

Help: Database query Action Plug-in

Version

0.1.5

Overview

This plug-in provides the ability to execute an SQL query on a database and format its result.

Description

Saved Parameters Description

- **Name:** The name used to identify a saved DatabaseQuery configured instance.
- **Comments:** Some comments about this saved DatabaseQuery configured instance.
- **Database Type:** The type of the external Database to connect to for the execution of the query. Supported RDBMS types are MySQL, Oracle and all ODBC compatible RDBMS.
- **Database host:** In the case of ODBC and Oracle, the DB host where the RDBMS is executing is specified in the datasource. In the case of MySQL it needs to be specified explicitly if the DB is not hosted on localhost
- **Database port:** The port to connect to the database. As for the host, the port to be used is specified in the datasource configuration for ODBC and Oracle, hence this parameter only makes sense for MySQL
- **Database name (or source name):** For MySQL, the database name needs to be specified, for ODBC and Oracle, the data source name is used instead.
- **Database user:** The user name for the external database being queried
- **Database password:** The password associate to the database user
- **Database query:** The SQL query to execute. Stored procedure can be used. Dynamic attributes are supported. For example `'select name from users where login = "<%=user_login %>"` would expand an input variable name 'user_login' and return the matching database rows.
- **Row formatting code:** This code will be executed for each row of the result set. Each row is represented in a variable name 'row' as a hash indexed by the database column name (as a symbol), for example `{:name=>'Maurice',:login=>'momo'}`. The content of the row variable can be modified to alter formatting. For example, `row = "#{row[:login]} (#{row[:name]})"`, would change the content of row to be a string, here `'momo (Maurice)'`.
- **Results formatting code:** This code will be executed after each row has been post-processed with the code specified in the 'row formatting code' box. A variable 'results' containing the array of the formatted rows can be manipulated to modify the action result. For example, `results = results.join(", ")` combined with the previous row formatting code, would mean results (and hence the output of the action) would be a string. Here `'momo (Maurice), baba (Bill)'`
- **Result Type:** The type of the action output. By default, the action returns an array of hash. With each matching row found represented by a hash, indexed by the database column name (as a symbol), for example `[{:name=>'Maurice',:login=>'momo'},{:name=>'Bill',:login=>'baba'}]`. The type chose should match the formatting code specified. No automatic type translation will be done, so all transformation needs to be explicit in the row and results formatting code boxes. In the example above, the output type should be set to 'string'

Inputs description

The list of inputs depends on the configuration of the Database Query action template as the query and formatting fields support dynamic inputs.

Outputs description

- *Query_results*: The formatted results of the query. If no formatting is provided, the results is an array of hash, otherwise the type of this outputs will depends on the specified output type.
- *Inserted Id*: Only if the query is an insert operation, the ID of the new record is returned.

Supported Actions

None

Dependencies

None

Operating Instructions

Notes: in order to be able to use Oracle or ODBC, some additional installation steps are required.

For Oracle: The Oracle client needs to be installed on the Orchestrator server.

The datasource to be accessed must be set up in the tnsnames.ora file. Before attempting a connection through Orchestrator, ensure that connectivity can be achieved and queries can be executed from the command line.

Once the oracle client is installed, the ruby-oci8 gem needs to be installed.

```
cd /opt/aspera/orchestrator      gem install
vendor/gems_linux-gnu/ruby-oci8-2.1.4.gem
```

For ODBC: The ODBC driver needs to be installed on the Orchestrator server (FreeTDS can be used on Linux).

The ODBC source to be accessed must be set up. Before attempting a connection through Orchestrator, ensure that connectivity can be achieved and queries can be executed from the command line.

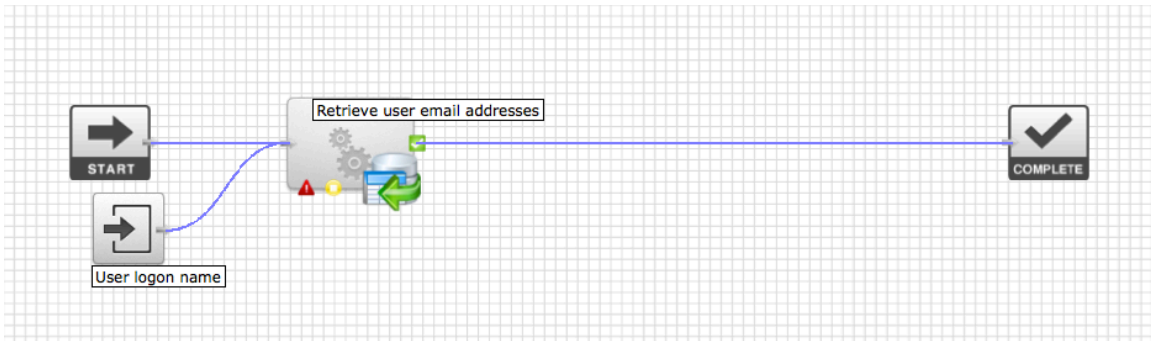
Once the ODBC drivers are installed, the ruby-odbc and dbd-odbc gems need to be installed.

```
cd /opt/aspera/orchestrator      gem install
vendor/gems_linux-gnu/ruby-odbc-0.99994.gem      gem install
vendor/gems_linux-gnu/dbd-odbc-0.2.5.gem
```

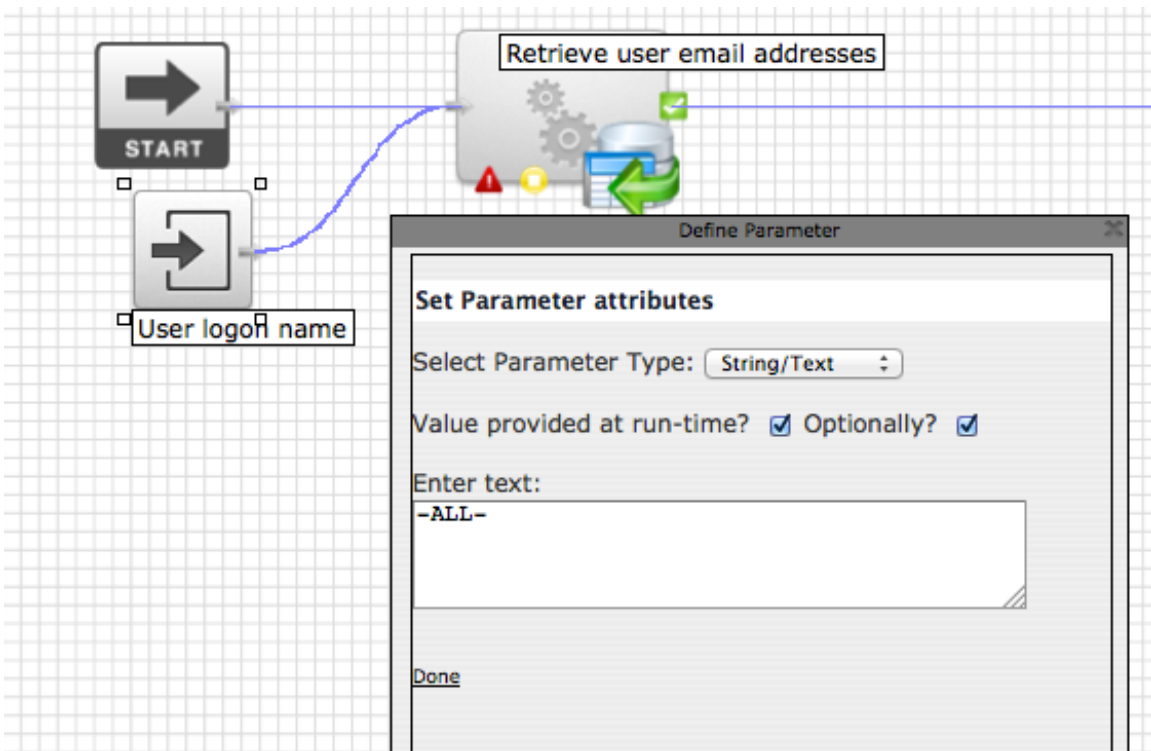
Sample workflows

We will create a small workflow whose goal is to retrieve information about active user from a database. For simplicity here we will use the Orchestrator database.

The workflow just as a single step for now:



The parameter is a runtime string and is defaulted to a value indicating no filtering is required.



If a value is provided, the goal is to only provide information for the specified user.

The database query action configuration is set as follow:

Database query configuration

Name	<input type="text" value="Retrieve user email addresses"/>
Comments	<input type="text"/>
Database type	<input type="text" value="Mysql"/>
Database host (mysql only)	<input type="text" value="localhost"/>
Database port (mysql only)	<input type="text" value="3306"/>
Database name (or odbc source)	<input type="text" value="orchestrator"/>
Database user	<input type="text" value="root"/>
Database password	<input type="password" value="....."/>
re-enter	<input type="password" value="....."/>
Database query	<input type="text" value="SELECT firstName, lastName, email FROM users WHERE"/>
Toggle editor	<input type="checkbox"/>
Row formatting code	<input type="text"/>
Toggle editor	<input type="checkbox"/>
Results formatting code	<input type="text"/>
Toggle editor	<input type="checkbox"/>
Result type	<input type="text" value="array"/>


The string used can contain static text and variable expanded at run-time. Variables are indicated using the syntax `<%=variable_name%>` embedded into static text. For example `'/tmp/<%=basename(full_name)%>'`
The row is represented as a variable 'row' and is a hash with the column name as a symbol key (i.e.: `row[:first_name]=>'Maurice'`)

The full query is:

```
SELECT firstName, lastName, email FROM users WHERE isActive = true <%=  
AND_REFINE_STATEMENT %>
```

Note the dynamic attribute, if it is blank, the query will return the data for all active user. To narrow the scope to a specific user login, for example ‘**admin**’, **AND_REFINE_STATEMENT** would need to be set to “**AND login = ‘admin’**”

If we run this workflow with the default runtime value, we get the following outputs for that step:




Retrieve user email addresses (Database query)

Status (attempt #): Complete (2)
Query returned 10 rows at 2014-09-08T11:19:10-07:00
Step Configuration [W343 Retrieve user email addresses](#)
Work Order: 8699

Start time: Mon Sep 08 11:19:10 -0700 2014
End time: Mon Sep 08 11:19:10 -0700 2014
Last update: Mon Sep 08 11:19:10 -0700 2014
Elapsed time: 127 sec
Processing time: 0

Inputs

Name	Value	Type
AND_REFINE_STATEMENT	-ALL-	String 

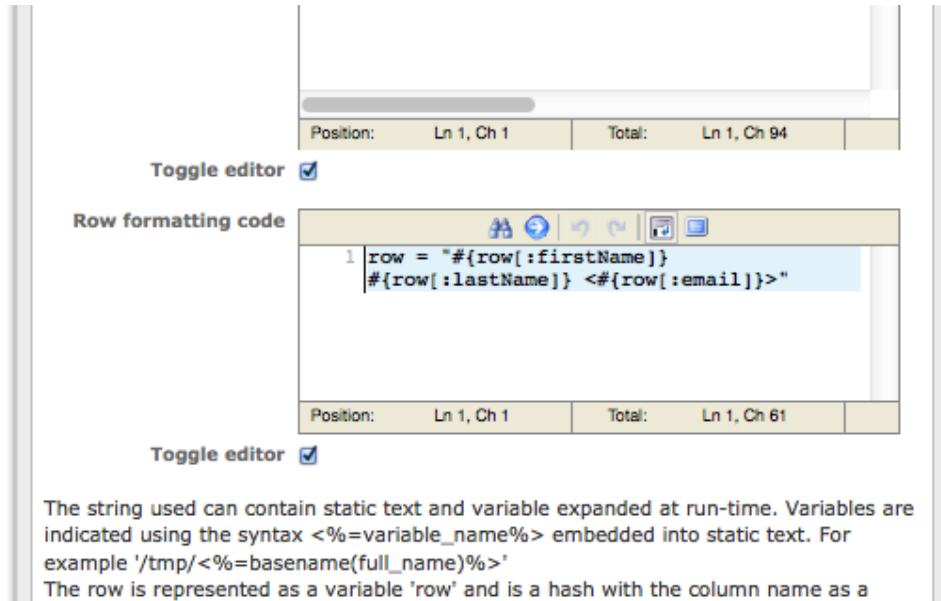
Outputs

Name	Value	Type
Query_results	<pre>{:email=>"mlusinchi@asperasoft.com", :lastName=>"admin", :firstName=>""}, {:email=>"system.workflow@asperasoft.com", :lastName=>"system", :firstName=>""}, {:email=>"", :lastName=>"ob", :firstName=>"b"}, {:email=>"" ,:lastName=>"" , :firstName=>""}, {:email=>"" ,:lastName=>"" , :firstName=>"other_op"}, {:email=>"mlusinchi@asperasoft.com", :lastName=>"Lusinchi", :firstName=>"Marc"}, {:email=>"amy.sita@asperatech.com", :lastName=>"Sita", :firstName=>"Amy"}, {:email=>"" ,:lastName=>"" ,:firstName=>""}, {:email=>"" ,:lastName=>"Lusinchi", :firstName=>"Marc-Olivier"}, {:email=>"marc@asperasoft.com", :lastName=>"dropbox", :firstName=>"log"}}</pre>	Array

Note that the result is an array of hash, with each hash representing the data for one row. Symbols are used as key, not string. So to get a value of a row, the syntax would be **output_value[0][:email]** (value here would be the string “mlusinchi@asperasoft.com”). And **output_value[0]['email']** would be nil.

Now let's format the row. Let's say we want the email address in rich format: **FirstName LastName** **<email@address>**

This is done by modifying the **row formatting** code field of the action configuration:



The screenshot shows a configuration window with two sections. The top section has a status bar with "Position: Ln 1, Ch 1" and "Total: Ln 1, Ch 94". Below it is a "Toggle editor" checkbox which is checked. The bottom section is titled "Row formatting code" and contains a text editor with the following code:

```
1 row = "#{row[:firstName]}  
  #{row[:lastName]} <#{row[:email]}>"
```

The status bar for this section shows "Position: Ln 1, Ch 1" and "Total: Ln 1, Ch 61". Below it is another "Toggle editor" checkbox which is checked. Below the editor, there is explanatory text:

The string used can contain static text and variable expanded at run-time. Variables are indicated using the syntax `<%=variable_name%>` embedded into static text. For example `'/tmp/<%=basename(full_name)%>'`
The row is represented as a variable 'row' and is a hash with the column name as a

The result is now



Retrieve user email addresses (Database query)

Status (attempt #): Complete (4)	Start time: Mon Sep 08 11:33:40 -0700 2014
Query returned 10 rows at 2014-09-08T11:33:40-07:00	End time: Mon Sep 08 11:33:40 -0700 2014
Step Configuration W343 Retrieve user email addresses	Last update: Mon Sep 08 11:33:40 -0700 2014
Work Order: 8699	Elapsed time: 997 sec
	Processing time: 0

Inputs		
Name	Value	Type
AND_REFINE_STATEMENT	-ALL-	String

Outputs		
Name	Value	Type
Query_results	<pre>[* admin <mlusinchi@asperasoft.com>", " system <system.workflow@asperasoft.com>", "b ob <>", " <>", "other_op <>", "Marc Lusinchi <mlusinchi@asperasoft.com>", "Amy Sita <amy.sita@asperatech.com>", " <>", "Marc-Olivier Lusinchi <>", "log dropbox <marc@asperasoft.com>"]</pre>	Array

Now let's assume we want to get the result as an aggregated string and we want to filter out the record with improper information.

example `'/tmp/<%=basename(full_name)%>'`

The row is represented as a variable 'row' and is a hash with the column name as a symbol key (i.e.: `row[:first_name]=>'Maurice'`)

Results formatting code

```
1 results = results.reject{|item|
  item.match(/.*<.+@.+\.+>/) ==
  nil}.join("<%=separator%>")
```

Position: Ln 1, Ch 91 Total: Ln 1, Ch 90

Toggle editor

As above, the string used can contain static text and variable expanded at run-time. The results are represented in an array name 'results' of which each line is by default a hash, unless the formatting as been changed above.

Result type string

The reject statement will cause all the entries not matching the provided format to be skipped. The join statement converts the array to a delimited string. The

separator to be used was made a dynamic attribute and would thus be whatever is specified as a string input “separator”.

Note that **the result type** was set to **string** to match what the **results formatting code** is returning.

Now since we have an extra input, we need to map it

The screenshot shows a workflow editor with a 'Retrieve user email addresses' step. The step is connected to 'START' and 'User logon name' inputs. A 'COMPLETE' button is visible. An 'Input Mappings' dialog is open, showing the mapping for the 'separator' input to a value of '.'.

Input Name	Type	Required?	Maps to	Pre-Processing
AND_REFINE_STATEMENT	string	Yes	User logon name (Parameter)	Set Pre-processing
separator	string	Yes	[Value]	Set Pre-processing

Here the mapping is done to inline value of “,” (comma then space)

If we relaunch, this time we get the following results:

Retrieve user email addresses (Database query)

Status (attempt #): Complete (1)
Query returned 10 rows at 2014-09-08T13:14:23-07:00

Step Configuration [W343 Retrieve user email addresses](#)
Work Order: 8700

Start time: Mon Sep 08 13:14:23 -0700 2014
End time: Mon Sep 08 13:14:23 -0700 2014
Last update: Mon Sep 08 13:14:23 -0700 2014
Elapsed time: 0 sec
Processing time: 0

Inputs

Name	Value	Type
AND_REFINE_STATEMENT	-ALL-	String
separator	,	String

Outputs

Name	Value	Type
Query_results	admin <mlusinchi@asperasoft.com>, system <system.workflow@asperasoft.com>, Marc Lusinchi <mlusinchi@asperasoft.com>, Amy Sita <amy.sita@asperatech.com>, log dropbox <marc@asperasoft.com>	String