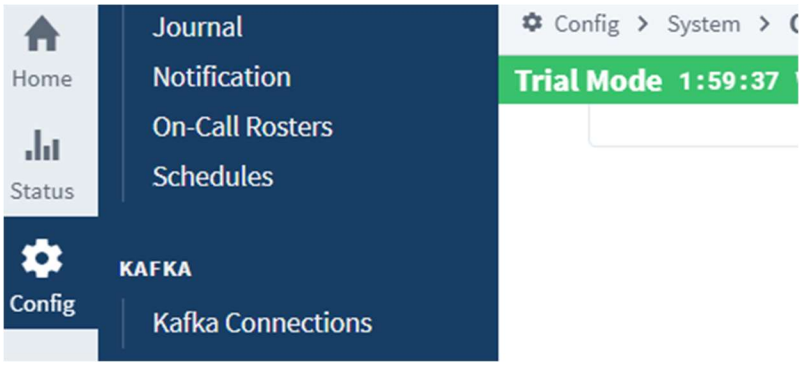
Kafka Connection Settings

To consume or publish records, users will need to first set up a Kafka connection. This can be done in the gateway after installing the module, under Config -> Kafka -> Kafka Connections.



Click the Create new KafkaConnectionSettings link, and a form will appear with multiple fields. The following properties are available to be filled:

Main	
Connection name	Connection name, must be unique
SSL Settings	
Keystore File Path	Path to keystore. Optional, leave blank if SSL disabled
Password	Password for the keystore. If SSL disabled, leave blank
Truststore File Path	Path to truststore. Optional, leave blank if SSL disabled
Password	Password for truststore. If SSL disabled, leave blank
SSL Enabled	Checkbox to enable or disable SSL
Custom Configurations	
Custom Consumer Configurations	Space separated key value pairs for consumer configuration. Must follow convention of key1=value1 key2=value2. Can be left blank. Will take a max of 4096 characters
Custom Consumer Configurations	Space separated key value pairs for producer configuration. Must follow convention of key1=value1 key2=value2. Can be left blank. Will take a max of 4096 characters

Each connection name needs to be unique. If SSL is desired, the SSL Enabled checkbox must be checked, the absolute path for the truststore/keystores file locations must be provided (e.g. /usr/local/bin/ignition/keystore.jks), and their respective password fields must be filled in as well.

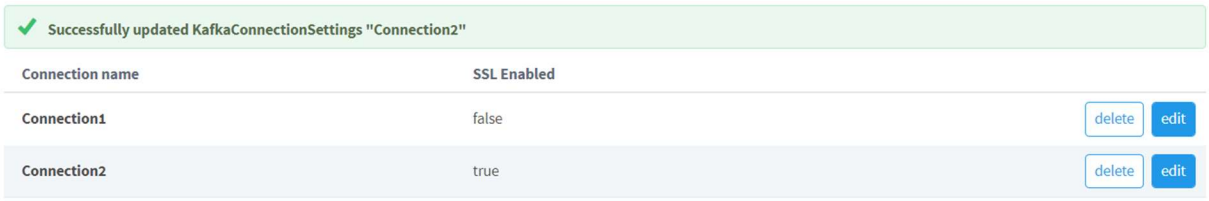
If any custom configurations for the consumers/producers are desired, they must follow the convention of

key1=value1 key2=value2

where the first key/value pair is paired with an equals sign without any spaces, and is separated from the next key/value pair by a single space

e.g. auto.commit.interval.ms=5000 max.poll.interval.ms=300000

If neither SSL nor custom configurations are desired, the name of the connection is sufficient. Click on the Create New KafkaConnectionSettings button, and the connection will then be generated.



In this way, multiple connections with different credential requirements can be generated.

system.kafka.getConsumer()

Creates a server connection and polls the Kafka server for recent messages. If SSL is enabled in the Kafka connection settings, the connection will be encrypted.

Syntax:

system.kafka.getConsumer(kafkaConnectionName, address, topic, groupname)

Parameters:

String kafkaConnectionName – The name of a configured Kafka connection setting, as defined in the gateway. If the connection name can't be found, the module will use the first viable Kafka connection to use instead.

String address – Kafka server address, including port.

String topic – Kafka topic.

String groupname – Kafka group id.

Returns:

List of dictionary objects, each representing a message. Each message object contains keys for 'value', 'timestamp', 'partition', 'offset','key' and 'headers'.

Scope:

Gateway, Perspective

Usage:

#python code

connName = "kafka"

addr = "192.168.1.1:9092" # if SSL is enabled, use the correct port

topic = "testTopic"

group = "somerandomuser"

toReturn = system.kafka.getConsumer(connName, addr, topic, group)

system.kafka.publish()

Creates a server connection and writes to the Kafka server. If SSL is enabled in the Kafka connection settings, the connection will be encrypted.

Syntax:

system.kafka.publish(kafkaConnectionName, address, topic, partition, timestamp, recordKey, recordValue, headerKeys, headerValues)

Parameters

String kafkaConnectionName – The name of a configured Kafka connection setting, as defined in the gateway. If the connection name can't be found, the module will use the first viable Kafka connection to use instead.

String address – Kafka server address, including port.

String topic – Kafka topic to send to.

int partition – Kafka partition to send to.

long timestamp – Record timestamp, in milliseconds since epoch.

String recordKey – Record key.

String recordValue – Record value.

String[] headerKeys – The header keys for the record, in list form.

String[] headerValues – The header values for the record, in list form. To match the header values, keep the lists in the same order.

Returns:

A dictionary object that contains some of the RecordMetadata properties from a successful write. Keys are 'offset', 'partition', 'timestamp' and 'topic'.

Scope:

Gateway, Perspective

Usage:

#python code

connName = "kafka"

addr = "192.168.1.1:9092" # if SSL is enabled, use the correct port

```
topic = "testTopic"

partition = 0

timestamp = system.date.toMillis(system.date.now())

key = "key1"

value = "value1"

headerKey = ["headerKey1"]

headerVal = ["headerVal1"]

system.kafka.publish(connName, addr, topic, partition, timestamp, key, value, headerKey,
headerVal)
```

system.kafka.seek()

Creates a server connection and retrieves the last N number of messages from a topic and partition. If SSL is enabled in the Kafka connection settings, the connection will be encrypted. Do not use this function if the same group is consuming data elsewhere at the same time, as there is a possibility for data loss.

Syntax:

> system.kafka.seek(kafkaConnectionName, address, topic, groupname, partition, numberOfMessages)

Parameters:

> String kafkaConnectionName – The name of a configured Kafka connection setting, as defined in the gateway. If the connection name can't be found, the module will use the first viable Kafka connection to use instead.
>
> String address – Kafka server address, including port.
>
> String topic – Kafka topic.
>
> String groupname – Kafka group id.
>
> int partition – Kafka partition to seek from.
>
> long numberOfMessages – Previous number of messages to read from.

Returns:

> List of dictionary objects, each representing a message. Each message object contains keys for 'value', 'timestamp', 'partition', 'offset', 'key' and 'headers'.

Scope:

> Gateway, Perspective

Usage:

#python code

connName = "kafka"

addr = "192.168.1.1:9092" # if SSL is enabled, use the correct port

topic = "testTopic"

group = "somerandomuser"

partition = 0

```
numMsgs = 5

toReturn = system.kafka.seek(connName, addr, topic, group, partition, numMsgs)
```

system.kafka.getTopicPartitions()

Grabs the partition numbers in a topic.

Syntax:

system.kafka.getTopicPartitions(kafkaConnectionName, address, topic, groupname)

Parameters:

String kafkaConnectionName – The name of a configured Kafka connection setting, as defined in the gateway. If the connection name can't be found, the module will use the first viable Kafka connection to use instead.

String address – Kafka server address, including port.

String topic – Kafka topic.

String groupname – Kafka group id.

Returns:

An ordered list of partition numbers..

Scope:

Gateway, Perspective

Usage:

```
#python code
connName = "kafka"
addr = "192.168.1.1:9092" # if SSL is enabled, use the correct port
topic = "testTopic"
group = "somerandomuser"
toReturn = system.kafka.getTopicPartitions(connName, addr, topic, group)
```