Kafka transport handler

The Kafka transport handler allows the reading and writing from/to Kafka.

Needs the feature `de.uniol.inf.is.odysseus.wrapper.kafka.feature`

Options

Mandatory for reading and writing:

- **topic:** String defining the kafka topic to read/write
- **messagetype:** either 'string' or 'bytearray'; defines the type of values written to/read from Kafka
- **bootstrap.servers:** comma separated list of Kafka brokers (host:port)

Optional:

- All other properties possible for Kafka Producers and Consumers respectively (see Kafka documentation for possible properties)

Please note that the parameters are directly forwarded to kafka, so, please also check the Kafka documentation.

Example

PQL for writing

File Transport Handler

```
#PARSER PQL
#RUNQUERY
input = ACCESS({
            source='Input',
            wrapper='GenericPull',
            transport='file',
            protocol='SimpleCSV',
            datahandler='Tuple',
            options=[
              ['filename', '${PROJECTPATH}/input.csv'],
              ['csv.delimiter', ','],
              ['csv.trim', 'true']
            ],
            schema=[
              ['text', 'String']

]


}

        )

output = SENDER({
              protocol = 'SimpleCSV',
              transport = 'Kafka',
              sink = 'Test',
              wrapper = 'GenericPush',
              options = [
                ['topic', 'test'],
                ['messagetype', 'string'],
                ['bootstrap.servers', 'localhost:9092']
              ]
            },
            input
          )
```

PQL for reading

Important: There is no GenericPull version of the transport handler. So alway us GenericPush!

File Transport Handler

```
#PARSER PQL
#ADDQUERY
input := ACCESS({
            source='test',
            wrapper='GenericPush',
            transport='Kafka',
            protocol='SimpleCSV',
            datahandler='Tuple',
            options=[
              ['topic', 'test'],
              ['messagetype', 'string'],
              ['bootstrap.servers', 'localhost:9092']
            ],
            schema=[
              ['text', 'String']

]


}

        )
```

Szenario: Reading from one Kafka and writing to another

Odysseus can be used to copy all input of one Kafka cluster to another:

In this example we use two docker-compose files (e.g. in folder kafka1 and kafka2):

```
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - 22181:2181

  kafka:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper
    ports:
      - 29092:29092
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
```

and

```
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - 22182:2181

  kafka:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper
    ports:
      - 29093:29093
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:29093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
```

Start first this query: This reads from kafka1 and writes to kafka2

```
#PARSER PQL
#QNAME ReadingAndWrinting
#ADDQUERY
input = ACCESS({
              protocol = 'Odysseus',
              transport = 'Kafka',
              source = 'ReadFromKafka1',
              wrapper = 'GenericPush',
              datahandler = 'Tuple',
              READMETADATA = true,
              options = [
                ['topic', 'test'],
                ['messagetype', 'bytearray'],
                ['metadata.broker.list', 'localhost:29092'],
                ['bootstrap.servers','localhost:29092'],
                ['group.id','marco1']
              ],
              SCHEMA = [
                      ['time','STARTTIMESTAMP']
              ]
            })

output = SENDER({
              protocol = 'Odysseus',
              transport = 'Kafka',
              sink = 'WriteToKafka2',
              wrapper = 'GenericPush',
              WRITEMETADATA = true,
              options = [
                ['topic', 'test_converted'],
                ['messagetype', 'bytearray'],
                ['metadata.broker.list', 'localhost:29093'],
                ['bootstrap.servers','localhost:29093'],
                ['group.id','marco1']
              ]
            },
            input
          )
```

And after that, start a query, that writes into the first kafka cluster.

```
#PARSER PQL
#QNAME Writing
#RUNQUERY

input = TIMER({PERIOD = 1000, TIMEFROMSTART = true, SOURCE = 'source'})

output = SENDER({
              protocol = 'Odysseus',
              transport = 'Kafka',
              sink = 'WriteToKafka1',
              wrapper = 'GenericPush',
              WRITEMETADATA = true,
              options = [
                ['topic', 'test'],
                ['messagetype', 'bytearray'],
                ['metadata.broker.list', 'localhost:29092'],
                ['bootstrap.servers','localhost:29092'],
                ['group.id','marco1']
              ]
            },
            input
          )
```