How to create Scheduler jobs that are triggered based on events

Revision 20221018

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 Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 GOAL

This document describes how to create Scheduler jobs that are triggered based on events.

2 SCENARIO

Normally, Denodo Scheduler jobs are triggered manually, through the Scheduler Client API or by time based triggers configured for each individual job.

Consider a scenario where caching is required because live access to a data source is not allowed. However, we must be aware of when to refresh the cache in order to access up-to-date data. Triggering a cache-refresh job at a specific time is not always possible, for instance, there is an ETL process that updates the data source and we do not have information about when it finishes, or, in general, data is not updated with a predefined schedule.

In this example, we will suppose that we have an external system (ETL) that updates a file in the file system when the process that updates the data source is complete. We want to have the option to launch a Scheduler job whenever this file is generated or modified (i.e) trigger a Scheduler job based on events.

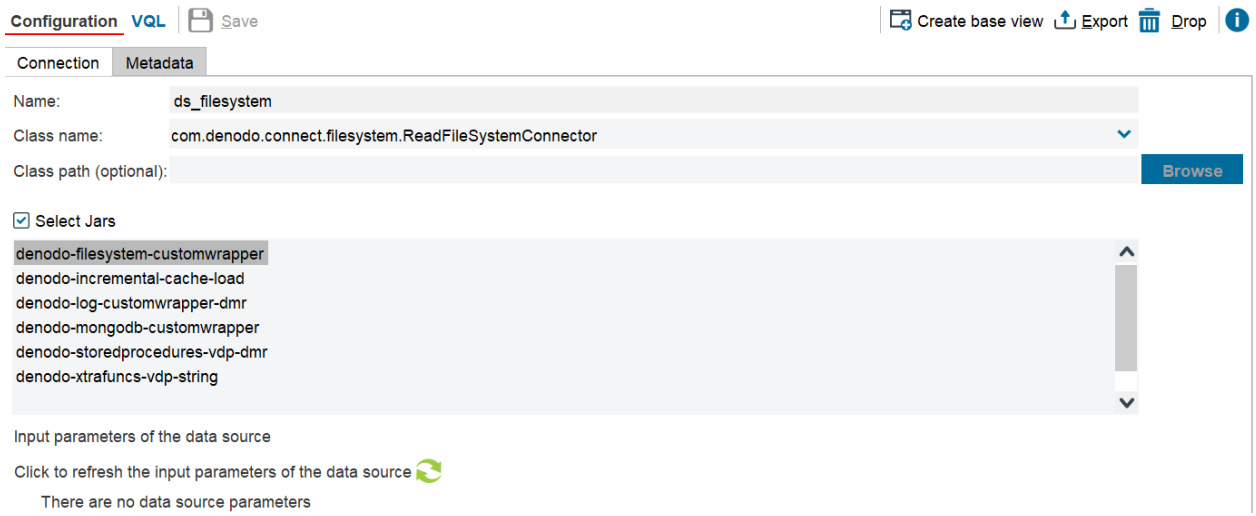
Detailed steps to perform this activity have been described in this document. We assume that we have a file named `fileExists.txt` which we are going to use to trigger the Scheduler job.

3 USING THE DENODO FILE SYSTEM CUSTOM WRAPPER

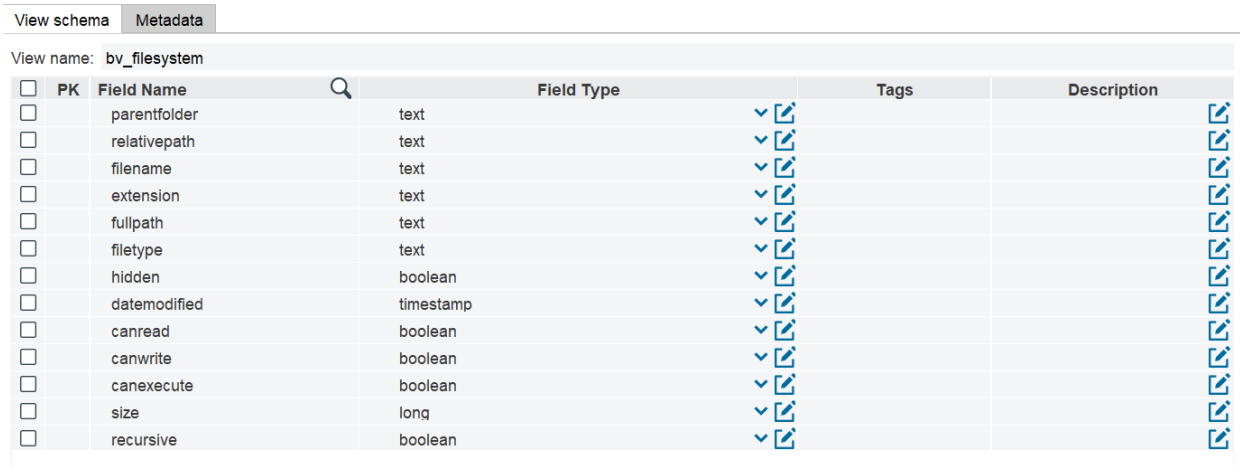
The Denodo File System Custom Wrapper enables Virtual DataPort to retrieve information from the filesystem. This allows you to inspect local, network, and FTP server-accessible folders and retrieve lists of files (in a single folder or recursively), and filter files using any of their metadata (file name, file size, last modification date, etc). The custom wrapper can also read their contents --in text or binary form-- and allows creating, updating and deleting text files.

In order to access the file that would trigger the Scheduler job, you can configure a custom data source in the Virtual DataPort server by following the below steps:

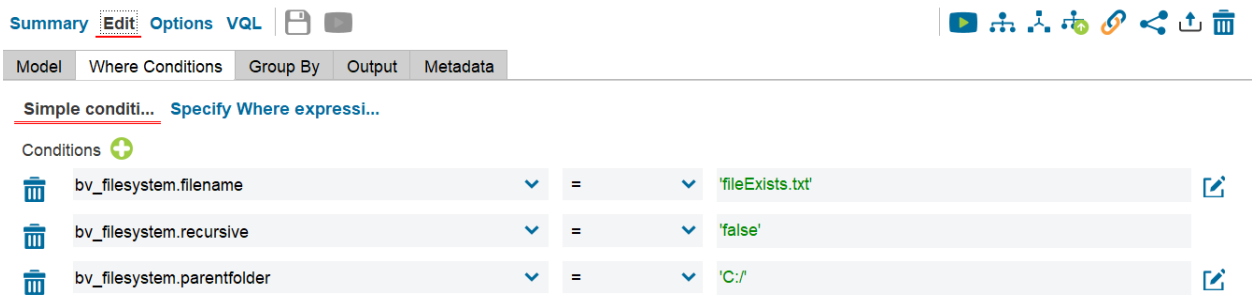
- In order to use the `denodofile-customwrapper` first, it is necessary to import its JAR file into the Denodo Virtual DataPort server. For this, login to the Virtual DataPort Administration tool, open the "`File > Extensions`" menu option and import the jar "`denodo-filesystem-customwrapper-8.0-<version>-jar-with-dependencies`" that is included in the distribution of the FileSystem custom wrapper.
- After this, create a new data source using "`File > New > DataSource > Custom`" menu option and fill the appropriate parameters:



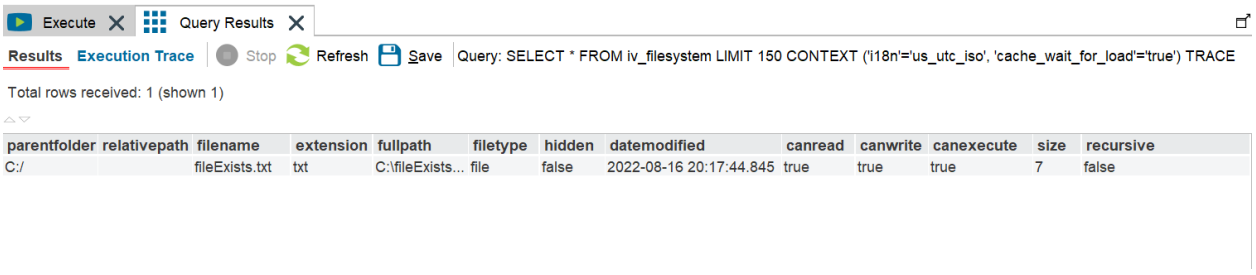
- Once the datasource (`ds_filesystem`) has been created, you can now create base views (`bv_filesystem`) on it and the schema of this base view would contain the following columns:



- Create a selection view (`iv_filesystem`) over the base view to provide the values for the `parentfolder`, `filename` and `recursive` columns in the '`Where Conditions`' tab choosing the files that we want to access. In this example we select the filename which is going to trigger the job:



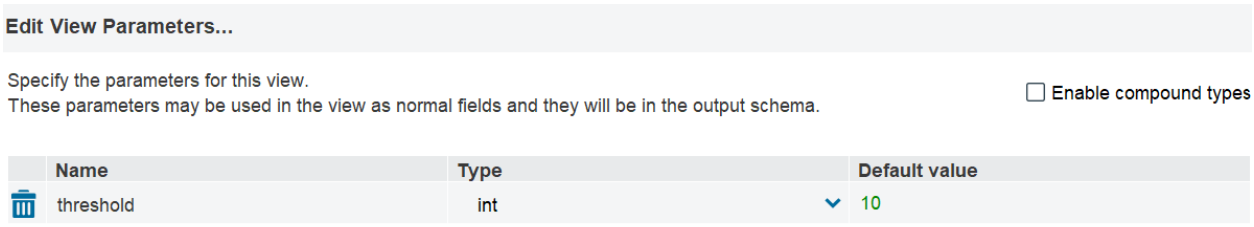
- In the following image we can see the result of the execution of this selection view and this will retrieve that file with its metadata:



4 DETECTING IF THE FILE IS MODIFIED OR REPLACED

In order to trigger a job based on an event, we need to first find whether the file has been modified or not. This can be achieved by finding the last modified date of that file. To do that, you can follow the below steps:

- Modify the derived view (`iv_filesystem`) with a new input view parameter. For instance, create a new parameter called '`threshold`' with int data type and set any default value. For this, navigate to the view and click on `Edit > Model tab > View parameters`. In this dialog, click `Add new parameter` to add the new view parameter.



- After setting the view parameter, navigate to the "`Output Tab`" and add a new int field called '`flag`' with the field expression as below:

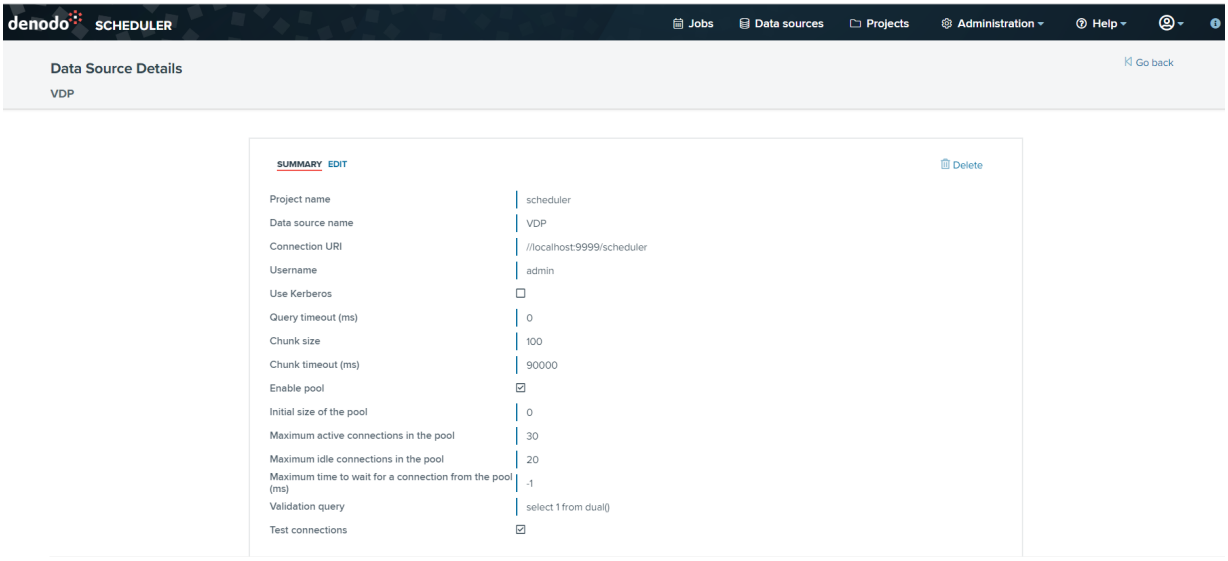
```
CASE
WHEN (now() > addminute(bv_filesystem.datemodified, threshold))
THEN 1
ELSE 0
END
```

- This condition will help to find whether the file was modified recently or not. The now() function returns the current date and time and addminute() function returns the datetime passed as parameter with its field minute rolled up by the amount specified (`threshold`). If the modified date of the file is greater than the current datetime, then the value of this field will be 1 otherwise 0.

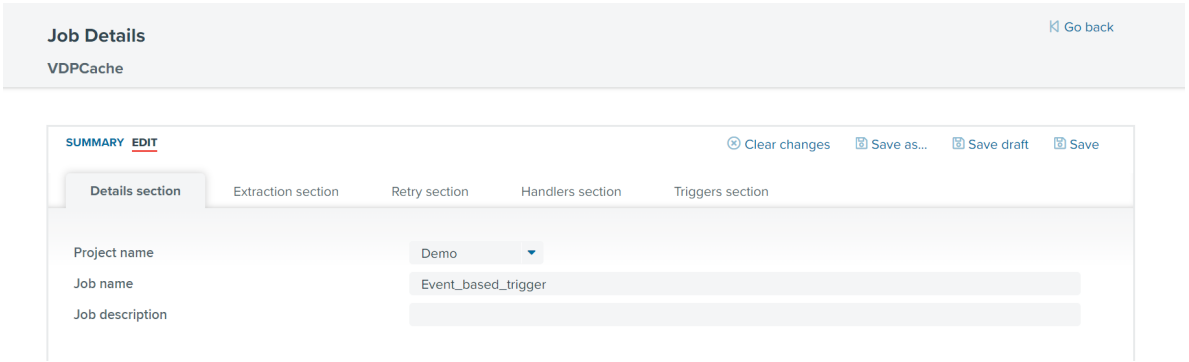
5 DEFINING AN EVENT TRIGGERED SCHEDULER JOB

The objective of this section is to create a `VDPCache` job for the cached view and trigger this job based on the file modification.

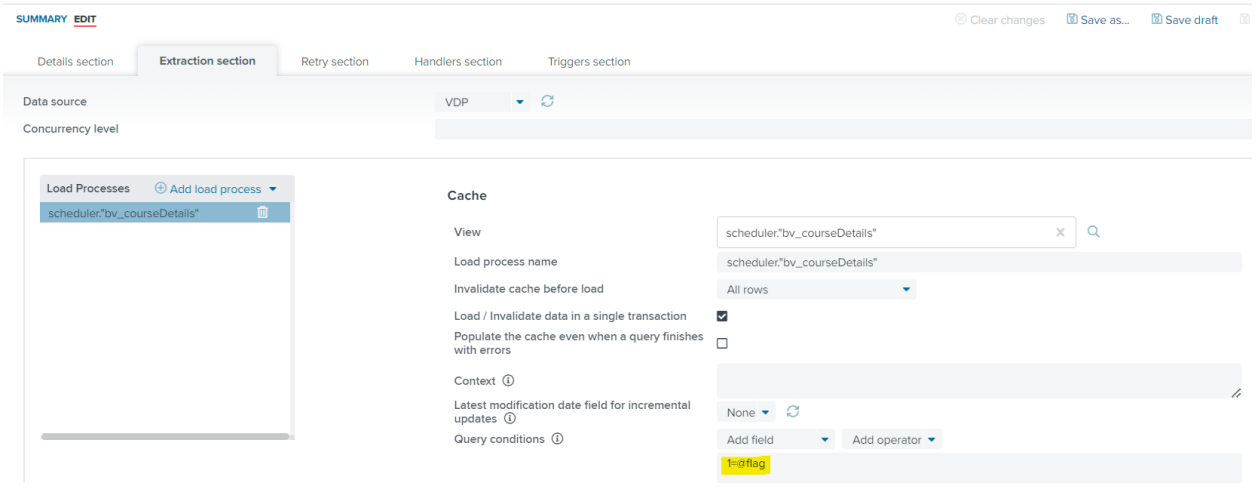
- Configure a Virtual DataPort data source in Denodo Scheduler Administration Tool by navigating to "`Data sources > Add data source`" option and select VDP from the drop-down. This prompts a new tab to create a VDP data source.



- Once the data source is created successfully, you can create a [VDPCache job] using this data source. To do this, navigate to the "`Jobs option`" in the **header menu bar** and select "`Add jobs > VDPCache`".
- In the `Details` section, you need to choose the project, set the job name and describe the job.

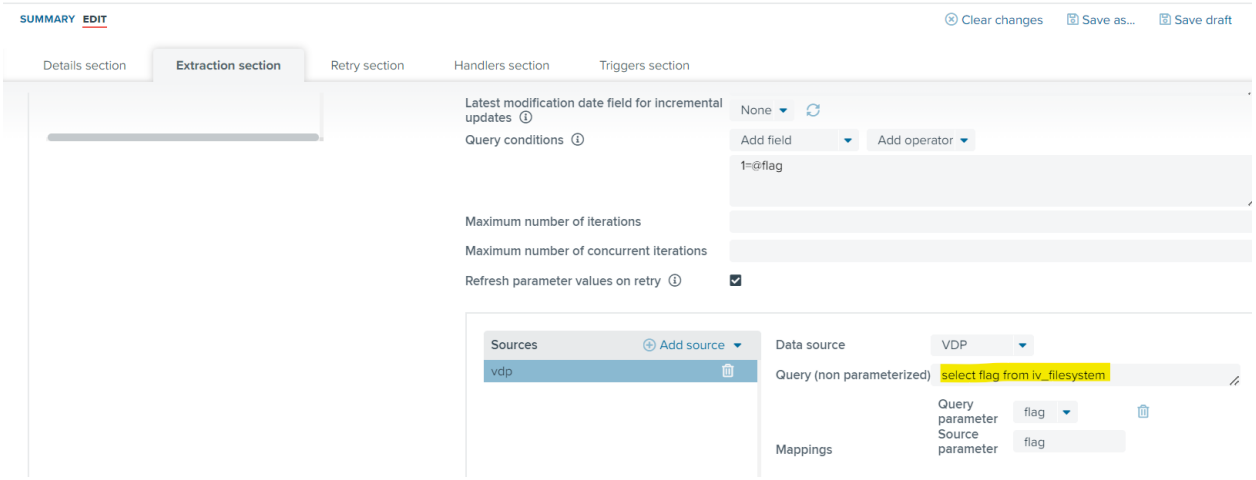


- Navigate to the `Extraction Section` tab, choose the VDP data source from the drop down which we configured as '`vdp`' and load the cached view.



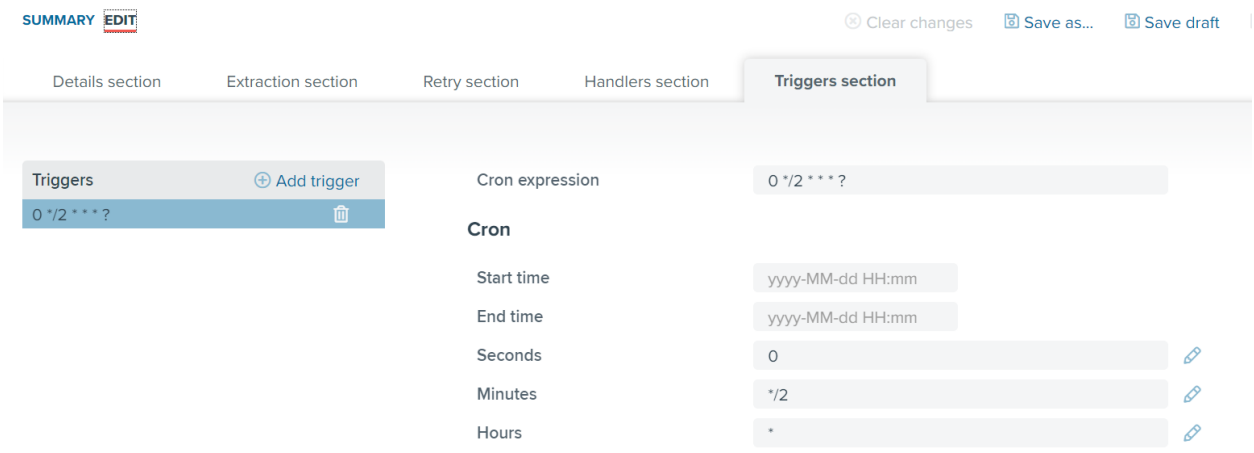
- For the `Parameterized query` field we will use the query we are going to run to see if the file has been modified or not:

```
SELECT flag FROM iv_filesystem
```

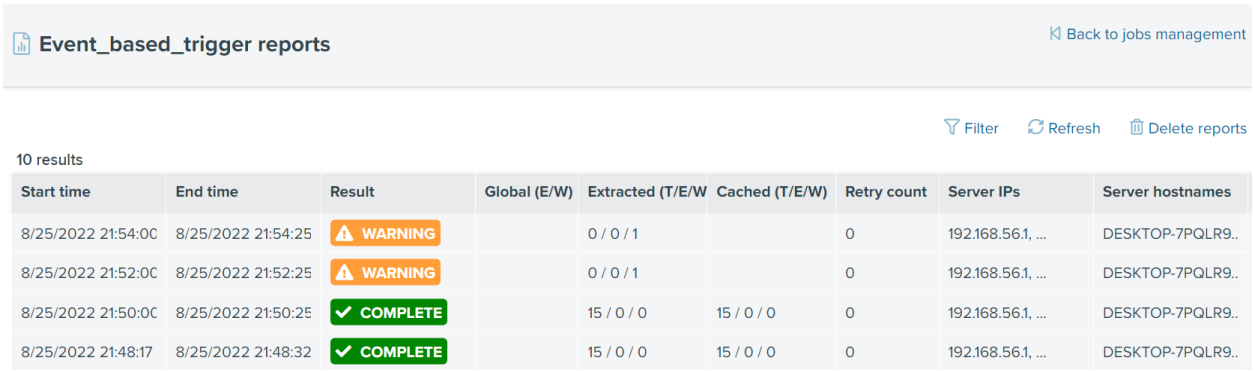


Here when the condition "`1 = @flag`" is met (when it meets 1 = 1) then the job will actually refresh the cache of the view..

- Navigate to the `Trigger section` tab and click on the "`Add trigger`" option. The trigger can be set for the job to be executed periodically. The following cron expression triggers this job "`event_based_trigger`" to run for every two minutes.



- Save the newly created job. Once saved, the job will now be executed automatically based on the cron expression provided in the Trigger. For this scenario, this will check every two minutes for any modifications done in the configured file. If the file has been modified then it will execute the job and cache the data with "`Complete`" status. Otherwise, the job shows the "`Warning`" status. This will be listed in the report section of the job.



6 REFERENCES

Denodo FileSystem CustomWrapper - User Manual
Creating a Denodo Scheduler Job