Denodo Log Custom Wrapper - User Manual

Revision 20210422

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

CONTENTS

1 INTRODUCTION

`log-customwrapper` is a Virtual DataPort custom wrapper created to analyze and extract information from log files which content is ordered by date. Although this can be done using the DF wrapper, in some cases log files are too big, and the DF wrapper is too slow: The DF wrapper reads the whole file line by line from the beginning (Even if you specify a begin/end delimiter), while the `log-customwrapper` allows to analyze only the part of the file needed, allowing to follow rolling log files.

2 ARCHITECTURE AND FEATURES

The `log-customwrapper` was developed using the VDP custom wrapper API for Denodo Platform. It allows to extract information from the log files by using an extractor pattern (A regular expression) to get the desired data and another pattern, used to extract the date from each entry. These patterns are created activating the `DOTALL` mode, thus the expression matches any character, including a line terminator.

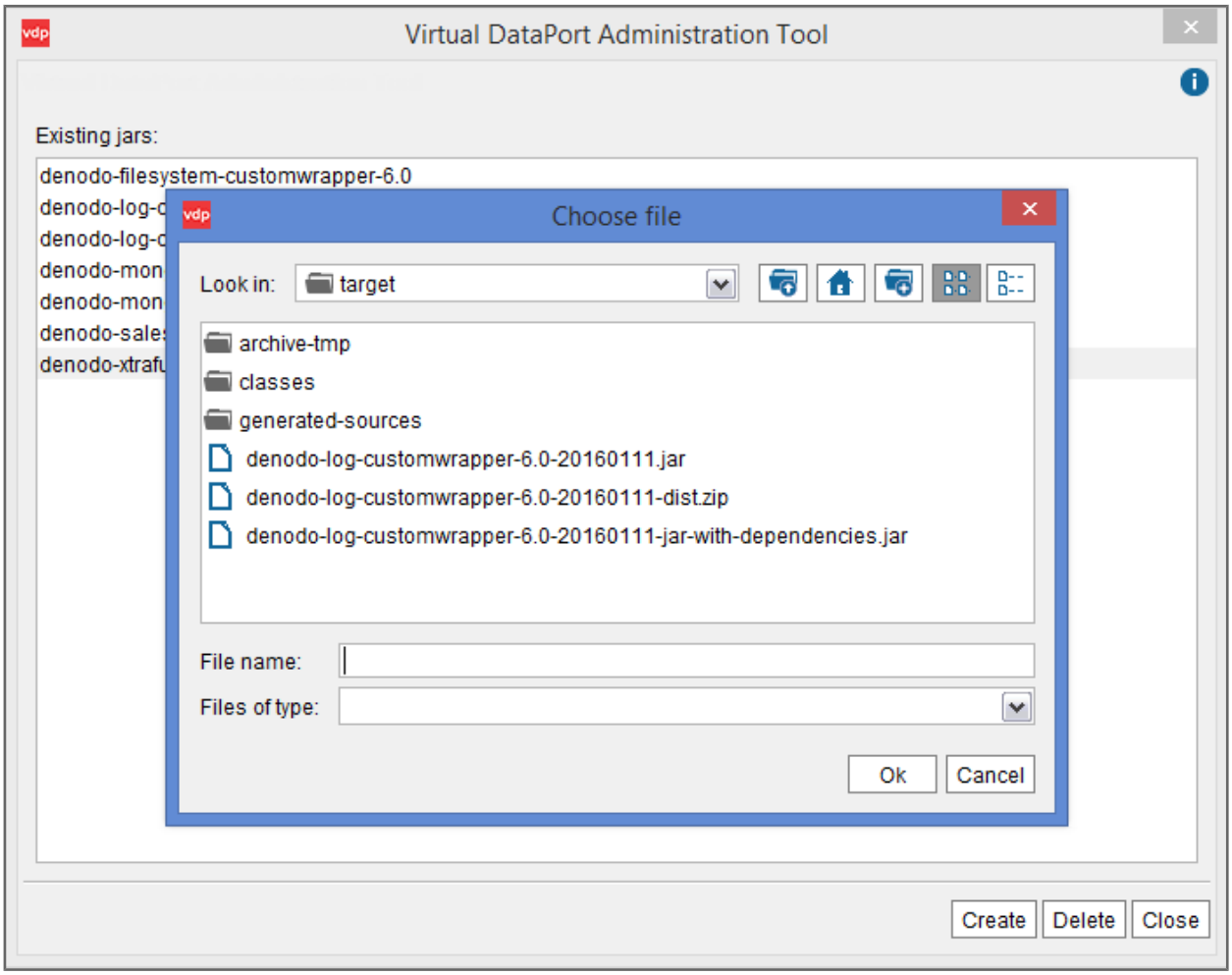
The custom wrapper uses a linear search algorithm to find the first entry of the log file where the relevant information is located and then reads the file entry by entry. When the log file is bigger than 50MB a binary search is used instead of the linear one. A parameter is used to specify the interval when the binary search must finish (When the difference between the specified and the read date is less than this interval.). We should modify this parameter depending on the frequency of writing of the log file. The compressed files only support linear search.

The log custom wrapper uses by default the same rolling pattern as the Denodo Platform. A different one can be used creating a different pattern that implements the `ILogRollingPolicy` interface.

3 USAGE

3.1 IMPORTING THE CUSTOM WRAPPER INTO VDP

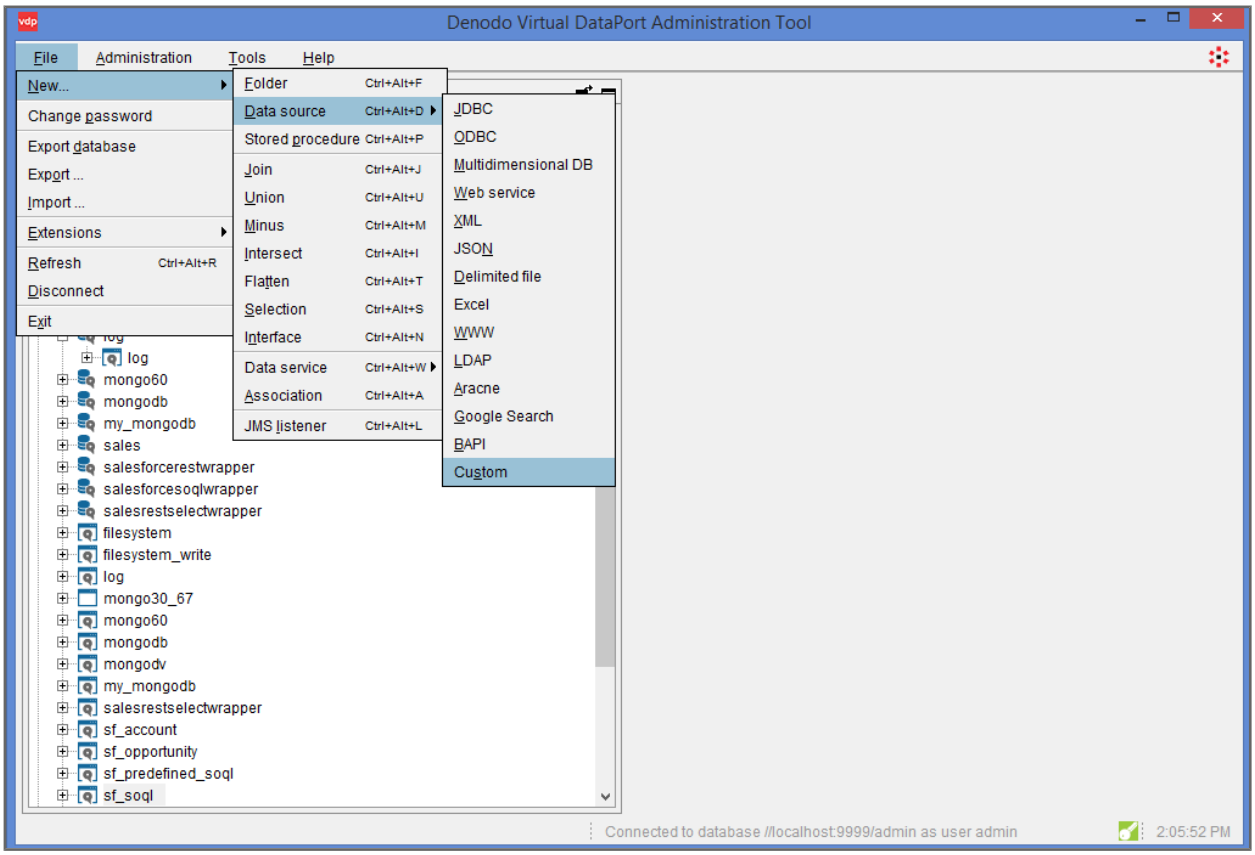
In order to use the Log Custom Wrapper in VDP, we must configure the Admin Tool to import the extension: `File → Extensions → Jar management`
From the `log-customwrapper` distribution, we will select the jar file and upload it to VDP.

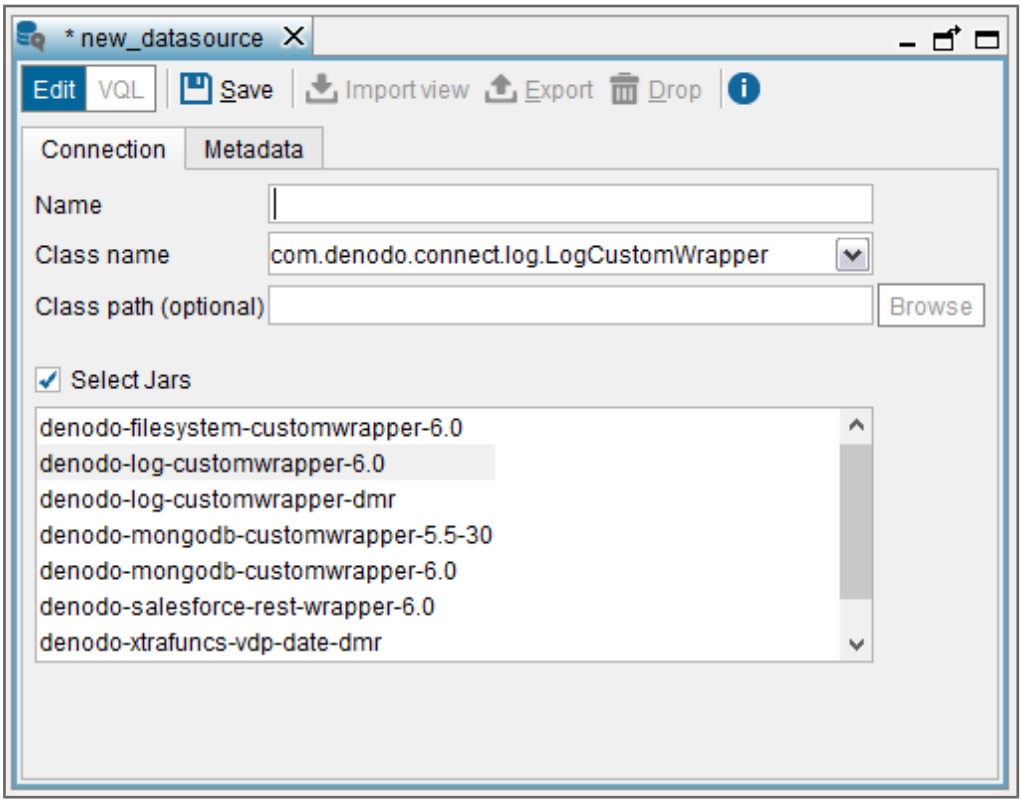


3.2 CREATING A DATASOURCE

Once the custom wrapper jar file has been uploaded to VDP using the Admin Tool, we can create new data sources for this custom wrapper --and their corresponding base views-- as usual.

Go to `New → Data Source → Custom` and specify the wrapper's class name `com.denodo.connect.log.LogCustomWrapper`. Also check 'Select Jars' and select the jar file of the custom wrapper.





3.3 CREATING A BASE VIEW

Once the custom wrapper has been registered, we will be asked by VDP to import a view for it.

In order to create our base view, we will have to specify:

- **Date pattern**: The format of the date, that allows to transform the string of the date, in a date format that the custom wrapper can read.
- **Date extractor pattern**: The regular expression to extract the date from a log line, we need to catch a group that contains the date string of every line of the log, this field and the previous one are necessary so the custom wrapper can search log lines by date.
- **Content extractor pattern**: The regular expression to extract the desired fields from the log, catching the groups that we desire, this parameter determines the number of fields of each base view.
- **Filepath**: The path to the log file (Including the name).
- **Sequential search interval**: The wrapper searches a specific date inside the log file. If a log line is read with a date that differs less than this interval, the sequential search starts.

Optionally

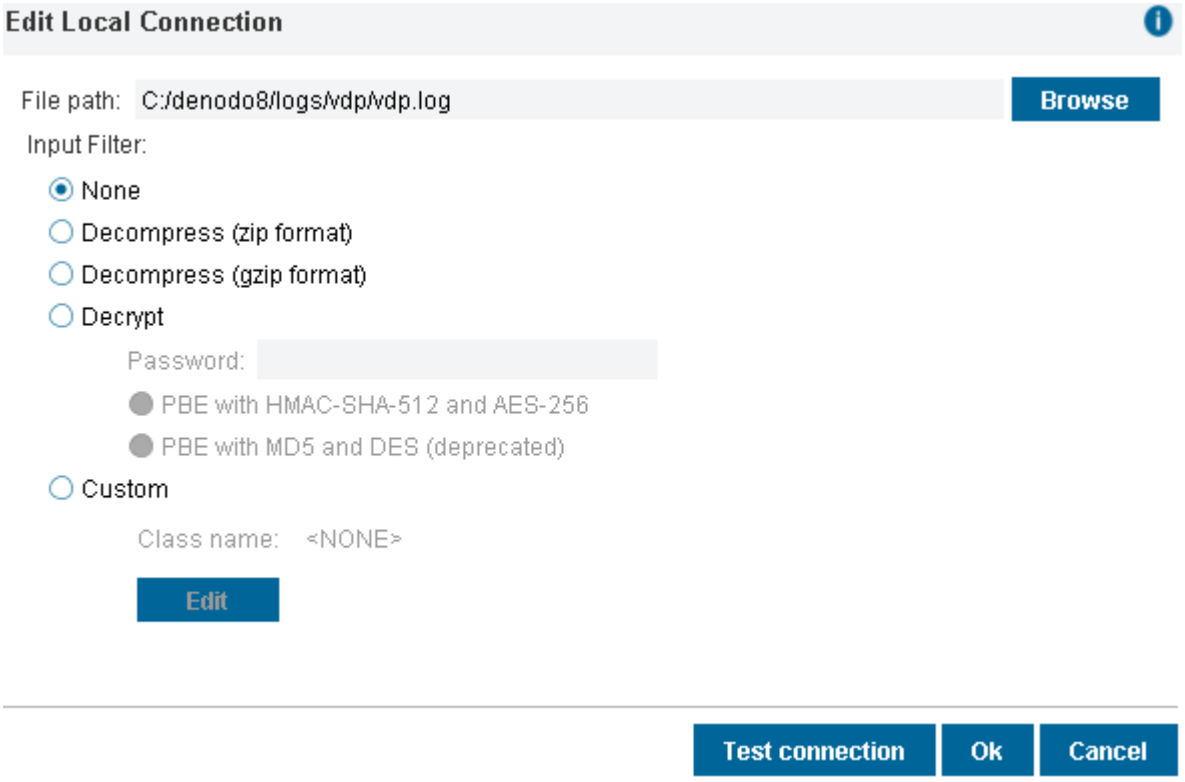
- **Log timezone**: The time zone that the custom wrapper has to use in order to get the dates from a log line. By default, the custom wrapper uses the time zone for the host where it is running.
- **File Encoding**: By default the custom wrapper uses the encoding `ISO-8859-1` to read the file. But with this field the user can choose the encoding of the file.
- **Rolling implementation class name**: The name of the class with a different rolling policy than the default one, that implements the `ILogRollingPolicy` interface.
- **Max Processed Entry Length:** Entries with more characters than this parameter will not be processed. Its value by default is 10000. If the file contains huge entries, a search could be very slow, with this parameter we could speed up the search. If the value of the parameter is `-1`, this limit is ignored.

NOTE: If you enter a literal that contains one of the special characters used to indicate interpolation variables ("@", "{" or "}"), in a parameter that accepts interpolation variables, you have to escape these characters with "\".

In the example:

- **Date pattern**: `yyyy-MM-dd'T'HH:mm:ss.SSS` (`yyyyMMddHHmmssSSS` for versions <6.0)
- **Date extractor pattern**: `(?:.*?)(\\d\{4\}-\\d\{2\}-\\d\{2\}T\\d\{2\}:\\d\{2\}:\\d\{2\}.\\d\{3\})(?:.*?)`
- **Log timezone**: `PST`
- **Content extractor pattern**: `(?:.*?)(.*?)(ERROR)(\\d*-\\d*-\\d*T\\d*:\\d*:\\d*\\.\\d*)(.*?)`
- **Filepath**: `C:/vdp/vdp.log`
- **Encoding File:** `UTF-8`
- **Sequential search interval**: `10000`
- **Rolling implementation class name**: `com.denodo.connect.log.rollingpolicy.DefaultLogRollingPolicy`

By clicking on the `Configure` button, you can choose the path of the log file. Note that, if you choose any of the `Decompress` options, you will most probably need to change the Rolling policy and in addition, the binary search can´t be used. The decrypt option is not supported by Denodo Log Custom Wrapper.



The custom wrapper detects the output fields according to the groups of the extractor pattern.

For the sake of this example, we will be integrating a VDP log, which format is as follows:

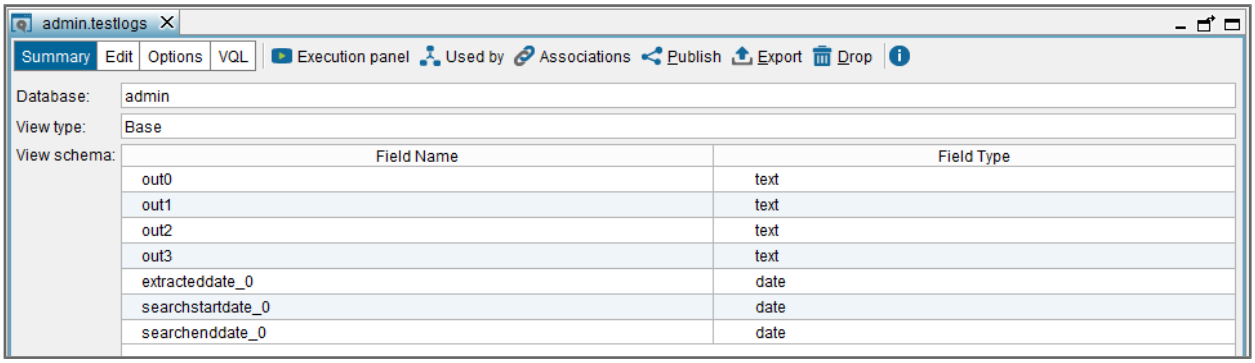
```
1      [RMI(179)-192.168.0.20-12]      ERROR      2013-10-10T12:02:16.189
com.denodo.vdb.catalog.view.function.    FunctionValueFunctionWrapperVisitor
-    Function    'rawtohex'    not    found.:    null    com.denodo.vdb.catalog.
metadata.vo.FunctionNotFoundException: Function 'rawtohex' with arity 1 not
found
```

where the date is `2013-10-10T12:02:16.189`, so the `Date pattern` field would have value `yyyy-MM-dd'T'HH:mm:ss.SSS` and the `Date extract pattern` field should be able to catch that string to be able to access and search the lines of the log that we want, in this case we have chosen `(?:.*?)(\\d\{4\}-\\d\{2\}-\\d\{2\}T\\d\{2\}:\\d\{2\}:\\d\{2\}.\\d\{3\})(?:.*?)`, because we observe see that the date is composed of a sequence of 17 numbers.

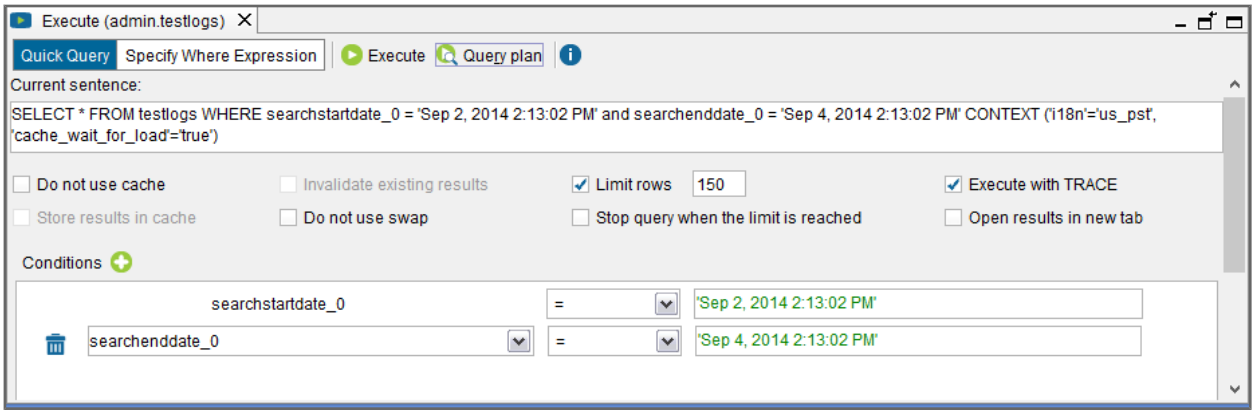
For the `Content Extract Pattern`, we have chosen which fields we want to extract from the log with this regular expression: `(?:.*?) (.*?)(ERROR) (\\d*-\\d*-\\d*T\\d*:\\d*:\\d*\\.\\d*)(.*?)`, where we catch a group with some text before "`ERROR`" (out0), other group with the word "`ERROR`" (out1), another one with the date (out2) and the last with the rest of the line (out3).

In the line of the example, these are the extracted groups:

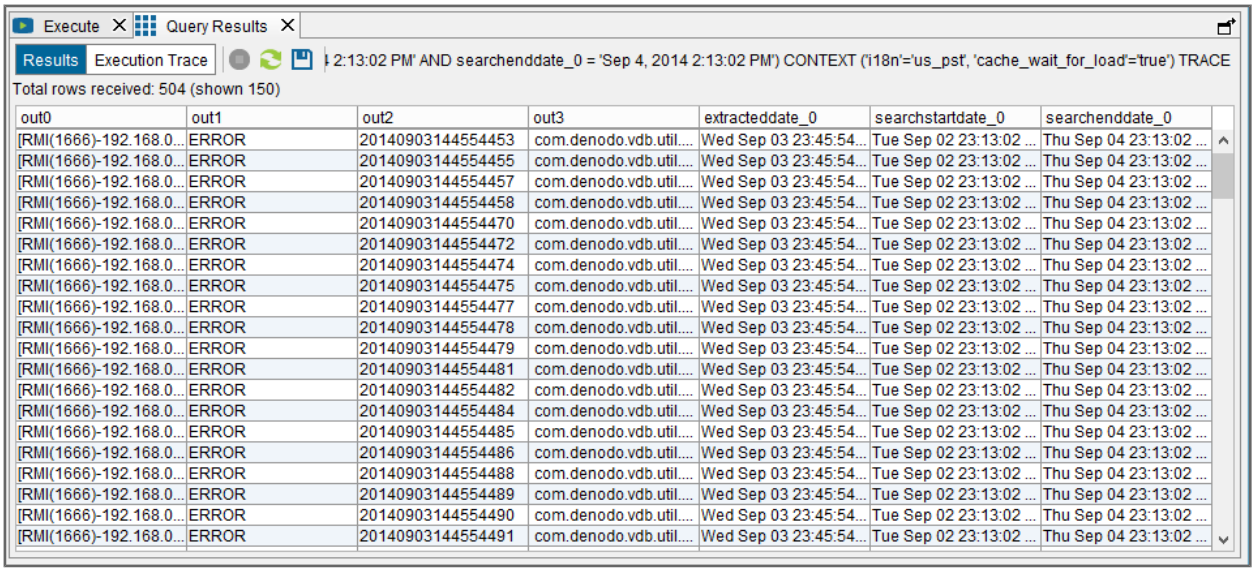
- out0: `[RMI(179)-192.168.0.20-12]`
- out1: `ERROR`
- out2: `2013-10-10T12:02:16.189`
- out3:`com.denodo.vdb.catalog.view.function.FunctionValueFunctionWrapperVisitor - Function 'rawtohex' not found.: null …`



When executing the base view, there are two parameters, the `start date` (mandatory) and the `end date` (optional).



Below, the results of the execution of a query on the base view:



out0	out1	out2	out3	extracteddate_0	searchstartdate_0	searchenddate_0
[RMI(1666)-192.168.0...	ERROR	20140903144554453	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554455	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554457	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554458	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554470	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554472	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554474	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554475	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554477	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554478	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554479	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554481	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554482	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554484	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554485	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554486	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554488	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554489	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554490	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...
[RMI(1666)-192.168.0...	ERROR	20140903144554491	com.denodo.vdb.util....	Wed Sep 03 23:45:54...	Tue Sep 02 23:13:02 ...	Thu Sep 04 23:13:02 ...

4 ROLLING POLICY IMPLEMENTATION

A RollingPolicy specifies the actions taken on a logging file rollover, every log has its own rolling policy, and though the `Rolling Implementation Class Name`, you can select the suitable policy for a particular log. You can use the implementations included in this custom wrapper or you can implement your own policies.

The rolling policy have to implement the interface `com.denodo.connect.log.rollingpolicy.ILogRollingPolicy`, that has two methods which should be implemented: `getNextLog` and `getPreviousLog`. These functions should determine the next and previous filename, the input of the method is the current log filename.

4.1 ROLLING POLICY INCLUDED

4.1.1 DefaultLogRollingPolicy

In this policy the current filename ends with `.log` and the others file names ends with `.log.(number)`. The `getPreviousLog` method, if the input filename ends with `.log` add `.1` to the filename. Otherwise the input filename ends with a number, it sums one to this number (for instance `filename.log.2` returns `filename.log.3`) . The `getNextLog` is the inverse, so if the filename ends with `.1`, delete this `.1`, and if ends with other number, it subtracts one to his number (for instance `filename.log.3` returns `filename.log.2`)

4.1.2 DefaultLogInverseRollingPolicy

In this policy the current filename ends with `.log` and the others file names ends with `.log.(number)` This policy is similar to DefaultLogRollingPolicy but in the inverse order. The `getPreviousLog` method, if the input filename ends with `.log` add `.{maxIndex}` to the filename. Otherwise the input filename ends with a number, it subtracts one to this number (for instance `filename.log.2` returns `filename.log.1`) . The `getNextLog` is the inverse, so if the filename ends with `.{maxIndex}`, delete this `.{maxIndex}`, and if ends with other number, it sums one to his number (for instance `filename.log.2` returns `filename.log.3`)

4.1.3 DateLogRollingPolicy

In this policy the current filename ends with .log, too. The other file names ends with `.log.yyyyMMddHH`. The `getPreviousLog` extract the part of the date from the filename `yyyyMMddHH` and it subtracts an hour, and the `getNextLog` also extracts the part of the date and sums 1 while the filename, that it returns, exists.

4.1.4 ByDayLogRollingPolicy

In this policy the current filename ends with `.log`, too. The other file names ends with `.log.yyyy-MM-dd`, but the dates don't have to be consecutive. The `getPreviousLog` finds among the files stored in the folder, if there are any file with a previous date to

the current one, and the `getNextLog` also finds a file with a next date to the current one. This format of log is used by the tool **Denodo Monitor**.

4.1.5 Compressed files

If the first file is compressed, all the previous policies would be applied adding the extension of that file. The supported extensions, in the current rolling policies, are `.zip`, `.tgz`, `.gz` or `.tar.gz`.

4.2 DEVELOPING CUSTOM ROLLING POLICY

The necessary interface for creating new custom rolling policy are located in the package `com.denodo.connect.log.rollingpolicy` whose name is `ILogRollingPolicy` that is included in the Denodo Log Custom Wrapper.

A custom rolling policy has to implement the interface `ILogRollingPolicy`, that contains the following methods that are used to search the next entry in the log that matches with the date of the search, it is possible to search in more than one file, so it is necessary these methods to change of log file while the search is performed.

- `public String getNextLog(final String currentLogFileName)`: This method should return the name of the next log file, if it exists, where the entries are immediately later. Otherwise it should return null.

- `public String getPreviousLog(final String currentLogFileName)`: This method should return the name of the previous log file, if it exists, where the entries are immediately earlier. Otherwise it should return null.

When you have finalized the class of the new custom rolling policy, you should import the jar that includes this class into DataPort (see **section Importing Extensions of the Administration Guide [ADMIN_GUIDE]**). In addition, when you create a new Data Source using the Denodo Log Custom Wrapper, you should select the jar of this custom wrapper and the jar of the custom rolling policy. Finally you can reference the custom rolling policy writing the name of its own class in the field `Rolling Implementation Class Name` when you create a base view.

Below, you can see an empty rolling policy:

```java
package com.denodo.connect.log.rollingpolicy;

public class ExampleLogRollingPolicy implements ILogRollingPolicy {

    public ExampleLogRollingPolicy() {
        super();
    }
    @Override
    public String getNextLog(String currentLogFileName) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getPreviousLog(String currentLogFileName) {
        // TODO Auto-generated method stub
        return null;
    }

}
```

5 LIMITATIONS

- The encrypted files are not supported.
- The compressed files only support the sequential search.
- The files with huge entries or lines affect the search performance owing to expression matching system of this wrapper. The parameter Max Processed Entry Length avoids that huge entries will be processed.