



Starting a Denodo Scheduler Job based on Kafka Events

Revision 20230525

NOTE

This document is confidential and proprietary of **Denodo Technologies**.
No part of this document may be reproduced in any form by any means without prior written authorization of **Denodo Technologies**.

Copyright © 2024
Denodo Technologies Proprietary and Confidential

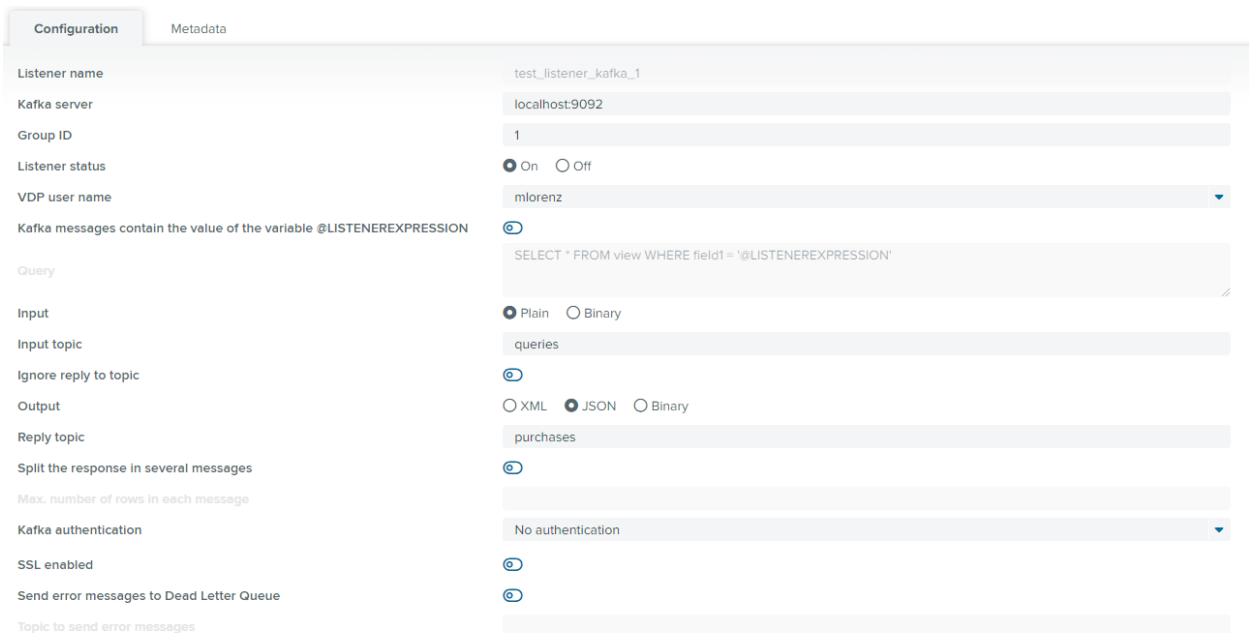
Denodo's Kafka Listener allows Denodo to listen to Kafka Events in real time. These Kafka events can be VQL Queries that Denodo can execute. The Goal is to use VQL Query Kafka events to query a view in Denodo that will call the Scheduler REST API to do a certain task, in our case starting a Scheduler job.

1 CREATE A KAFKA LISTENER

To be able to listen to Kafka events in the first place, it is necessary to configure a Kafka Listener.

- In either Denodo Design Studio or Denodo Virtual Dataport Administration Tool go to File > New > Listener > Kafka Listener
- Configure your Kafka Listener, for this use case there are not many requirements:
 - Make sure Listener status is turned on.
 - Select the VDP User under whom you want to execute the queries passed through Kafka.
 - Specify your Input Topic and Reply Topic.

For more detailed information on how to configure a Kafka listener go to the Section [Creating Kafka Listeners](#) of the Virtual DataPort Administration Guide



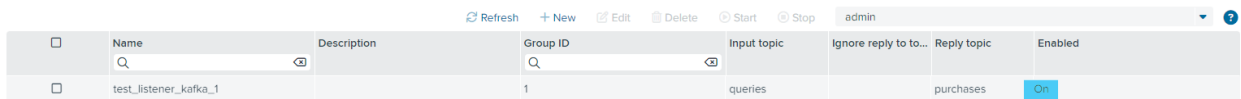
The screenshot shows the configuration interface for a Kafka listener. The 'Configuration' tab is active, and the listener name is 'test_listener_kafka_1'. The Kafka server is set to 'localhost:9092' and the Group ID is '1'. The listener status is 'On'. The VDP user name is 'mlorenz'. The Kafka messages contain the value of the variable '@LISTENEREXPRESSION'. The query is 'SELECT * FROM view WHERE field1 = '@LISTENEREXPRESSION''. The input is 'Plain' and the input topic is 'queries'. The output is 'JSON' and the reply topic is 'purchases'. The Kafka authentication is 'No authentication'. The SSL enabled checkbox is unchecked. The 'Send error messages to Dead Letter Queue' checkbox is checked. The topic to send error messages is empty.

When creating the data sources and views to call the REST API you will have the **option to work with an interpolation variable**. This means you only need to create one view instead of multiple views if you want to start multiple scheduler jobs. You will find more information about that later in this topic. If you decide to do so you should also specify the SELECT statement by checking the option Kafka messages contain the variable @LISTENEREXPRESSION

```
SELECT * FROM <DenodoDataBase>.rnschedulerjob WHERE post_body = '@LISTENEREXPRESSION'
```

This allows you to only insert the Post_Body of the PUT REST API into the Kafka Events as parameters.

To make sure your Kafka Listener is running go to Tools > Listeners > Kafka.



The screenshot shows a table of Kafka listeners in the Denodo interface. The table has columns for Name, Description, Group ID, Input topic, Ignore reply to..., Reply topic, and Enabled. A single listener named 'test_listener_kafka_1' is listed with Group ID '1', Input topic 'queries', and Reply topic 'purchases'. The 'Enabled' column for this listener shows a blue 'On' button.

	Name	Description	Group ID	Input topic	Ignore reply to...	Reply topic	Enabled
<input type="checkbox"/>	test_listener_kafka_1		1	queries		purchases	On

- Under the Enabled Column you can see the status of the Listener

2 DENODO SCHEDULER REST API

Next we need to work with the Scheduler REST API. You can access the documentation under the following URL:

```
http://<host>:<port>/webadmin/denodo-scheduler-admin/swagger-ui/index.html
```

Your Scheduler Server and Scheduler Administration Tool have to be started to access the URL.

You can for example run the

```
GET /webadmin/denodo-scheduler-admin/public/api/jobs
```

Method to retrieve the id's and project id's of your jobs, which you will need to start them.

To trigger a Scheduler job there are two PUT APIs available. For changing the status of a single job it does not really matter which one you pick. However you can only use one of them to change the status of multiple jobs at the same time

- PUT `/webadmin/denodo-scheduler-admin/public/api/projects/{projectID}/jobs/{jobID}/status`
 - This API endpoint changes the job status of a single job.
- PUT `/webadmin/denodo-scheduler-admin/public/api/projects/{projectID}/jobs/status`
 - You can use this endpoint to change the status of multiple jobs, e.g. starting multiple jobs at the same time.

You can use the site to prebuild your Request URL which you need in the next section of this guide. In the example below we choose the action start to trigger a Scheduler Job with the ID 108.

PUT /webadmin/denodo-scheduler-admin/public/api/projects/{projectID}/jobs/status Changes the jobs status by starting, stopping, disabling or enabling them. Expects a JSON body: {"action": "<action>", "IDs": "<comma separated list of job ids>"} where action can be: "start", "start_with_state", "start_with_dependencies", "stop", "enable" or "disable"

Parameters Cancel

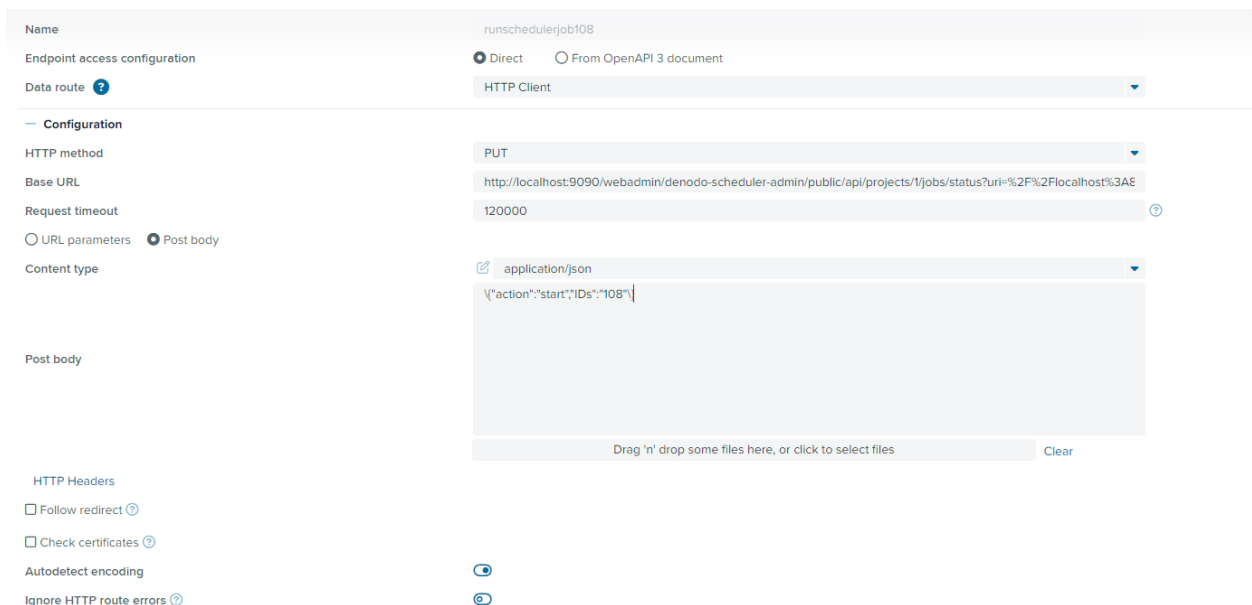
Name	Description
jobsToChangeStatus * required string (body)	jobsToChangeStatus Edit Value Model
	<pre>{"action": "start", "IDs": "108"}</pre>
	Cancel
	Parameter content type application/json
projectID * required integer(\$int32) (path)	projectID 1
uri string (query)	Scheduler server //localhost:8000

Request URL
http://localhost:9090/webadmin/denodo-scheduler-admin/public/api/projects/1/jobs/status?uri=%2Flocalhost%3A8000

3 CREATING A JSON DATA SOURCE

With the Request URL you can now create a JSON data source in Denodo.

- Got to File > New > Data source > JSON.
- Provide a Name to the connection.
- Choose HTTP Client as your Data route.
- Choose PUT as your HTTP Client.
- Post your Request URL in the Base URL field.
- In the Post body section choose application/json as your Content type.
 - Enter the JSON Body as described in the REST API documentation
 - IMPORTANT: add the backslash “\” character before your curly braces “{” / “}”
 - For reference look at the screenshot below
 - This is only needed when specifying the post_body directly in Content Type. If you want to work with interpolation variables this is not needed.
 - You can also specify an interpolation variable. This is **recommended** as you will not have to create a separate view for every single Post Body. An example is shown below in a screenshot.
- In the Authentication section provide the authentication details of your Denodo User.
- **NOTE:** When testing the connection, Denodo will call the API and therefore run the scheduler jobs if the connection details are correct. Keep that in mind in case that is unwanted. You might want to change the action to enable/disable/stop to test the connection without triggering the job(s).



The screenshot shows the Denodo configuration interface for a data source named 'runschedulertjob108'. The configuration is as follows:

- Name:** runschedulertjob108
- Endpoint access configuration:** Direct (selected), From OpenAPI 3 document
- Data route:** HTTP Client
- Configuration:**
 - HTTP method:** PUT
 - Base URL:** http://localhost:9090/webadmin/denodo-scheduler-admin/public/api/projects/1/jobs/status?uri=%2F%2Flocalhost%3A
 - Request timeout:** 120000
 - Content type:** application/json
 - Post body:** \{"action": "start", "IDs": "108"}
- HTTP Headers:**
 - Follow redirect:
 - Check certificates:
 - Autodetect encoding:
 - Ignore HTTP route errors:

Connection Metadata

Name runschedulerjob108

Endpoint access configuration Direct From OpenAPI 3 document

Data route

— Configuration

HTTP method

Base URL

Request timeout

URL parameters Post body

Content type

Post body

Drag 'n' drop some files here, or click to select files

HTTP Headers

Follow redirect

Check certificates

Autodetect encoding

Ignore HTTP route errors

In the Screenshot above `@{post_body}` represents an interpolation variable.

— Authentication

Authentication

Pass-through session credentials

Login

Password

+ Filters

Ignore route errors

4 CREATING A VIEW

Since the Scheduler API PUT Methods described above return no content but only change the jobs status, we cannot create the base view in the conventional graphical way.

We **first have to create a JSON Wrapper** with some output schema. After that we can create the views.

- You can use the VQL Code below as a template.
- Provide a Wrapper name of your choosing.
- The DATASOURCENAME should be set to your JSON data source created before.
- You can specify any OUTPUTSCHEMA. We need it to be able to create a base view since we can not create a base view without an Output.

If you have specified an interpolation variable in the data source creation, your VQL code has to be adjusted. Templates and Examples are provided below.

#For creating a view without an interpolation variable:

```
CREATE OR REPLACE WRAPPER JSON <WRAPPER_NAME>
  DATASOURCENAME=<DATA SOURCE NAME>
  TUPLEROOT '/JSONFile'
  OUTPUTSCHEMA (
    <output_name>=<output>:'<Java_data_type>
  );
```

#Example:

```
CREATE OR REPLACE WRAPPER JSON runschedulerjob
  DATASOURCENAME=runschedulerjob108
  TUPLEROOT '/JSONFile'
  OUTPUTSCHEMA (
    iss = 'iss' : 'java.lang.String'
  );
```

#For creating a view with an interpolation variable:

```
CREATE OR REPLACE WRAPPER JSON <WRAPPER_NAME>
  DATASOURCENAME=<DATA SOURCE NAME>
  TUPLEROOT '/JSONFile'
  ROUTE HTTP 'http.CommonsHttpClientConnection,120000' PUT ''
  POSTBODY '{@post_body}'
  OUTPUTSCHEMA (
    post_body = 'post_body' : 'java.lang.String' (OBL)
  (DEFAULTVALUE='{"action": "<default_action>","IDs": "<default_IDs>"})')
```

```
EXTERN
  );
```

#Example:

```
CREATE OR REPLACE WRAPPER JSON runschedulerjob108
  DATASOURCENAME=runschedulerjob108
  TUPLEROOT '/JSONFile'
  ROUTE HTTP 'http.CommonsHttpClientConnection,120000' PUT ''
  POSTBODY '@{post_body}'
  OUTPUTSCHEMA (
    post_body = 'post_body' : 'java.lang.String' (OBL)
  (DEFAULTVALUE='{"action": "start","IDs": "108,99,1"}') EXTERN
  );
```

Next we can create a base view using the Wrapper we just created.

- Use the VQL Code below as a template again.
- Provide a view name.
- Choose your I18N setting.

#For creating a view without an interpolation variable:

```
CREATE OR REPLACE TABLE <VIEW_NAME> I18N <INTERNATIONALIZATION> (
  iss:text
)
CACHE OFF
TIMETOLIVEINCACHE DEFAULT
ADD SEARCHMETHOD <WRAPPER_NAME>(
  I18N <INTERNATIONALIZATION>
  CONSTRAINTS (
    ADD iss NOS ZERO ()
  )
  OUTPUTLIST (iss)
  WRAPPER (<WRAPPER_NAME>)
);
```

#Example

```
CREATE OR REPLACE TABLE runschedulerjob108 I18N de_euro (
  iss:text
)
CACHE OFF
TIMETOLIVEINCACHE DEFAULT
ADD SEARCHMETHOD runschedulerjob108(
  I18N de_euro
  CONSTRAINTS (
    ADD iss NOS ZERO ()
  )
  OUTPUTLIST (iss)
  WRAPPER (json runschedulerjob108)
);
```

#Example for creating a view with interpolation variables

```
CREATE OR REPLACE TABLE runschedulerjob I18N de_euro (
  post_body:text (extern)
)
CACHE OFF
TIMETOLIVEINCACHE DEFAULT
ADD SEARCHMETHOD runschedulerjob(
  I18N de_euro
  CONSTRAINTS (
    ADD post_body (=) OBL ONE
  )
  WRAPPER (json runschedulerjob)
);
```

5 RUN THE SCHEDULER JOB VIA KAFKA EVENT

You have now set up everything necessary to be able to run Scheduler jobs via Kafka events.

Create a new Event under the topic you have subscribed to under the Kafka Listener.

Kafka Event Example for working without the interpolation variable.

```
PS C:\Denodo Projects\Kafka Event Driven Scheduler Jobs\kafka-java-getting-started> docker exec -it broker bash  
[appuser@75b5c6bb1778 ~]$ kafka-console-producer --topic queries --bootstrap-server broker:9092  
>SELECT * FROM schedulercw.runschedulerjob108
```

Kafka Event Example for working with the interpolation variable and the @LISTENEREXPRESSION set up in the Kafka Listener.

```
>{"action":"start","IDs":"108"}  
>{"action":"start","IDs":"108"}  
>|
```

If the Kafka Listener is running and picking up the Event, the Scheduler job will be started.

VDP	RUNNING	11/05/2023 13:03:18
-----	----------------	---------------------

6 REFERENCES

[Creating Kafka Listeners](#)
[Scheduler REST Client API](#)
[JSON Sources](#)