

NiFi-Example-Workflow

Introduction

Prerequisites

[Step 1 - Cloning the repository](#)

[Step 2 - Setting up kafka server](#)

[Step 3 - Produce data to Kafka](#)

[Step 4 - Consume kafka to NiFi](#)

[Step 5 - Extract the contents of flowfile into attributes](#)

[Step 6 - Update the attributes of flowfile to add processedAtTimestamp](#)

[Step 7 - Rename and pull the values back to flowfile contents using Jolt processor](#)

[Step 8 - Route the flowfiles to different destination based on attributes](#)

Introduction

This document is for sending data to MySQL from NiFi. This workflow is designed for covering multiple processors.

It talks about consuming data from kafka and send it to multiple destinations like MySQL, PostgreSQL, ElasticSearch, Kafka etc.

Prerequisites

Make sure python3, pip3, git, docker and docker compose are installed on your system.

If not, run as root the following on Ubuntu based system

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
```

```

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/ke
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install git python3-pip docker-ce docker-ce-cli co

```

For debian systems, follow

```

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/ke
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install git python3-pip docker-ce docker-ce-cli co

```

Refer to here <https://docs.docker.com/engine/install/> for installing docker components on other systems.

Step 1 - Cloning the repository

Clone the git repository which contains everything related to this lab example.

```
git clone https://github.com/cloudpoet-in/nifi-training.git
```

Step 2 - Setting up kafka server

Setup kafka server if you dont have it already.

In git repo, navigate to directory `nifi-training/sandbox-deployments/kafka` and then run

```
docker compose up -d
```

This will start a kafka server which will listen on `localhost:9092`

Step 3 - Produce data to Kafka

Produce some data to kafka using simulator script.

Simulator script requires a python library called as `confluent-kafka` which can be installed by running

```
pip3 install confluent-kafka

# if this doesnt work, run
# pip3 install confluent-kafka --break-system-packages
```

Once dependency is ready, navigate to `nifi-training/kafka-producer-simulator` and run

```
python3 kafka_send.py
```

This will continuously produce some dummy records to kafka running at `localhost:9092` in topic named `nifi-dummy` . Keep the script running in the background as it will produce data every 5 seconds. Feel free to update it as per needs.

Successful logs looks like this

```

root@nifi-2:~/nifi-training/simulator# python3 kafka_send.py
Message produced: {"source_type": "warehouse", "source_name": "Warehouse 1", "temperature": -2.21, "pressure": 980.54, "timestamp": 1727170984}
Message produced: {"source_type": "city", "source_name": "City Center", "temperature": 3.26, "pressure": 1035.9, "timestamp": 1727170990}
Message produced: {"source_type": "warehouse", "source_name": "Warehouse 1", "temperature": -11.15, "pressure": 1009.63, "timestamp": 1727170995}
Message produced: {"source_type": "city", "source_name": "City Center", "temperature": 16.29, "pressure": 975.28, "timestamp": 1727171000}
Message produced: {"source_type": "city", "source_name": "City Center", "temperature": 34.84, "pressure": 1020.96, "timestamp": 1727171005}
Message produced: {"source_type": "factory", "source_name": "Factory A", "temperature": -15.37, "pressure": 958.01, "timestamp": 1727171010}
Message produced: {"source_type": "city", "source_name": "City Suburb", "temperature": -14.69, "pressure": 1026.51, "timestamp": 1727171015}
Message produced: {"source_type": "warehouse", "source_name": "Warehouse 2", "temperature": -9.9, "pressure": 986.02, "timestamp": 1727171020}
Message produced: {"source_type": "factory", "source_name": "Factory B", "temperature": 21.71, "pressure": 970.93, "timestamp": 1727171025}
Message produced: {"source_type": "warehouse", "source_name": "Warehouse 1", "temperature": 9.71, "pressure": 993.14, "timestamp": 1727171030}
Message produced: {"source_type": "factory", "source_name": "Factory B", "temperature": 36.37, "pressure": 953.31, "timestamp": 1727171035}
Message produced: {"source_type": "factory", "source_name": "Factory A", "temperature": 26.05, "pressure": 955.45, "timestamp": 1727171040}
Message produced: {"source_type": "city", "source_name": "City Center", "temperature": 9.52, "pressure": 967.77, "timestamp": 1727171045}
Message produced: {"source_type": "city", "source_name": "City Center", "temperature": -2.29, "pressure": 976.77, "timestamp": 1727171050}
Message produced: {"source_type": "city", "source_name": "City Suburb", "temperature": 24.37, "pressure": 1048.46, "timestamp": 1727171055}
Message produced: {"source_type": "factory", "source_name": "Factory A", "temperature": -9.72, "pressure": 978.79, "timestamp": 1727171060}

```

Step 4 - Consume kafka to NiFi

Drag a processor in nifi named **ConsumeKafka_2_6** and configure it as shown. Add unique GroupID.

Configure Processor | ConsumeKafka_2_6 1.27.0

Invalid

SETTINGS | SCHEDULING | **PROPERTIES** | RELATIONSHIPS | COMMENTS

Required field

Property	Value
Kafka Brokers	localhost:9092
Topic Name(s)	nifi-dummy
Topic Name Format	names
Group ID	testing123
Commit Offsets	true
Max Uncommitted Time	1 secs
Honor Transactions	true
Message Demarcator	No value set
Separate By Key	false
Security Protocol	PLAINTEXT
SASL Mechanism	GSSAPI
Kerberos Credentials Service	No value set

CANCEL APPLY

The property named **offset Reset** can be changed to consume from end of topic or start of the topic.

Configure Processor | ConsumeKafka_2_6 1.27.0

Invalid

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property

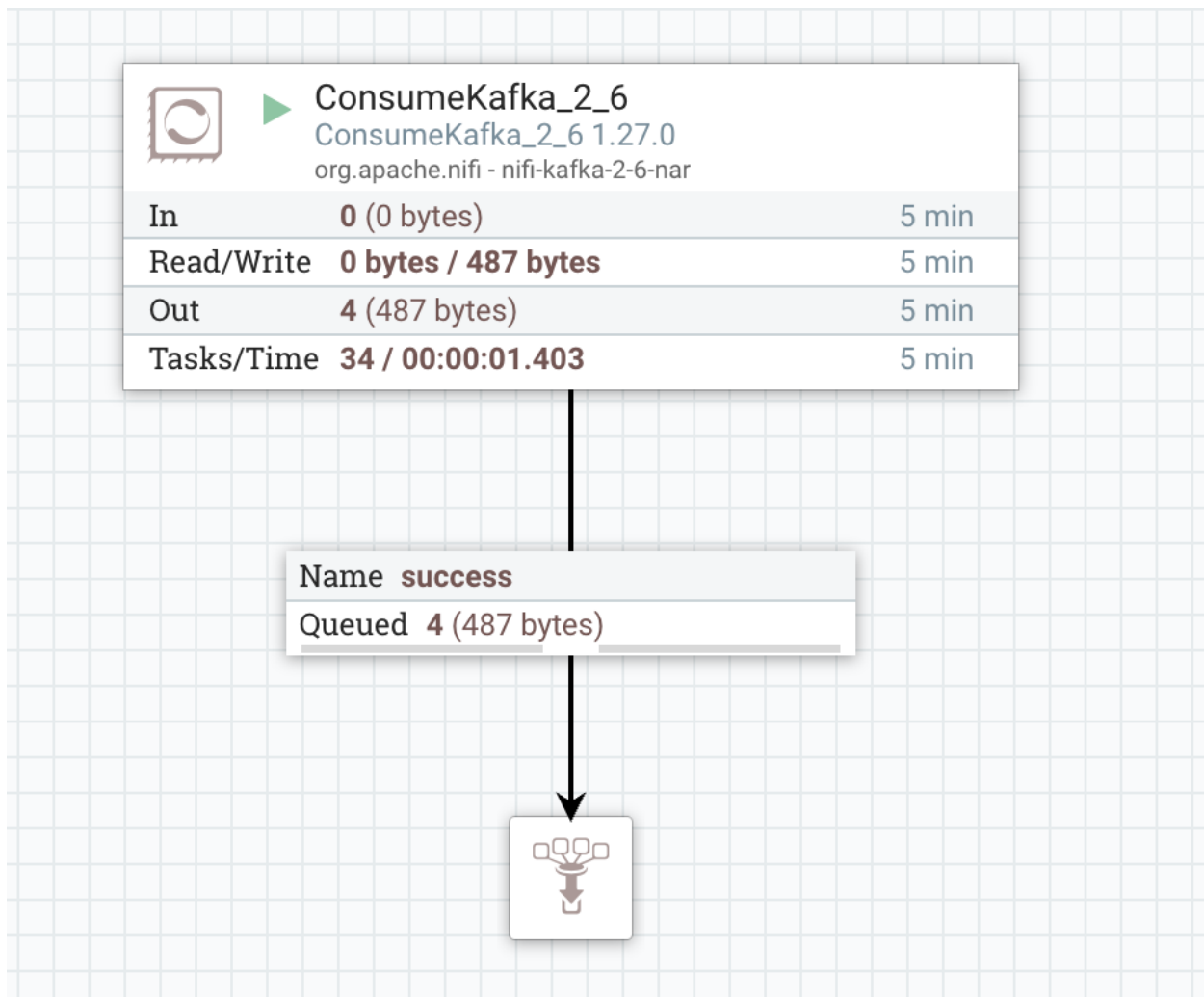
Value

Kerberos Credentials Service	No value set
Kerberos User Service	No value set
Kerberos	Allows you to manage the condition when there is no initial offset in Kafka or if the current offset does not exist any more on the server (e.g. because that data has been deleted). Corresponds to Kafka's 'auto.offset.reset' property.
Kerberos	Default value: latest
Kerberos	No value set
SSL Context	Expression language scope: Not Supported
Key Attribute	Sensitive property: false
Offset Reset	latest
Message Header Encoding	UTF-8
Headers to Add as Attributes (Regex)	No value set
Max Poll Records	10000
Communications Timeout	60 secs

CANCEL

APPLY

Make sure you are able to consume from kafka as shown



Step 5 - Extract the contents of flowfile into attributes

Our records contain 3 types of data i.e. factory, warehouse, city based.

e.g.

Sample - 1

```
{  
  "source_type": "factory",  
  "source_name": "Factory A",  
  "temperature": 20.1,  
  "pressure": 1040.88,  
}
```

```
"timestamp": 1727171070
}
```

Sample - 2

```
{
  "source_type": "warehouse",
  "source_name": "Warehouse 1",
  "temperature": 0.37,
  "pressure": 957.9,
  "timestamp": 1727171080
}
```

Sample - 3

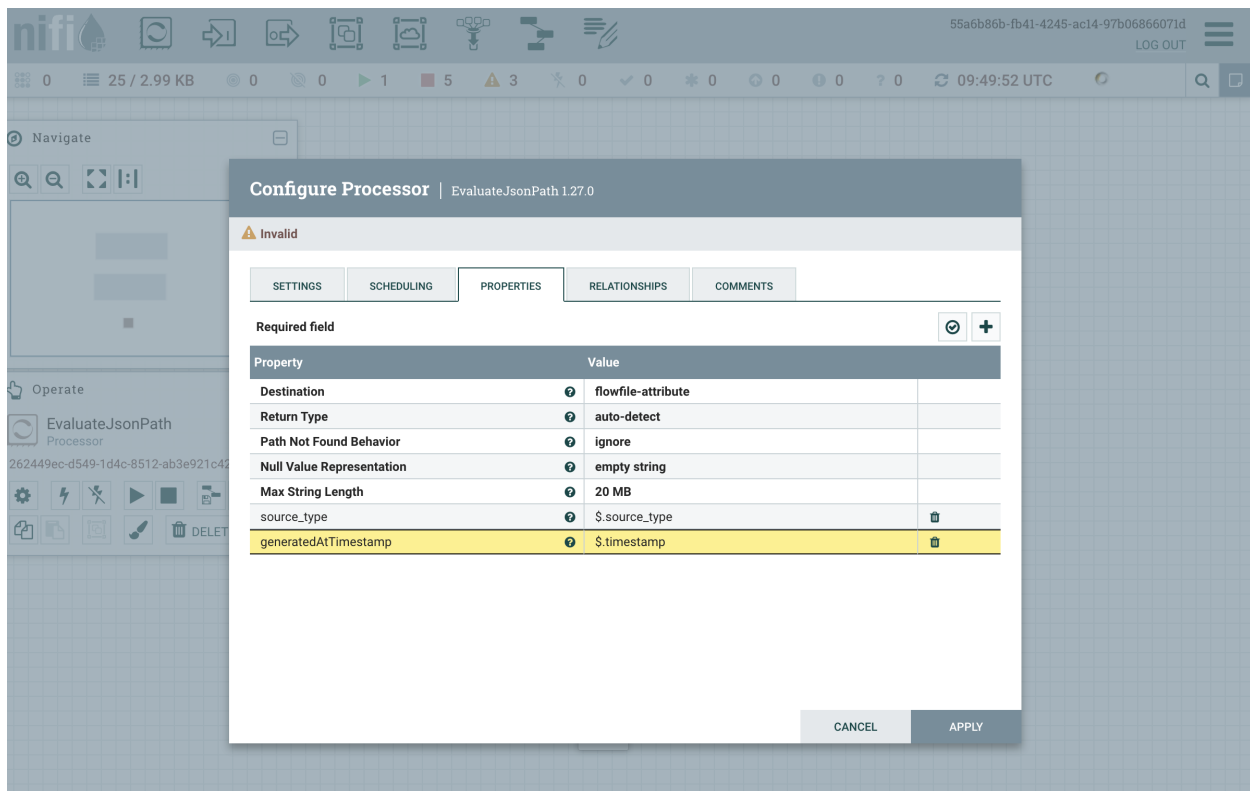
```
{
  "source_type": "city",
  "source_name": "City Center",
  "temperature": 25.23,
  "pressure": 1024.26,
  "timestamp": 1727171105
}
```

Next, we need to extract keys from these json into the flowfile attributes.

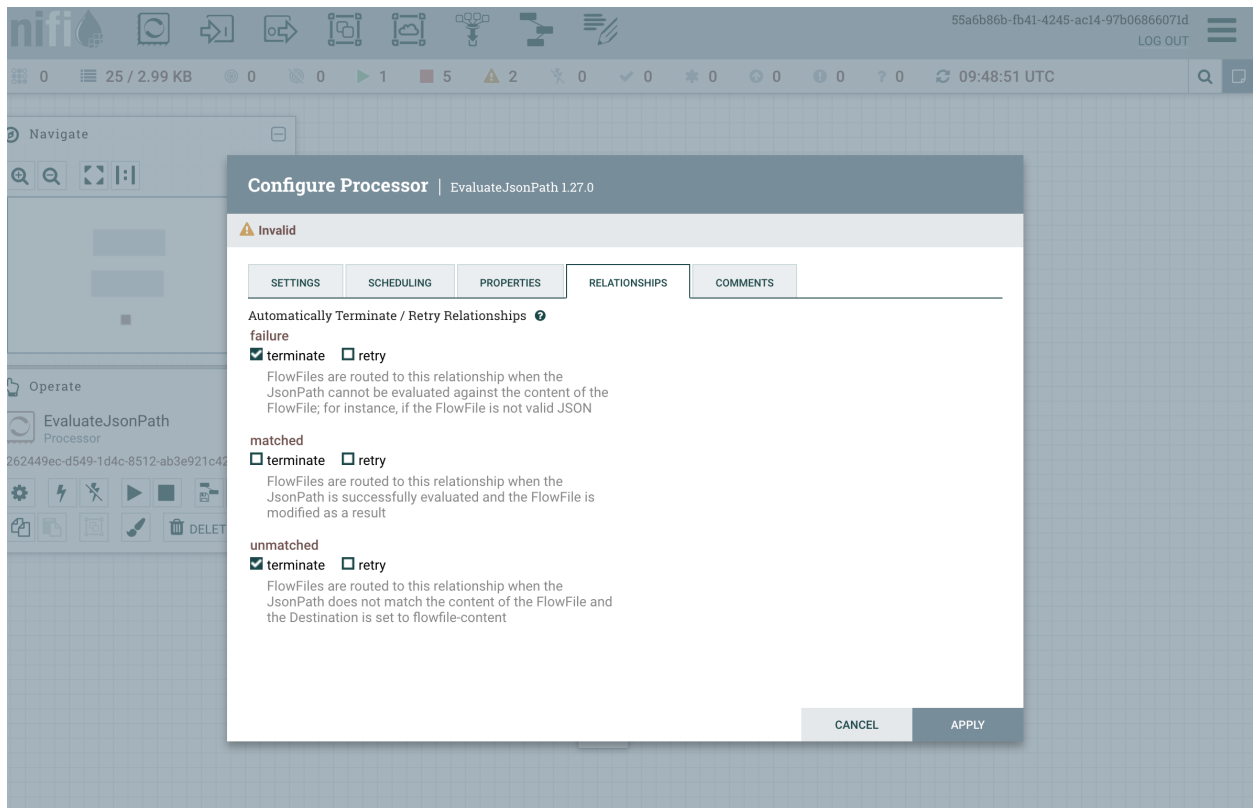
For that, we need to use a processor called as `EvaluateJsonPath`

Let us extract `timestamp` and `source_type` into the flowfile attributes by configuring processor as shown. Add properties manually by clicking + icon in the top right of the processor


Set the destination to `flowfile-attribute`



Terminate the failure and unmatched relationship since we dont need to process those records at all in this example.



Make sure everything looks on UI like shown and there are no errors.




▶

ConsumeKafka_2_6
ConsumeKafka_2_6 1.27.0
org.apache.nifi - nifi-kafka-2-6-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	297 / 00:00:03.200	5 min

Name **success**

Queued 0 (0 bytes)



▶

EvaluateJsonPath
EvaluateJsonPath 1.27.0
org.apache.nifi - nifi-standard-nar

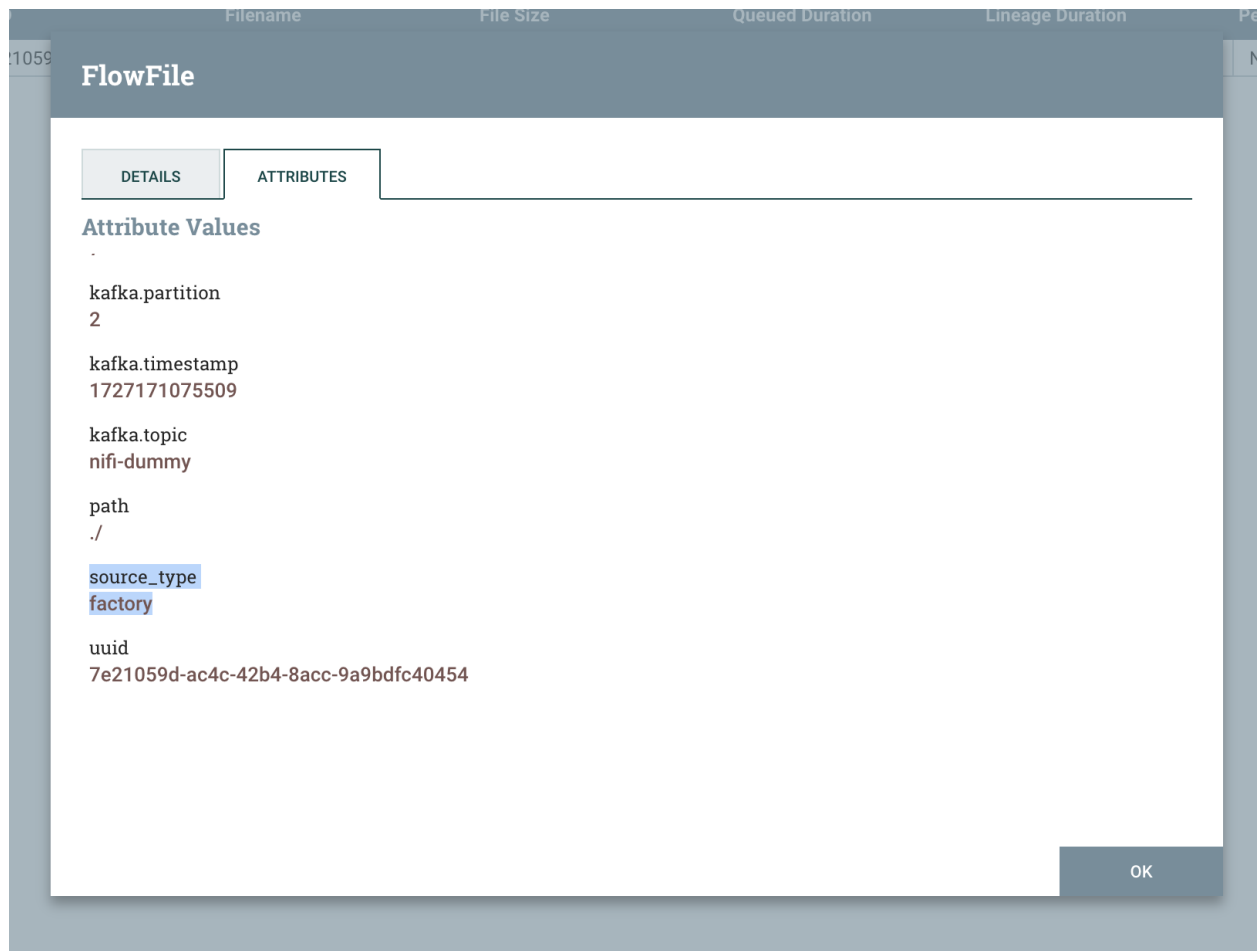
In	25 (2.99 KB)	5 min
Read/Write	2.99 KB / 0 bytes	5 min
Out	25 (2.99 KB)	5 min
Tasks/Time	25 / 00:00:00.050	5 min

Name **matched**

Queued 25 (2.99 KB)



Also check the attributes of processed flowfiles to see if they contain our custom attributes.



Step 6 - Update the attributes of flowfile to add processedAtTimestamp

Add a processor named UpdateAttributes and configure it as shown. Add a function `${now()}`

Configure Processor | UpdateAttribute 1.27.0

Invalid

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

✓

+

Property		Value	
Delete Attributes Expression	?	No value set	
Store State	?	Do not store state	
Stateful Variables Initial Value	?	No value set	
Cache Value Lookup Cache Size	?	100	
processedAtTimestamp	?	\${now()}	<div>✕</div>

⚙️

ADVANCED

CANCEL

APPLY

Make sure it looks good



ConsumeKafka_2_6

ConsumeKafka_2_6 1.27.0

org.apache.nifi - nifi-kafka-2-6-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	298 / 00:00:03.206	5 min

Name **success**

Queued 0 (0 bytes)



EvaluateJsonPath

EvaluateJsonPath 1.27.0

org.apache.nifi - nifi-standard-nar

In	25 (2.99 KB)	5 min
Read/Write	2.99 KB / 0 bytes	5 min
Out	25 (2.99 KB)	5 min
Tasks/Time	25 / 00:00:00.050	5 min

Name **matched**

Queued 0 (0 bytes)



UpdateAttribute

UpdateAttribute 1.27.0

org.apache.nifi - nifi-update-attribute-nar

In	25 (2.99 KB)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	25 (2.99 KB)	5 min
Tasks/Time	25 / 00:00:00.040	5 min

Name **success**

Queued 25 (2.99 KB)



Check the attributes if it contains our custom attribute.

The screenshot shows the NiFi interface with a FlowFile selected. The 'ATTRIBUTES' tab is active, displaying a list of key-value pairs. The 'processedAtTimestamp' attribute is highlighted, showing the value 'Tue Sep 24 09:54:25 UTC 2024'. Other attributes include 'kafka.timestamp', 'kafka.topic', 'path', 'source_type', 'factory', and 'uuid'. The background shows a table of FlowFiles with columns for UUID, Filename, File Size, Queued Duration, Lineage Duration, and Penalty.

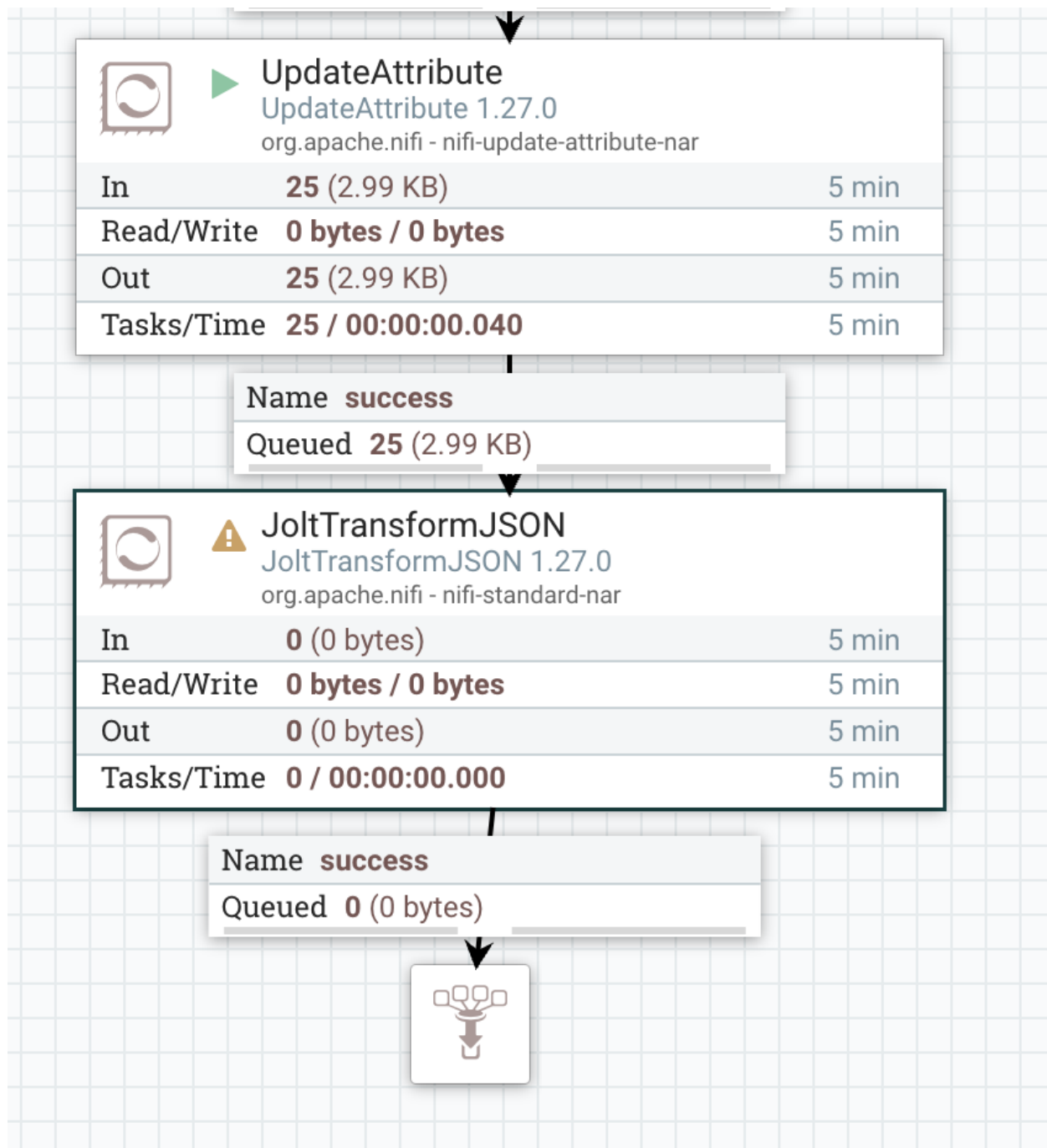
UUID	Filename	File Size	Queued Duration	Lineage Duration	Penalty
7e21059					No
fe8406a					No
9f2677e					No
4a6a8f0					No
3a50bde					No
9b26d93					No
b6e99d2					No
0b68354					No
d234221					No
d7b98ec					No
cb691eb					No
322df4a					No
9c2fb67					No
7c4a514					No
18897a4					No
c540a30					No
2b446ac					No
51bdb4b					No
b4adc9b					No
20356e3					No
a1c7558					No
b519ea5					No
e40d414					No
c9dc2194-2f78-40f3-89ba-...	c9dc2194-2f78-40f3-89ba-...	122.00 bytes	00:00:25.884	00:08:20.681	No
00401b30-c760-4b0d-00e4-...	00401b30-c760-4b0d-00e4-...	122.00 bytes	00:00:25.884	00:08:15.600	No

The timestamp is in format `Tue Sep 24 09:54:25 UTC 2024` . **We might need our own format sometimes to support different systems.**

In that case, `${now()}` function can be updated as `${now():format('yyyy-MM-dd HH:mm:ss')}`

Step 7 - Rename and pull the values back to flowfile contents using Jolt processor

Use a processor named `JoltTransformRecord` next in line.



Jolt is very powerful processor to manipulate JSON. It required Jolt specification which is an expression language of its own.

To learn more about Jolt, refer <https://jolt-demo.appspot.com/#inception>

In our case, we want to manipulate JSON as shown where

INPUT

```
{
  "source_type": "factory",
  "source_name": "Factory A",
  "temperature": 20.1,
  "pressure": 1040.88,
  "timestamp": 1727171070
}
```

Transformed OUTPUT

```
{
  "SOURCE_TYPE": "factory",
  "SOURCE_NAME": "Factory A",
  "TEMPERATURE": 20.1,
  "PRESSURE": 1040.88,
  "TIMESTAMP": 1727171070
}
```

For development, click the advanced button in the bottom left corner.

Configure Processor | JoltTransformJSON 1.27.0

Invalid

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

✓

+

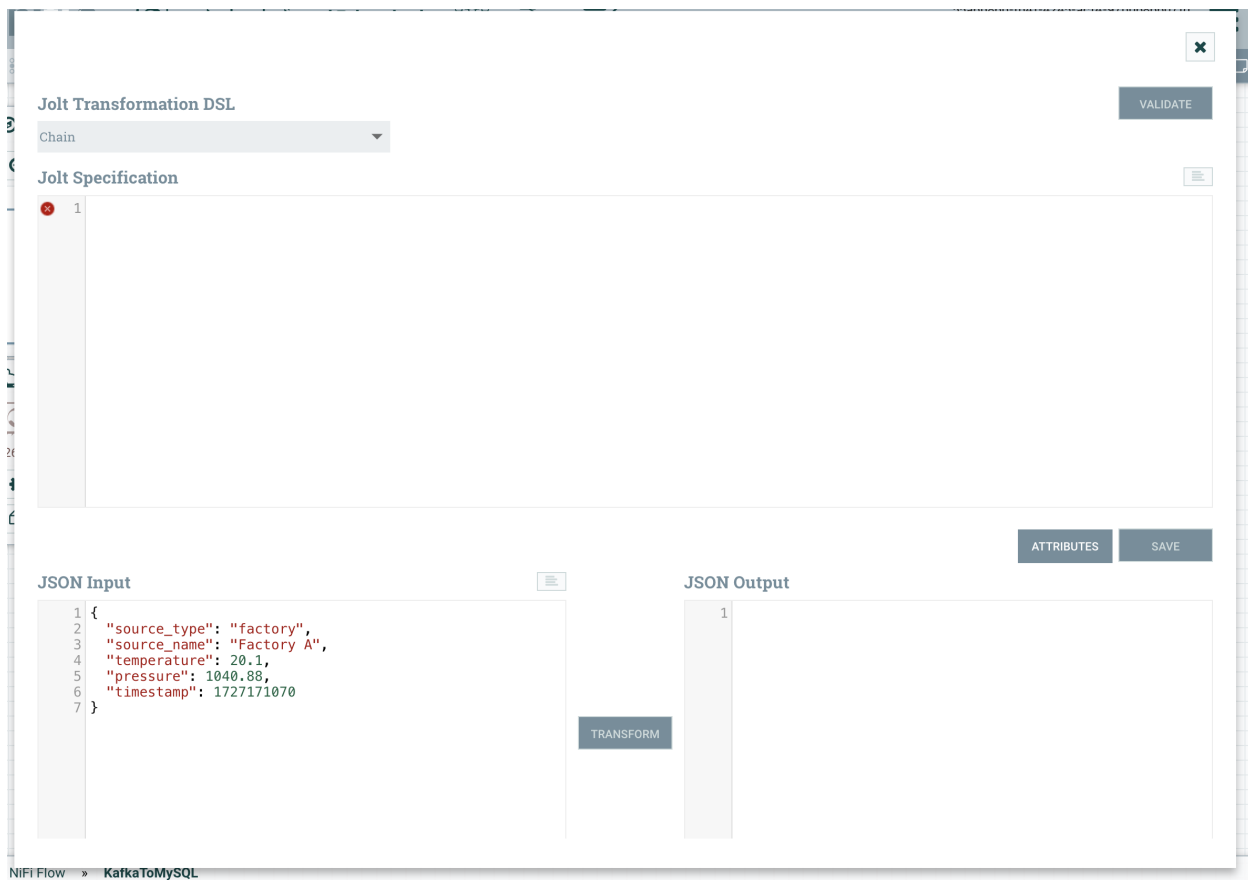
Property		Value
Jolt Transformation DSL	?	Chain
Jolt Specification	?	No value set
Transform Cache Size	?	1
Pretty Print	?	false
Max String Length	?	20 MB

⚙️ ADVANCED

CANCEL

APPLY

It will open Jolt DSL window where we can feed our custom JSON and put some specification to test it out.



e.g. to rename our keys, we can put specification

```
[{
  "operation": "shift",
  "spec": {
    "source_type": "SOURCE_TYPE",
    "source_name": "SOURCE_NAME",
    "temperature": "TEMPERATURE",
    "pressure": "PRESSURE",
    "timestamp": "TIMESTAMP"
  }
}]
```

So results will look like,

✕

Jolt Transformation DSL

Specification is Valid VALIDATE

Chain ▾

Jolt Specification

☰

```

1 [{
2   "operation": "shift",
3   "spec": {
4     "source_type": "SOURCE_TYPE",
5     "source_name": "SOURCE_NAME",
6     "temperature": "TEMPERATURE",
7     "pressure": "PRESSURE",
8     "timestamp": "TIMESTAMP"
9   }
10 }]

```

JSON Input

☰

JSON Output

ATTRIBUTES

SAVE

```

1 {
2   "source_type": "city",
3   "source_name": "City Center",
4   "temperature": 25.23,
5   "pressure": 1024.26,
6   "timestamp": 1727171105
7 }

```

TRANSFORM

```

1 {
2   "SOURCE_TYPE": "city",
3   "SOURCE_NAME": "City Center",
4   "TEMPERATURE": 25.23,
5   "PRESSURE": 1024.26
6 }

```

Now, we have to fetch some extra properties from flowfile attributes. It can be done by adding attribute field to specifications using default operation as shown

```

[ {
  "operation": "shift",
  "spec": {
    "source_type": "SOURCE_TYPE",
    "source_name": "SOURCE_NAME",
    "temperature": "TEMPERATURE",
    "pressure": "PRESSURE",
    "timestamp": "TIMESTAMP"
  }
}, {
  "operation": "default",
  "spec": {
    "EntryProcessedAt": "${processedAtTimestamp}",

```

```

    "KAFKA_TIMESTAMP": "${kafka.timestamp:format('yyyy-MM-dd')}
  }
}]

```

here, `processedAtTimestamp` and `kafka.timestamp` are the names of attribute that we want inside content. We are processing them inside JOLT.

With above specs, our records will look like this

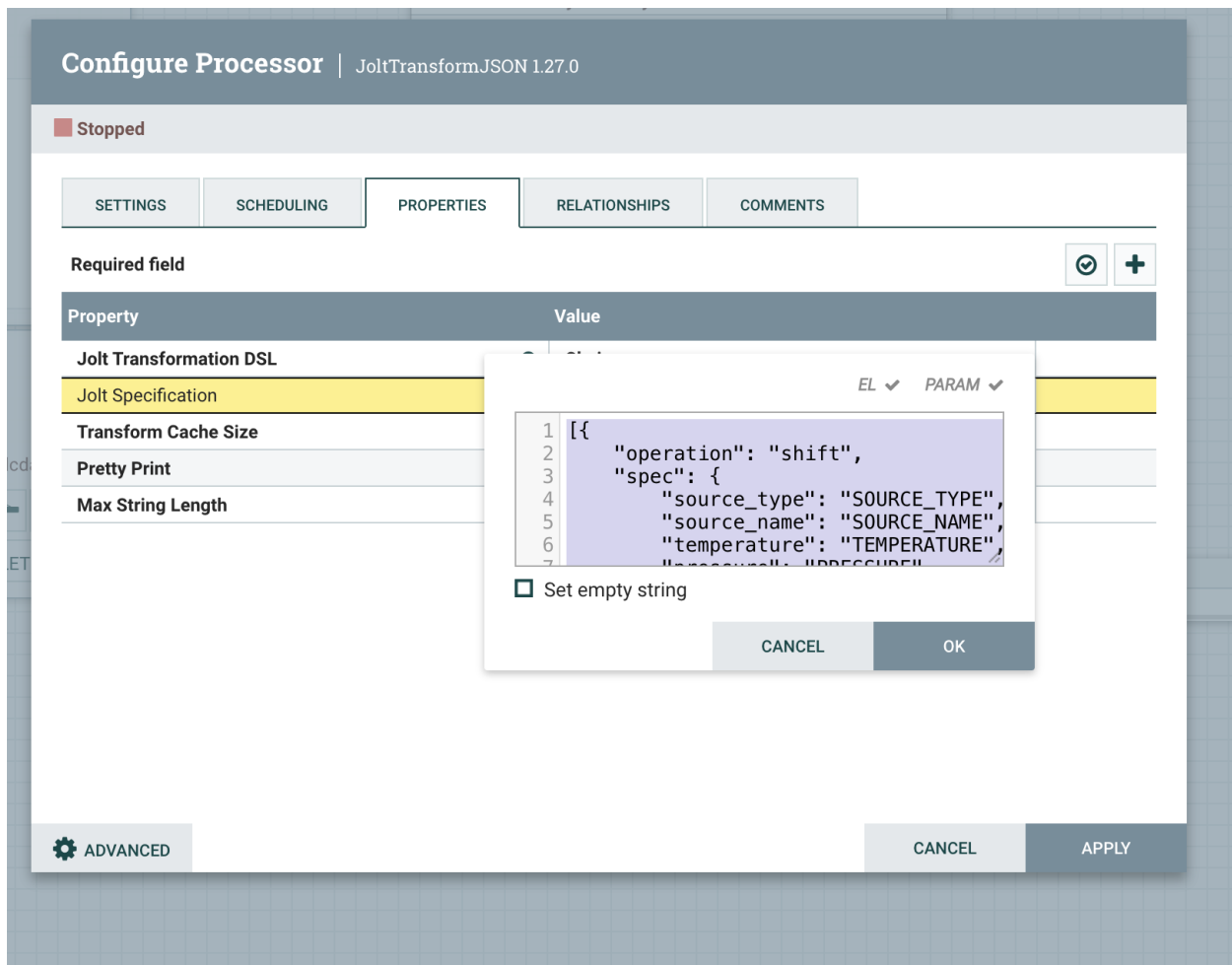
View as: **formatted** ▼

```

1 {
2   "SOURCE_TYPE" : "factory",
3   "SOURCE_NAME" : "Factory A",
4   "TEMPERATURE" : -13.77,
5   "PRESSURE" : 985.06,
6   "TIMESTAMP" : 1727171075,
7   "KAFKA_TIMESTAMP" : "2024-09-24 09:44:35",
8   "EntryProcessedAt" : "Tue Sep 24 09:54:25 UTC 2024"
9 }

```

Jolt processor, put the spec here.



Step 8 - Route the flowfiles to different destination based on attributes

Add processor named `RouteOnAttribute` and make its properties as shown.

Out0 (0 bytes)5 min

Configure ProcessorRouteOnAttribute 1.27.0

Stopped

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property

Value

Property	Value	
Routing Strategy	Route to Property name	
city	<code>\${source_type:contains("city")}</code>	
warehouse	<code>\${source_type:contains("warehouse")}</code>	
factory	<code>\${source_type:contains("factory")}</code>	

CANCEL

APPLY