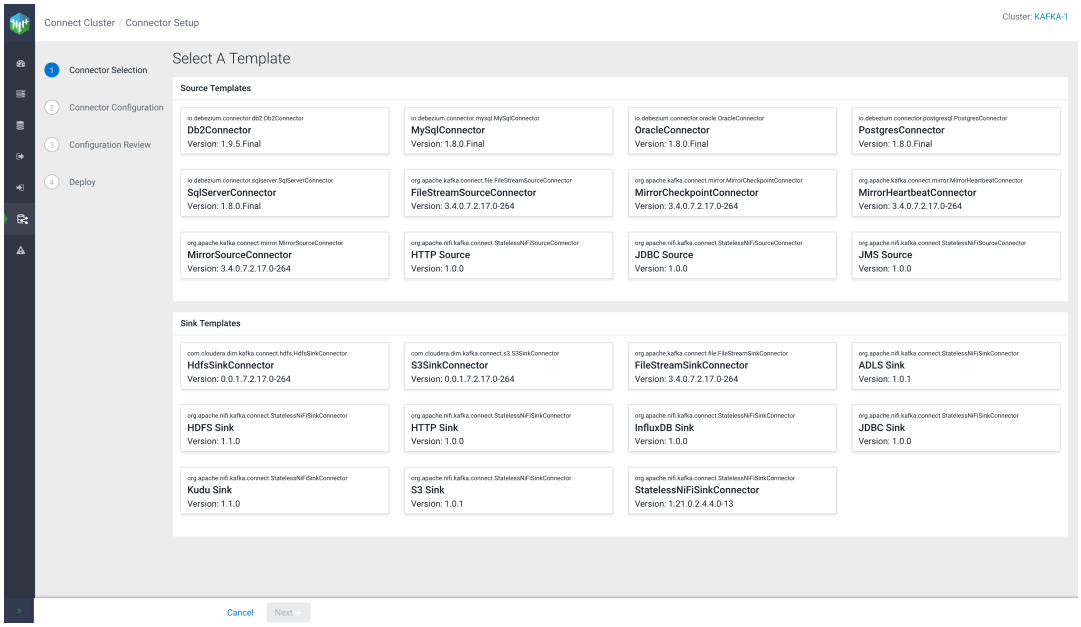
The Stateless NiFi Source and Sink connectors are generic connectors that act as the base for Stateless NiFi in Kafka Connect. The idea behind these connectors is that you develop your own dataflow in NiFi, and deploy that dataflow as a Kafka Connect connector using either Stateless Nifi Source or Sink connectors. This way, you can create virtually any connector without having to write a single line of code. All of the heavy lifting is done on the NiFi UI.

Predefined dataflows and Stateless NiFi-based connectors

In addition to the base Stateless NiFi Source and Sink connectors, there are a number of readily-available connectors that are based on Stateless NiFi. Specifically, Cloudera ships a set of predefined dataflows. These dataflows are developed and maintained by Cloudera. Each dataflow covers a common data movement use case. These predefined flows are presented as standalone connectors on the Streams Messaging Manager UI. Using these dataflows/connectors, you can immediately start using Stateless NiFi without having to develop your own dataflow.

You can find a full list of the Stateless NiFi-based connectors in Connectors on page 42. In addition, you can also use the Streams Messaging Manager UI to check which of the connectors are Stateless NiFi-based. When adding a new connector using Streams Messaging Manager, you are presented with a number of cards that represents the connectors available for deployment. Each card includes the following information about the connector:

- The connector's fully qualified class name.
- The connector's display name. If no display name is available, the card includes the unqualified classname.
- The version of the connector.



Notice how the fully qualified class name for many of the default available connectors is either `org.apache.nifi.kafka.connect.StatelessNiFiSourceConnector` or `org.apache.nifi.kafka.connect.StatelessNiFiSinkConnector`. Even though these are standalone connectors and have a unique display name, they are all Stateless NiFi connectors. Each of them runs a predefined flow designed for a specific use case. For example, JDBC Source runs a dataflow that reads records from a database table and transfers each record to Kafka.

Note: Unlike the base Stateless NiFi connectors, connectors running predefined flows **do not** accept a custom dataflow with flow.snapshot. The dataflow they use is specified by the meta.smm.predefined.flow.name and meta.smm.predefined.flow .version properties. While configuring these properties is possible, Cloudera does not recommend that you do so.

Dataflow development best practices for Stateless NiFi

By leveraging Stateless NiFi, you can build your own dataflows and use those as Kafka Connect connectors. However, there are some principles to keep in mind when building such a dataflow in NiFi.

General recommendations and criteria

Learn about the general recommendations and criteria that you need to follow when building a dataflow that you will run as a Kafka Connect connector.

- If your dataflow depends on external resources (like a JDBC driver), the resource must be made available on all Kafka Connect role instance nodes (Connect workers). Additionally, the resource must be in the same location on all nodes with proper file permissions. Cloudera also recommends parameterizing the file path so that you can specify it when deploying the connector.
- The dataflow should be designed in its dedicated process group and should have a parameter context assigned to its process group. This way, you have the ability to parameterize your connector, which enables you to provide custom property values when deploying the connector using Streams Messaging Manager.
- A source connector flow should always have a single source. This means that the flow should have only one processor with no incoming connections.
- A sink connector flow should always have a single destination. If you want to transfer the data consumed from the Kafka topic to two different destinations at the same time, it is preferred to have two distinct sink connectors with different consumer groups. If using a single sink connector, you must take into consideration the possibility of creating duplicates. If one destination is working as expected but not the other, the Kafka message is not acknowledged and consumed again at a later time while the data was still successfully sent to one destination.

- If you require specific connectors to flush data to the target system at specific intervals, configure the cloudera.offset.flush.interval.ms connector property. This property overrides the role-level Offset Flush Interval (offset.flush.interval.ms) on a per-connector basis.

Dataflow execution and scheduling

Learn how Stateless NiFi Source and Sink connectors handle dataflow execution and scheduling.

Stateless NiFi does not consider the scheduling settings of processors. When a Stateless NiFi connector is deployed, Kafka Connect starts an instance of the Stateless NiFi engine that executes the dataflow. The following sections discuss in detail how the dataflow is executed by both source and sink connectors

Source connectors

A Kafka Connect source connector obtains data from an external (source) data store and delivers the data to a Kafka service. The architecture of a source connector is as follows.

Figure 3: Source Connector Architecture

When a Stateless NiFi Source connector instance is deployed, a number of tasks are started for the connector instance. The number of tasks is determined by the tasks.max property, which can be set during connector deployment. Each task consists of the following:

- `SourceTask`

  In Stateless NiFi connectors, the `SourceTask` is responsible for executing one instance of the connector's dataflow.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.

- Kafka Producer

  The Kafka producer is responsible for delivering messages to the configured Kafka topics.

With the Stateless NiFi Source connector, the `SourceTask` is the component executing an instance of the dataflow specified in the connector's flow.snapshot property. Any source connector flow you build should not attempt to deliver directly to Kafka itself using a processor such as PublishKafka. Instead, the data should be routed to a NiFi output port in the dataflow. Any flowfile delivered to that output port is obtained by the connector and delivered to Kafka.

The dataflow is triggered continuously in one thread within the `SourceTask`. This means that in one iteration, the entire flow is executed (one processor after another), and the data it fetches from the source data store in that execution ends up on the flow's output port in the form of flowfiles. These flowfiles are forwarded by the Kafka Connect framework to the task's producer as messages (one flowfile equals one message). At this point the flow's execution is finished, and the flow is triggered again.

The producer is a regular Kafka producer that is responsible for delivering the messages to the Kafka topics. The producer is running on a separate thread. This means that while the dataflow is running, the producer can work on delivering the messages it received at the end of the flow's previous execution.

Each flowfile is delivered as a single Kafka message. Therefore, if a flowfile contains thousands of JSON records, totaling 5 MB for example, it is important to split those flowfiles into individual records using a SplitRecord processor before transferring the data to the output port. Otherwise, this results in a single Kafka message that is 5 MB while the default message size for Kafka is 1 MB.

Scheduling settings of the processors in the dataflow are ignored. The dataflow is executed in one thread from the first processor through the output port. Afterward, the flow is retriggered. One execution of the dataflow takes as long as necessary to fetch and process the data that accumulated in the source data store since the previous execution. However, the execution has a default timeout of one minute. This timeout can be configured on a connector level with the dataflow.timeout property.

Transactional sources

Unlike with traditional NiFi, Stateless NiFi keeps the contents of flowfiles in memory. As long as the source of the data is replayable and transactional, there is no concern over data loss. This is handled by treating the entire dataflow as a single transaction. Once data is obtained from a source component, the data is then transferred to the next processor in the flow. At this point, in traditional NiFi, the processor would acknowledge the data and NiFi would take ownership of that data, having persisted it to disk.

With Stateless NiFi, however, the data is not yet acknowledged. Instead, data is transferred from one processor to the next with each processor performing and completing its task. This is repeated until all data is queued up at the output port. At this point, the NiFi connector provides the data to Kafka. Only after Kafka has acknowledged the records does the connector acknowledge this to NiFi. Only at that point is the session committed, allowing the source processor to acknowledge receipt of the data.

As a result, if NiFi is restarted while processing some piece of data, the source processor will not have acknowledged the data and is able to replay the data, resulting in no data loss.

Initial data load

Stateless NiFi connectors are intended to run continuously. This means that the dataflow is retriggered immediately after a single execution finishes. This also means that the dataflow is only supposed to process as many messages at a time that accumulated in the source data store since the previous execution.

Depending on the connector, if there is already a significant amount of data in the source data store and the connector is not configured properly, you might run into problems with the dataflow's initial execution. In such a case, the dataflow tries to fetch all the data from the source data store during the initial execution. If there is too much data, one of the following happens:

- No data appears in the target Kafka topics, and the log file includes an `InterruptedException` entry.

  This means that loading and processing the data took more time than the timeout configured in dataflow.timeout.
- Not data appears in the target Kafka topics, and the connector throws an `OutOfMemory` error.

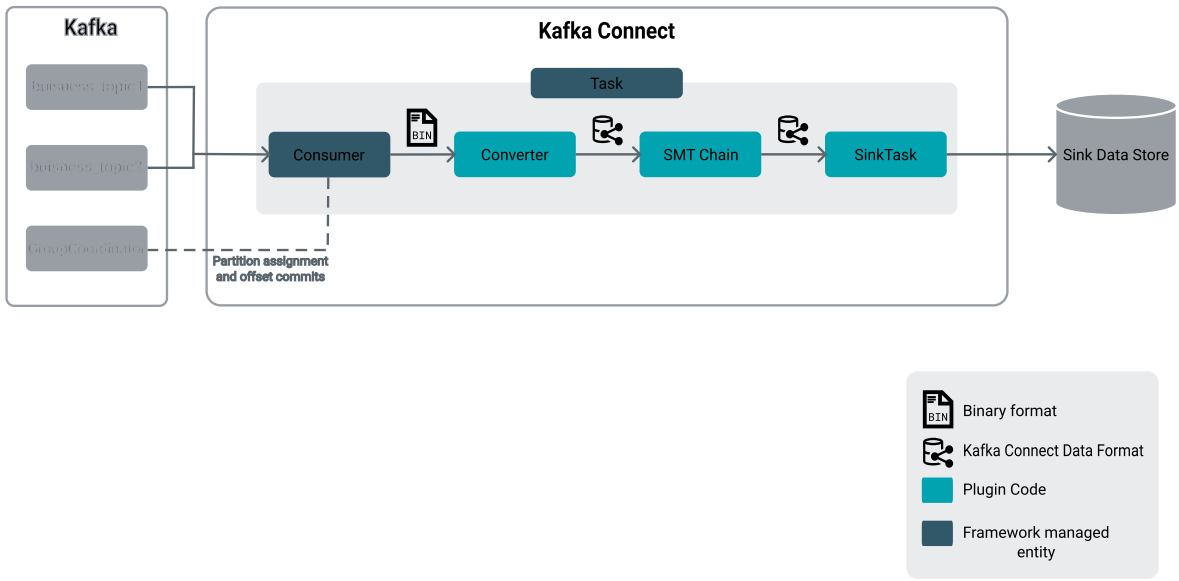
  This can happen because the Stateless NiFi engine, which runs the dataflow, stores all data in memory. Trying to read too much data can exhaust the available memory capacity.

If you need to handle a large initial data load, Cloudera recommends that you process and transfer the initial load using another tool, for example regular NiFi. Afterward, you can configure and start the Kafka connector to handle subsequent changes in the source system.

Sink connectors

A Kafka Connect sink connector obtains data from a Kafka service and delivers that data to an external (sink) data store. The architecture of a sink connector is as follows.

Figure 4: Sink Connector Architecture



When a Stateless NiFi Sink connector instance is deployed, a number of tasks are started for the connector instance. The number of tasks is determined by the tasks.max property, which can be set during connector deployment. Each task consists of the following:

- Kafka Consumer

  The consumer is responsible for fetching the messages from the Kafka topics and delivering them to the converter.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

  The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.
- `SinkTask`

  In Stateless NiFi connectors, the `SinkTask` is responsible for executing one instance of the connector's dataflow.

The consumer is a regular Kafka consumer reading messages from the configured topics and delivering them to the `SinkTask`. Messages are delivered through the converter and the SMT chain.

In a Stateless NiFi Sink connector, the `SinkTask` is the one executing an instance of the dataflow configured in the connector's flow.snapshot property. Any sink connector flow you build should not attempt to source data directly from Kafka using a processor such as ConsumeKafka. Instead, the data should be received from an input port within the dataflow. Each Kafka record is queued into the outbound connection of the input port so that the next processor in the flow is able to process it. The entire task (consumer + `SinkTask`) is executed in one thread. There are two important timeout parameters relevant for sink connectors.

- Offset Flush Interval (offset.flush.interval.ms)

  This is the interval, in milliseconds, at which Kafka Connect attempts to commit task offsets.
- dataflow.timeout

  Specifies the maximum amount of time to wait for the dataflow to complete.



Important: The configuration scope of the two properties is different. Offset Flush Interval (offset.flush.interval.ms) is a Kafka Connect role property that you set in Cloudera Manager. The value set for this property is **global** and affects **all** connectors in your deployment. You can override the role-level value on a per-connector basis by configuring cloudera.offset.flush.interval.ms in the connector configuration. dataflow.timeout is a connector-level property. You set it during connector deployment and it is configured on a per-connector basis.

The execution of a `SinkTask` is the following:

1. The consumer consumes messages from the Kafka topics for the period of time determined by Offset Flush Interval (default is one minute), and puts the messages on the input of the `SinkTask`, that is, the input port of the connector's dataflow.
2. After Offset Flush Interval is reached, the consumer stops consuming messages and the flow is triggered. Once triggered, the flow processes all messages that accumulated on its input port. Processing time is not infinite. The flow will time out once the value set in dataflow.timeout is reached.
3. After flow is finished running, the process restarts with step 1.

Notice that the flow is not triggered for each Kafka message individually. Instead, it is triggered for a batch of messages that the consumer put on its input port in the time specified by Offset Flush Interval. One execution (trigger) of the flow is one transaction. This means that the flow either manages to process all messages within the batch successfully (delivers them to the target system and commits the transaction), or rolls back the session. If the session is rolled back, the consumer is going to put the same batch of messages on the input of the flow in the next execution of step 1. Therefore, in the next execution of step 2, the flow is going to try and process the same batch of messages.

Each Kafka message is delivered to the target system as a single flowfile. Depending on the destination, Cloudera recommends that you consider merging your Kafka records together into bigger chunks before delivering them. For example, assume you are delivering data to HDFS. Because HDFS is not built for accessing and handling many small files, it is not recommended to send each individual Kafka message to HDFS as a separate file.

Merging considerations for sink connectors

Learn about the limitations and considerations when using MergeContent and MergeRecord processors with Stateless NiFi.



Tip: Before you continue, ensure that you read Dataflow execution and scheduling first. You must have an understanding of how dataflow execution works to understand the examples and settings discussed here.

NiFi supports many processors that can be used as sinks for Kafka data. If the destination service operates well with many small messages, then the data from Kafka can often be delivered directly to a sink processor. For example, a PublishJMS processor can easily handle many small messages. However, other services, like S3 or HDFS, perform much better if the data is first batched or merged together. For this reason, the MergeContent and MergeRecord processors are extremely popular in NiFi. They make it possible to merge many small flowfiles into one larger flowfile.

With MergeContent and MergeRecord in traditional NiFi, you can set a minimum and maximum size for the merged data along with a timeout. However, with Stateless NiFi and Kafka Connect, this might not work. This is because only a limited number of flowfiles are made available to the processor. You can still use these processors to merge the data together, but some limitations apply.

If MergeContent or MergeRecord is triggered but not enough flowfiles are present to create a batch, the processor does nothing. Therefore, Cloudera recommends that you set Minimum Number of Entries to 1 and Minimum Group Size to 0 B (these are default values). With these settings, the MergeContent and MergeRecord processors create a merged flowfile from the input flowfiles available in the given execution of the connector and do not wait for more data to arrive. Configuring Max Bin Age is not required in this case.

The Maximum Number of Entries and Maximum Group Size processor properties can also be used to limit the maximum size of the merged flowfile. For example, in a NiFi Stateless Sink connector the consumer running inside the connector puts a group of messages on the input flow. Afterward, the flow is executed. This execution of the flow can only work with the amount of messages that it has on its input when it gets triggered. If it had 115 messages on its input, and MergeContent processor's Maximum Number of Entries is 100, the flow is going to produce two outgoing messages, one with 100 entries and another one with 15.

To control how many messages are placed on the input of the flow for one execution, you can configure the Offset Flush Interval (offset.flush.interval.ms) Kafka Connect role property. You can override this role-level value at the connector level by configuring cloudera.offset.flush.interval.ms in the connector. The lower the interval, the less time the consumer has to put messages on the input of the flow.

NiFi Stateless Sink merging example

Let's look at a merging example with a NiFi Stateless Sink connector. Assume that the dataflow contains the following elements in the specified order:

```
input port -> MergeContent -> PutHDFS
```

Additionally, assume the following configurations:

- Offset Flush Interval is set to one minute (default). In this timeframe the internal consumer puts 1100 messages on the input port of the dataflow.
- dataflow.timeout is 90 seconds.
- The MergeContent processor has the following properties:
  - Maximum Number of Entries = 1
  - Minimum Group Size = 0B
  - Maximum Number of Entries = 100



Important: Offset Flush Interval (offset.flush.interval.ms) is a Kafka Connect role property. It is configured in Cloudera Manager and affects all connectors. You can override this value per connector by configuring cloudera.offset.flush.interval.ms during connector deployment, without changing the global role-level setting. dataflow.timeout is a connector property. It is configured during connector deployment, for example in Streams Messaging Manager. MergeContent's properties are processor properties configured in NiFi when you build the dataflow.

In this case, MergeContent creates 11 flowfiles, and PutHDFS has to upload these 11 files to HDFS within what is left of dataflow.timeout (90 seconds). Assume that PutHDFS manages to upload six files when dataflow.timeout is up. Since the dataflow did not finish execution within the dedicated timeframe, it gets interrupted, and the transaction is rolled back. The consumer starts putting messages on the input of the dataflow for a minute again.

However, since the previous transaction was rolled back, the messages that the dataflow tried to process were not committed. Therefore, the consumer puts the same 1100 messages on the input port of the dataflow. This behavior can result in an endless loop. Additionally, HDFS was not notified that the six files that were uploaded successfully should be removed. This results in these files being reuploaded in the next execution. You can resolve an issue like this by doing the following:

1. Decrease the offset flush interval for this connector.

Configure cloudera.offset.flush.interval.ms in the connector to override the role-level value. For example, setting this to 30000 (30 seconds) instead of 60000 (one minute) halves the number of messages that are put on the input port of the dataflow. Fewer messages on the input port means that the dataflow needs to process fewer messages in one execution. 550 instead of 1100. Using the connector-level property avoids affecting other connectors.

2. Set Maximum Number of Entries to a value that is larger than the number of messages you expect the dataflow will receive.

   Increasing this value to, for example, 1,000,000,000 ensures that all messages are put in one HDFS file. This way all data is either successfully uploaded to HDFS, or the upload is interrupted and rolled back, but without leftover bits in the target system.

Considerations for listen-type source connectors

There are a number of limitations that apply to listen-type connectors that you should be aware of.

A listen-type source connector consists of a NiFi dataflow in which the first processor is a ListenX processor. These processors listen on specific interfaces and specific ports on the host where they are running. For example, a flow starting with a ListenHTTP processor listens on a specific port for HTTP requests made by external clients.

The Kafka Connect framework does not provide the concept of task affinity mapping, meaning that if a connector is deployed with multiple tasks, it is not possible to ensure that tasks are assigned to different Kafka Connect workers (Kafka Connect role instances). Two tasks started on the same worker in a listen-type connector results in a failure. This happens because both connectors try to bind to the same port. For this reason, you must ensure that listen-type source connectors are deployed with a single task. That is, you must set tasks.max=1 when configuring the connector.

A common deployment model is to have a load balancer in front of the Kafka Connect workers and a mapping rule linking to the port on each Kafka Connect worker. This way, the load balancer FQDN can be provided to the client without assuming which worker the task is going to be deployed on.

If multiple tasks are required for performance, you can choose to have multiple standalone Kafka Connect workers and deploy the connector independently on each worker. However, in such a case, Cloudera recommends using Cloudera Flow Management and NiFi as they provide a significantly more powerful environment for this type of use cases.

Mapping of NiFi features

There are some features that exist in NiFi that translate seamlessly in the context of Kafka Connect. The following discusses some of these NiFi features and how they can be utilized in Kafka Connect.

State management

In NiFi, a processor is capable of storing state about the work that it has accomplished. This is particularly important for source components such as the ListS3 processor. This processor keeps state about the data that it has already seen so that it does not list the same files repeatedly. State in NiFi can be stored local to the node (local state), or across all nodes in a cluster (cluster-wide state).

When using the Stateless NiFi Source connector, the state that is stored by these processors is provided to Kafka Connect and is stored within Kafka itself as source offsets and source partition information. This allows a tasks to restart and resume where they left off. If a processor stores local state in NiFi, it will be stored in Kafka using a source partition that corresponds directly to that task. As a result, each task is analogous to a node in a NiFi cluster. If the processor stores cluster-wide state in NiFi, the state will be stored in Kafka using a source partition that corresponds to a cluster-wide state.



Note: Kafka Connect does not allow for state to be stored for a sink task.

Primary node

NiFi provides several processors that are expected to run only on a single node in the cluster. This type of execution is configured by setting Execution to Primary Node Only in the SCHEDULING tab on the NiFi UI when configuring a processor.

When using the Stateless NiFi Source connector, if any source processor in the dataflow is set to run on a primary node only, only a single task will ever run, even if the tasks.max connector property is set to a value higher than one. If you attempt to use multiple tasks for a dataflow that has Primary Node Only set, a warning is logged.

Because processors should only be scheduled with Primary Node Only if they are sources of data, this setting is ignored for all sink tasks and for any processor in a source task that has incoming connections.

Processor yielding

When a processor determines that it is not capable of performing any work (for example, because the system that the processor is pulling from has no more data available), it might choose to yield. This means that the processor stops running for some amount of time. You can configure how long the processors stops running in a dataflow by configuring Processor Configure SETTINGS Yield Duration in NiFi. When using a source connector, if a processor chooses to yield, the source connector pauses for the configured amount of time before triggering the dataflow to run again.

Kafka Connect worker assignment and Stateless NiFi

Learn how NiFi Stateless connectors and tasks get assigned to Connect workers, in what cases you need to be aware of worker assignments, and how you can look up worker assignments.

When running Kafka Connect in a cluster environment, each cluster node in the Kafka Connect cluster has a Kafka Connect worker running on it. In Cloudera, when the cluster is managed by Cloudera Manager, Kafka Connect workers are represented by Kafka Connect role instances, which are deployed under your Kafka service. This means that the number of workers available in the cluster is identical to the number of Kafka Connect role instances.

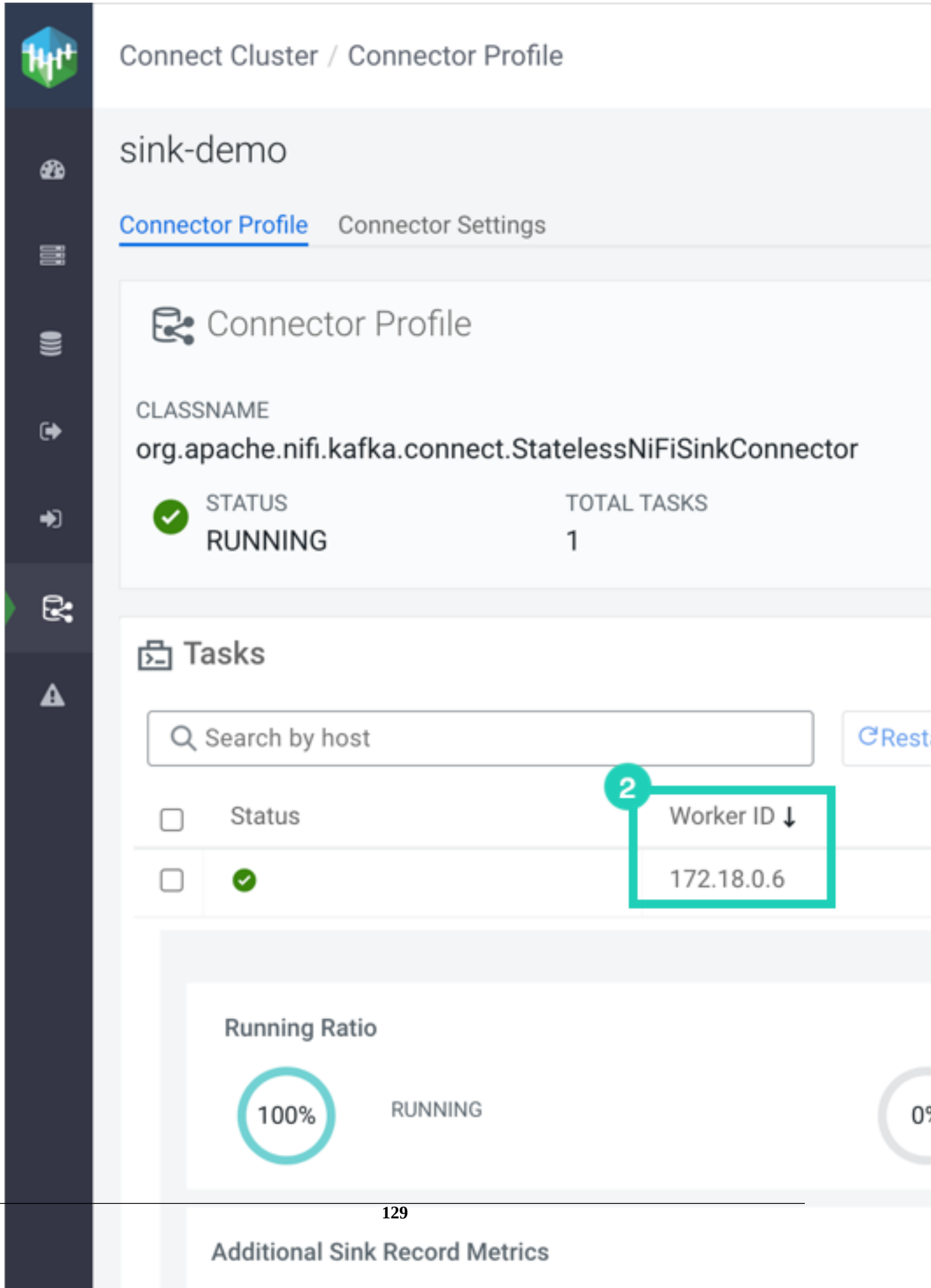
When you deploy a connector, the connector is assigned to a worker. Additionally the task or tasks of the connector are also assigned to workers. The workers that the connector and tasks are assigned to might be the same, but might also be different. If the connector has multiple tasks (tasks.max>1), each task might be assigned to the same worker, or they might be assigned to different workers. In addition, if a Kafka Connect worker is stopped, the connectors and tasks that were assigned to it are moved to different workers in the Connect cluster. In general, you have no agency over worker assignment, it is decided by the Connect Framework, and it is not configurable. Additionally, worker assignments might change dynamically, however, this is a normal and expected behavior of Kafka Connect.



Note: In a test environment you can force deploy connectors and tasks to a specific worker by stopping all other workers In the Connect cluster.

With Stateless NiFi connectors, knowing exact worker assignments is required when you are running a listen-type connector (for example, HTTP Source, Syslog TCP Source, or Syslog UDP Source). With these types of connectors you must ensure that you send messages to the host of the task's worker and not the connector's worker.

You can look up worker assignments in Streams Messaging Manager on the Connector Profile page of the connector. The worker assigned to the connector is displayed in the **Connector Profile** section under **ASSIGNED WORKER** (1). The worker assigned to the task is displayed in the **Tasks** section under **Worker ID** (2).


Connect Cluster / Connector Profile
sink-demo
Connector Profile
Connector Settings
Connector Profile
CLASSNAME
org.apache.nifi.kafka.connect.StatelessNiFiSinkConnector
STATUS
RUNNING
TOTAL TASKS
1
Tasks
Search by host
2
Status
Worker ID
172.18.0.6
Running Ratio
100%
RUNNING
Additional Sink Record Metrics

Related Information

Kafka Connect log files and Stateless NiFi

Kafka Connect log files and Stateless NiFi

Learn how and where Stateless NiFi connectors produce their logs.

Each cluster host that runs a Kafka Connect worker (Kafka Connect role) has a log file specific to Kafka Connect. The logs are stored by default in /var/log/kafka. The Connect related logs are named kafka-connect-*[***HOSTNAME***]*.log. The log directory path is specified by the Kafka Connect Log Directory Kafka Connect role property. This property is configured in Cloudera Manager.

Every log entry produced by a NiFi dataflow that is running in a Kafka Connect connector is directed to the Kafka Connect log file. Specifically, the dataflow log entries are written to the log file located on the cluster host that is running the Kafka Connect worker that is executing the connector's task.

For example, if your connector has a single task, which is executed by worker-1 running on host-1, then the log for that connector will be located on host-1. If a connector has multiple tasks, each running on a different worker, then each task's dataflow writes logs to the respective worker's log file.

If a Kafka Connect worker is stopped, its tasks are moved to other workers. As a result, it is possible that a specific connector's log entries first appear in the log file located on one host, but later on appear in another log file located on a different host.

Related Information

Kafka connect worker assignment and Stateless NiFi

Kafka Connect tasks and Stateless NiFi

Learn which Stateless NiFi connectors you can or cannot run with multiple Kafka Connect tasks (task.max>=2).

By default all connectors that you deploy start with a single Kafka Connect task. The number of tasks deployed for a connector can be configured using the tasks.max connector property, but not all types of connectors are meant to run using more than a single task.

Specifically with Stateless NiFi, there are a handful of source connectors that either cannot run with multiple tasks or function incorrectly when multiple tasks are configured. The following provides information on which connectors can or cannot be run with multiple tasks and what effects configuring multiple tasks has on both sink and source connectors.

Source connectors

Notice that the connector's task is made up of multiple components, these are as follows:

- `SourceTask`

  In Stateless NiFi connectors, the `SourceTask` is responsible for executing one instance of the connector's dataflow.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

  The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.
- Kafka Producer

  The Kafka producer is responsible for delivering messages to the configured Kafka topics.

If you deploy a source connector with more than one task, then each task is going to have its own instance of the task components. Because the dataflow in the `SourceTask` is responsible for fetching the data, rather than the source data store providing the data to the `SourceTask`, it is not always possible to start a connector with more than one task.

For example, starting a JDBC Source connector, which is a Stateless NiFi-based connector, with two tasks would mean that two instances of the dataflow are running in parallel. Both are fetching data from the same table of the same database. These two tasks would be delivering the messages to the same Kafka topic. However, the two dataflow instances running in parallel would have no way of knowing whether a certain record was already been fetched by the other instance. This results in the same piece of data appearing in multiple Kafka messages.

Because of this, there are a number of limitations that you must keep in mind when using the Stateless NiFi Source connectors. These limitations apply when you are running a custom dataflow with a Stateless NiFi Source connector, as well when running any of the Stateless NiFi-based source connectors that Cloudera develops.

Listen-type connectors

A listen-type connector consists of a NiFi dataflow in which the first processor is a ListenX processor. Listen-type processors start a server with one or more background threads, listening on a defined network port. Stateless NiFi-based connectors of this category are:

- HTTP Source
- Syslog TCP Source
- Syslog UDP Source

For example, the HTTP Source connector's flow uses a ListenHTTP NiFi processor to start an HTTP server listening on a configured port. Deploying this connector with two tasks would mean that two ListenHTTP processors are running simultaneously, both listening on the same network port. Since there is no user-defined mapping between tasks and Kafka Connect workers, it is possible that these two instances get deployed to the same worker. This means that two HTTP servers would be listening on the same port on the same cluster node. This results in port collision. To avoid issues like this, you must ensure that listen-type connectors are always deployed with exactly one task.

Primary node only connectors

Connectors in this category are connectors that use a dataflow with a NiFi processor that is configured to run on the primary node only if the flow was running on a regular NiFi cluster. For example, the QueryDatabaseTableRecord processor is a processor that is intended to run on primary nodes only. This processor is used in the JDBC Source connector. Stateless NiFi-based connectors of this category are:

- JDBC Source
- SFTP Source

For these connectors, the configured value of the tasks.max property is ignored. The connector is always started with exactly one task.

All other connectors

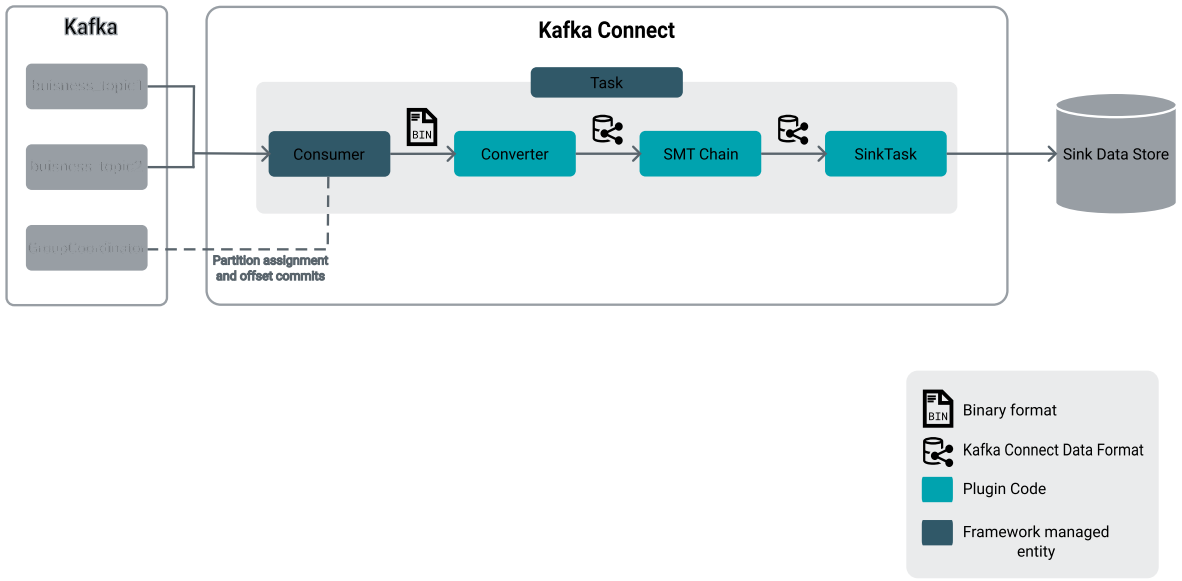
Connectors in this category are connectors that are neither listen-type or primary node only. Stateless NiFi-based connectors of this category are:

- JMS Source
- MQTT Source

There is no limitation to the number of tasks for these connectors. In the case of these connectors, it is the responsibility of the source data store system to make sure that one piece of data is only given to one of the connector's tasks

Sink Connectors

Figure 5: Sink Connector Architecture



Notice that the connector's task is made up of multiple components, these are as follows:

- Kafka Consumer

  The consumer is responsible for fetching the messages from the Kafka topics and delivering them to the converter.
- SMT Chain

  The Single Message Transforms (SMT) chain can modify and filter the data on a single message basis.
- Converter

  The converter is responsible for converting Kafka message keys and values with a configured conversion mechanism. However, Stateless NiFi connectors currently only use keys of string type and values of byte array type. As a result, the converter does not perform any meaningful conversion.
- `SinkTask`

  In Stateless NiFi connectors, the `SinkTask` is responsible for executing one instance of the connector's dataflow.

If you deploy a sink connector with more than one task, each task is going to have its own instance of the four task components.

With sink connectors, there is no limitation to using more than one task for a connector. When a connector is started with multiple tasks, each task has its own Kafka consumer, but these consumers belong to the same Kafka consumer group. Kafka makes sure that a message fetched by one consumer is not given to another consumer within the same consumer group.

There is no user-defined mapping between tasks and Kafka Connect workers, meaning that once deployed, the tasks of a connector may or may not be executed on the same worker.

Developing a dataflow for Stateless NiFi

Learn about the recommended process of building a dataflow that you can deploy with the Stateless NiFi Sink or Source connectors. This process involves building and designing a parameterized dataflow within a process group and then downloading the dataflow as a flow definition.

About this task

The general steps for building a dataflow are identical for both source and sink connector flows. For ease of understanding a dataflow example for a simple MQTT Source and MQTT Sink connector is provided without going into details (what processors to use, what parameters and properties to set, what relationships to define and so on).

Before you begin

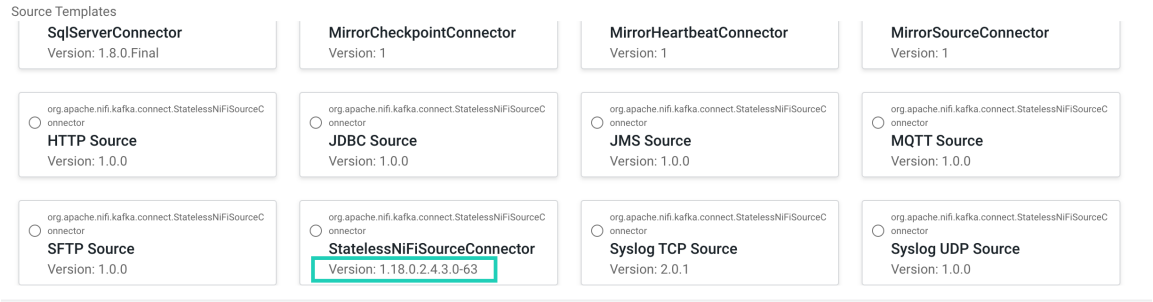
- Ensure that you reviewed Dataflow development best practices for Stateless NiFi.
- You have access to a running instance of NiFi. The NiFi instance does not need to run on a cluster. A standalone instance running on **any machine**, like your own computer, is sufficient.
- Ensure that the version of your NiFi instance matches the version of the Stateless NiFi plugin used by Kafka Connect on your cluster.

  You can look up the Stateless NiFi plugin version either by using the Streams Messaging Manager UI, or by logging into a Kafka Connect host and checking the Connect plugin directory. In addition, ensure that you note down the plugin version. You will need to manually edit the flow definition JSON and replace the NiFi version with the plugin version.

For Use Streams Messaging Manager UI

1. Access the Streams Messaging Manager UI, and click **Connect** in the navigation sidebar.
2. Click the New Connector option.
3. Locate the StatelessNiFiSourceConnector or StatelessNiFiSinkConnector cards. The version is located on the card.

   The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version on the card is 1.18.0.2.4.3.0-63, then you must NiFi 1.18.0 to build your flow.



Important: Using a NiFi instance that is running a different version of NiFi than the Stateless NiFi plugin is not supported. Interoperability across versions is not guaranteed.

For Check Connect plugin directory

1. Using SSH, log in to one of your Kafka Connect hosts.
2. Navigate to the directory where your Kafka Connect plugins are located.

   The default directory is /var/lib/kafka. If you are using a custom plugin directory, go to Cloudera Manager Kafka service Configuration and search for the Plugin Path property. This property specifies the directory where Kafka Connect plugins are stored.
3. List the contents of the directory and look for a nifi-kafka-connector-*[***VERSION***]* entry.

   The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version in the name of the entry is 1.18.0.2.4.3.0-63, then you must use NiFi 1.18.0 to build your flow.

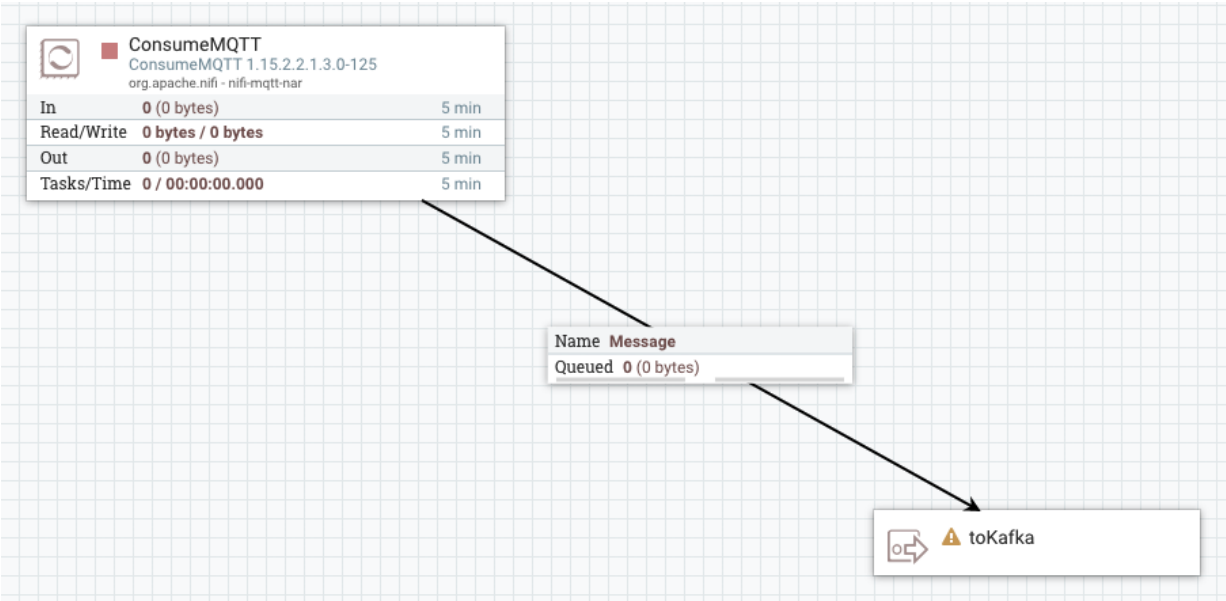
   Important: Using a NiFi instance that is running a different version of NiFi than the Stateless NiFi plugin is not supported. Interoperability across versions is not guaranteed.

Procedure

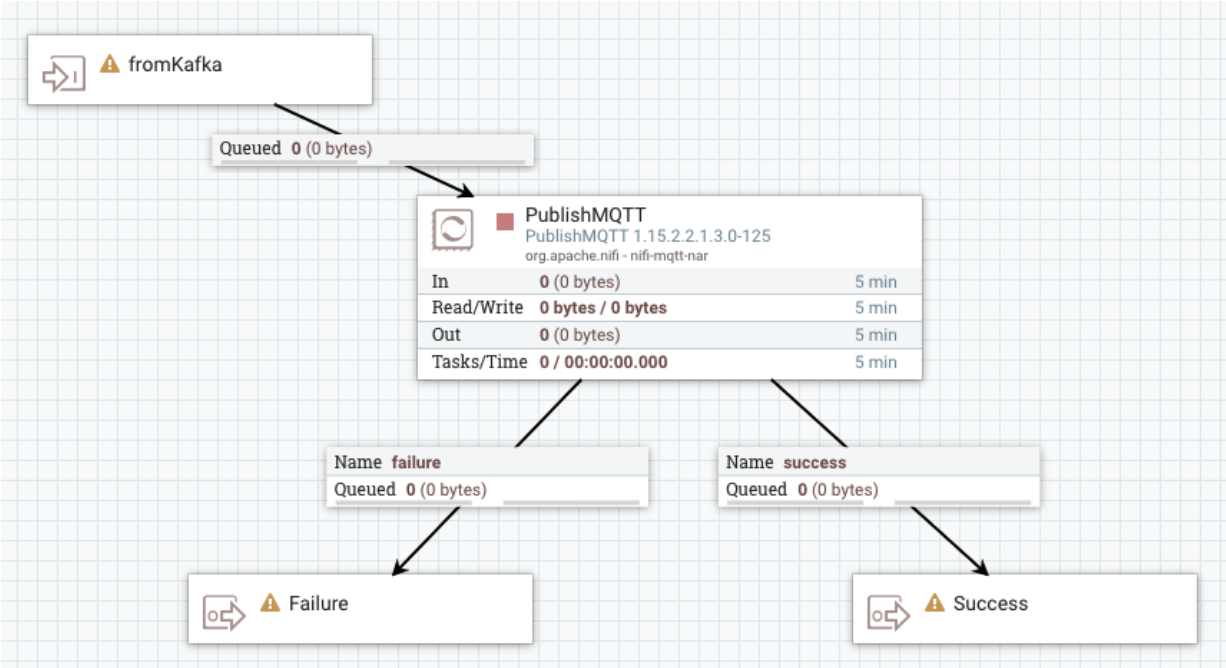
1. Access the NiFi UI.
2. Create a new process group.

3. Right-click the process group and select Configure.
4. Create a new parameter context and assign it to the process group:
   a) Select Process Group Parameter Context Create new parameter context...
   b) Enter a name for the parameter context.
   c) Click Apply.
5. Click Apply and then click OK.
6. Close the configuration dialog.
7. Double-click the process group you created.
8. Design and parameterize your dataflow.

   For example, you can build a dataflow that you deploy as a source connector. The following is an example dataflow for a simple MQTT source connector. It consists of a ConsumeMQTT processor and an output port (representing a Kafka topic).

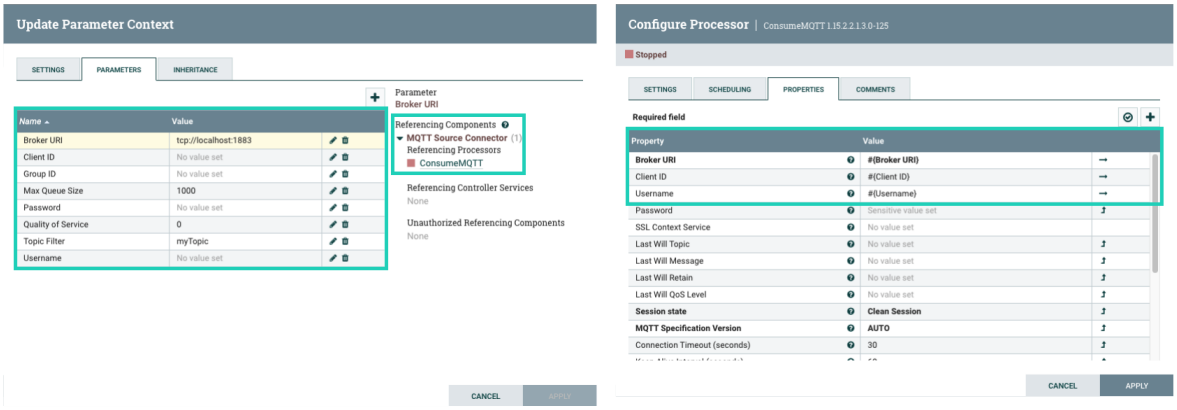


   Alternatively, you can also build a dataflow to deploy as a sink connector. The following is an example dataflow for a simple MQTT sink connector. It consists of an input port (representing a Kafka topic), a PublishMQTT processor, and two output ports.



In the case of this example, a failure output port is required. This is done so that in case the destination is not available, the session can be rolled back and the Kafka message can be declined. The failure port is specified in the configuration of the connector when deploying it through Streams Messaging Manager.

When designing and building your dataflow, ensure that you update your parameter context and reference the parameters in your components. For example:



Keep in mind that sensitive properties in processors or controller services must always be parametrized. That is, they must get their values from a parameter in the process group's parameter context. This is because the values of sensitive properties are not exported with the process group, therefore, they must be configurable when deploying the connector.

Additionally, sensitive processor or controller service properties can only reference sensitive parameter context parameters. That is, the parameter must be marked as sensitive during creation.

For more information on building dataflows, parameters, and referencing parameters, see the *Flow Management* library.

9. Exit the process group.
10. Right-click the process group and select Download flow definition Without external service .

If you followed these steps, your process group is not referencing controller services that are outside of the process group. As a result, you can choose to download the flow without external services included.

11. Open the JSON file you downloaded in a text editor, and replace all occurrences of the NiFi version with the Stateless NiFi plugin version used in your cluster.

    For example, if the NiFi version is 1.18.0 and your plugin version is 1.18.0.2.4.3.0-63, then every occurrence of 1.18.0 in the JSON must be replaced with 1.18.0.2.4.3.0-63.

Results

The dataflow is downloaded as a flow definition (JSON file).

What to do next

Deploy the dataflow as a Kafka Connect connector using the Stateless NiFi Source or Sink connectors. Continue with *Deploying a dataflow using stateless NiFi*.

Related Information

Deploying a dataflow using Stateless NiFi

Flow Management

Deploying a dataflow using Stateless NiFi

A custom developed NiFi dataflow can be deployed with the Stateless NiFi Source or Sink connectors using Streams Messaging Manager.

About this task

After building and downloading your dataflow, you can deploy it in Kafka Connect as a source or sink connector. This can be done using Streams Messaging Manager. The following list of steps walk you through the process of deploying a Stateless NiFi Sink or Source connector and provide examples on how dataflows are configured using Streams Messaging Manager.

These steps are for the Streams Messaging Manager UI, however, the actions described here can also be completed using the Streams Messaging Manager REST API. For more information, see *Streams Messaging Manager REST API Reference*.

Before you begin

- A Cloudera cluster containing Streams Messaging Manager and Kafka is available.
- You have access to the Streams Messaging Manager UI.
- The flow definition of the dataflow that you want to deploy is available to you. For more information on how to design, build, and export a dataflow using the NiFi UI, see Developing a dataflow for Stateless NiFi .
- The properties of Kafka consumers and producers used internally by the connector (Connect-managed clients) can be configured on a per-connector basis. The following steps give an example of how this can be done. However, in order for consumer and producer property overrides to take effect, you must ensure that the Connector Kafka Client Configuration Override Policy Kafka service property is set to All in Cloudera Manager. By default this property is set to None, which means that no overrides are allowed.

Procedure

1. Access the Streams Messaging Manager UI.
2. Click (Connect) on the navigation sidebar.
3. Click + New Connector to add a new connector.
4. Select a connector.

   Which connector you select depends on the type of dataflow you want to run. Select StatelessNiFiSourceConnector found on the **Source Templates** tab if your dataflow collects (sources) data from

systems and publishes it to Kafka. Select the StatelessNiFiSinkConnector from the Sink **Templates** tab if your dataflow delivers (sinks) Kafka data into other systems.

5. Enter a name for the connector in the Enter Name field.

   Ensure that you add a unique and easily identifiable name. The name of the connector cannot be changed once the connector is deployed.

6. Configure the connector

   Connector configuration can be broken into three steps. First, you configure the properties available in the default configuration template. Second, you configure other connector properties. Finally, you specify the flow parameters specific to your dataflow. These are the parameters that you set up in the parameter context when you built the flow in NiFi. Even though the following breaks the process up into multiple substeps, all properties are configured on the same page in the Streams Messaging Manager UI.

   a) Configure the properties available in the default template.

      The following properties from the default template are the ones that you should configure. Other properties in the template that are not highlighted here have working default values configured. Configuring the properties that are present in the template but not highlighted in the following list is not recommended by Cloudera.

      flow.snapshot

      Specifies the dataflow to run. You have multiple options on how to specify the value of this property. You can copy and paste the contents of the flow definition, upload the flow definition, or reference a file path. For more information on flow.snapshot configuration, see *Configuring flow.snapshot for Stateless NiFi connectors*.

      input.port

      The name of the input port in the NiFi dataflow that Kafka records are sent to by the connector's Kafka consumer. This property is specific to the Stateless NiFi Sink connector and is a port in NiFi terminology, not a network port. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple Input Ports, this property must be specified.

      output.port

      The name of the output port in the NiFi dataflow that is used to deliver the messages from the dataflow to the connector's Kafka producer. This property is specific to the Stateless NiFi Source connector and is a port in NiFi terminology, and not a network port. If the dataflow contains exactly one port, this can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

      topics

      For source connectors, this is the name of the topic to deliver data to. For sink connectors, this can be a comma-separated list of topics to fetch data from.

      krb5.file

      Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. This property is optional and is only required if a connection is established with a Kerberized service. Use the default value if an appropriate krb5.conf is located in /etc.

   b) Add other connector properties.

      There are many other properties that you can set. These include the properties specific to the Stateless NiFi connectors as well as the Kafka Connect properties that come from the base framework. Which of these you require depends on your dataflow and use case.

      In Stateless NiFi connectors, the Kafka message's key is always a string, its value is always a byte array. As a result, the key and value converters of the connector should not be modified. Use the default values, which are as follows:

      ```
      "key.converter": "org.apache.kafka.connect.storage.StringConverter"
      ```

```
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er"
```

A comprehensive list of all other properties that the connectors accept can be found in *Stateless NiFi Source connector reference*, *Stateless NiFi Sink Connector reference*, and the *Apache Kafka documentation*.

c) Configure dataflow-specific parameters.

If you followed the recommendations provided in *Dataflow development best practices for Stateless NiFi* and *Developing a dataflow for Stateless NiFi*, your dataflow is parameterized. The parameters that you defined within the dataflow can be assigned values using the configuration pane in Streams Messaging Manager. This is done by adding a configuration entry for each of the dataflow parameters that you want to configure. Dataflow parameters can be added to the configuration as follows:



Tip: Dataflow-specific parameters can also be automatically imported by clicking the Save and Enhance option when specifying the flow.snapshot property. For more information, see *Configuring flow.snapshot for Stateless NiFi connectors*.

```
"parameter.[***PARAMETER NAME***]": "[***VALUE***]"
```

For example, assume that you have a the following configuration entry:

```
"parameter.Directory": "/mydir"
```

In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory.

You can also apply a parameter to a specific parameter context. This is done by prefixing the parameter name with the name of the parameter context followed by a colon.

```
"parameter.[***PARAMETER CONTEXT NAME***]:[***PARAMETER NAME***]":
 "[***VALUE***]"
```

For example, assume you have the following configuration entry:

```
"parameter.My Context:Directory": "/mydir"
```

In a case like this, only the parameter context called My Context gets the specified value for the Directory parameter.

Once all properties are configured, your configuration should look similar to the following example:

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSource
Connector",
 "flow.snapshot": "[***FLOW DEFINITION JSON***]",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConve
rter",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.MQTT Source Parameters:MQTT Broker URI": "tcp
://[***HOST***]:[***PORT***]",
 "parameter.MQTT Source Parameters:MQTT Quality of Service": "0",
 "parameter.MQTT Source Parameters:MQTT Topics": "[***MQTT TOPIC
 NAME***]"
}
```

This example is for a custom built MQTT source connector. The example uses additional Connect properties such as key.converter and value.converter. The example also specifies a number of flow parameters such as MQTT Broker URI and MQTT Topics.

d) Configure Kafka producer and consumer properties.

Source connectors use internal Kafka producers to write message to topic, sink connectors use internal consumers to fetch messages from topics. The properties of these producers and consumers can be modified on a per-connector basis. This is done by adding the required properties with the producer.override and consumer .override prefixes to the connector configuration.

For example, if you want to set the consumer of your sink connector to use auto.offset.reset=earliert, you need to add the following configuration entry to the configuration of the connector.

```
consumer.override.auto.offset.reset=earliest
```

7. Click Validate.

   The validator displays any JSON errors in your configuration. Fix any errors that are displayed. If your JSON is valid, the **JSON is valid** message is displayed in the validator.

8. Click Next.
9. Review your connector configuration.
10. Click Deploy.

Related Information

Streams Messaging Manager REST API Reference
Apache Kafka documentation
Dataflow development best practices for Stateless NiFi
Developing a dataflow for Stateless NiFi
Configuring flow.snapshot for Stateless NiFi connectors

Downloading and viewing the predefined Stateless NiFi dataflows shipped in Cloudera

Learn how to download and view the predefined dataflows used by the Stateless NiFi-based connectors shipped in Cloudera. Predefined dataflows are downloaded from Cloudera cluster hosts. The downloaded files can be uploaded to NiFi to examine the dataflow's structure.

About this task

A number connectors shipped and developed by Cloudera are based on Stateless NiFi and run predefined dataflows. For example, the JDBC Source connector is a Stateless NiFi-based connector that runs a predefined dataflow.

To better understand how Stateless NiFi-based connectors like JDBC Source work, and how they move and manipulate Kafka data, it can be useful to view the dataflow that the connectors use.

These dataflows are installed on every cluster host that is running a Kafka Connect service role. You can fetch the dataflows from your cluster hosts, load them into NiFi, and look at their flow structure.

The flow definition JSON files of these dataflows are present on the cluster hosts at /etc/kafka_connect_ext/flow-definitions. However, you cannot simply load these flow definition files as is into NiFi. They must be edited first.

Before you begin

- You have access to a running Cloudera cluster that has a Kafka service and Kafka Connect service roles deployed.
- You have access to a running instance of NiFi. The NiFi instance does not need to run on a cluster. A standalone instance running on **any machine**, like your own computer, is sufficient.

Procedure

1. Fetch the flow definition of the chosen connector from the cluster.

The flow definition files are located on the cluster hosts at /etc/kafka_connect_ext/flow-definitions. How you complete this step depends on your cluster environment and what utilities are available to you. For example, if using scp, you can run the following command.

```
scp [***USER***]@[***CLUSTER HOST***]:/etc/kafka_connect_ext/flow-definit
ions/[***CONNECTOR NAME***]-[***VERSION***].json .
```

2. Open .json file in a text editor.
3. Copy contents of the "flow" JSON element to a separate .json file.
4. In the .json you created, replace the value of each "version" element with the version of NiFi you will use to view the dataflow.
5. In the NiFi UI, drag a new process group to the canvas.
6. In the Add Process Group modal, click  and browse for your .json file.
7. Click Add.

Results

The dataflow is imported to the process group with all of its parameters.

What to do next

After the dataflow is imported, you can view the structure of the flow to gain a better understanding of how the dataflow operates. If you want to, you can make modifications and redeploy the modified dataflow as a custom connector.

Related Information

Developing a dataflow for Stateless NiFi

Deploying a dataflow for Stateless NiFi

Configuring flow.snapshot for Stateless NiFi connectors

Learn about the options you have in Streams Messaging Manager when configuring the flow.snapshot Stateless NiFi connector property as well as the Save and Enhance feature that can be used to automatically import NiFi parameters into the connector configuration.

Important: Configuring the flow.snapshot property is only required when you are deploying your custom developed dataflows with the Stateless NiFi Source or Sink connectors. Cloudera developed connectors powered by Stateless NiFi do not require flow.snapshot to be set. If the connector you are deploying does not by default have a flow.snapshot entry in its configuration template in Streams Messaging Manager, you do not need to configure it.

NiFi dataflows within Kafka Connect are deployed using the Stateless NiFi Source and Sink connectors. When you deploy your custom developed dataflows with the Stateless NiFi connectors, you must configure the flow.snapshot property. This property specifies the dataflow that is run within the connector.

Because the value of this property is the dataflow itself, the value you add to the property is not a simple string or integer. Instead, the value is the full contents of a flow definition JSON file, which depending on the dataflow, can be considerable in size.

As a result of this, the property is handled in a unique way by Streams Messaging Manager. You can use multiple configuration methods to specify value. Additionally, you can choose to automatically enhance (amend) the configuration with the NiFi parameters that are available within the flow definition.

Configuration methods

You can use the following configuration methods when specifying a value for flow.snapshot. The following instructions assume that a flow definition JSON file is available to you and that you are logged in to the **Connect** tab of the Streams Messaging Manager UI.

Note:

If the flow definition is larger than 64 KB, the following message appears:

```
Too large config, to view flow.snapshot click the edit button...
```

In a case like this, the dataflow is added, but you are not able to edit it directly in the **Properties** section.

Instead, you must click and edit or review the flow definition in the modal window that appears.

Copy and paste the contents of the flow definition

This method is similar to configuring any other connector property. The only difference is that the value you specify is considerably longer than other property value.

1. Open the flow definition in an editor.
2. Copy the full contents of the file.
3. Paste the contents of the file in the value field of the flow.snapshot property.

Upload the flow definition

This method involves uploading the flow definition from your own machine. This method is convenient if the flow definition JSON is available on the machine you are using to access the Streams Messaging Manager UI.

1. Click next to flow.snapshot.
2. Click Browse... in the modal window that appears.
3. Select the flow definition file that you want to use.
4. Click Save or Save and Enhance.

Reference a file path

The flow.snapshot property accepts file paths. This means that instead of copy-pasting or uploading the full contents of the flow definition JSON, you can also specify a location on disk where the file is available. Cloudera recommends that you use this method if the flow definition is larger than 1 MB.

1. Deploy the flow definition file on all Kafka Connect hosts.

   Important: Ensure that the file is deployed to all hosts at the same location and that the user that Kafka Connect is running as, which defaults to kafka, has read access to the file.
2. Set the file path as the value of the flow.snapshot property.

Using Save and Enhance

NiFi dataflows that you run within Kafka Connect are in the majority of cases parameterized. You can assign values to the parameters available within the dataflow on the Streams Messaging Manager UI during connector configuration. This is done by adding each parameter, with specific prefixes, in the **Properties** section. However, this can take a considerable amount of time if done manually. This is especially true if the dataflow has many parameters and multiple parameter contexts.

In cases like this, you can choose to enhance the configuration by clicking the Save and Enhance option after a value for the flow.snapshot property is added. If this option is used, Streams Messaging Manager parses the dataflow specified in flow.snapshot, extracts all parameters that are available in the dataflow, and adds them to properties of

the connector. The Save and Enhance option is available on the modal window that appears after you click next to the property.

Tutorial: developing and deploying a JDBC Source dataflow in Kafka Connect using Stateless NiFi

A step-by step tutorial that walks you through how you can create a JDBC Source dataflow and how to deploy the dataflow as a Kafka Connect connector using the Stateless NiFi Source connector. The connector/dataflow presented in this tutorial reads records from an Oracle database table and forwards them to Kafka in JSON format.

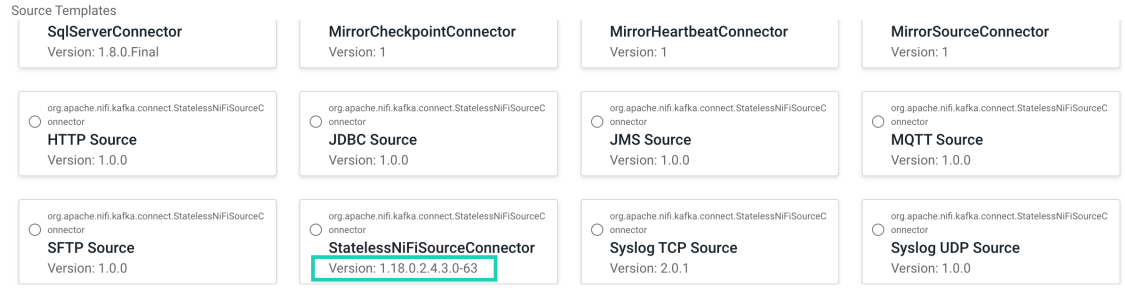
Before you begin

- Look up the Stateless NiFi plugin version either by using the Streams Messaging Manager UI, or by logging into a Kafka Connect host and checking the Connect plugin directory.

  #### For Use Streams Messaging Manager UI

  1. Access the Streams Messaging Manager UI, and click **Connect** in the navigation sidebar.
  2. Click the New Connector option.
  3. Locate the StatelessNiFiSourceConnector or StatelessNiFiSinkConnector cards. The version is located on the card.

     The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version on the card is 1.18.0.2.4.3.0-63, then you must NiFi 1.18.0 to build your flow.

     Important: Using a NiFi instance that is running a different version of NiFi than the Stateless NiFi plugin is not supported. Interoperability across versions is not guaranteed.

  #### For Check Connect plugin directory

  1. Using SSH, log in to one of your Kafka Connect hosts.
  2. Navigate to the directory where your Kafka Connect plugins are located.

     The default directory is /var/lib/kafka. If you are using a custom plugin directory, go to Cloudera Manager Kafka service Configuration and search for the Plugin Path property. This property specifies the directory where Kafka Connect plugins are stored.
  3. List the contents of the directory and look for a nifi-kafka-connector-*[***VERSION***]* entry.

     The version is made up of multiple digits. The first three represent the NiFi version. For example, if the version in the name of the entry is 1.18.0.2.4.3.0-63, then you must use NiFi 1.18.0 to build your flow.

     Important: Using a NiFi instance that is running a different version of NiFi than the Stateless NiFi plugin is not supported. Interoperability across versions is not guaranteed.

- Download and start NiFi. You can download NiFi from https://archive.apache.org/dist/nifi/. This example uses NiFi 1.18.0 (nifi-1.18.0-bin.zip).

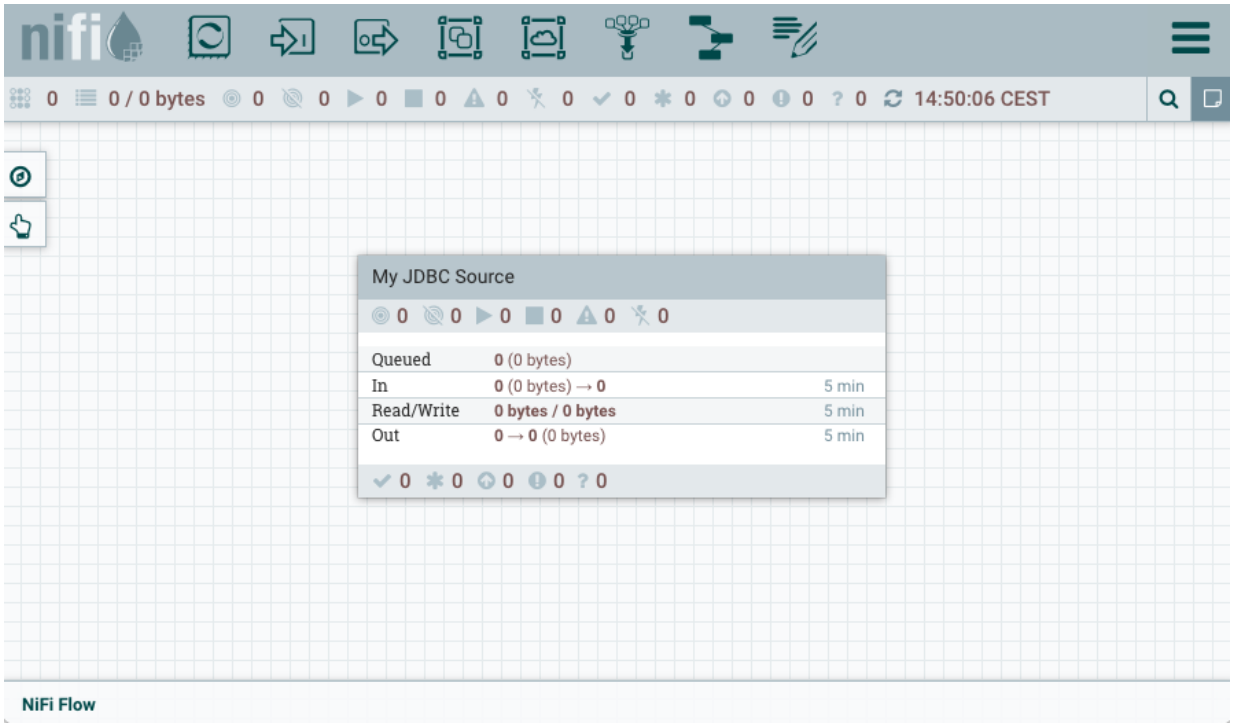
- The connector/dataflow developed in this tutorial requires the Oracle JDBC driver to function. Ensure that the driver JAR is deployed and available on every Kafka Connect host under the same path with correct file permissions. Note down the location where you deploy the driver, you will need to set the location as a property value during connector deployment. For example:

```
cp ./ojdbc8-[***VERSION***].jar /var/lib/kafka_connect_jdbc/oracle-connector-java.jar
```

```
chmod 644 /var/lib/kafka_connect_jdbc/oracle-connector-java.jar
```

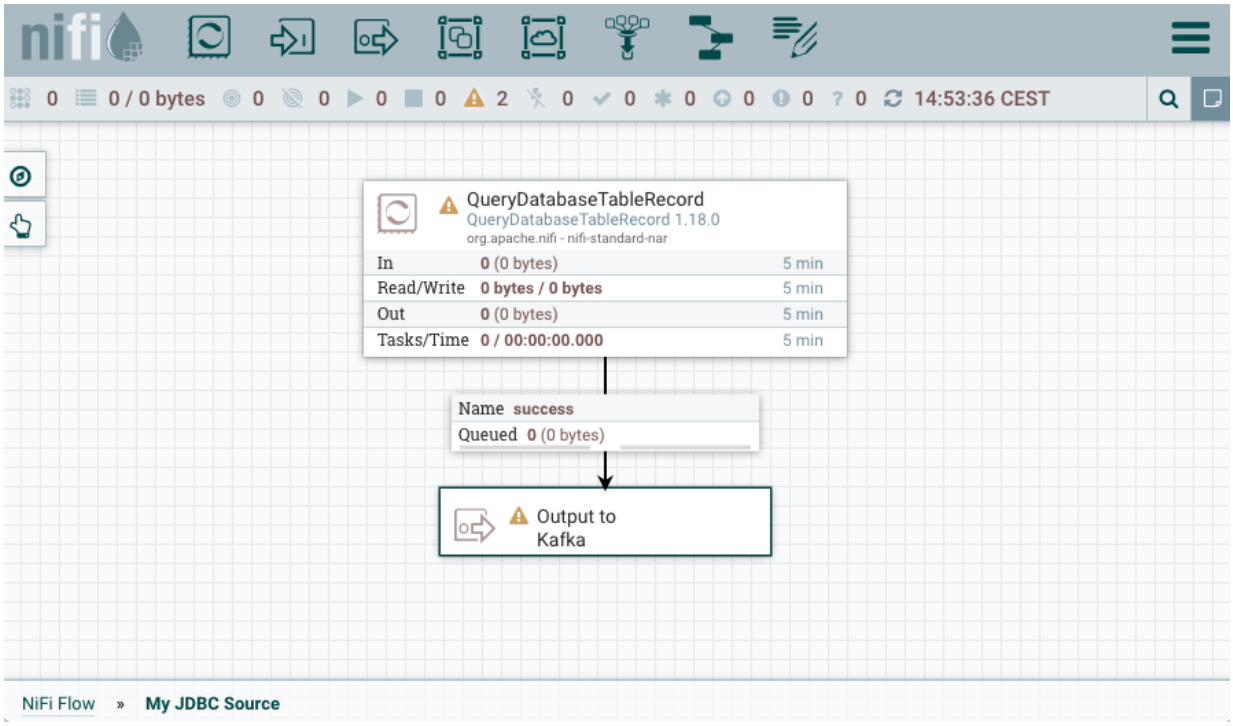
Procedure

1. In NiFi, create a process group and give it a name.



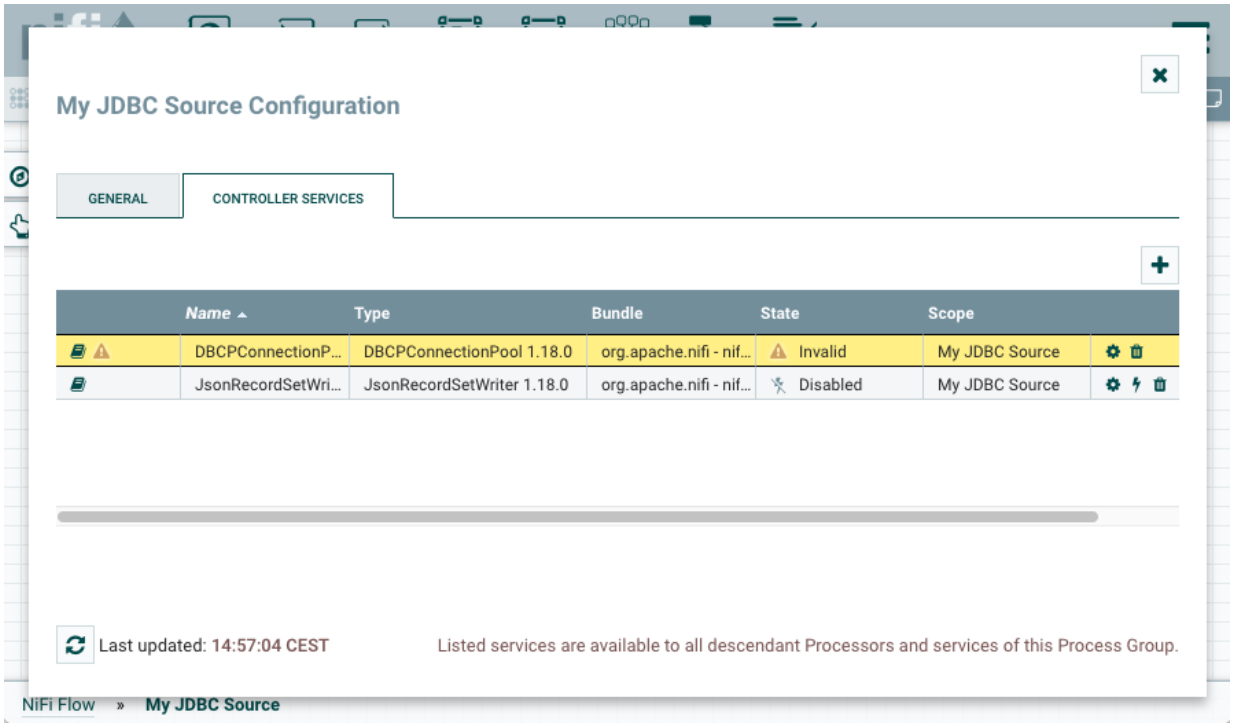
2. Add a QueryDatabaseTableRecord processor and an output port to the process group.

   Note down the name of the output port, you will need to set the name as the value of a connector property during connector deployment.



3. Add a DBCPConnectionPool and a JsonRecordSetWriter controller service to the process group.

   The controller services are required for the QueryDatabaseTableRecord processor to access the database and to convert records.



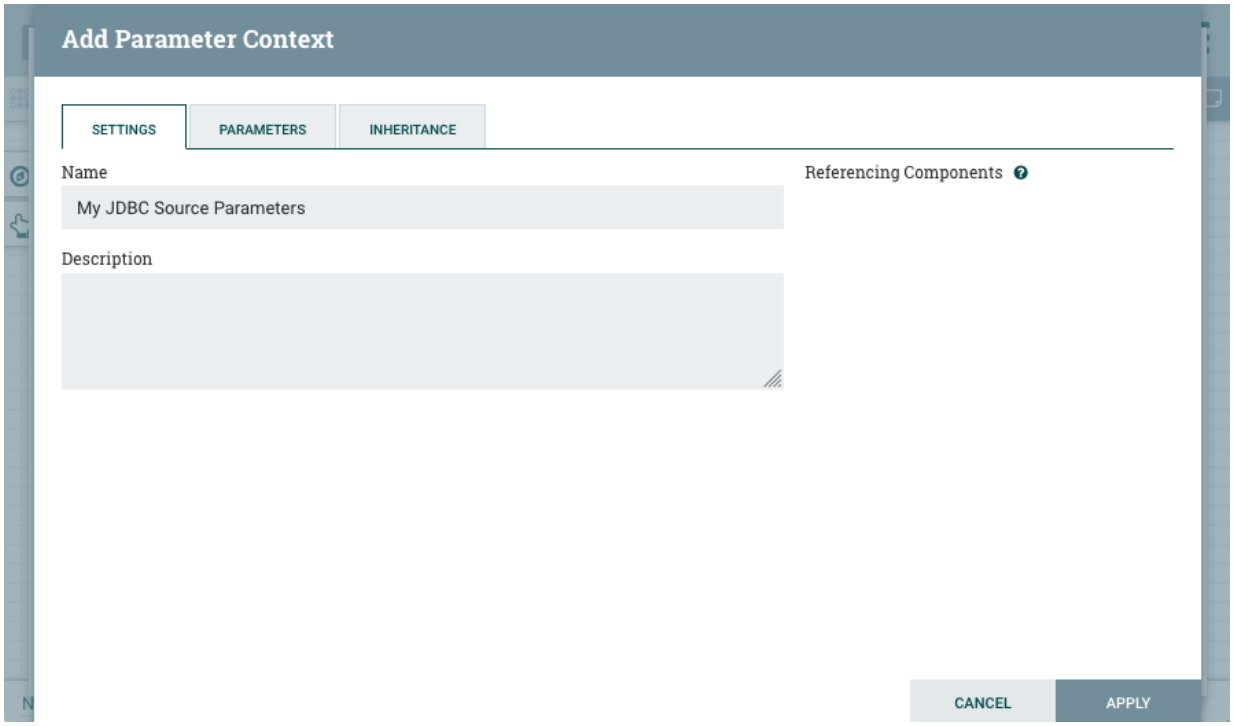
4. Examine the properties of the QueryDatabaseTable processor and the two controller services. Collect the properties that you want to parameterize.

   Parameterized properties can be set when deploying the connector. In this example, the following properties are parametrized:

   - Database Type

- Table Name
- Maximum-value Columns
- Initial Load Strategy
- Database Connection URL
- Database Driver Class Name
- Database Driver Location
- Database User
- Database Password

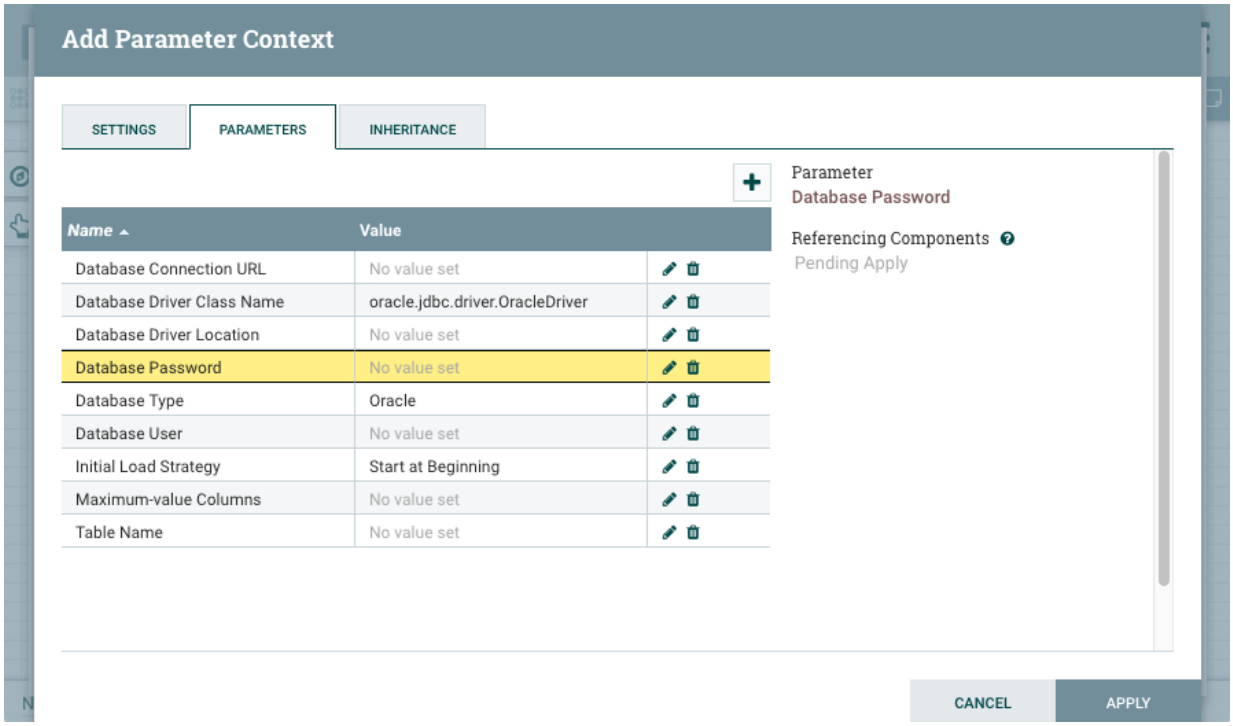
5. Create a parameter context, give it a name, and add the above mentioned parameters to the parameter context.



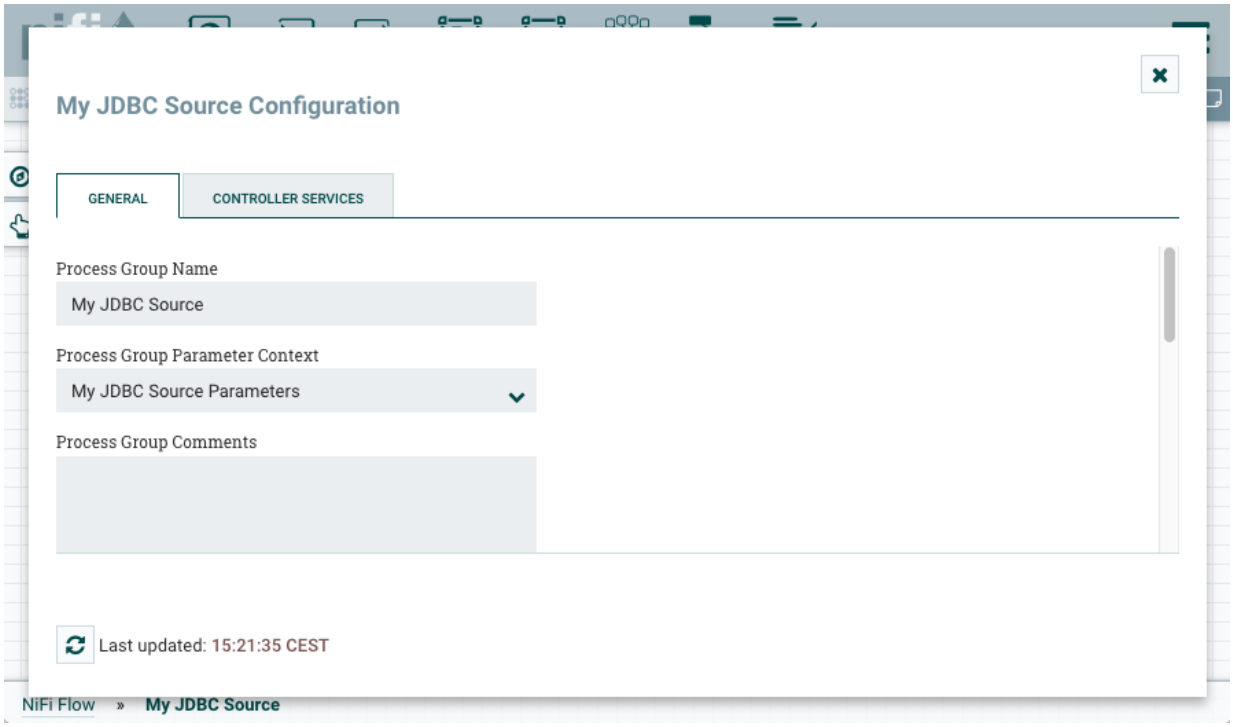
You can assign values to the parameters in this step. The values you assign here serve as default values for the parameters when the flow is deployed as a connector. Assigning values is not mandatory as all parameters will be configurable during connector deployment. However, assigning sensible default values is a good practice and makes the dataflow easier to deploy in Kafka Connect.

In this example, the connector will connect to an Oracle database. Therefore, values are specified for Database Driver Class Name and Database Type parameters. Additionally, Initial Load Strategy is set to Start at Beginning.

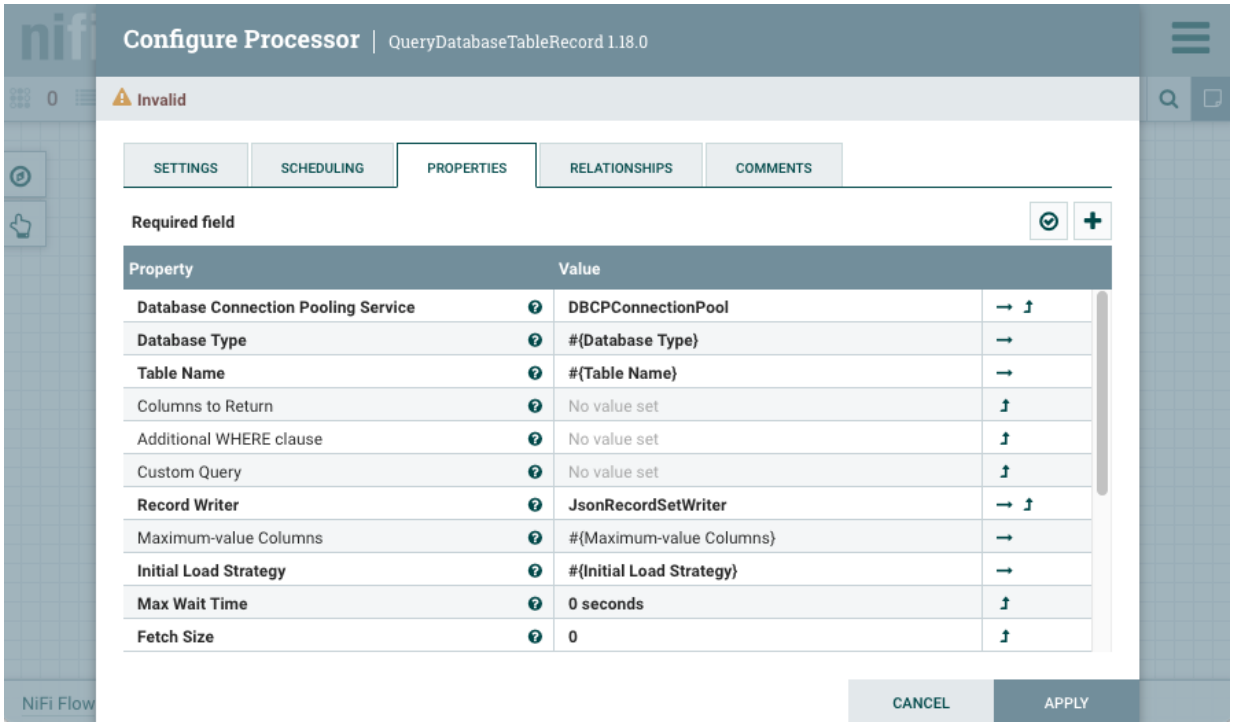
Ensure that you mark the Database Password parameter as a Sensitive Value during creation. This is because only sensitive parameters can be referenced from sensitive processor or controller service properties.

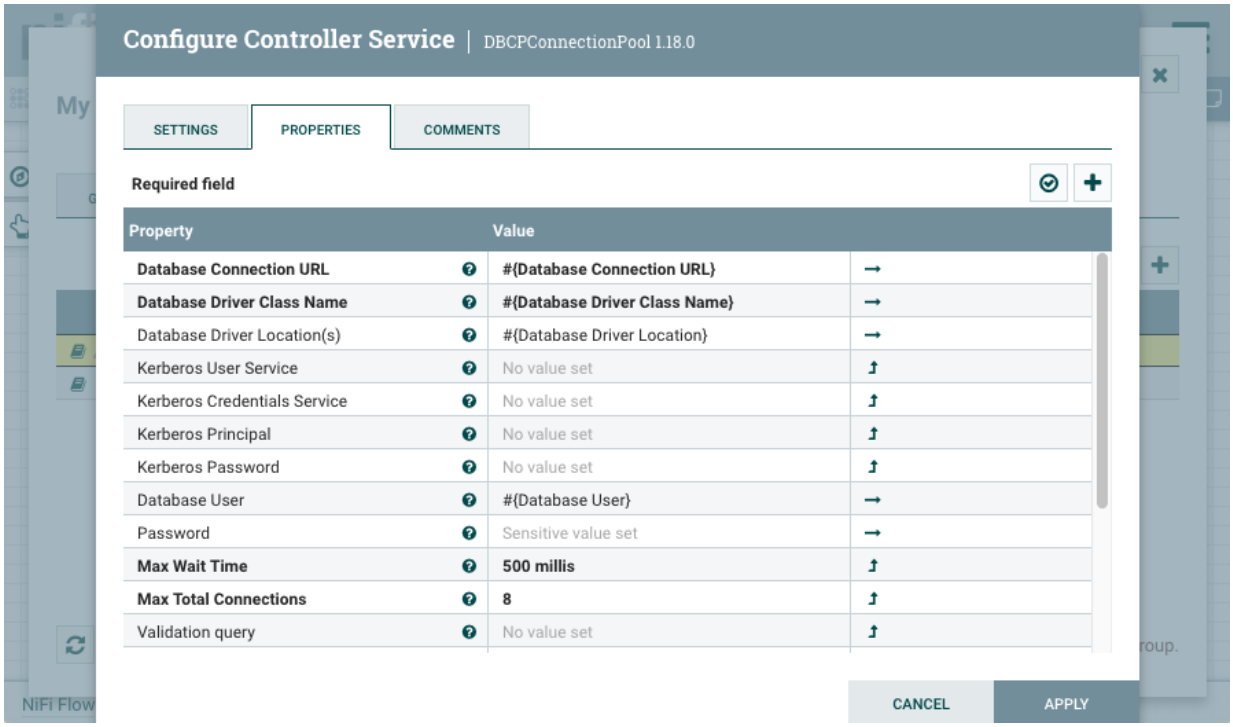


6. Assign the parameter context to the process group.



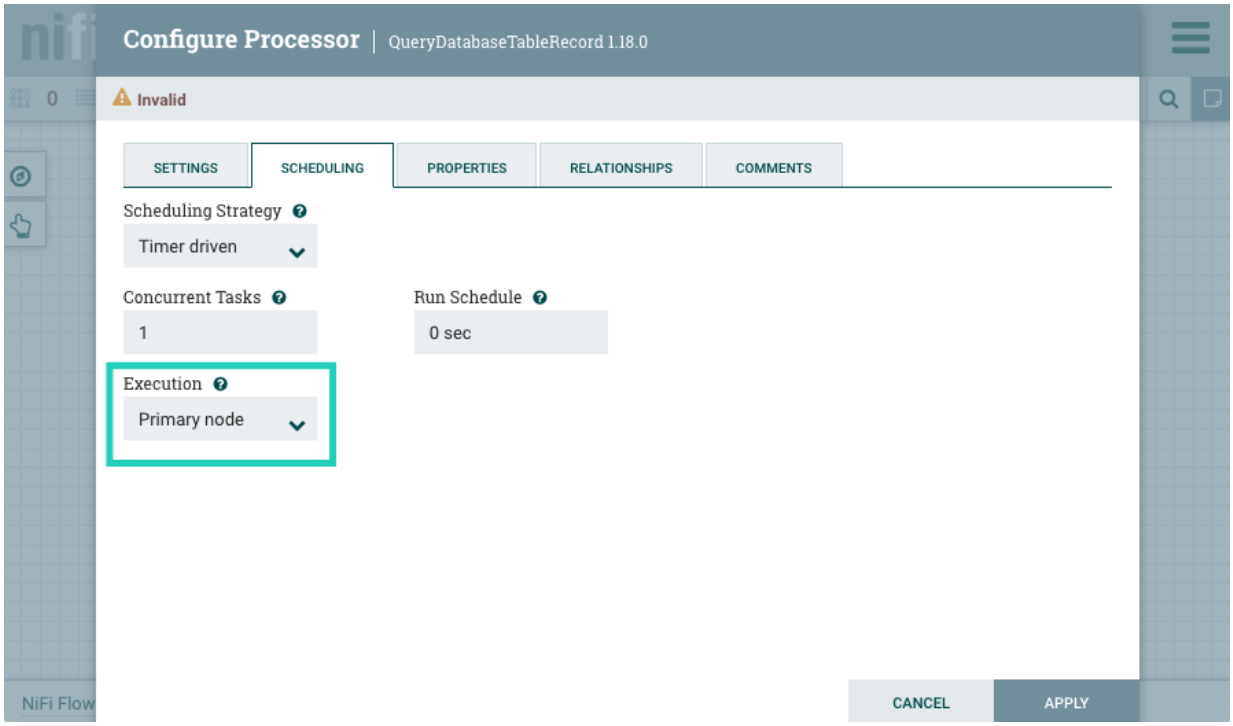
7. Reference the parameters from the processor and the controller services.





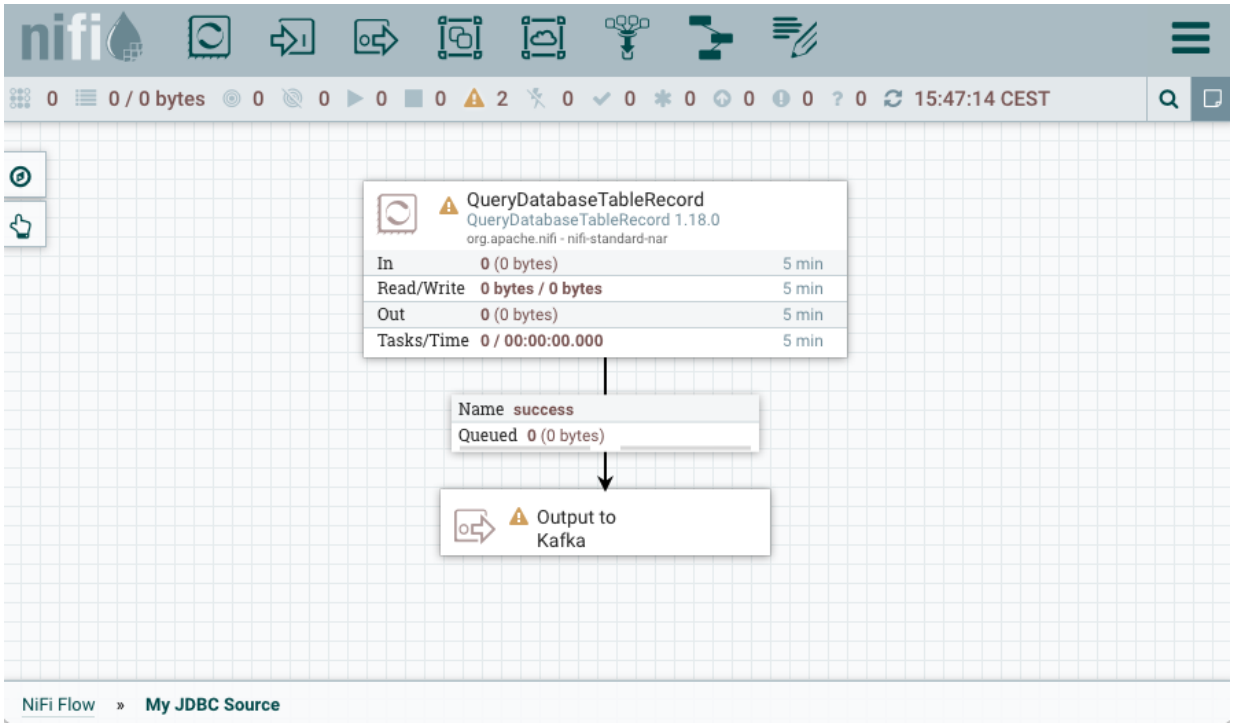
At this point, one controller service is invalid, and the other is disabled, but that is expected. Neither processors, nor controller services need to be valid, started, or enabled at this point.

8. Open the configuration of the QueryDatabaseTableRecord processor, and go to the SCHEDULING tab.



Notice that Execution is set to Primary node (which is the only option for this particular processor). For Stateless NiFi Kafka connectors, this setting means that the connector is going to be deployed with one task, regardless of the value of the tasks.max connector property.

9. At this point, your dataflow is in the following state.



Notice how the dataflow is not running, and there are warnings present. This is expected. The dataflow does not need to run in order for you to be able to download the flow definition. The properties you must configure to make the dataflow function will be set later on during connector deployment.

10. Right-click the process group and select Download flow definition Without external service .

If you followed these steps, your process group is not referencing controller services that are outside of the process group. As a result, you can choose to download the flow without external services included.

11. Open the downloaded JSON file, and replace **every occurrence** of the version number (1.18.0) with the version number of your Stateless NiFi plugin.



12. Access the Streams Messaging Manager UI.
13. Click **Connect** in the navigation sidebar.
14. Click the New Connector option.
15. Select StatelessNiFiSourceConnector from the available connectors.

    The Connector Configuration page appears.
16. Configure the following connector properties.

    - name=My Custom JDBC Connector
    - output.port=Output to Kafka

      This property must be set to the name of the output port in the dataflow. The output port name was specified in Step 2 on page 144.
    - topics=jdbc_example

      This is the name of the target topic that the connector writes data to.
17. Configure the flow.snapshot property.

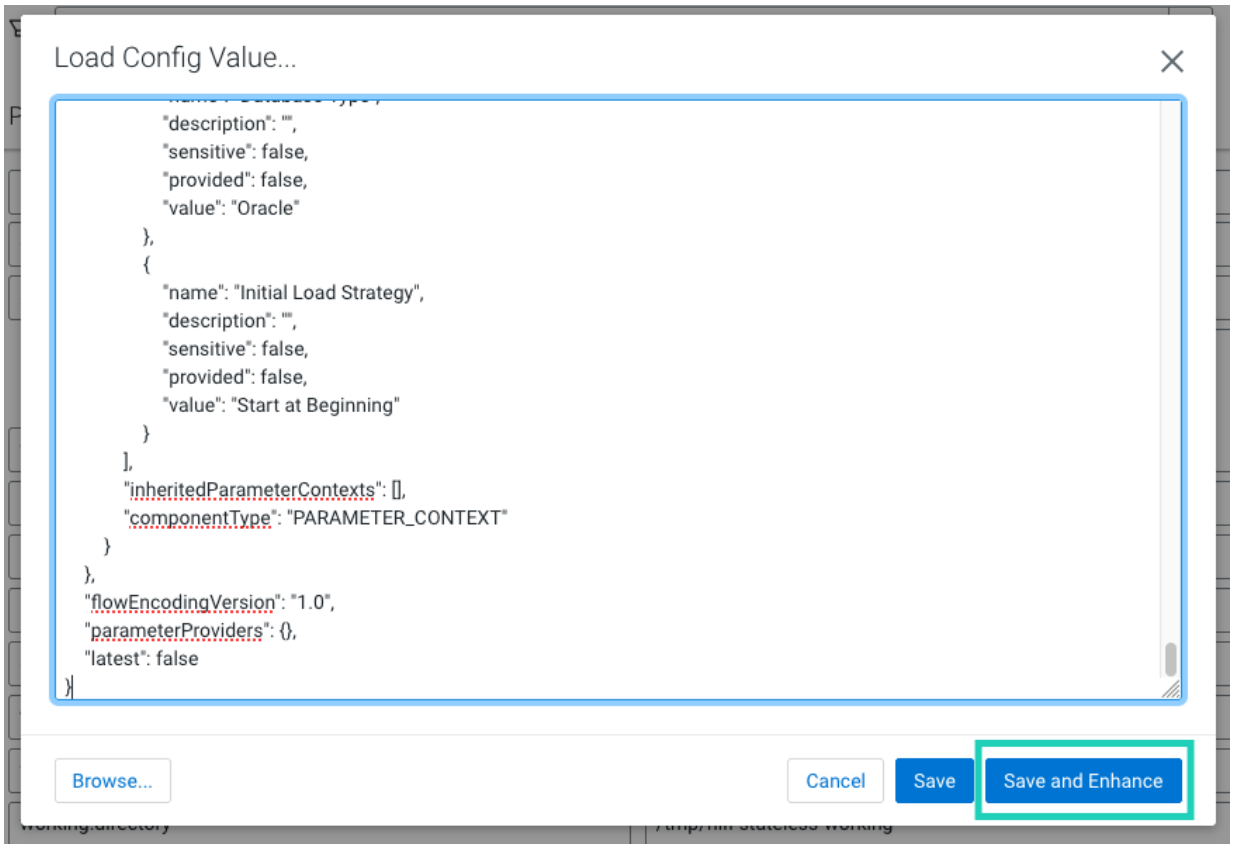
The flow.snapshot property specifies the dataflow that the connector runs. This is the dataflow that you developed and exported as flow definition JSON in the previous steps. While you have multiple options when configuring this property, this tutorial uses the Import and Enhance option in Streams Messaging Manager.

a) Click ⋮ found next to flow.snapshot, and then click Edit.

b) Paste the contents of the flow definition JSON, or click Browse... to upload the file from your machine.

c) Click Save and Enhance.

Save and Enhance is a unique Streams Messaging Manager feature only available for this property. If this option is used, Streams Messaging Manager parses the dataflow specified in flow.snapshot, extracts all parameters that are available in the dataflow, and adds them to connector configuration.

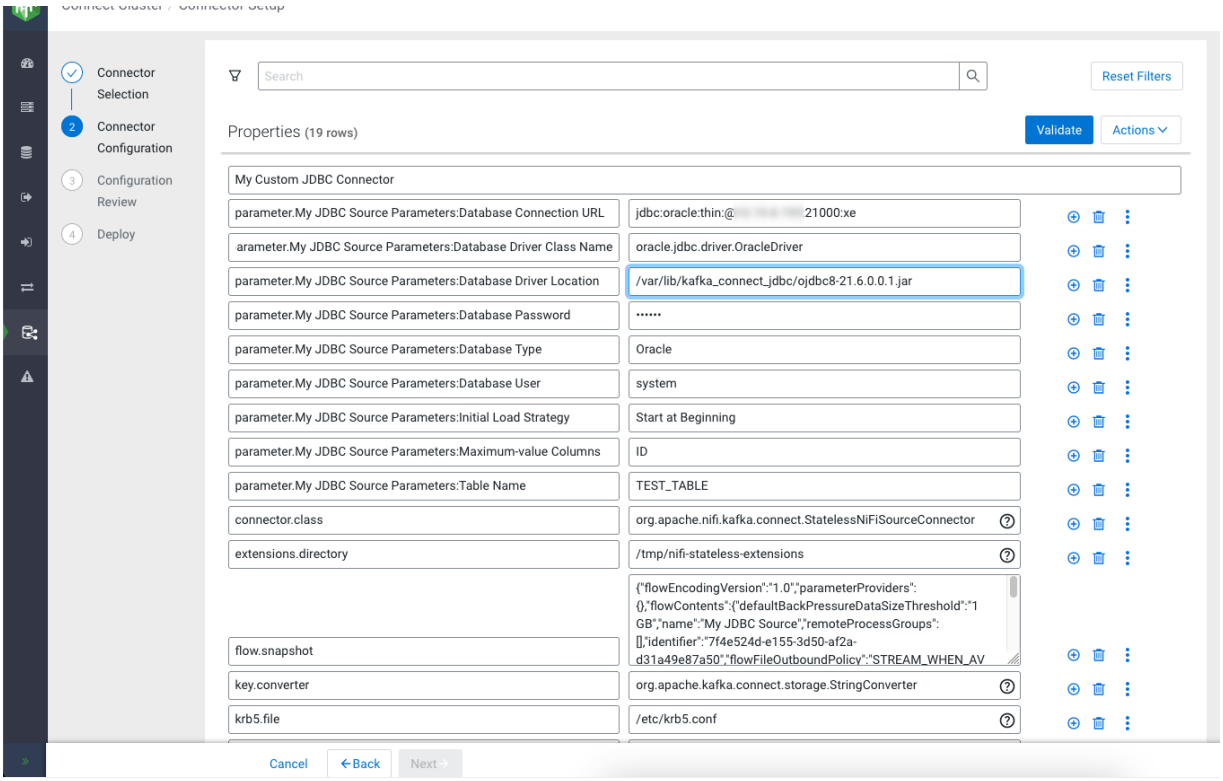


18. Configure the dataflow parameters.

These are the properties that have the parameter. prefix. In this specific example, values for Database Class N ame, Database Type, and Initial Load Strategy are loaded automatically because values were specified within the dataflow in Step 5 on page 146. If needed, you can change the values.

Set Database Driver Location to the path where you deployed the JDBC driver.

19. Review your configuration.



20. Click Validate.

21. Click Next.

22. Click Deploy to deploy the connector.

Connector deployment is not instantaneous. You might need to wait a few seconds for deployment to finish.

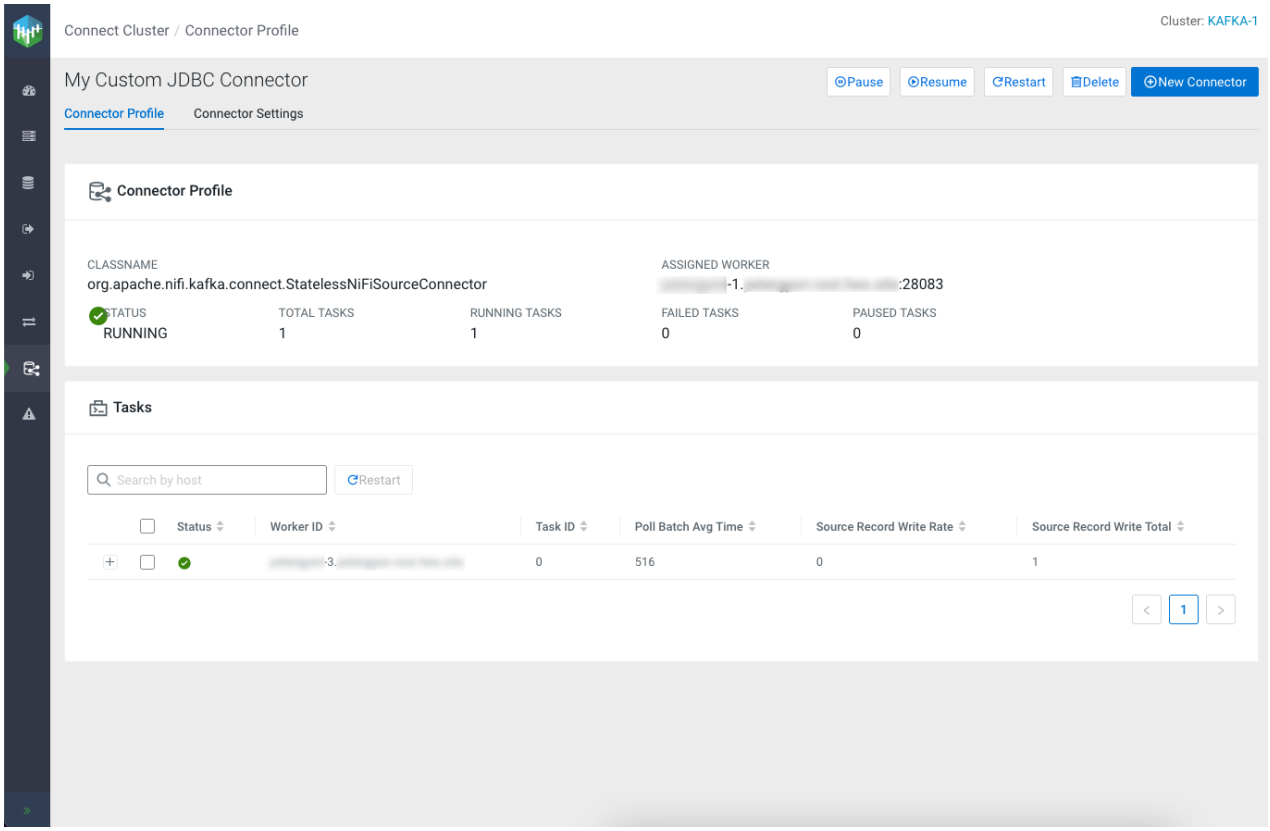
Results

Your custom developed flow is deployed in Kafka Connect and is running as a Kafka Connect connector.

What to do next

Once deployment finishes, navigate to the Connector Profile page of the connector. On this page you can view various details regarding the connector. Most importantly, you can view what the connector's status is, and which workers the connector and its task were assigned to.

Note: It might take some time for tasks to appear on this page.



In this particular case, the connector was assigned to the worker running on cluster host 1, whereas the task was assigned to worker running on cluster host 3. This means that dataflow log entries will be present in the Kafka Connect log file on host 3.

In addition to monitoring the connector, you can also check the contents of the topic that the connector is writing to. This can be done in Streams Messaging Manager by navigating to the Topics page, searching for the topic, and clicking . This opens the Streams Messaging Manager Data Explorer in a modal window. You can use the Data Explorer to sample the data that is flowing through the topic. The database table's records appear in the topic in JSON format.

Related Information

Kafka Connect tasks and Stateless NiFi

Syslog TCP Source connector

The Syslog TCP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The Syslog TCP Source connector listens on a port for syslog messages over TCP and transfers them to Kafka. The connector accepts messages in one of the following formats: Syslog 3164, Syslog 5424, or Grok. If the input messages are in Grok format, the connector can either derive the schema using the field names from the value of the Grok Expression property or read the schema from Schema Registry.

The connector can write messages into Kafka in one of the following formats: Avro, JSON, or text. If the output format is text, the raw message is transferred to Kafka. If the output format is JSON, the input message is processed and converted into JSON format before it is transferred to Kafka. If the output format is Avro, the input message is processed and converted into Avro format before it is transferred to Kafka. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on. The Kerberos keytab that is used to access Schema Registry must also be on the same cluster node that the connector runs on.

The connection to Schema Registry can be secured by TLS. The truststore file necessary for securing the connection must be on the same cluster node that the connector runs on. Mutual TLS for securing the communication between the message sender and the connector itself is also supported. The keystore and truststore files necessary for securing the connection must be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Source property reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the Syslog TCP Source properties reference.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- The Syslog TCP Source connector must use at least one-way SSL. It cannot be used without SSL.
- The Schema Registry URL property is mandatory even if Schema Registry is not used. If Schema Registry is not used, use the default value, or completely remove the property from the configuration.
- Schemas are only read from Schema Registry if Input Data Format is set to GROK and Schema Access Strategy is set to Schema Registry.
- If Output Format is AVRO, the schema of the records can be embedded in the output data. Whether the schema is embedded is determined by the Avro Schema Write Strategy property.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example

In this example, the connector uses mutual TLS to receive data in Syslog 3164 format which is then transferred to Kafka in JSON format.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "Syslog TCP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.Syslog TCP Source Parameters:Port": "[***PORT***]",
 "parameter.Syslog TCP Source Parameters:Input Data Format": "Syslog 3164",
 "parameter.Syslog TCP Source Parameters:Output Format": "JSON",
 "parameter.Syslog TCP Source Parameters:Client Authentication": "REQUIRED",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Filename": "[***THE
 FULLY-QUALIFIED FILENAME OF THE KEYSTORE***]",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Key Password":
 "[***KEYSTORE KEY PASSWORD***]",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Password":
 "[***KEYSTORE PASSWORD***]",
 "parameter.Syslog TCP Source Parameters:SSL Keystore Type": "[***KEYSTORE
 TYPE***]",
 "parameter.Syslog TCP Source Parameters:SSL Truststore Filename": "[***THE
 FULLY-QUALIFIED FILENAME OF THE TRUSTSTORE***]",
 "parameter.Syslog TCP Source Parameters:SSL Truststore Password":
 "[***TRUSTSTORE PASSWORD***]",
 "parameter.Syslog TCP Source Parameters:SSL Truststore Type":
 "[***TRUSTSTORE TYPE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Port

The port that the connector listens on for incoming messages.

Input Data Format

Determines what format incoming messages are expected in.

Output Format

Determines the format in which messages are transferred to Kafka.

Client Authentication

Determines if one-way or two-way SSL is used. In this example, this property is set to REQUIRED, meaning that two-way SSL is used.

Keystore *

These are the properties for accessing the keystore containing the keypair used for secure communication.

Truststore *

These are the parameters for accessing the truststore containing the message sender's certificate used for secure communication.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a

colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to / tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

Syslog TCP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Syslog TCP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source property reference.

Authorized Issuer DN Pattern

Description

A regular expression that can be applied against the Issuer's Distinguished Name of incoming TLS connections.

Default Value

.*

Accepted Values

Required

false

Authorized Subject DN Pattern

Description

A regular expression that can be applied against the Subject's Distinguished Name of incoming TLS connections.

Default Value

.*

Accepted Values

Required

false

Avro Schema Write Strategy

Description

Specifies how the schema is attached to the outgoing Avro messages. This property only takes effect if the Output Format is AVRO.

- If set to Embed Avro Schema then the schema is embedded in every output Avro message.
- If set to Do Not Write Schema then no schema information is attached to the output Avro messages.
- If set to HWX Content-Encoded Schema Reference then a reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Character Set

Description

The character set used in the input as well as the output data.

Default Value

UTF-8

Accepted Values

Required

true

Client Authentication

Description

The client authentication policy used for SSL.

Default Value

REQUIRED

Accepted Values

NONE, WANT, REQUIRED

Required

true

Date Format

Description

Specifies the format used for writing date fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Grok Expression

Description

Specifies the format of a line in Grok format. This allows the connector to understand how to parse each line in the input file. If a line in the file does not match this pattern, the line is handled according to the Grok No Match Behavior. A valid Grok expression must be specified using this property even if Grok format is not used.

Default Value

%{GREEDYDATA:message}

Accepted Values

Required

true

Grok No Match Behaviour

Description

Specifies how to handle lines that do not match the pattern set in the Grok Expression property.

- If set to append-to-previous-message, non-matching lines are appended to the last field of the previous message.
- If set to skip-line, non-matching lines are skipped. If set to raw-line, non-matching lines are only added to the _raw field.

Default Value

append-to-previous-message

Accepted Values

append-to-previous-message, skip-line, raw-line

Required

true

Input Data Format

Description

The format of incoming messages.

Default Value

Syslog 3164

Accepted Values

Syslog 3164, Syslog 5424, Grok

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Max Batch Size

Description

The maximum number of messages to add to a single batch of messages. If multiple messages are available, they are concatenated with a new line character up to this configured maximum number of messages.

Default Value

1

Accepted Values

Required

true

Max Number of Worker Threads

Description

he maximum number of worker threads available for handling TCP connections.

Default Value

2

Accepted Values

Required

true

Output Format

Description

The format of the messages written to Kafka.

Default Value

AVRO

Accepted Values

TEXT, AVRO, JSON

Required

true

Output Grouping for JSON

Description

Specifies how JSON objects are grouped in the connector output.

- If set to output-array, the output will consist of an array of JSON objects.
- If set to output-oneline, each line of the output data will be one JSON object. That is, each JSON object occupies one line in the output.

Default Value

output-oneline

Accepted Values

output-array, output-oneline

Required

true

Port

Description

The port to listen on for communication.

Type

int

Default Value

Accepted Values

Required

true

SSL Keystore Filename

Description

The fully-qualified filename of a keystore. This keystore is used to establish a secure connection between this connector and its clients using SSL.

Default Value

Accepted Values

Required

true

SSL Keystore Key Password

Description

The password used to access the key stored in the keystore file configured in SSL Keystore Fil ename.

Default Value

Accepted Values

Required

true

SSL Keystore Password

Description

The password used to access the contents of the keystore configured in the SSL Keystore Filen ame property.

Default Value

Accepted Values

Required

true

SSL Keystore Type

Description

The type of the keystore configured in the SSL Keystore Filename property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

SSL Truststore Filename

Description

The fully-qualified filename of a truststore. It can be used for establishing connections using mutual TLS. When using one-way SSL (Client Authentication parameter set to NONE), this parameter must be removed completely from the configuration.

Default Value

Accepted Values

Required

false

SSL Truststore Password

Description

The password used to access the contents of the truststore configured in the SSL Truststore Filen ame property. When using one-way SSL (Client Authentication parameter set to NONE), this parameter must be removed completely from the configuration.

Default Value

Accepted Values

Required

false

SSL Truststore Type

Description

The type of the truststore configured in the SSL Truststore Filename property. When using one-way SSL (Client Authentication parameter set to NONE), this parameter must be removed completely from the configuration.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the input message. This property only takes effect if the input data format is GROK.

- If set to Schema Registry then the schema is read from Schema Registry.

- If set to Field Names from Grok Expression then the schema is determined using the field names in the Grok Expression property.

Default Value

Schema Registry

Accepted Values

Schema Registry, Field Names From Grok Expression

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

- If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.
- If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V

ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format used for writing time fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format used for writing timestamp fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

Syslog UDP Source connector

The Syslog UDP Source connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The Syslog UDP Source connector listens on a port for syslog messages over UDP and transfers them to Kafka. The connector accepts messages in one of the following formats: Syslog 3164, Syslog 5424, or Grok. If the input messages are in Grok format, the connector can either derive the schema using the field names from the value of the Grok Expression property or read the schema from Schema Registry.

The connector can write messages into Kafka in one of the following formats: Avro, JSON, or text. If the output format is text, the raw message is transferred to Kafka. If the output format is JSON, the input message is processed and converted into JSON format before it is transferred to Kafka. If the output format is Avro, the input message is processed and converted into Avro format before it is transferred to Kafka. How the record schema is attached to these Avro messages depends on the value of the Avro Schema Write Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on.

The Kerberos keytab that is used to access Schema Registry must also be on the same cluster node that the connector runs on. The connection to Schema Registry can be secured by TLS. The truststore file necessary for securing the connection must also be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Source properties

These are the properties that are specific to the Stateless NiFi Source connector. All Stateless NiFi Source connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Source property reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the Syslog UDP Source properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- The Schema Registry URL property is mandatory even if Schema Registry is not used. If Schema Registry is not used, use the default value, or completely remove the property from the configuration.
- Schemas are only read from Schema Registry if Input Data Format is set to GROK and Schema Access Strategy is set to Schema Registry.
- If Output Format is AVRO, the schema of the records can be embedded in the output data. Whether the schema is embedded is determined by the Avro Schema Write Strategy property.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example

In this example, the connector receives data in Syslog 3164 format and transfers it to Kafka in JSON format.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSourceConn
ector",
 "meta.smm.predefined.flow.name": "Syslog UDP Source",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.Syslog UDP Source Parameters:Port": "[***PORT***]",
 "parameter.Syslog UDP Source Parameters:Input Data Format": "Syslog 3164",
```

```
 "parameter.Syslog UDP Source Parameters:Output Format": "JSON",
 "parameter.Syslog UDP Source Parameters:Output Grouping for JSON": "output-
oneline"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic that the connector sends messages to.

Port

The port that the connector listens on for incoming messages.

Input Data Format

Determines what format incoming messages are expected in.

Output Format

Determines the format in which messages are transferred to Kafka.

Related Information

Apache Kafka documentation

Stateless NiFi Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Source connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

header.attribute.regex

Description

A Java regular expression that is evaluated against all flowfile attribute names. Any attribute name matching the regular expression is converted into a Kafka message header. The name of the attribute is used as the header key, the value of the attribute is used as the header value. If not specified, headers are not added to the Kafka record.

Default Value

Accepted Values

Required

false

header.name.regex

Description

A Java regular expression that will be evaluated against all flowfile attribute names. For any attribute whose name matches the regular expression, the Kafka record will have a header whose name matches the attribute name and whose value matches the attribute value. If not specified, the Kafka record will have no headers added to it.

Default Value

Accepted Values

Required

false

key.attribute

Description

Specifies the name of a flowfile attribute that should be used to specify the key of the Kafka record. If not specified, the Kafka record will not have a key associated with it. If specified, but the attribute does not exist on a particular flowfile, it will also have no key associated with it.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

True

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

output.port

Description

The name of the output port in the NiFi dataflow to pull data from. If the dataflow contains exactly one port, this property is optional and can be omitted. However, if the dataflow contains multiple ports (for example, a success and a failure port), this property must be specified. If any flowfile is sent to any port other than the specified port, it is considered as a failure. The session is rolled back and no data is collected.

Default Value

Accepted Values

Required

false

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific Parameter Contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the parameter context named My Context.

Default Value

Accepted Values

Required

false

topic.name.attribute

Description

Specifies the name of a flowfile attribute to use for determining which Kafka topic a flowfile is sent to. Either the topics or topic.name.attribute property must be specified. If both are specified, topi c.name.attribute takes precedence. However, if a flowfile does not have the specified attribute name, then the connector falls back to using the topics property.

Default Value

Accepted Values

Required

false

topics

Description

The name of the topic to deliver data to. All flowfiles are delivered to the topic specified here. However, it is also possible to determine the topic individually for each flowfile. To do this, ensure that the dataflow specifies the topic name in an attribute, and then use topic.name.attribute to specify the name of the attribute instead of topic name. For example, if you wanted a separate Kafka topic for each data source, you can omit the topics property and instead specify the attribute (for example, datasource.hostname) corresponding to the topic using the topic.name.attribute property.

Default Value

Accepted Values

Required

true

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

Syslog UDP Source properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Syslog UDP Source connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Source connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Source property reference.

Avro Schema Write Strategy

Description

Specifies how the schema is attached to the outgoing Avro messages. This property only takes effect if the Output Format is AVRO.

- If set to Embed Avro Schema then the schema is embedded in every output Avro message.
- If set to Do Not Write Schema then no schema information is attached to the output Avro messages.
- If set to HWX Content-Encoded Schema Reference then a reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Character Set

Description

The character set used in the input as well as the output data.

Default Value

UTF-8

Accepted Values

Required

true

Date Format

Description

Specifies the format used for writing date fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Grok Expression

Description

Specifies the format of a line in Grok format. This allows the connector to understand how to parse each line in the input file. If a line in the file does not match this pattern, the line is handled according to what is set in the Grok No Match Behavior property. A valid Grok expression must be specified using this property even if Grok format is not used.

Default Value

%{GREEDYDATA:message}

Accepted Values

Required

true

Grok No Match Behaviour

Description

Specifies how to handle lines that do not match the pattern set in the Grok Expression property.

- If set to append-to-previous-message, non-matching lines are appended to the last field of the previous message.
- If set to skip-line, non-matching lines are skipped. If set to raw-line, non-matching lines are only added to the _raw field.

Default Value

append-to-previous-message

Accepted Values

append-to-previous-message, skip-line, raw-line

Required

true

Input Data Format

Description

The format of incoming messages.

Default Value

Syslog 3164

Accepted Values

Syslog 3164, Syslog 5424, Grok

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Output Format

Description

The format of the messages written to Kafka.

Default Value

AVRO

Accepted Values

TEXT, AVRO, JSON

Required

true

Output Grouping for JSON

Description

Specifies how JSON objects are grouped in the connector output.

- If set to output-array, the output will consist of an array of JSON objects.
- If set to output-oneline, each line of the output data becomes one JSON object. That is, each JSON object occupies one line in the output.

Default Value

output-oneline

Accepted Values

output-array, output-oneline

Required

true

Port

Description

The port to listen on for communication.

Default Value

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the input message. This property only takes effect if the input data format is GROK.

- If set to Schema Registry then the schema is read from Schema Registry.
- If set to Field Names from Grok Expression then the schema is determined using the field names in the Grok Expression property.

Default Value

Schema Registry

Accepted Values

Schema Registry, Field Names From Grok Expression

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

- If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.
- If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format used for writing time fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format used for writing timestamp fields if the Output Format is JSON. Otherwise this parameter is not used.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

ADLS Sink connector

The ADLS Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka connect framework. Learn about the connector, its properties, and configuration.

The ADLS Sink connector fetches messages from Kafka and uploads them to ADLS. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages can contain unstructured (character or binary) data or they can be in Avro or JSON format.

If the input is unstructured data, record processing is disabled. In a case like this, multiple messages can be concatenated into a single output file on ADLS using a demarcator (for example, newline for text messages). Merging is optional, large binary data can be forwarded to Azure at a 1:1 ratio, one Kafka message equals a single ADLS file.

If the input is either Avro or JSON, record processing is enabled. In a case like this, the schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The output can be Avro, JSON or CSV. Multiple records are typically merged into a single output file before uploading to ADLS.

The connector can authenticate to Azure using an Account Key, SAS Token, Service Principal, or a Managed Identity.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Sink properties reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the ADLS Sink properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.

- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example for fetching unstructured data

In this example, the connector fetches unstructured messages containing single line character data from Kafka. The connector concatenates the messages using newline characters as the demarcator. The files that the connector creates and uploads to ADLS are maximum 10 MB in size. The files are named according to the following pattern: mess ages_*[***UUID***]*. The connector authenticates to Azure using an Account Key.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "ADLS Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutAzureDataLakeStorage Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Account Name": "[***ACCOUNT NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Account Key": "[***ACCOUNT KEY***]",
 "parameter.ADLS Sink Parameters:ADLS Filesystem Name": "[***FILESYSTEM
 NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Directory Name": "[***DIRECTORY
 NAME***]",
 "parameter.ADLS Sink Parameters:Maximum File Size": "10 MB",
 "parameter.ADLS Sink Parameters:Kafka Message Data Format": "Raw",
 "parameter.ADLS Sink Parameters:Output File Demarcator": "\n",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "messages_${file
name.uuid}"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

ADLS Account Name

The Storage Account Name to use for authentication to Azure. This is also the target account where the output file is uploaded.

ADLS Account Key

The Storage Account Key to use for authentication to Azure.

ADLS Filesystem Name

The filesystem (or container) in the Storage Account to upload data to.

ADLS Directory Name

The target directory in the filesystem.

Maximum File Size

The maximum size of the output data file. In this example, the maximum size is 10 MB.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Raw. This means that the connector expects unstructured text or binary data.

Output File Demarcator

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages. In this example, the property is set to \n (newline). This means that the newline character is used to separate the single line character data of the Kafka messages.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid}.

messages_ is a string literal. ${filename.uuid} is an expression that inserts a generated UUID in the filename. As a result, the files are named according to the following pattern: mess ages_*[***UUID***]*.

Configuration example for fetching JSON data

In this example, the connector fetches messages in JSON format from Kafka. The connector parses the JSON records and converts them to Avro format using the schema inferred from the JSON data. The schema is also embedded in the output file. The connector merges multiple records (all messages that are available for a single execution of the connector) into an Avro file, generates a name for the file using the following pattern data_*[***TIMESTAMP***]_[***SEQUENCE***]*.avro, and uploads the file to ADLS. The connector authenticates to Azure using a Service Principal.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "ADLS Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutAzureDataLakeStorage Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Account Name": "[***ACCOUNT NAME***]",
 "parameter.ADLS Sink Parameters:Azure Service Principal Tenant ID":
 "[***SERVICE PRINCIPAL TENANT ID***]",
 "parameter.ADLS Sink Parameters:Azure Service Principal Client ID":
 "[***SERVICE PRINCIPAL CLIENT ID***]",
 "parameter.ADLS Sink Parameters:Azure Service Principal Client Secret":
 "[***SERVICE PRINCIPAL CLIENT SECRET***]",
 "parameter.ADLS Sink Parameters:ADLS Filesystem Name": "[***FILESYSTEM
 NAME***]",
 "parameter.ADLS Sink Parameters:ADLS Directory Name": "[***DIRECTORY
 NAME***]",
 "parameter.ADLS Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.ADLS Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.ADLS Sink Parameters:Avro Schema Write Strategy": "Embed Avro
Schema",
 "parameter.ADLS Sink Parameters:Output File Data Format": "Avro",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "data_${filename.
timestamp}_${filename.sequence}.avro",
```

```
 "parameter.ADLS Sink Parameters:Output Filename Timestamp Format": "yyyy
MMdd_HHmmss_SSS"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

ADLS Account Name

The Storage Account Name to use for authentication to Azure. This is also the target account where the output file is uploaded.

Service Principal *

The properties of the Service Principal to use for authentication to Azure.

ADLS Filesystem Name

The filesystem (or container) in the Storage Account to upload data to.

ADLS Directory Name

The target directory in the filesystem.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects JSON data.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Infer Schema, meaning that the schema is determined (inferred) from the JSON data.

Avro Schema Write Strategy

Specifies whether the record schema is written to the output data file. In this example, this property is set to Embed Avro Schema, meaning that the schema is embedded in the output Avro file.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to Avro, meaning that the output file format is Avro.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to data_${filename.timestamp}_${filename.sequence}.avro.

data_, the underscore (_), and .avro are string literals. ${filename.timestamp} and ${filename.sequence} are expressions. ${filename.timestamp} inserts the current timestamp in the filename. ${filename.sequence} inserts an incrementing sequence value in the filename. As a result, the files are named according to the following pattern: data*_[***TIMESTAMP***]_[***SEQUENCE***]*.avro.

Output Filename Timestamp Format

Specifies the timestamp format used in the ${filename.timestamp} expression. The expression is used when generating the output filename.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

ADLS Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the ADLS Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

ADLS Account Key

Description

The Storage Account Key to use for authentication to Azure. Required when Account Key authentication is used.

Default Value

Accepted Values

Required

false

ADLS Account Name

Description

The name of the Azure Storage Account to use for authentication to Azure and the target account where the output file is uploaded.

Default Value

Accepted Values

Required

true

ADLS Directory Name

Description

The name of the directory where the output file is uploaded. Use an empty string to specify the root directory.

Default Value

Accepted Values

Required

true

ADLS Endpoint Suffix

Description

Specifies the endpoint suffix used for accessing the ADLS service.

Default Value

dfs.core.windows.net

Accepted Values

Required

true

ADLS Filesystem Name

Description

The name of the filesystem (or container) in the Azure Storage Account where the output file is uploaded.

Default Value

Accepted Values

Required

true

ADLS SAS Token

Description

The SAS Token to use for authentication to Azure. Required when SAS Token authentication is used.

Default Value

Accepted Values

Required

false

Avro Schema Write Strategy

Description

Specifies how the record schema is attached to the output data file. Applicable only for Avro output (Output File Data Format is set to Avro).

Do Not Write Schema

Neither the schema nor reference to the schema is attached to the output Avro messages.

Embed Avro Schema

The schema is embedded in every output Avro message.

HWX Content-Encoded Schema Reference

A reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Azure Service Principal Client ID

Description

The Client ID of the Service Principal to use for authentication to Azure. Required when Service Principal authentication is used.

Default Value

Accepted Values

Required

false

Azure Service Principal Client Secret

Description

The Client Secret of the Service Principal to use for authentication to Azure. Required when Service Principal authentication is used.

Default Value

Accepted Values

Required

false

Azure Service Principal Tenant ID

Description

The Tenant ID of the Service Principal to use for authentication to Azure. Required when Service Principal authentication is used.

Default Value

Accepted Values

Required

false

Date Format

Description

Specifies the format to use when writing date fields to JSON or CSV.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If set to Avro or JSON, record processing is enabled. Raw can be used for unstructured text or binary data.

Default Value

Avro

Accepted Values

Avro, JSON, Raw

Required

true

Kerberos Keytab

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum File Size

Description

The maximum size of the output data file. No size limit is applied if this property is not specified. Example values: 100 MB, 1 GB.

Default Value

Accepted Values

Required

false

Maximum Number of Entries

Description

The maximum number of entries in the output data file. In the context of this property, entry can mean one of two things. If record processing is enabled (Output File Data Format is set to Avro or JSON), an entry is a record. Otherwise, entry means a Kafka message. Set this property to 1 if you want to create a separate output file for each Kafka message.

Default Value

1000000

Accepted Values

Required

true

Output File Data Format

Description

Specifies the format of the records written to the output file. Required when record processing is enabled (Kafka Message Data Format is set to Avro or JSON).

Default Value

Avro

Accepted Values

Avro, JSON, CSV

Required

false

Output File Demarcator

Description

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages (no record processing). This property can only be used if Kafka Message Data Format is set to Raw. If you want to use newline as the delimiter, set this property to \n.

Default Value

Accepted Values

Required

false

Output Filename Pattern

Description

Specifies the structure of the name of the output file. The pattern can contain string literal (fixed text) parts and one or more of the following expressions:

- ${filename.uuid}: Inserts a generated UUID in the filename.

- ${filename.timestamp}: Inserts the current timestamp in the filename.
- ${filename.sequence}: Inserts an incrementing sequence value in the filename.

In order to generate unique filenames, either ${filename.uuid} or ${filename.sequence} must be used in the pattern.

Examples:

- data_${filename.uuid}.json
- records_${filename.timestamp}_${filename.sequence}.avro

Default Value

${filename.uuid}

Accepted Values

Required

false

Output Filename Sequence Initial Value

Description

This property is used to configure the initial value of the ${filename.sequence} expression. The value you set in this property is not the initial value of the sequence. The sequence starts at the value of this property +1. For example, if you set this property to 0, the sequence starts at 1.

Default Value

0

Accepted Values

Required

false

Output Filename Sequence Padding Length

Description

Specifies the length of the ${filename.sequence} expression in characters. If the sequence has fewer characters than the value set in this property, it is padded with zeros (0). Padding is added to the left of the sequence.

Default Value

6

Accepted Values

Required

false

Output Filename Timestamp Format

Description

The timestamp format to use for the ${filename.timestamp} expression. For example, yyyyMMdd _HHmmss_SSS.

Default Value

Accepted Values

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Kafka Message Data Format is set to Raw).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when writing Time fields to JSON or CSV.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when writing Timestamp fields to JSON or CSV.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Use Azure Managed Identity

Description

Turns on authentication using Managed Identity.

Default Value

false

Accepted Values

true, false

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

Amazon S3 Sink

Learn more about the Amazon S3 Sink connector

The Amazon S3 Sink connector allows users to stream Kafka data into S3 buckets.

Configuration example

A simple configuration example for the Amazon S3 Sink connector.

The following is a simple configuration example for the Amazon S3 Sink connector. Short descriptions of the properties set in this example are also provided. For a full properties reference, see the Amazon S3 Sink properties reference.

```
{
    "aws.s3.bucket": "bring-me-the-bucket",
    "aws.s3.service_endpoint": "http://myendpoint:9090/",
    "aws.access_key_id": "EXAMPLEID",
    "aws.secret_access_key": “EXAMPLEKEY",
    "connector.class": "com.cloudera.dim.kafka.connect.s3.S3SinkConnector",
    "tasks.max": 1,
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConver
ter",
    "value.converter.passthrough.enabled": true,
    "value.converter.schema.registry.url": "http://schema-registry:9090/api/
v1",
    "topics": "avro_topic",
    "output.storage": "com.cloudera.dim.kafka.connect.s3.S3PartitionStorage"
,
    "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.avr
o.AvroPartitionWriter",
    "output.avro.passthrough.enabled": true
  }
```

aws.s3.bucket

Target S3 bucket name.

aws.s3.service_endpoint

Target S3 host and port.

aws.access_key_id

The AWS secret key ID used for authentication.

aws.secret_access_key

The AWS secret access key used for authentication.

connector.class

Class name of the Amazon S3 Sink connector.

tasks.max

Maximum number of tasks.

key.converter

The converter capable of understanding the data format of the key of each record on this topic.

value.converter

The converter capable of understanding the data format of the value of each record on this topic.

Note: When the AvroConverter is used, you can specify Schema Registry properties to be used by the AvroConverter's Schema Registry client. This is done by adding the required Schema Registry property as a suffix to the value.converter property. For example, value.converter.schema.registry.url. Properties defined this way are passed on to the Schema Registry client used by the AvroConverter.

value.converter.passthrough.enabled

This property controls whether or not data is converted into the Kafka Connect intermediate data format before writing into an output file. Because in this example the input and output format is the same, the property is set to true, that is, data is not converted.

value.converter.schema.registry.url

The URL to Schema Registry. This is a mandatory property if the topic has records encoded in Avro format.

topics

List of topics to consume data from.

output.storage

The S3 storage implementation class.

output.writer

Determines the output file format. Because in this example the output format is Avro, AvroPartitio nWriter is used.

output.avro.passthrough.enabled

This property has to match the configuration of the value.converter.passthrough.enabled property because both the input and output formats are Avro.

Amazon S3 Sink properties reference

Amazon S3 Sink connector properties reference.

The following table collects connector properties that are specific for the Amazon S3 Sink Connector. For properties common to all sink connectors, see the upstream Apache Kafka documentation.

Property Name	Description	Type	Default Value	Accepted Values	Recommended Value
aws.s3.bucket	The target S3 bucket name.	String	none	Any valid S3 bucket name.	
aws.s3.service_endpoint	The target S3 host and port.	String	none	Any valid S3 endpoint.	
aws.access_key_id	The AWS secret key ID to authenticate.	String	none	Any valid secret key issued by AWS.	
aws.secret_access_key	The AWS secret access key to authenticate.	String	none	Any valid access key issued by AWS.	
value.converter	Value conversion class.	String	none		com.cloudera.dim.kafka.connect.converts
value.converter.passthrough.enabled	Configures whether the AvroConverter translates an Avro record into Kafka Connect Data or transparently passes the Avro encoded bytes as payload.	Boolean	true	true, false	True if input and output are both Avro.

Property Name	Description	Type	Default Value	Accepted Values	Recommended Value
value.converter.schema.registry.url	The URL to the Schema Registry server.	String	none		
output.storage	The S3 storage implementation class.	String	none		com.cloudera.dim.kafka.connect.s3.S3Pa
output.writer	The output file writer which determines the type of file to be written. The value of this property should be the FQCN of a class that implements the PartitionWriter interface.	String	none	• com.cloudera.dim.kafka.connect.partition.writers.avro.AvroP • com.cloudera.dim.kafka.connect.partition.writers.json.JsonPa • com.cloudera.dim.kafka.connect.hdfs.parquet.ParquetPartitio • com.cloudera.dim.kafka.connect.partition.writers.txt.TxtParti	com.cloudera.dim.kafka.connect.partitio
output.avro.passthrough.enabled	Configures Whether the output writer expects an Avro encoded Kafka Connect data record. Must match the configuration ofvalue.converter.passthrough.enabled.	Boolean	none	true, false	True if input and output are both Avro.
cloudera.offset.flush.interval.ms	Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.	Long	Inherits role-level offset.flush.interval.ms when not set		

HDFS Sink

Learn more about the HDFS Sink connector.

The HDFS Sink connector can be used to transfer data from Kafka topics to files on HDFS clusters. Each partition of every topic results in a collection of files named in the following pattern:

```
{topic name}_{partition number}_{end_offset}.{file extension}
```

For example, running the HDFS Sink connector on partition 0 of a topic named sourceTopic can yield the following series of files:

```
sourceTopic_0_50.avro - for record 0 ~ 50
sourceTopic_0_79.avro - holding record 51 ~ 79
...
```

The HDFS Sink connector periodically commits records to final result files. Each commit results in a separate "chunk" file.

Configuration example for writing data to HDFS

A simple configuration example for the HDFS Sink connector.

The following is a simple configuration example for the HDFS Sink connector. Short descriptions of the properties set in this example are also provided. For a full properties reference, see the HDFS Sink properties reference.

```
{
    "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnec
tor",
    "tasks.max": 1,
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConve
rter",
    "value.converter.passthrough.enabled": true,
    "value.converter.schema.registry.url": "http://localhost:9090/api/v1",
    "topics": "avro_topic",
    "hdfs.uri": "hdfs://my-host.my-realm.com:8020",
    "hdfs.output": "/topics_output/",
    "output.writer": "com.cloudera.dim.kafka.connect.partition.writers.avro
.AvroPartitionWriter",
    "output.avro.passthrough.enabled": true,
    "hdfs.kerberos.authentication": true,
    "hdfs.kerberos.user.principal": "user_account@MY-REALM.COM",
    "hdfs.kerberos.keytab.path": "/path/to/user_account.keytab",
    "hdfs.kerberos.namenode.principal": "hdfs/_HOST@MY-REALM.COM",
    "hadoop.conf.path": "/etc/hadoop/"
  }
```

connector.class

Class name of the HDFS Sink connector.

key.converter

The converter capable of understanding the data format of the key of each record on this topic.

value.converter

The converter capable of understanding the data format of the value of each record on this topic.



Note: When the AvroConverter is used, you can specify Schema Registry properties to be used by the AvroConverter's Schema Registry client. This is done by adding the required Schema Registry property as a suffix to the value.converter property. For example, value.converter.schema.registry.url. Properties defined this way are passed on to the Schema Registry client used by the AvroConverter.

value.converter.passthrough.enabled

This property controls whether or not data is converted into the Kafka Connect intermediate data format before writing into an output file. Because in this example the input and output format is the same, the property is set to true, that is, data is not converted.

value.converter.schema.registry.url

The URL to Schema Registry. This is a mandatory property if the topic has records encoded in Avro format.

topics

List of topics to consume data from.

hdfs.uri

The URI to the namenode of the HDFS cluster.

hdfs.output

The destination folder on the HDFS cluster where output files will reside.

output.writer

Determines the output file format. Because in this example the output format is Avro, AvroPartitio nWriter is used.

output.avro.passthrough.enabled

This property has to match the configuration of the value.converter.passthrough.enabled property because both the input and output formats are Avro.

hdfs.kerberos.authentication

Enables or disables kerberos authentication.

hdfs.kerberos.user.principal

The user principal that the Kafka Connect role will use.

hdfs.kerberos.keytab.path

The path to the kerberos keytab file.

hdfs.kerberos.namenode.principal

The Kerberos principal used by the namenode. This is necessary when the HDFS cluster has data encryption turned on.

hadoop.conf.path

The path to the hadoop configuration files. This is necessary when the HDFS cluster has data encryption turned on.

Configuration example for writing data to Ozone FS

A simple configuration example for the HDFS Sink connector that writes data to the Ozone FS.

The following is a simple configuration example for the HDFS Sink connector. In this example data is written to the Ozone FS. Short descriptions of the properties set in this example are also provided. For a full properties reference, see the HDFS Sink properties reference.

```
{
    "connector.class": "com.cloudera.dim.kafka.connect.hdfs.HdfsSinkConnec
tor",
    "tasks.max": 1,
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "com.cloudera.dim.kafka.connect.converts.AvroConve
rter",
    "value.converter.passthrough.enabled": true,
    "value.converter.schema.registry.url": "http://localhost:9090/api/v1",
    "topics": "avro_topic",
    "hdfs.uri": "ofs://ozone1/volume1/bucket1/",
    "hdfs.output": "/topics_output/",
    "output.writer": "com.cloudera.dim.kafka.connect.hdfs.avro.AvroPartiti
onWriter",
    "output.avro.passthrough.enabled": true,
    "hdfs.kerberos.authentication": true,
    "hdfs.kerberos.user.principal": "user_account@MY-REALM.COM",
    "hdfs.kerberos.keytab.path": "/path/to/user_account.keytab",
    "hadoop.conf.path": "/etc/hadoop/"
  }
```

connector.class

Class name of the HDFS Sink connector.

key.converter

The converter capable of understanding the data format of the key of each record on this topic.

value.converter

The converter capable of understanding the data format of the value of each record on this topic.



Note: When the AvroConverter is used, you can specify Schema Registry properties to be used by the AvroConverter's Schema Registry client. This is done by adding the required Schema Registry property as a suffix to the value.converter property. For example, value.converter.schema.registry.url. Properties defined this way are passed on to the Schema Registry client used by the AvroConverter.

value.converter.passthrough.enabled

This property controls whether or not data is converted into the Kafka Connect intermediate data format before writing into an output file. Because in this example the input and output format is the same, the property is set to true, that is, data is not converted.

value.converter.schema.registry.url

The URL to Schema Registry. This is a mandatory property if the topic has records encoded in Avro format.

topics

List of topics to consume data from.

hdfs.uri

The Ozone FS (ofs) URI.

hdfs.output

The destination folder on the HDFS cluster where output files will reside.

output.writer

Determines the output file format. Because in this example the output format is Avro, AvroPartitio nWriter is used.

output.avro.passthrough.enabled

This property has to match the configuration of the value.converter.passthrough.enabled property because both the input and output formats are Avro.

hdfs.kerberos.authentication

Enables or disables kerberos authentication.

hdfs.kerberos.user.principal

The user principal that the Kafka Connect role will use.

hdfs.kerberos.keytab.path

The path to the kerberos keytab file.

hadoop.conf.path

The path to the hadoop configuration files. This is necessary when the HDFS cluster has data encryption turned on.

HDFS Sink properties reference

HDFS Sink connector properties reference.

The following table collects connector properties that are specific for the HDFS Sink Connector. For properties common to all sink connectors, see the upstream Apache Kafka documentation.

Property Name	Description	Type	Default Value	Accepted Values	Recommended Values
hdfs.uri	The file system URI to connect to on the destination cluster. This property supports any valid Hadoop-compatible filesystem (HCFS, For example, HDFS or ofs) URI.	String	None		
hdfs.output	The root directory on the HDFS cluster where all the output files will reside. The sub path has the following pattern: {topic}/{topic}_{partition}_{endoffset}.{file extension}	String	/tmp		Any path on the HDFS file system where the role has read write permission.
hdfs.kerberos.authentication	Enables or disables secure access to the HDFS cluster by authenticating with Kerberos.	Boolean	false	true or false	
hdfs.kerberos.user.principal	The kerberos user principal.	String	null	The host-dependent Kerberos principal assigned to the Kafka Connect role.	
hdfs.kerberos.keytab.path	The path to the Kerberos keytab file.	String	null		In a Cloudera Manager provisioned environment, it's recommended to use the Cloudera Manager Config Provider to automatically provision the path.
hdfs.kerberos.namenode.principal	The kerberos name node principal. Required when the HDFS cluster has data encryption on.	String	null		
hadoop.conf.path	The path to the site specific Hadoop configuration XML files. Required when the HDFS cluster has data encryption on.	String	null		
output.writer	The output file writer which determines the type of file to be written to the HDFS cluster. The value of this property should be the FQCN of a class that implements the PartitionWriter interface.	String	null	• com.cloudera.dim.kafka.connect.partition.writers.avro.AvroP • com.cloudera.dim.kafka.connect.partition.writers.json.JsonPa • com.cloudera.dim.kafka.connect.hdfs.parquet.ParquetPartitio • com.cloudera.dim.kafka.connect.partition.writers.txt.TxtParti	com.cloudera.dim.kafka.connect.partitio

Property Name	Description	Type	Default Value	Accepted Values	Recommended Values
output.avro.passthrough.enabled	Configures whether the output writer expects an Avro encoded Kafka Connect data record. Must match the configuration of value.converter.passthrough.enabled.	Boolean	true	true or false	True if input and output are both Avro.
value.converter	The converter to be used to translate the value field of the source Kafka record into Kafka Connect Data format.	String	Inherited from Kafka Connect worker properties.	• org.apache.kafka.connect.json.JsonConverter • org.apache.kafka.connect.storage.StringConverter • com.cloudera.dim.kafka.connect.converts.AvroConverter	com.cloudera.dim.kafka.connect.converts
value.converter.schema.registry.url	The URL to the Schema Registry server.	String	null	true or false	
value.converter.passthrough.enabled	Configures whether the AvroConverter translates an Avro record into Kafka Connect Data or transparently passes the Avro encoded bytes as payload.	Boolean	true	true or false	True if input and output are both Avro.
cloudera.offset.flush.interval.ms	Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.	Long	Inherits role-level offset.flush.interval.ms when not set		

HDFS Stateless Sink connector

The HDFS Stateless Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. Learn about the connector, its properties, and configuration.

The HDFS Stateless Sink Connector fetches messages from Kafka and stores them in HDFS. The topic this connector receives messages from is determined by the value of the topics parameter in the configuration. The messages can contain unstructured data (character or binary) or they can be in Avro or JSON format.

If the input is unstructured data, record processing is disabled. Multiple messages can be concatenated into a single output file on HDFS using a demarcator (for example newline for text messages). Merging is optional, large binary data can be forwarded to HDFS at a 1:1 ratio, one Kafka message equals a single HDFS file. However, in case of large files, this usage is encouraged only to avoid small file problems in HDFS.

If the input is either Avro or JSON, record processing is enabled. In a case like this, the schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The output can be Avro, JSON, CSV, or Parquet. Multiple records are typically merged into a single output file to store in HDFS.

The connector supports connecting to Kerberos enabled HDFS and Schema Registry services.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Sink properties reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the HDFS Stateless Sink properties.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- If HDFS is running on a different cluster than the connector, the HDFS configuration resource files must be copied to the Kafka Connect node that the connector runs on.

Configuration example for unstructured data

In this example, the connector fetches unstructured messages from Kafka. The connector concatenates the messages using newline characters as the demarcator. The files that the connector creates and stores in HDFS are maximum 10 MB in size. The files are named according to the following pattern: messages_*[***UUID***]*.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
"connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnect
or",
"meta.smm.predefined.flow.name": "HDFS Sink",
"meta.smm.predefined.flow.version": "1.0.0",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
"tasks.max": "1",
"nexus.url": "https://repository.cloudera.com/artifactory/repo/",
"working.directory": "/tmp/nifi-stateless-working",
"failure.ports": "",
"topics": "[***TOPIC NAME***]",
```

```
"parameter.HDFS Sink Parameters:Hadoop Configuration Resources": "/etc/ha
doop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml",
"parameter.HDFS Sink Parameters:Kafka Message Data Format": "Raw",
"parameter.HDFS Sink Parameters:Output Directory Pattern": "[***OUTPUT
 DIRECTORY PATH***]",
"parameter.HDFS Sink Parameters:Output Filename Pattern": "messages_${fil
ename.uuid}",
"parameter.HDFS Sink Parameters:Maximum File Size": "10 MB",
"parameter.HDFS Sink Parameters:Output File Demarcator": " \n"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Hadoop Configuration Resources

A comma delimited list of Hadoop configuration files (core-site.xml, hdfs-site.xml, and so on). If HDFS is running on a different cluster than the connector, ensure that you copy the configuration files to the Kafka Connect node that the connector runs on.



Note: In the majority of cases, these files can be found in/etc/hadoop/conf.

Output Directory Pattern

Specifies the path to the output directory in HDFS.

Maximum File Size

The maximum size of the output data file. In this example, the maximum size is 10 MB.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Raw. This means that the connector expects unstructured text or binary data and that record processing is disabled.

Output File Demarcator

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages. In this example, the property is set to \n (newline). This means that the newline character is used to separate the single line character data of the Kafka messages.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid}. The word messages and the underscore (_) are string literals. ${filename.uuid} is an expression, which inserts a Universal Unique Identifier (UUID). As a result, the files are named according to the following pattern: messages_*[***UUID***]*.

Configuration example for Avro input and JSON output

In this example, the connector fetches messages in Avro format from Kafka. The connector parses the Avro records and converts them to JSON format using the schema definition stored in Schema Registry. The Avro file does not contain any schema reference. This is the reason why the schema must be specified in the connector parameters. The connector merges multiple records (all messages that are available for a single execution of the connector) into a JSON file, generates a name for the file using the following pattern: messages_*[***UUID***]*.json, and stores the file in HDFS.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
"connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnect
or",
"meta.smm.predefined.flow.name": "HDFS Sink",
"meta.smm.predefined.flow.version": "1.0.0",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
"tasks.max": "1",
"nexus.url": "https://repository.cloudera.com/artifactory/repo/",
"working.directory": "/tmp/nifi-stateless-working",
"failure.ports": "",
"topics": "[***TOPIC NAME***]",
"parameter.HDFS Sink Parameters:Hadoop Configuration Resources": "/etc/ha
doop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml",
"parameter.HDFS Sink Parameters:Kafka Message Data Format": "Avro",
"parameter.HDFS Sink Parameters:Output Directory Pattern": "[***OUTPUT
 DIRECTORY NAME***]",
"parameter.HDFS Sink Parameters:Output File Data Format": "JSON",
"parameter.HDFS Sink Parameters:Output Filename Pattern": "messages_${filen
ame.uuid}.json",
"parameter.HDFS Sink Parameters:Schema Access Strategy": "Schema Registry",
"parameter.HDFS Sink Parameters:Schema Name": "[***SCHEMA NAME***]"
"parameter.HDFS Sink Parameters:Schema Registry URL": "[***SCHEMA REGISTRY
 SERVER URL***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Hadoop Configuration Resources

A comma delimited list of Hadoop configuration files (core-site.xml, hdfs-site.xml, and so on). If HDFS is running on a different cluster than the connector, ensure that you copy the configuration files to the Kafka Connect node that the connector runs on.

Note: In the majority of cases, these files can be found in/etc/hadoop/conf.

Output Directory Pattern

Specifies the path to the output directory in HDFS.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Avro. This means that the connector expects Avro data and that record processing is enabled.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to JSON, meaning that the output file format is JSON.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid }.json. The word messages, the underscore (_), and .json are string literals. ${filename.uuid} is an expression, which inserts a Universal Unique Identifier (UUID). As a result, the files are named according to the following pattern: messages_*[***UUID***]*.json.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Schema Registry, meaning that the schema definition used to parse input data is taken from the Schema Registry.

Schema Name

Specifies the schema name to look up in Schema Registry.

Schema Registry URL

The URL of the Schema Registry server.

Note: Although not shown in this example, if Schema Registry is deployed on a secure cluster, the following Kerberos and/or TLS/SSL properties must also be set:

- Truststore Filename for Schema Registry
- Truststore Password for Schema Registry
- Truststore Type for Schema Registry
- Kerberos Keytab for Schema Registry
- Kerberos Principal for Schema Registry

Configuration example for JSON input and Parquet output

In this example, the connector fetches messages in JSON format from Kafka. The connector parses the JSON records and converts them to Parquet format using the schema inferred from the JSON data. The connector merges multiple records (all messages that are available for a single execution of the connector) into a Parquet file, generates a name for the file using the following pattern: messages*_[***TIMESTAMP***]_[***SEQUENCE***]*.parquet.gz, and stores the files in HDFS. The output directory path contains a fixed name and a timestamp appended to it: /outputDir/*[***TIMESTAMP***]*. The connector authenticates to HDFS using Kerberos.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
"connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnect
or",
"meta.smm.predefined.flow.name": "HDFS Sink",
"meta.smm.predefined.flow.version": "1.0.0",
"key.converter": "org.apache.kafka.connect.storage.StringConverter",
"value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
"tasks.max": "1",
"nexus.url": "https://repository.cloudera.com/artifactory/repo/",
"working.directory": "/tmp/nifi-stateless-working",
"failure.ports": "",
"topics": "[***TOPIC NAME***]",
"parameter.HDFS Sink Parameters:Hadoop Configuration Resources": "/etc/ha
doop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml",
"parameter.HDFS Sink Parameters:Kafka Message Data Format": "JSON",
"parameter.HDFS Sink Parameters:Output Directory Pattern": "/outputDir/${di
rectory.timestamp}",
"parameter.HDFS Sink Parameters:Output Directory Timestamp Format":"yyMMdd",
"parameter.HDFS Sink Parameters:Output File Data Format": "Parquet",
"parameter.HDFS Sink Parameters:Output Filename Pattern": "messages_${fil
ename.timestamp}_${filename.sequence}.parquet.gz",
"parameter.HDFS Sink Parameters:Output Filename Timestamp Format”: “HHmmss”,
"parameter.HDFS Sink Parameters:Schema Access Strategy": "Infer Schema",
"parameter.HDFS Sink Parameters:Kerberos Keytab for HDFS”:”[***KEYTAB
 LOCATION***]”,
"parameter.HDFS Sink Parameters:Kerberos Principal for HDFS”:”[***PRINCIPAL
 NAME***]”,
"parameter.HDFS Sink Parameters:Compression Codec for Parquet”:”GZIP”
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Hadoop Configuration Resources

A comma delimited list of Hadoop configuration files (core-site.xml, hdfs-site.xml, and so on). If HDFS is running on a different cluster than the connector, ensure that you copy the configuration files to the Kafka Connect node that the connector runs on.



Note: In the majority of cases, these files can be found in/etc/hadoop/conf.

Output Directory Pattern

Specifies the path to the output directory in HDFS. In this example, the path is set to /outputD ir/${directory.timestamp}. As a result, the directory path in this example resolves to /outputD ir/*[***TIMESTAMP***]*. The timestamp format is configured in the Output Directory Timestamp F ormat property.

Output Directory Timestamp Format

Specifies the timestamp format used in the ${directory.timestamp} expression, which can be used in the Output Directory Pattern property's value. Directory Time Based Partitioning can be implemented using this expression. For example, if Output Directory Pattern is set to /outputDir/ ${directory.timestamp} and the timestamp format specified is yyMMdd, the output files will be generated in the /outputDir/220101/ directory if the date is January 1, 2022.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects JSON data and that record processing is enabled.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to Parquet, meaning that the output file format is Parquet.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.timestam p}_${filename.sequence}.parquet.gz.The word messages, the underscores (_), and parquet.gz are string literals. ${filename.timestamp} and ${filename.sequence} are expressions. ${filename.times tamp} inserts the current timestamp in the filename. ${filename.sequence} inserts an incrementing sequence value in the filename. As a result, the files are named according to the following pattern: messages*_[***TIMESTAMP***]_[***SEQUENCE***]*.parquet.gz.

Output Filename Timestamp Format

Specifies the timestamp format used in the ${filename.timestamp} expression. The expression is used when generating the output filename. In this example, the format is HHmmss.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Infer Schema, meaning that the schema is determined (inferred) from the JSON data.

Kerberos Keytab for HDFS

Specifies the keytab used to authenticate to HDFS with Kerberos.

Kerberos Principal for HDFS

Specifies the principal used to authenticate to HDFS with Kerberos.

Compression Codec

Specifies the codec used to compress files and store them in compressed format in HDFS, in the example the GZIP codec is used.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interva l.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

HDFS Stateless Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the HDFS Stateless Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the

accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Avro Schema Write Strategy

Description

Specifies how the record schema is attached to the output data file. Applicable only for Avro output (Output File Data Format is set to Avro).

Do Not Write Schema

Neither the schema nor reference to the schema is attached to the output Avro messages.

Embed Avro Schema

The schema is embedded in every output Avro message.

HWX Content-Encoded Schema Reference

A reference to the schema (identified by Schema Name) within Schema Registry is encoded in the content of the outgoing Avro messages.

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

Compression Codec

Description

The codec used for file compression in HDFS. Use this property to set the codec if the output file format is JSON, Avro, or CSV.

Default Value

NONE

Accepted Values

NONE, DEFAULT, BZIP, GZIP, LZ4, LZO, SNAPPY, AUTOMATIC

Required

true

Compression Codec for Parquet

Description

Codec used for file compression in HDFS. Use this property to set the codec if the output file format is Parquet.

Default Value

UNCOMPRESSED

Accepted Values

UNCOMPRESSED,SNAPPY, GZIP, LZO

Required

true

Date Format

Description

Specifies the format to use when writing date fields to JSON or CSV.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Hadoop Configuration Resources

Description

A comma separated list of files which contains the Hadoop file system configuration.

Default Value

/etc/hadoop/conf/core-site.xml, /etc/hadoop/conf/hdfs-site.xml

Accepted Values

Required

false

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If set to Avro or JSON, record processing is enabled. Raw can be used for unstructured text or binary data.

Default Value

Avro

Accepted Values

Avro, JSON, Raw

Required

true

Kerberos Keytab for HDFS

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing HDFS.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for HDFS

Description

The Kerberos principal used for authenticating to HDFS.

Default Value

default

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum File Size

Description

The maximum size of the output data file. No size limit is applied if this property is not specified. Example values: 100 MB, 1 GB.

Default Value

Accepted Values

Required

false

Maximum Number of Entries

Description

The maximum number of entries in the output data file. In the context of this property, entry can mean one of two things. If record processing is enabled (Output File Data Format is set to Avro or JSON), an entry is a record. Otherwise, entry means a Kafka message. Set this property to 1 if you want to create a separate output file for each Kafka message.

Default Value

1000000

Accepted Values

Required

true

Output Directory Pattern

Description

Specifies the full path of output HDFS directory. The pattern can contain string literals (fixed text) as well as the ${directory.timestamp} expression, which inserts the current timestamp in the directory name.

Default Value

Accepted Values

Required

true

Output Directory Timestamp Format

Description

The timestamp format to use for the ${directory.timestamp} expression. For example: yyyyMMdd.

Default Value

Accepted Values

Required

false

Output File Data Format

Description

Specifies the format of the records written to the output file. Required when record processing is enabled (Kafka Message Data Format is set to Avro or JSON).

Default Value

Avro

Accepted Values

Avro, JSON, CSV, Parquet

Required

false

Output File Demarcator

Description

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages (no record processing). This property can only be used if Kafka Message Data Format is set to Raw. If you want to use newline as the delimiter, set this property to \n.

Default Value

Accepted Values

Required

false

Output Filename Pattern

Description

Specifies the structure of the name of the output file. The pattern can contain string literal (fixed text) parts and one or more of the following expressions:

- ${filename.uuid}: Inserts a generated UUID in the filename.
- ${filename.timestamp}: Inserts the current timestamp in the filename.
- ${filename.sequence}: Inserts an incrementing sequence value in the filename.

In order to generate unique filenames, either ${filename.uuid} or ${filename.sequence} must be used in the pattern.

Examples:

- data_${filename.uuid}.json
- records_${filename.timestamp}_${filename.sequence}.avro

Default Value

${filename.uuid}

Accepted Values

Required

false

Output Filename Sequence Initial Value

Description

This property is used to configure the initial value of the ${filename.sequence} expression. The value you set in this property is not the initial value of the sequence. The sequence starts at the value of this property +1. For example, if you set this property to 0, the sequence starts at 1.

Default Value

0

Accepted Values

Required

false

Output Filename Sequence Padding Length

Description

Specifies the length of the ${filename.sequence} expression in characters. If the sequence has fewer characters than the value set in this property, it is padded with zeros (0). Padding is added to the left of the sequence.

Default Value

6

Accepted Values

Required

false

Output Filename Timestamp Format

Description

The timestamp format to use for the ${filename.timestamp} expression. For example, yyyyMMdd _HHmmss_SSS.

Default Value

Accepted Values

Required

false

Output Files by Kafka Partitions

Description

Controls how fetched messages are merged together in the output file. If set to true, only messages fetched from the same Kafka partition get merged together in one output file (provided that Maxi mum Number of Entries is greater than 1). If set to false, messages are merged together in the order they are fetched from the Kafka topic. This property has no effect if Maximum Number of Entries is 1.

Default Value

false

Accepted Values

true, false

Required

false

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Kafka Message Data Format is set to Raw).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration JSON.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when reading or writing Time fields to JSON or CSV.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading or writing Timestamp fields to JSON or CSV.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

HTTP Sink connector

The HTTP Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The HTTP Sink connector obtains messages from a Kafka topic and transfers their content in a HTTP POST requests to a specified endpoint. The topic the connector receives messages from is determined by the value of the topics parameter in the configuration. The connector can forward the data it reads from Kafka as is (raw data) or can be configured to execute record processing. When record processing is enabled, the connector expects the incoming data in Avro or JSON format. In case of Avro, the connector can either read the record schema from the Avro file it receives (provided that the schema is embedded) or it can fetch the schema from Schema Registry. In case of JSON, it can either infer the schema or fetch it from Schema Registry. What strategy is used to retrieve the schema is determined by the Schema Access Strategy property.

If Schema Registry is used, and it is on a Kerberized cluster, the krb5.file property must point to the krb5.conf file that provides access to the cluster on which Schema Registry is present. This means that the krb5.conf file must be on the same cluster node that the connector runs on. The connections to Schema Registry and the HTTP server can be secured by TLS. The keystore and truststore files necessary for securing these connections must also be on the same cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Sink properties reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the HTTP Sink properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- The value of the Schema Access Strategy property is not independent of the value of the Kafka Message Data F ormat property. As a result, you must exercise caution when configuring Schema Access Strategy.
  - If the value of Kafka Message Data Format is AVRO, the possible values for Schema Access Strategy are Schema Registry, Embedded Schema or HWX Content-Encoded Schema Reference.
  - If the value of Kafka Message Data Format is JSON, the possible values for Schema Access Strategy are Sch ema Registry or Infer Schema.

Configuration example

In this example, the connector receives data in any format and forwards the raw data as the content of an HTTP POST request.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "HTTP Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "input.port": "Input from Kafka",
 "failure.ports": "Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.HTTP Sink Parameters:Forward Raw Data": "true",
 "parameter.HTTP Sink Parameters:Remote URL": "http://[***SERVER
 HOSTNAME***]:[***PORT***]/[***PATH***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Forward Raw Data

Specifies whether messages from Kafka should be forwarded as is or converted to JSON. In this example, the property is set to true, meaning that the connector does not process any records. It forwards incoming data as is.

Remote URL

Identifies the HTTP endpoint that receives the messages sent by this connector. In this example, SSL is not used. As a result, the URL starts with http not https. For example, http://my.http-server.com:22000/contentListener.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

HTTP Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the HTTP Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Basic Authentication Password

Description

The password to be used for authentication when connecting to the URL specified in the Remote URL property. If an authentication method other than Basic Authentication is used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Basic Authentication Username

Description

The username used for authentication when connecting to the URL specified in the Remote URL property. Cannot include control characters (0-31), ':', or DEL (127). If an authentication method other than Basic Authentication is used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Content-Type

Description

The Content-Type to specify in the HTTP header of the POST request sent by this connector.

Default Value

application/octet-stream

Accepted Values

Required

true

Date Format

Description

Specifies the format to use when reading date fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the date fields in the output JSON message.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Forward Raw Data

Description

Specifies whether messages from Kafka should be forwarded as is or converted to JSON. If set to false the Kafka Message Data Format parameter must be specified.

Default Value

true

Accepted Values

true, false

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If the Forward Raw Data property is set to true then this property is ignored. However, even in a case like this, this property must be assigned a valid value.

Default Value

AVRO

Accepted Values

AVRO, JSON

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Keystore Filename for Secure HTTP

Description

The fully-qualified filename of a keystore. This keystore is used to establish a secure connection with the HTTP server using mutual TLS.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Key Password for Secure HTTP

Description

The password used to access the key stored in the keystore file configured in the Keystore Filename for Secure HTTP property.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Password for Secure HTTP

Description

The password used to access the contents of the keystore configured in the Keystore Filename for Secure HTTP property.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Keystore Type for Secure HTTP

Description

The type of the keystore configured in the Keystore Filename for Secure HTTP property.

If the HTTP server does not require client certificate authentication, this property must be completely removed from the configuration.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

false

Output Grouping for JSON

Description

Specifies how JSON objects are grouped in the connector output.

If set to output-array, the output will consist of an array of JSON objects.

f set to output-oneline, each line of the output data becomes one JSON object. That is, each JSON object occupies one line in the output.

This property is only used if forwarding raw data is disabled (Forward Raw Data is set to false). This is because when forwarding raw data is disabled, the data coming from Kafka gets converted to JSON. However, this property must still be set even if forwarding raw data is enabled.

Default Value

output-oneline

Accepted Values

output-array, output-oneline

Required

true

Remote URL

Description

The remote URL to connect to. Must include scheme, host, port, and path.

Default Value

https://localhost:22000/contentListener

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Forward Raw Data is set to true).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.

If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when reading time fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the time fields in the output JSON message.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading timestamp fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the timestamp fields in the output JSON message.

Default Value

HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connection.

Accepted Values

Required

true

Truststore Filename for Secure HTTP

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with the HTTP server using TLS

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Password for Secure HTTP

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Secure HTTP property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

Truststore Type for Secure HTTP

Description

The type of the truststore configured in the Truststore Filename for Secure HTTP property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

InfluxDB Sink connector

The InfluxDB Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect Framework. Learn about the connector, its properties, and configuration.



Important: The InfluxDB Sink connector only supports InfluxDB v1.

The InfluxDB Sink Connector fetches messages from Kafka and loads them to InfluxDB. The topic this connector receives messages from is determined by the value of the topics parameter in the configuration. The messages the connector receives from Kafka can be in either Avro or JSON format.

If the connector input is in Avro format, then it can either read the schema from the Avro file it receives (provided that the schema is embedded) or it can fetch the schema from Schema Registry. If the connector's input is in JSON format, then it can either infer the schema or fetch it from Schema Registry. The strategy that is used to retrieve the schema is determined by the Schema Access Strategy property.

The connector can authenticate to InfluxDB using username and password.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Sink properties reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the InfluxDB Sink properties reference.

Notes and limitations

- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- The value of the Schema Access Strategy property is not independent of the value of the Kafka Message Data F ormat property. As a result, you must exercise caution when configuring Schema Access Strategy.
  - If the value of Kafka Message Data Format is AVRO, the possible values for Schema Access Strategy are Schema Registry, Embedded Schema or HWX Content-Encoded Schema Reference.
  - If the value of Kafka Message Data Format is JSON, the possible values for Schema Access Strategy are Sch ema Registry or Infer Schema.
- The Line Protocol Query parameter contains a query expression which creates a string from input record fields. The query depends on the record schema. The result of the query must comply with the line protocol format.

Configuration example

In this example, the connector receives data in JSON format (which is the default setting), transforms the data to line protocol format, and inserts it to InfluxDB. Schema Registry is not used in the example. As a result, this example does not include Schema Registry key and truststore configurations.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "InfluxDB Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "extensions.directory": "/tmp/stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo/",
 "failure.ports": "Retry from PutInfluxDB",
 "topics": "[***TOPIC NAME***]",
 "parameter.InfluxDB Connection URL": "http://[**SERVER
 HOSTNAME**]:[**PORT**]",
 "parameter.InfluxDB Database Name": "[***DATABASE NAME***]",
 "parameter.InfluxDB User Name": "[***USER NAME***]",
 "parameter.InfluxDB User Password": "[***USER PASSWORD***]",
 "parameter.Line Protocol Query": "select 'temp,device_id=' || device_id ||
 ',device_state=' || device_state || ',part_id=' || part_id || ',code=' ||
 code || ',result=' || result_code || ' temperature=' || temperature || ' '
|| ts || '000000' as payload from FLOWFILE",
 "parameter.Schema Access Strategy":"Infer Schema"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

InfluxDB Connection URL

The URL used to connect to InfluxDB.

InfluxDB Database Name

The name of the InfluxDB database.

InfluxDB User Name

The username used to connect to InfluxDB.

InfluxDB User Password

The password used to connect to InfluxDB.

Line Protocol Query

Specifies a query based on the Avro schema of the records and the measurement used.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, the property is set to Infer Schema. This means that the schema is determined (inferred) from the JSON data.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interva l.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

InfluxDB Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the InfluxDB Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Batch Size

Description

Maximum size of records merged together and put to InfluxDB in one batch. Example values: 100 MB, 1 GB

Default Value

1 MB

Accepted Values

Required

true

Consistency Level

Description

Specifies the InfluxDB consistency level

Default Value

ONE

Accepted Values

ONRE ANY, ALL, QUORUM

Required

true

Date Format

Description

Specifies the format to use when reading date fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the date fields in the output JSON message.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

InfluxDB Connection URL

Description

Specifies the InfluxDB URL to connect to.

Default Value

http://localhost:8086

Accepted Values

Required

true

InfluxDB Database Name

Description

The name of the InfluxDB database to connect to.

Default Value

Accepted Values

Required

true

InfluxDB User Name

Description

The username for accessing InfluxDB.

Default Value

Accepted Values

Required

false

InfluxDB User Password

Description

The password for InfluxDB user.

Default Value

Accepted Values

Required

false

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If the Forward Raw Data property is set to true then this property is ignored. However, even in a case like this, this property must be assigned a valid value.

Default Value

AVRO

Accepted Values

AVRO, JSON

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the Kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Line Protocol Query

Description

A record schema based query which returns the attributes in line protocol format.

Default Value

Accepted Values

Required

true

Retention Policy

Description

Specifies the retention policy for saving records in InfluxDB.

Default Value

autogen

Accepted Values

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry.

If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name.

If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when reading time fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the time fields in the output JSON message.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading timestamp fields from JSON. If Forward Raw Data is set to false, the format defined here also applies to the timestamp fields in the output JSON message.

Default Value

HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connection.

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

JDBC Sink connector

The JDBC Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The JDBC Sink connector fetches messages from Kafka and loads them into a database table. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages the connector receives from Kafka can be in either Avro or JSON format and must contain records that can be inserted into the database table. The schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The strategy that is used to retrieve the schema is determined by the Schema Access Strategy property.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Sink properties reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the JDBC Sink properties reference.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.

Configuration example

In this example, the connector fetches messages in JSON format from Kafka, translates it to a SQL INSERT statement in order to load the data into a PostgreSQL database. The schema of the records must match the schema of the database table. The record schema in this example is inferred from the input JSON.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "JDBC Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "Retry from PutDatabaseRecord,Failure from PutDatabaseReco
rd",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.JDBC Sink Parameters:Database Connection URL": "[***JDBC
 URL***]",
 "parameter.JDBC Sink Parameters:Database Driver Location": "[***PATH TO
 JDBC DRIVER***]",
 "parameter.JDBC Sink Parameters:Database Driver Class Name": "org.postgr
esql.Driver",
```

```
 "parameter.JDBC Sink Parameters:Database Type": "PostgreSQL",
 "parameter.JDBC Sink Parameters:Database User Name": "[***USERNAME***]",
 "parameter.JDBC Sink Parameters:Database User Password":
 "[***PASSWORD***]",
 "parameter.JDBC Sink Parameters:Database Table Name": "[***TABLE NAME***]",
 "parameter.JDBC Sink Parameters:Database Schema Name": "[***SCHEMA
 NAME***]",
 "parameter.JDBC Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.JDBC Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.JDBC Sink Parameters:Quote Table Name": "true",
 "parameter.JDBC Sink Parameters:Quote Column Names": "true"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

Database Connection URL

The JDBC URL of the PostgreSQL database. For example, jdbc:postgresql://myhost:5432/mydb.

Database Driver Location

The path to the PostgreSQL JDBC driver JAR file (or the directory containing the JAR).

Database Driver Class Name

The Java class name of the PostgreSQL Driver implementation.

Database Type

The type of the database. Because this is a PostgreSQL example, this property is set to PostgreSQL.

Database User Name

The username used for authenticating to the database.

Database User Password

The name of the database table to load data to.

Kafka Message Data Format

The format of the messages the connector receives from Kafka.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, the property is set to Infer Schema. This means that the schema is determined (inferred) from the JSON data.

Quote Table Name and Quote Column Name

These properties specify whether to place double quotes around the table or column names in the SQL statement used by the connector. Some databases, like Oracle and PostgreSQL are case-sensitive for table and column names. By default, Oracle converts all table column names to uppercase if they are not double-quoted upon creation. Similarly, PostgreSQL converts names to lowercase. If tables or columns are double-quoted upon creation and have camel cased names, set these properties to true so that double quotes are also used around the table or column names in the SQL statement used by the connector. If this is not done, the connector will not find the tables on the database.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

JDBC Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the JDBC Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Database Connection URL

Description

The database specific connection URL used for connecting to the database. For example, jdbc:postgresql://localhost:5432/postgres.

Default Value

Accepted Values

Required

true

Database Driver Class Name

Description

The database driver class name. For example, org.postgresql.Driver.

Default Value

Accepted Values

Required

true

Database Driver Location

Description

A comma-separated list of files or folders containing the JDBC client libraries.

Default Value

Accepted Values

Required

true

Database Schema Name

Description

The name of the database schema containing the table to load data.

Default Value

Accepted Values

Required

false

Database Table Name

Description

The name of the database table to load data.

Default Value

Accepted Values

Required

true

Database Type

Description

The database type used for generating database specific SQL queries.

Default Value

Generic

Accepted Values

Generic, Oracle, Oracle 12+, MS SQL 2008, MS SQL 2012+, MySQL, PostgreSQL

Required

true

Database User Name

Description

The database user name. If username/password authentication is not required by the database server, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Database User Password

Description

The database user password. If username/password authentication is not required by the database server, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Date Format

Description

Specifies the format to use when reading date fields from JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka.

Default Value

Avro

Accepted Values

Avro, JSON

Required

true

Kerberos Keytab

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Quote Column Names

Description

Specifies whether to use quotes around the column names in the generated SQL statement.

Default Value

false

Accepted Values

true, false

Required

true

Quote Table Name

Description

Specifies whether to use quotes around the table name in the generated SQL statement.

Default Value

false

Accepted Values

true, false

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro

messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from

the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Statement Type

Description

Specifies the type of SQL statement that is generated to put data into the database.

Default Value

INSERT

Accepted Values

INSERT, INSERT_IGNORE, UPDATE, UPSERT

Required

true

Time Format

Description

Specifies the format to use when reading Time fields from JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading Timestamp fields from JSON.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

Kudu Sink connector

The Kudu Sink connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The Kudu Sink connector fetches messages from Kafka and loads them into a table in Kudu. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages the connector receives from Kafka can be in either Avro or JSON format and must contain records that can be inserted into a Kudu table. If the connector input is in Avro format, then it can either read the schema from the Avro file it receives (provided that the schema is embedded) or it can fetch the schema from Schema Registry. If the connector's input is in JSON format, then it can either infer the schema, or fetch it from Schema Registry. The strategy that is used to retrieve the schema is determined by the Schema Access Strategy property.

Kudu is expected to be on a Kerberized cluster. It is not mandatory to have Kudu and Schema Registry on the same cluster. In the connector's configuration the krb5.file property must point to the krb5.conf file that provides access to the cluster that Kudu and Schema Registry are on. The keytab and truststore files that are necessary to access Kudu and Schema Registry must be present on the cluster node that the connector runs on.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Sink properties reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the Kudu Sink properties reference.



Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- Schema Registry URL is mandatory even if no registry is used. In a case like this, use the default value, or remove the property completely from the configuration.
- Truststore parameters are mandatory even if connecting to a non-secure Schema Registry or no registry is used. In that case the default (empty) truststore can be used.
- Kerberos parameters for Schema Registry are mandatory even if connecting to a non-secure Schema Registry or no registry is used. In that case the default (empty) keytab can be used.
- Kudu is expected to be on a Kerberized cluster. As a result, Kerberos parameters for Kudu are mandatory.

Configuration example

In this example, the connector receives data in JSON format from a Kafka topic and forwards it to a Kudu table. The record schema is inferred based on the data.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "Kudu Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter
",
```

```
 "tasks.max": "1",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "input.port": "Input from Kafka",
 "failure.ports": "Failure",
 "krb5.file": "[***PATH TO KRB5 FILE***]",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.Kudu Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.Kudu Sink Parameters:Kerberos Keytab for Kudu": "[***PATH TO
 KUDU KEYTAB***]",
 "parameter.Kudu Sink Parameters:Kerberos Principal for Kudu": "[***KUDU
 PRINCIPAL***]",
 "parameter.Kudu Sink Parameters:Kudu Masters": "[***KUDU MASTER 1 WITH
 PORT***],[***KUDU MASTER 2 WITH PORT***],[***KUDU MASTER 3 WITH PORT***]",
 "parameter.Kudu Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.Kudu Sink Parameters:Table Name With Schema": "[***KUDU
 SCHEMA***].[***KUDU TABLE***]"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

krb5.file

The path to a krb5.conf file that contains the information needed to access the Kudu service on a Kerberized cluster

topics

The name of the Kafka topic that the connector fetches messages from.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects data in JSON format.

Kerberos Keytab for Kudu

The path to the keytab file that enables access to the Kudu service on a Kerberized cluster.

Kerberos Principal for Kudu

The principal used to access Kudu

Kudu Masters

A comma-separated list of server URLs that identify the Kudu service's master nodes. For example: localhost:7051,localhost:7151,localhost:7251

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, the property is set to Infer Schema. This means that the record schema is inferred based on the data the connector receives.

Table Name With Schema

The schema and name of the Kudu table in which the records get inserted.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an

endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

Kudu Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Kudu Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Date Format

Description

Specifies the format to use when reading Date fields from JSON.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Handle Schema Drift

Description

Specifies whether to handle Schema Drift. If set to true, when fields with names that are not in the target Kudu table are encountered, the Kudu table is altered to include new columns for those fields. If set to false, fields that only exist in the input data are skipped.

Default Value

false

Accepted Values

true, false

Required

true

Ignore Null

Description

Ignore NULL on Kudu Put Operation. If set to true, only non-null columns get updated.

Default Value

false

Accepted Values

true, false

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka.

Default Value

AVRO

Accepted Values

AVRO, JSON

Required

true

Kerberos Keytab for Kudu

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Kudu.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections

Accepted Values

Required

true

Kerberos Keytab for Schema Registry

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal for Kudu

Description

The Kerberos principal used for authenticating to Kudu.

Default Value

default

Accepted Values

Required

true

Kerberos Principal for Schema Registry

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Kudu Kerberos Service Name

Description

The SASL protocol name to use for authenticating with Kerberos. Must match the Kudu service principal name.

Default Value

kudu

Accepted Values

Required

false

Kudu Masters

Description

The comma separated addresses of the Kudu masters to connect to. For example: localhost: 7051 ,localhost:7151,localhost:7251.

Default Value

Accepted Values

Required

true

Kudu Operation Type

Description

Specifies what Kudu operation gets executed.

Default Value

INSERT

Accepted Values

INSERT, INSERT_IGNORE, UPSERT, UPDATE, DELETE, UPDATE_IGNORE, DELETE_IGNORE

Required

true

Lowercase Field Names

Description

Specifies whether to convert column names to lowercase when finding indexes of Kudu table columns.

Default Value

false

Accepted Values

true, false

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The schema name to look up in Schema Registry. If the Schema Access Strategy property is set to Schema Registry, this property must contain a valid schema name. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, use the default value.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Table Name With Schema

Description

The schema and name of the Kudu Table to put data into. For example: default.mytable. When using Impala: impala::default.mytable.

Default Value

Accepted Values

Required

true

Time Format

Description

Specifies the format to use when reading Time fields from JSON.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when reading Timestamp fields from JSON. The value must match the Java Simple Date Format. (For example, MM/dd/yyyy HH:mm:ss for a two-digit month, followed by a two-digit day, followed by a four-digit year, all separated by forward slashes. This is followed by a two-digit hour in 24-hour format, followed by a two-digit minute, followed by a two-digit second, all separated by colons. The resulting timestamp looks like this: 01/01/2017 18:04:15.)

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename for Schema Registry

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using TLS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections

Accepted Values

Required

true

Truststore Password for Schema Registry

Description

The password used to access the contents of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

password

Accepted Values

Required

true

Truststore Type for Schema Registry

Description

The type of the truststore configured in the Truststore Filename for Schema Registry property.

Default Value

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

S3 Sink connector

The S3 Sink Connector is a Stateless NiFi dataflow developed by Cloudera that is running in the Kafka Connect framework. Learn about the connector, its properties, and configuration.

The S3 Sink connector fetches messages from Kafka and uploads them to AWS S3. The topic this connector receives messages from is determined by the value of the topics property in the configuration. The messages can contain unstructured (character or binary) data or they can be in Avro or JSON format.

If the input is unstructured data, record processing is disabled. In a case like this, multiple messages can be concatenated into a single output file on S3 using a demarcator (for example, newline for text messages). Merging is optional, large binary data can be forwarded to S3 at a 1:1 ratio, one Kafka message equals a single S3 object.

If the input is either Avro or JSON, record processing is enabled. In a case like this, the schema of the records can be a predefined schema retrieved from Schema Registry (for both Avro and JSON data), it can be embedded in the Avro data, or inferred from the JSON data. The output can be Avro, JSON, CSV or Parquet. Multiple records are typically merged into a single output file before uploading to S3.

The connector can authenticate to AWS using an Access Key.

Properties and configuration

Configuration is passed to the connector during creation. The properties of the connector can be categorized into three groups. These are as follows:

Common connector properties

These are the properties of the Kafka Connect framework that are accepted by all connectors. For a comprehensive list of these properties, see the Apache Kafka documentation.

Stateless NiFi Sink properties

These are the properties that are specific to the Stateless NiFi Sink connector. All Stateless NiFi Sink connectors share and accept these properties. For a comprehensive list of these properties, see the Stateless NiFi Sink properties reference.

Connector/dataflow-specific properties

These properties are unique to this specific connector. Or to be more precise, unique to the dataflow running within the connector. These properties use the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

For a comprehensive list of these properties, see the S3 Sink properties reference.

Tip: When creating a new connector using the Streams Messaging Manager UI, all valid properties for this connector are presented in the default configuration template. You can use the template in the UI as a quick reference to view all accepted properties.

Notes and limitations

- Required properties must be assigned a valid value even if they are not used in the particular configuration. If a required property is not used, either leave its default value, or completely remove the property from the configuration.
- If a property that has a default value is completely removed from the configuration, the system uses the default value.
- Properties not marked as required must be completely removed from the configuration if not set.
- Schema Branch and Schema Version can not be specified at the same time.
- It is not recommended to configure the following combination of settings:
  - Kafka Message Data Format = JSON
  - Schema Access Strategy = Infer Schema
  - Maximum Number of Entries > 1
  - Output File Data Format = Avro

  In case the input records do not strictly match to one schema (for example, one record has a missing field while the other records do not), schema inference determines a schema for each record, while batching the schemas together assigns only one schema to the entire batch. Because of this, there might be records in the batch that do not match with the schema that is assigned to the batch. This can be avoided by disabling batching with the Maximum Number of Entries set to 1.

Configuration example for fetching unstructured data

In this example, the connector fetches unstructured messages containing single line character data from Kafka. The connector concatenates the messages using newline characters as the demarcator. The files that the connector creates and uploads to ADLS are maximum 10 MB in size. The files are named according to the following pattern: messages_*[***UUID***]*. The connector authenticates to AWS using an Access Key.

Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnector",
 "meta.smm.predefined.flow.name": "S3 Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConverter",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutS3Object Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.S3 Sink Parameters:AWS Access Key ID": "[***ACCESS KEY ID***]",
 "parameter.S3 Sink Parameters:AWS Secret Access Key": "[***SECRET ACCESS KEY***]",
```

```
 "parameter.S3 Sink Parameters:S3 Region": "[***S3 REGION***]",
 "parameter.S3 Sink Parameters:S3 Bucket": "[***S3 BUCKET***]",
 "parameter.S3 Sink Parameters:Maximum File Size": "10 MB",
 "parameter.S3 Sink Parameters:Kafka Message Data Format": "Raw",
 "parameter.S3 Sink Parameters:Output File Demarcator": "\n",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "messages_${fi
lename.uuid}"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

AWS Access Key ID

The Access Key ID to use for authentication to AWS.

AWS Secret Access Key

The Secret Access Key to use for authentication to AWS.

S3 Region

The AWS Region of the S3 Bucket to upload data to.

S3 Bucket

The S3 Bucket to upload data to.

Maximum File Size

The maximum size of the output data file. In this example, the maximum size is 10 MB.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to Raw. This means that the connector expects unstructured text or binary data.

Output File Demarcator

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages. In this example, the property is set to \n (newline). This means that the newline character is used to separate the single line character data of the Kafka messages.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to messages_${filename.uuid}.

messages_ is a string literal, ${filename.uuid} is an expression that inserts a generated UUID in the filename. As a result, the files are named according to the following pattern: mess ages_*[***UUID***]*.

Configuration example for fetching JSON data

In this example, the connector fetches messages in JSON format from Kafka. The connector parses the JSON records and converts them to Avro format using the schema inferred from the JSON data. The schema is also embedded in the output file. The connector merges multiple records (all messages that are available for a single execution of the connector) into an Avro file, generates a name for the file using the following pattern data_*[***TIMESTAMP***]_[***SEQUENCE***]*.avro, and uploads the file to S3. The connector authenticates to AWS using an Access Key.



Note: All properties that are not present in this example have suitable default values or are unnecessary for this specific use case.

```
{
```

```
 "connector.class": "org.apache.nifi.kafka.connect.StatelessNiFiSinkConnec
tor",
 "meta.smm.predefined.flow.name": "S3 Sink",
 "meta.smm.predefined.flow.version": "1.0.0",
 "key.converter": "org.apache.kafka.connect.storage.StringConverter",
 "value.converter": "org.apache.kafka.connect.converters.ByteArrayConvert
er",
 "nexus.url": "https://repository.cloudera.com/artifactory/repo",
 "extensions.directory": "/tmp/nifi-stateless-extensions",
 "working.directory": "/tmp/nifi-stateless-working",
 "failure.ports": "PutS3Object Failure",
 "topics": "[***KAFKA TOPIC NAME***]",
 "parameter.S3 Sink Parameters:AWS Access Key ID": "[***ACCESS KEY ID***]",
 "parameter.S3 Sink Parameters:AWS Secret Access Key": "[***SECRET ACCESS
 KEY***]",
 "parameter.S3 Sink Parameters:S3 Region": "[***S3 REGION***]",
 "parameter.S3 Sink Parameters:S3 Bucket": "[***S3 BUCKET***]"
 "parameter.ADLS Sink Parameters:Kafka Message Data Format": "JSON",
 "parameter.ADLS Sink Parameters:Schema Access Strategy": "Infer Schema",
 "parameter.ADLS Sink Parameters:Schema Write Strategy": "Embed Avro Schema
",
 "parameter.ADLS Sink Parameters:Output File Data Format": "Avro",
 "parameter.ADLS Sink Parameters:Output Filename Pattern": "data_${filenam
e.timestamp}_${filename.sequence}.avro",
 "parameter.ADLS Sink Parameters:Output Filename Timestamp Format": "yyyyMM
dd_HHmmss_SSS"
}
```

The following list collects the properties from the configuration example that must be customized for this use case.

topics

The name of the Kafka topic the connector fetches messages from.

AWS Access Key ID

The Access Key ID to use for authentication to AWS.

AWS Secret Access Key

The Secret Access Key to use for authentication to AWS.

S3 Region

The AWS Region of the S3 Bucket to upload data to.

S3 Bucket

The S3 Bucket to upload data to.

Kafka Message Data Format

The format of the messages the connector receives from Kafka. In this example, this property is set to JSON. This means that the connector expects JSON data.

Schema Access Strategy

Specifies the strategy used for determining the schema of the Kafka record. In this example, this property is set to Infer Schema, meaning that the schema is determined (inferred) from the JSON data.

Schema Write Strategy

Specifies whether the record schema is written to the output data file. In this example, this property is set to Embed Avro Schema, meaning that the schema is embedded in the output Avro file.

Output File Data Format

Specifies the format of the records written to the output file. In this example, this property is set to Avro, meaning that the output file format is Avro.

Output Filename Pattern

Specifies the structure of the name of the output file. This property accepts string literals (fixed text) as well as various expressions. In this example, the property is set to data_${filename.timestamp}_${filename.sequence}.avro.

data_, the underscore (_), and .avro are string literals. ${filename.timestamp} and ${filename.sequence} are expressions. ${filename.timestamp} inserts the current timestamp in the filename. ${filename.sequence} inserts an incrementing sequence value in the filename. As a result, the files are named according to the following pattern: data_*[***TIMESTAMP***]_[***SEQUENCE***]*.avro.

Output Filename Timestamp Format

Specifies the timestamp format used in the ${filename.timestamp} expression. The expression is used when generating the output filename.

Related Information

Apache Kafka documentation

Stateless NiFi Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the Stateless NiFi Sink connector.

In addition to the properties listed here, Stateless NiFi connectors also accept the properties of the Kafka Connect framework. For a comprehensive list of these properties, see the Apache Kafka documentation.

attribute.prefix

Description

The prefix to add to the key of each header that matches the regular expression specified in headers.as.attributes.regex. For example, if the header key is MyHeader, its value is MyValue, headers.as.attributes.regex is set to My.*, and this property is set to kafka, the flowfile that is created for the Kafka message will have an attribute named kafka.MyHeader with a value of MyValue.

Default Value

Accepted Values

Required

false

dataflow.timeout

Description

Specifies the maximum amount of time to wait for the dataflow to complete. If the dataflow does not complete before this timeout, the thread is interrupted and the dataflow is considered as a failure. The session is rolled back and the connector retriggers the flow. Defaults to 60 seconds if not specified.

Default Value

60 seconds

Accepted Values

Required

false

extensions.directory

Description

Specifies the directory that stores downloaded extensions. Extensions are the NAR (NiFi Archive) files containing the processors and controller services a flow might use. Since Stateless NiFi is only the NiFi engine, it does not contain any of the processors and controller services you might use

in your flow. When deploying the connector with the custom flow, the system needs to download the specific extensions that your flow uses from Nexus (unless they are already present in this directory). These extensions are stored in this directory. Because the default directory might not be writable, and to aid in upgrade scenarios, Cloudera recommends that you always specify an extensions directory.

Default Value

/tmp/nifi-stateless-extensions

Accepted Values

Required

true

failure.ports

Description

A comma separated list of output ports that are considered as failure conditions. If any flowfile is routed to an output port specified in this property, the dataflow is considered a failure and the session is rolled back. After a set amount of time, the dataflow reattempts to process the Kafka record. Any data transferred to an output port that is not in the list of failure ports is discarded.

Because of how Stateless NiFi Sink connectors behave, even if a single flowfile ends up in an output port that is marked as failure, the entire sessions is rolled back with all messages in the batch. Furthermore, if a flowfile ends up in a failure port in each subsequent iteration, the result is an endless loop. With some sink connectors (for example. MQTT Sink) this is the desired behavior. For more information regarding this behavior, see Dataflow execution and scheduling.

Default Value

Accepted Values

Required

false

flow.snapshot

Description

Specifies the dataflow to run. When using Streams Messaging Manager to deploy a connector, the value you set in this property must be a JSON object. URLs, file paths, or escaped JSON strings are not supported when using Streams Messaging Manager. Alternatively, if using the Kafka Connect REST API to deploy a connector, this can be a file containing the dataflow, a URL that points to a dataflow, or a string containing the entire dataflow as an escaped JSON. Cloudera however, does not recommend using the Kafka Connect REST API to interact with this connector or Kafka Connect.

Default Value

Accepted Values

Required

true

headers.as.attributes.regex

Description

A Java regular expression that is evaluated against all Kafka record headers. Headers are added to the flowfile as an attribute if the header key matches the regular expression. The header key is used as the attribute name. The header value is used as the attribute value. Additionally, the name of the attribute can also contain an optional prefix which is defined by the attribute.prefix property.

Default Value

Accepted Values

Required

false

input.port

Description

The name of the input port in the NiFi dataflow that Kafka records are sent to. If the dataflow contains exactly one input port, this property is optional and can be omitted. However, if the dataflow contains multiple input ports, this property must be specified.

Default Value

Accepted Values

Required

false

krb5.file

Description

Specifies the krb5.conf file to use if the dataflow interacts with any services that are secured using Kerberos. Defaults to /etc/krb5.conf if not specified.

Default Value

/etc/krb5.conf

Accepted Values

Required

false

name

Description

The name of the connector. On the Streams Messaging Manager UI, the connector names are specified using the Enter Name field. The name that you enter in the Enter Name field is automatically set as the value of the name property when the connector is deployed. Because of this, the name property is omitted from the configuration template provided in Streams Messaging Manager. If you manually add the name property to the configuration in Streams Messaging Manager, ensure that the value you set matches the connector name specified in the Enter Name field. Otherwise, the connector fails to deploy.

Default Value

Accepted Values

Required

true

nexus.url

Description

Specifies the Base URL of the Nexus instance to source extensions from. If configuring a Nexus instance that has multiple repositories, include the name of the repository in the URL. For example, https://nexus-private.myorganization.org/nexus/repository/my-repository/. If the property is not specified, the necessary extensions (the ones used by the flow) must be provided in the extensions directory before deploying the connector.

Default Value

Accepted Values

Required

true

parameter.*[***FLOW PARAMETER NAME***]*

Description

Specifies a parameter to use in the dataflow. For example, assume that you have the following entry in your connector configuration "parameter.Directory": "/mydir". In a case like this, any parameter context in the dataflow that has a parameter named Directory gets the specified value (/mydir). If the dataflow has child process groups, and those child process groups have their own parameter contexts, the value is used for all parameter contexts that contain a parameter named Directory. Parameters can also be applied to specific parameter contexts only. This can be done by prefixing the parameter name (Directory) with the name of the parameter context followed by a colon. For example, parameter.My Context:Directory only applies the specified value for the Dire ctory parameter in the Parameter Context named My Context.

Default Value

Accepted Values

Required

false

working.directory

Description

Specifies a directory on the Connect server that NiFi should use for unpacking extensions that it needs to perform the dataflow. The contents of extensions.directory are unpacked here. Defaults to /tmp/nifi-stateless-working if not specified.

Default Value

/tmp/nifi-stateless-working

Accepted Values

Required

false

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interva l.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false

S3 Sink properties reference

Review the following reference for a comprehensive list of the connector properties that are specific to the S3 Sink connector.

The properties listed in this reference must be added to the connector configuration with the following prefix:

```
parameter.[***CONNECTOR NAME***] Parameters:
```

In addition to the properties listed here, this connector also accepts certain properties of the Kafka Connect framework as well as the properties of the NiFi Stateless Sink connector. When creating a new connector using the Streams Messaging Manager UI, all valid properties are presented in the default configuration template. You can view the configuration template to get a full list of valid properties. In addition, for more information regarding the accepted properties not listed here, you can review the Apache Kafka documentation and the Stateless NiFi Sink properties reference.

Avro Schema Write Strategy

Description

Specifies whether the record schema is written to the output data file. Applicable only for Avro output (Output File Data Format is set to Avro).

Default Value

Embed Avro Schema

Accepted Values

Embed Avro Schema, Do Not Write Schema, HWX Content-Encoded Schema Reference

Required

false

AWS Access Key ID

Description

The Access Key ID to use for authentication to AWS.

Default Value

Accepted Values

Required

true

AWS Secret Access Key

Description

The Secret Access Key to use for authentication to AWS.

Default Value

Accepted Values

Required

true

Date Format

Description

Specifies the format to use when writing date fields to JSON or CSV.

Default Value

yyyy-MM-dd

Accepted Values

Required

true

Kafka Message Data Format

Description

Specifies the format of the messages the connector receives from Kafka. If set to Avro or JSON, record processing is enabled. Raw can be used for unstructured text or binary data

Default Value

Avro

Accepted Values

Avro, JSON, Raw

Required

true

Kerberos Keytab

Description

The fully-qualified filename of the kerberos keytab associated with the principal for accessing Schema Registry.

Default Value

The location of the default keytab which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Kerberos Principal

Description

The Kerberos principal used for authenticating to Schema Registry.

Default Value

default

Accepted Values

Required

true

Maximum File Size

Description

The maximum size of the output data file. No size limit is applied if this property is not specified. Example values: 100 MB, 1 GB.

Default Value

Accepted Values

Required

false

Maximum Number of Entries

Description

The maximum number of entries in the output data file. In the context of this property, entry can mean one of two things. If record processing is enabled (Kafka Message Data Format is set to Avro

or JSON), an entry is a record. Otherwise, entry means a Kafka message. Set this property to 1 if you want to create a separate output file for each Kafka message.

Default Value

1000000

Accepted Values

Required

true

Output File Data Format

Description

Specifies the format of the records written to the output file. Required when record processing is enabled (Kafka Message Data Format is set to Avro or JSON).

Default Value

Avro

Accepted Values

Avro, JSON, CSV, Parquet

Required

Output File Demarcator

Description

Specifies the character sequence for demarcating (delimiting) message boundaries when multiple Kafka messages are ingested into an output file as raw messages (no record processing). This property can only be used if Kafka Message Data Format is set to Raw. If you want to use newline as the delimiter, set this property to \n.

Default Value

Accepted Values

Required

false

Output Filename Pattern

Description

Specifies the structure of the name of the output file (S3 object name including the prefix). The pattern can contain string literal (fixed text) parts and one or more of the following expressions:

- ${filename.uuid}: Inserts a generated UUID in the filename.
- ${filename.timestamp}: Inserts the current timestamp in the filename.
- ${filename.sequence}: Inserts an incrementing sequence value in the filename.

In order to generate unique filenames, either ${filename.uuid} or ${filename.sequence} must be used in the pattern.

Examples:

- data_${filename.uuid}.json
- records_${filename.timestamp}_${filename.sequence}.avro

Default Value

${filename.uuid}

Accepted Values

Required

false

Output Filename Sequence Initial Value

Description

This property is used to configure the initial value of the ${filename.sequence} expression. The value you set in this property is not the initial value of the sequence. The sequence starts at the value of this property +1. For example, if you set this property to 0, the sequence starts at 1.

Default Value

0

Accepted Values

Required

false

Output Filename Sequence Padding Length

Description

Specifies the length of the ${filename.sequence} expression in characters. If the sequence has fewer characters than the value set in this property, it is padded with zeros (0). Padding is added to the left of the sequence.

Default Value

6

Accepted Values

Required

false

Output Filename Timestamp Format

Description

The timestamp format to use for the ${filename.timestamp} expression. For example, yyyyMMdd _HHmmss_SSS.

Default Value

Accepted Values

Required

false

Parquet Compression Type

Description

The type of compression used for writing parquet files. Required when Output File Data Format is Parquet.

Default Value

UNCOMPRESSED

Accepted Values

UNCOMPRESSED, SNAPPY, GZIP, LZO, BROTLI, LZ4, ZSTD

Required

false

S3 Bucket

Description

The name of the S3 Bucket where the output file is uploaded.

Default Value

Accepted Values

Required

false

S3 Region

Description

The Region of the S3 Bucket.

Default Value

Accepted Values

Required

true

S3 Storage Class

Description

The Storage Class to use for the output file on S3.

Default Value

Standard

Accepted Values

Standard, IntelligentTiering, StandardInfrequentAccess, OneZoneInfrequentAccess, Glacier, GlacierInstantRetrieval, DeepArchive, ReducedRedundancy

Required

true

Schema Access Strategy

Description

Specifies the strategy used for determining the schema of the Kafka record. The value you set here depends on the data format set in Kafka Message Data Format.

- If set to Schema Registry, the schema is read from Schema Registry. This setting can be used with both Avro and JSON formats.
- If set to Infer Schema, the schema is inferred based on the input file. This setting can only be used if Kafka Message Data Format is JSON.
- If set to Embedded Schema, the schema embedded in the input is used. This setting can only be used if Kafka Message Data Format is Avro.
- If set to HWX Content-Encoded Schema Reference, the schema is read from Schema Registry. This setting can only be used if Kafka Message Data Format is Avro. In this case the Avro messages are expected to have a reference to the schema in Schema Registry encoded within the message content.

This property is not used if record processing is disabled (Kafka Message Data Format is set to Raw).

Default Value

Schema Registry

Accepted Values

Schema Registry, Infer Schema, Embedded Schema, HWX Content-Encoded Schema Reference

Required

true

Schema Branch

Description

The name of the branch to use when looking up the schema in Schema Registry. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Name

Description

The name of the branch to use when looking up the schema in Schema Registry. Schema Branch and Schema Version cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Schema Registry URL

Description

The URL of the Schema Registry server. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

http://localhost:7788/api/v1

Accepted Values

Required

true

Schema Version

Description

The version of the schema to look up in Schema Registry. If Schema Registry is used and a schema version is not specified, the latest version of the schema is retrieved. Schema Branch and Schema V ersion cannot be specified at the same time. If one is specified, the other needs to be removed from the configuration. If Schema Registry is not used, this property must be completely removed from the configuration.

Default Value

Accepted Values

Required

false

Time Format

Description

Specifies the format to use when writing Time fields to JSON or CSV.

Default Value

HH:mm:ss

Accepted Values

Required

true

Timestamp Format

Description

Specifies the format to use when writing Timestamp fields to JSON or CSV.

Default Value

yyyy-MM-dd HH:mm:ss.SSS

Accepted Values

Required

true

Truststore Filename

Description

The fully-qualified filename of a truststore. This truststore is used to establish a secure connection with Schema Registry using HTTPS.

Default Value

The location of the default truststore which is empty and can only be used for unsecure connections.

Accepted Values

Required

true

Truststore Password

Description

The password used to access the contents of the truststore configured in the Truststore Filename property.

Default Value

password

Accepted Values

Required

true

Truststore Type

Description

The type of the truststore configured in the Truststore Filename property.

Default Value

JKS

Accepted Values

BCFKS, PKCS12, JKS

Required

true

cloudera.offset.flush.interval.ms

Description

Specifies the interval, in milliseconds, at which connector task offsets are committed for this connector. When set, this value overrides the role-level Offset Flush Interval (offset.flush.interval.ms) setting.

Default Value

Inherits role-level offset.flush.interval.ms when not set

Accepted Values

Required

false