



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

CONTENTS

1 GOAL.....	3
2 SCENARIO.....	4
3 USING THE DENODO FILE SYSTEM CUSTOM WRAPPER.....	5
4 DETECTING IF THE FILE IS MODIFIED OR REPLACED.....	7
5 DEFINING AN EVENT TRIGGERED SCHEDULER JOB.....	9
6 REFERENCES.....	12

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 `denodo-file-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:

The screenshot shows the 'Configuration' page in the Denodo Virtual DataPort Administration tool. The 'Metadata' tab is active. The 'Name' field is set to 'ds_filesystem'. The 'Class name' is set to 'com.denodo.connect.filesystem.ReadFileSystemConnector'. The 'Class path (optional)' field is empty, with a 'Browse' button next to it. Under the 'Select Jars' section, a list of JAR files is displayed, with 'denodo-filesystem-customwrapper' selected. Below the list, there is a section for 'Input parameters of the data source' which is currently empty, with a refresh button and the text 'There are no data source 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:

View schema Metadata

View name: bv_filesystem

<input type="checkbox"/>	PK	Field Name	Field Type	Tags	Description
<input type="checkbox"/>		parentfolder	text		
<input type="checkbox"/>		relativepath	text		
<input type="checkbox"/>		filename	text		
<input type="checkbox"/>		extension	text		
<input type="checkbox"/>		fullpath	text		
<input type="checkbox"/>		filetype	text		
<input type="checkbox"/>		hidden	boolean		
<input type="checkbox"/>		datemodified	timestamp		
<input type="checkbox"/>		canread	boolean		
<input type="checkbox"/>		canwrite	boolean		
<input type="checkbox"/>		canexecute	boolean		
<input type="checkbox"/>		size	long		
<input type="checkbox"/>		recursive	boolean		

- 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:

Summary Edit Options VQL

Model Where Conditions Group By Output Metadata

Simple condition... Specify Where expressi...

Conditions +

<input type="checkbox"/>	bv_filesystem.filename	=	'fileExists.txt'
<input type="checkbox"/>	bv_filesystem.recursive	=	'false'
<input type="checkbox"/>	bv_filesystem.parentfolder	=	'C:/'

- 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:

Execute Query Results

Results Execution Trace Stop Refresh Save Query: SELECT * FROM iv_filesystem LIMIT 150 CONTEXT ('i18n='us_utc_iso', 'cache_wait_for_load='true') TRACE

Total rows received: 1 (shown 1)

parentfolder	relativepath	filename	extension	fullpath	filetype	hidden	datemodified	canread	canwrite	canexecute	size	recursive
C:/		fileExists.txt	txt	C:\fileExists...	file	false	2022-08-16 20:17:44.845	true	true	true	7	false

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.

Edit View Parameters...

Specify the parameters for this view.

These parameters may be used in the view as normal fields and they will be in the output schema.

Enable compound types

Name	Type	Default value
 threshold	int	10

- 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
```

New Field

Field name

flag

Field expression

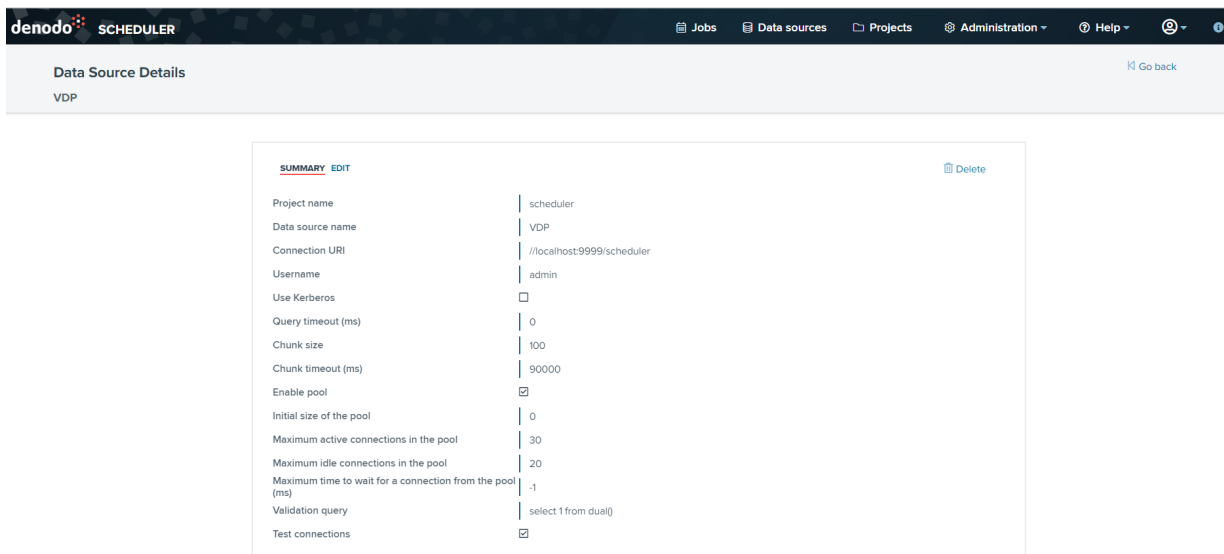
```
case WHEN (now() > addminute(bv_filesystem.datemodified, threshold)) THEN 1 ELSE 0 END
```

- 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.

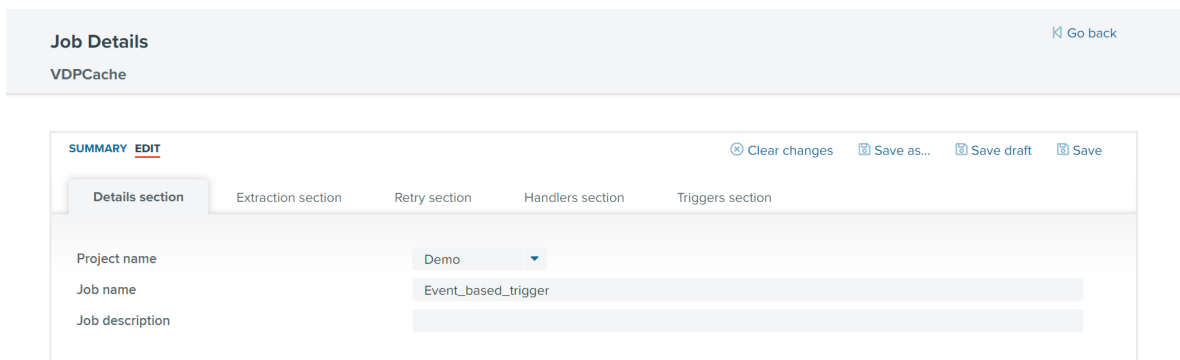
- Configure a Virtual DataPort data source in Denodo Scheduler Administration Tool by navigating to "Data sources > Add data source" option and select VDP from the drop-down. This prompts a new tab to create a VDP data source.



The screenshot shows the 'Data Source Details' page for a VDP data source. The page has a header with 'denodo SCHEDULER' and navigation links for Jobs, Data sources, Projects, Administration, and Help. Below the header, there's a 'Data Source Details' section with a 'Go back' link. The main content area shows a 'SUMMARY' tab with a 'Delete' button. The details are as follows:

Project name	scheduler
Data source name	VDP
Connection URI	//localhost:9999/scheduler
Username	admin
Use Kerberos	<input type="checkbox"/>
Query timeout (ms)	0
Chunk size	100
Chunk timeout (ms)	90000
Enable pool	<input checked="" type="checkbox"/>
Initial size of the pool	0
Maximum active connections in the pool	30
Maximum idle connections in the pool	20
Maximum time to wait for a connection from the pool (ms)	-1
Validation query	select 1 from dual()
Test connections	<input checked="" type="checkbox"/>

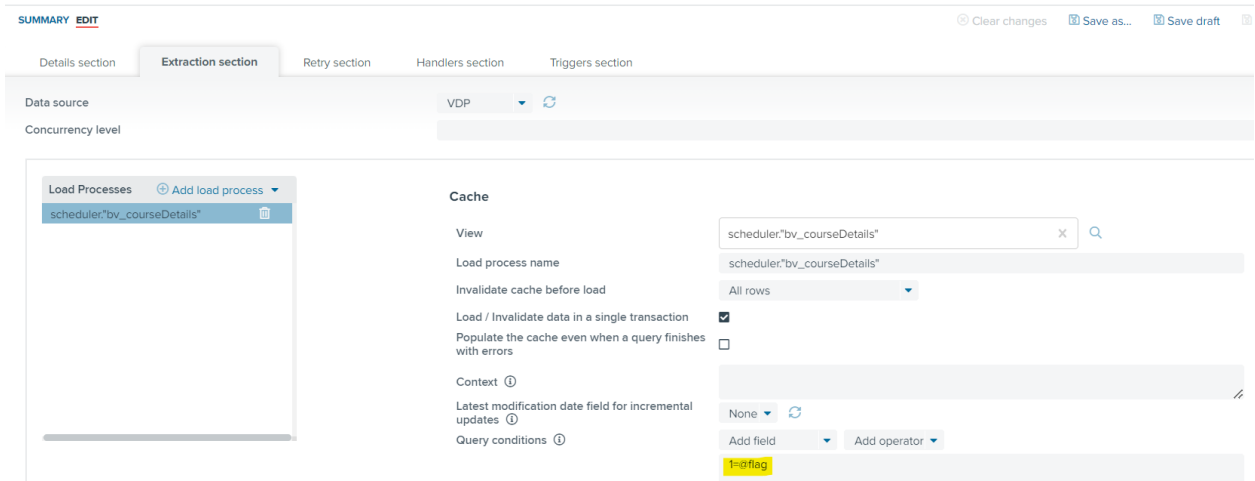
- Once the data source is created successfully, you can create a [VDPCache job](#) using this data source. To do this, navigate to the "Jobs option" in the **header menu bar** and select "Add jobs > VDPCache".
- In the Details section, you need to choose the project, set the job name and describe the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



The screenshot shows the 'Job Details' page for a VDPCache job. The page has a header with 'denodo SCHEDULER' and navigation links for Jobs, Data sources, Projects, Administration, and Help. Below the header, there's a 'Job Details' section with a 'Go back' link. The main content area shows a 'SUMMARY' tab with buttons for 'Clear changes', 'Save as...', 'Save draft', and 'Save'. There are five tabs: 'Details section', 'Extraction section', 'Retry section', 'Handlers section', and 'Triggers section'. The 'Details section' is active and shows the following fields:

Project name	Demo
Job name	Event_based_trigger
Job description	

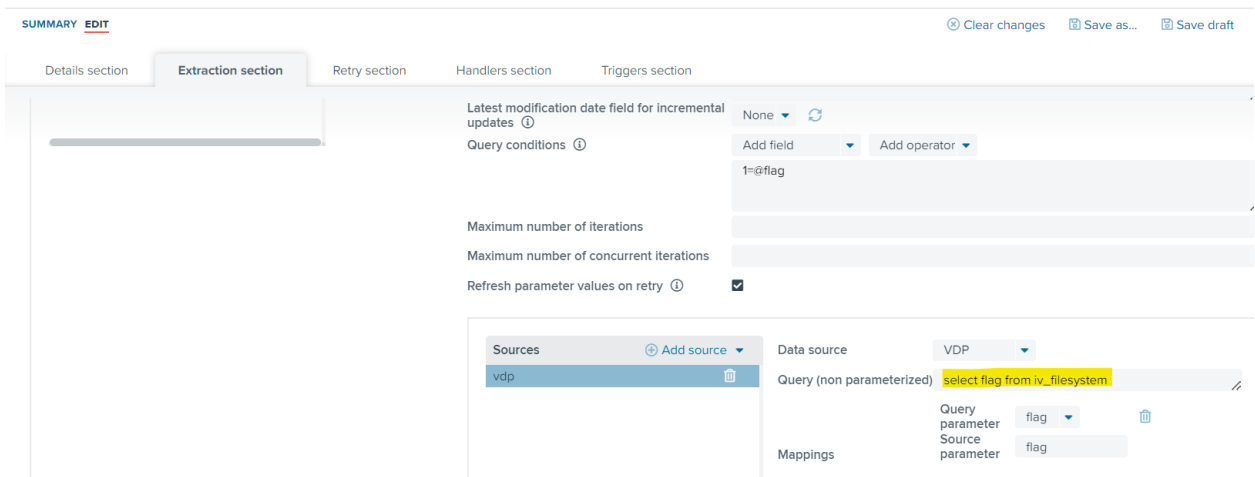
- Navigate to the Extraction Section tab, choose the VDP data source from the drop down which we configured as 'vdp' and load the cached view.



The screenshot shows the Denodo Scheduler configuration interface. The 'Extraction section' is active. Under the 'Cache' section, the 'View' is set to 'scheduler:"bv_courseDetails"'. The 'Load process name' is also 'scheduler:"bv_courseDetails"'. The 'Invalidate cache before load' is set to 'All rows'. The 'Load / Invalidate data in a single transaction' checkbox is checked. The 'Populate the cache even when a query finishes with errors' checkbox is unchecked. The 'Context' field is empty. The 'Latest modification date field for incremental updates' is set to 'None'. The 'Query conditions' field contains '1=@flag'.

- 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
```



The screenshot shows the Denodo Scheduler configuration interface. The 'Extraction section' is active. The 'Query conditions' field contains '1=@flag'. The 'Refresh parameter values on retry' checkbox is checked. The 'Sources' section shows a source named 'vdp'. The 'Data source' is set to 'VDP'. The 'Query (non parameterized)' field contains 'select flag from iv_filesystem'. The 'Query parameter' is set to 'flag' and the 'Source parameter' is also set to 'flag'.

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.

SUMMARY **EDIT** Clear changes Save as... Save draft

Details section Extraction section Retry section Handlers section **Triggers section**

Triggers + Add trigger

0 */2 ***? 🗑️

Cron expression 0 */2 ***?

Cron

Start time yyyy-MM-dd HH:mm

End time yyyy-MM-dd HH:mm

Seconds 0 🔗

Minutes */2 🔗

Hours * 🔗

- 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.

Event_based_trigger reports [Back to jobs management](#)

Filter Refresh Delete reports

10 results

Start time	End time	Result	Global (E/W)	Extracted (T/E/W)	Cached (T/E/W)	Retry count	Server IPs	Server hostnames
8/25/2022 21:54:00	8/25/2022 21:54:25	⚠️ WARNING		0 / 0 / 1		0	192.168.56.1, ...	DESKTOP-7PQLR9..
8/25/2022 21:52:00	8/25/2022 21:52:25	⚠️ WARNING		0 / 0 / 1		0	192.168.56.1, ...	DESKTOP-7PQLR9..
8/25/2022 21:50:00	8/25/2022 21:50:25	✅ COMPLETE		15 / 0 / 0	15 / 0 / 0	0	192.168.56.1, ...	DESKTOP-7PQLR9..
8/25/2022 21:48:17	8/25/2022 21:48:32	✅ COMPLETE		15 / 0 / 0	15 / 0 / 0	0	192.168.56.1, ...	DESKTOP-7PQLR9..

6 REFERENCES

[Denodo FileSystem CustomWrapper - User Manual](#)
[Creating a Denodo Scheduler Job](#)