# Odysseus Studio


- Odysseus Studio
  - Available sources
  - Controlling queries
  - PQL Operators
  - Outline
  - MEP Functions View
  - Users View
  - Operator Detail Info View
  - Configuration


## Odysseus Studio

**Important**:

- Currently, there is a bug in Studio: The menu entry "Show Stream Elements" is not working in Linux based systems. Under Windows you can simply ignore the grey and click on the menu entry.
- As a work around, the entries are now shows on the main menu level, images below are not updated!

Odysseus Studio is the RCP based frontend to develop and run persistent queries. Odysseus Studio needs no installation and can be extracted to any directory.

To run Odysseus a Java JDK 17 or higher needs to be installed. For a detailed tutorial for the installation look at How to install Odysseus

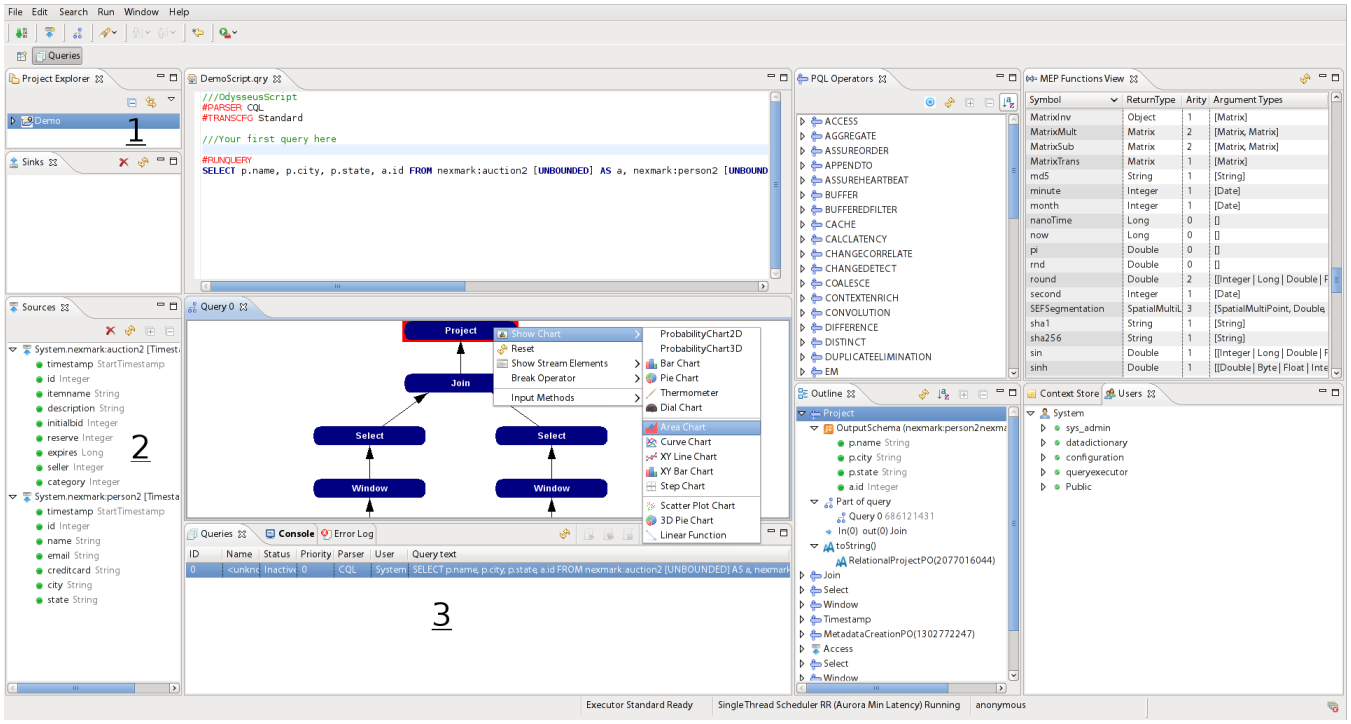
Start Odysseus Studio by running “studio” and choose a new workspace.

Hints:

- Do not use an existing Eclipse workspace.
- Avoid blanks in the workspace path.

If Odysseus Studio asks for a login, see How to install Odysseus for further information. You can check the mark to avoid typing login information again.

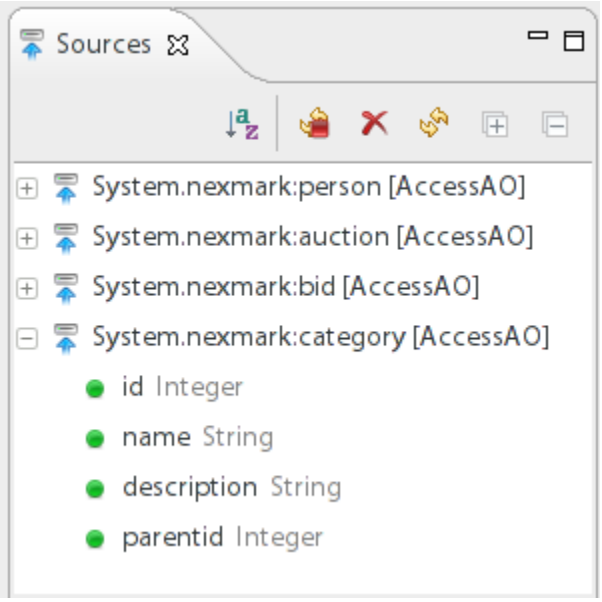
Initially, Odysseus creates a directory in the user’s home directory for configuration issues.




After the login process, the Odysseus RCP GUI starts up. In this GUI you can create a new Odysseus project by issuing a click with the right mouse button in area 1. When you have created a new project you can start creating Queries for your project using any soure listet in area 2 or creating own sources. You can start an Odysseus Query by a click on the Play-Button in the toolbar. All running queries are listet in area 3.

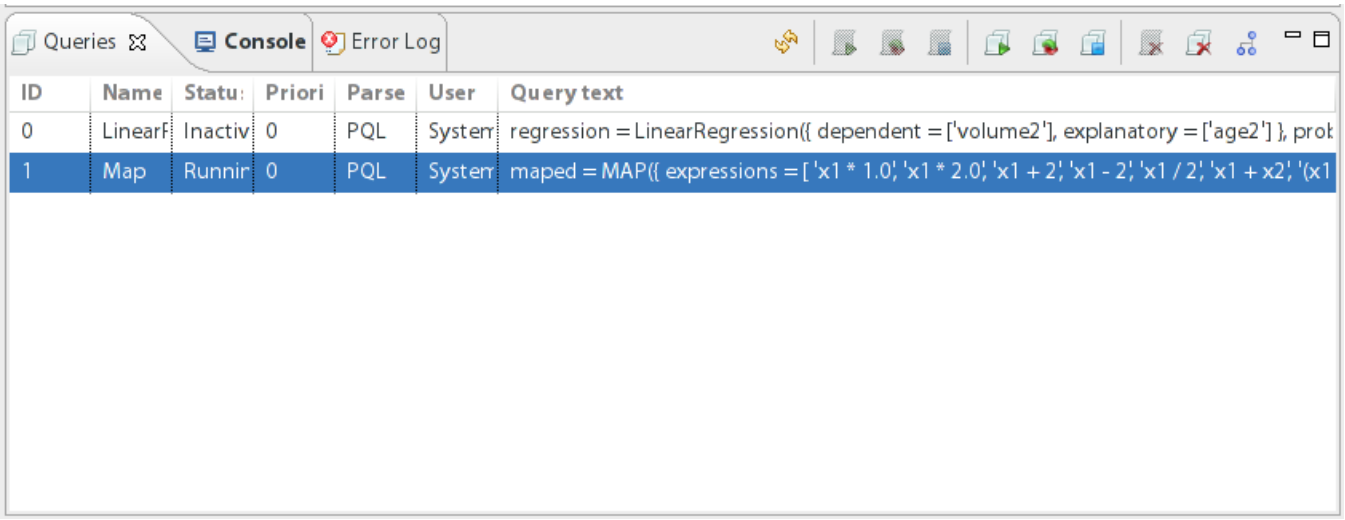
The Odysseus Studio provides a wide range of so called Views to make the definition, administration and controlling of processing queries very convenient. In the following, we will go through all available Views in Odysseus Studio.

## Available sources




The Sources View contains a list of available sources with their given name and schema information to be used in you processing query. Every time a new source is registered in Odysseus, a new source will be inserted in the list of available sources.

## Controlling queries

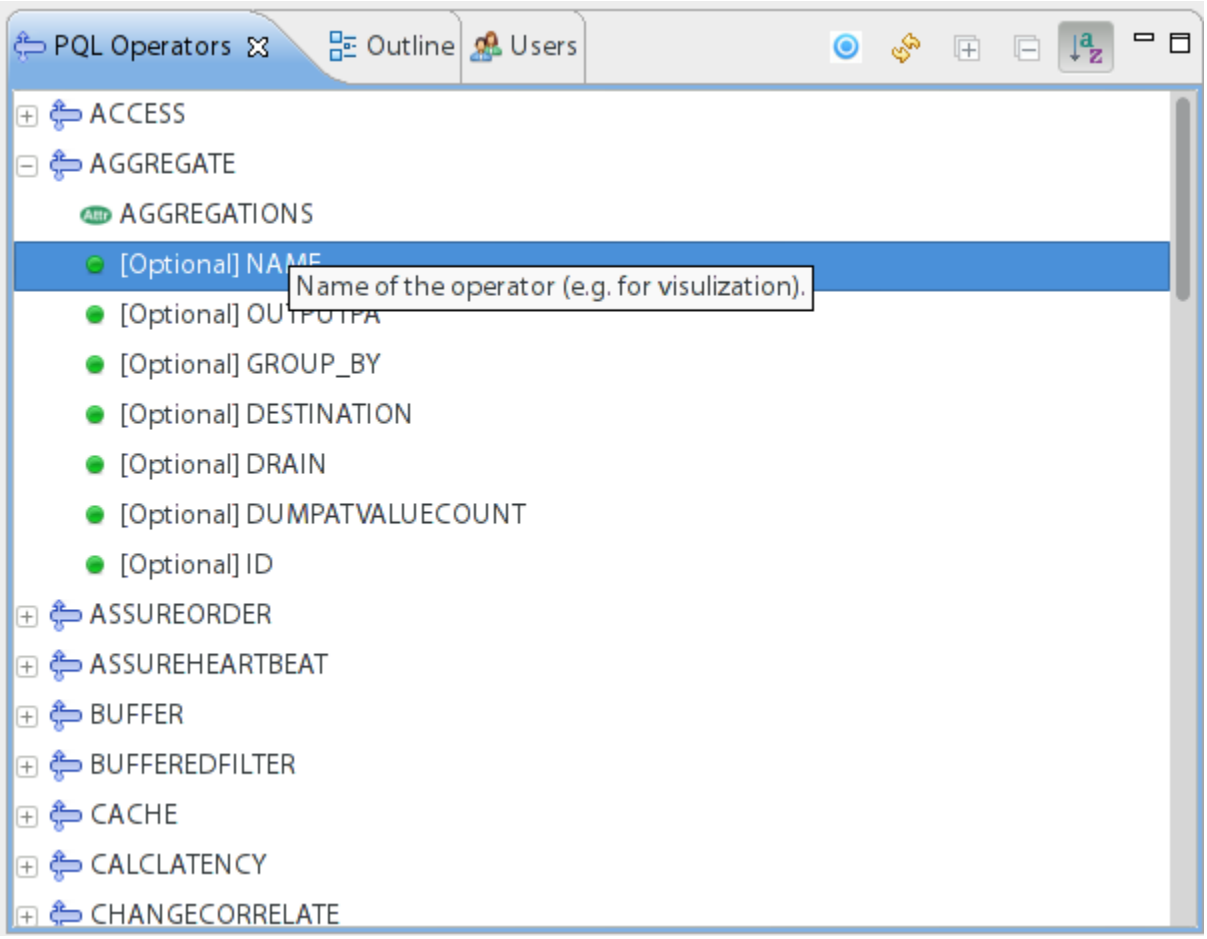



The Queries View allows you to control your processing queries from within Odysseus Studio. Using this view, you can start and stop queries, remove them from the list of queries or perform actions on multiple queries at once.

The View consists of 7 columns providing all necessary information about registered queries. In the first column, you have the unique identifier of you query. The second column holds the name of the query or the name of the last operator in the query if no name is set. The third column shows the status of the query, i.e., Running or Inactive. The fourth column holds the priority of the query. Column 5 shows the parser that is used to parse the query text, i. e., PQL or CQL. The sixth column shows the user who registered the query in Odysseus. Finally, the last column shows the query itself in the given language.

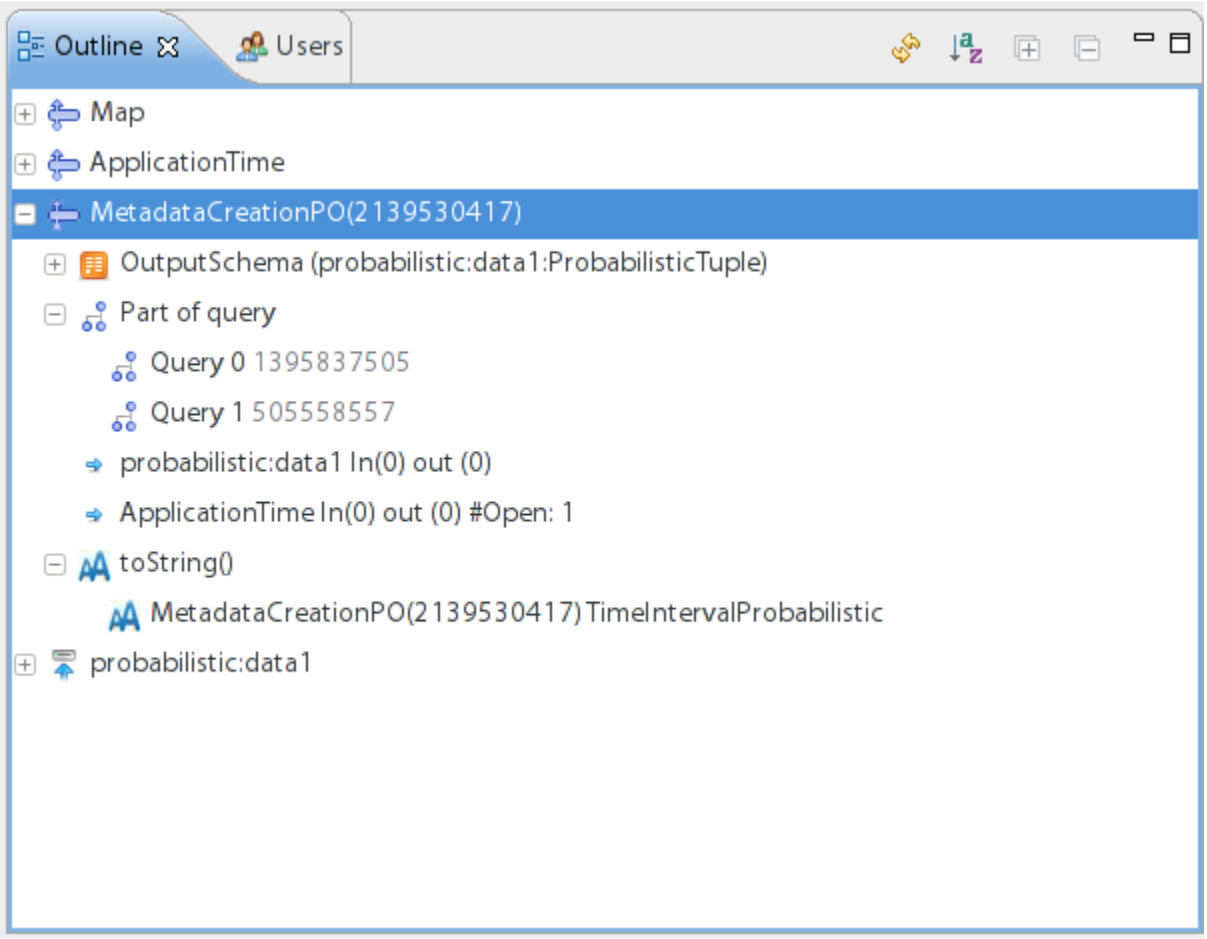
In the tool bar of the view you have a list of menu entries to control either the selected query, or performing actions on all queries. Further, the first entry in the tool bar can be used to reload the list of registered queries, and the last entry switches to the graph view to see the current selected query as a processing graph representation.

## PQL Operators



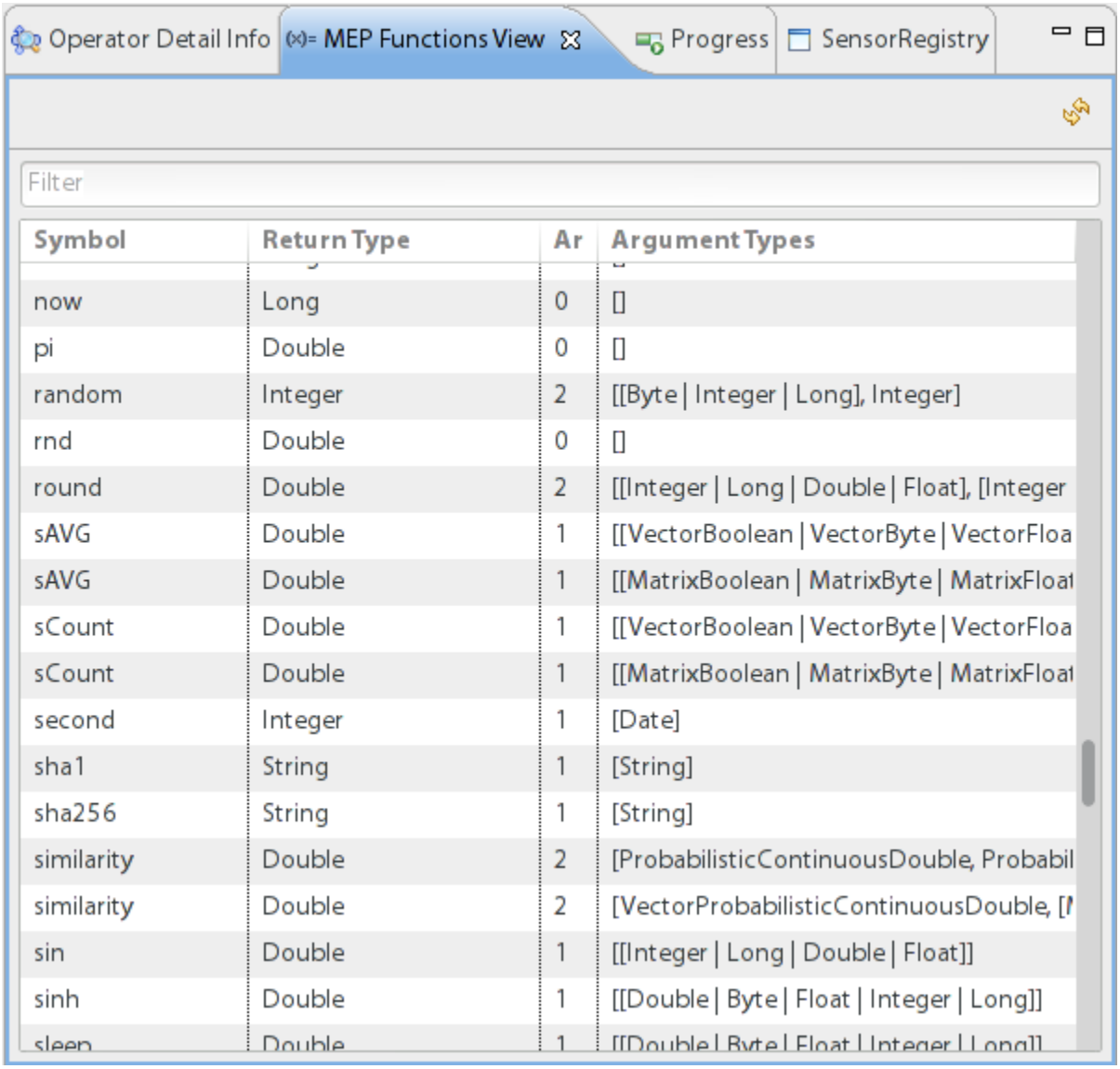

In the PQL Operators View you can see all available PQL Operators including their required and optional parameters. Further, you can get a description of each parameter by moving the mouse over it.

## Outline




The Outline View includes all operators of the current query including their output schema and queries they belong to. This is especially interesting in case of query sharing in which one operator is used in multiple queries to increase the processing performance.

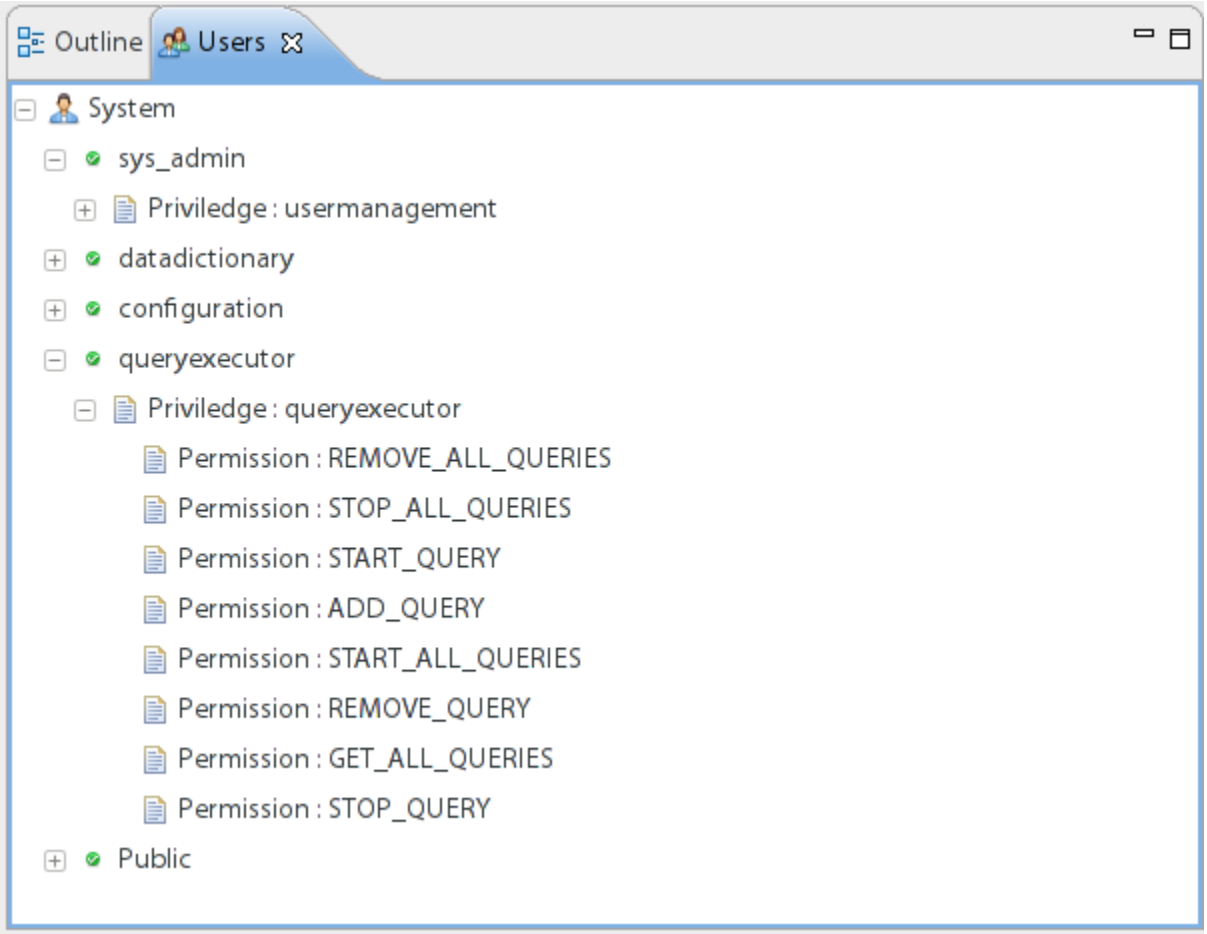
## MEP Functions View




| Symbol | Return Type | Ar | Argument Types |
|---|---|---|---|
| now | Long | 0 | [] |
| pi | Double | 0 | [] |
| random | Integer | 2 | [[Byte \| Integer \| Long], Integer] |
| rnd | Double | 0 | [] |
| round | Double | 2 | [[Integer \| Long \| Double \| Float], [Integer |
| sAVG | Double | 1 | [[VectorBoolean \| VectorByte \| VectorFloa |
| sAVG | Double | 1 | [[MatrixBoolean \| MatrixByte \| MatrixFloat |
| sCount | Double | 1 | [[VectorBoolean \| VectorByte \| VectorFloa |
| sCount | Double | 1 | [[MatrixBoolean \| MatrixByte \| MatrixFloat |
| second | Integer | 1 | [Date] |
| sha1 | String | 1 | [String] |
| sha256 | String | 1 | [String] |
| similarity | Double | 2 | [ProbabilisticContinuousDouble, Probabil |
| similarity | Double | 2 | [VectorProbabilisticContinuousDouble, [ |
| sin | Double | 1 | [[Integer \| Long \| Double \| Float]] |
| sinh | Double | 1 | [[Double \| Byte \| Float \| Integer \| Long]] |
| sleep | Double | 1 | [[Double \| Byte \| Float \| Integer \| Long]] |

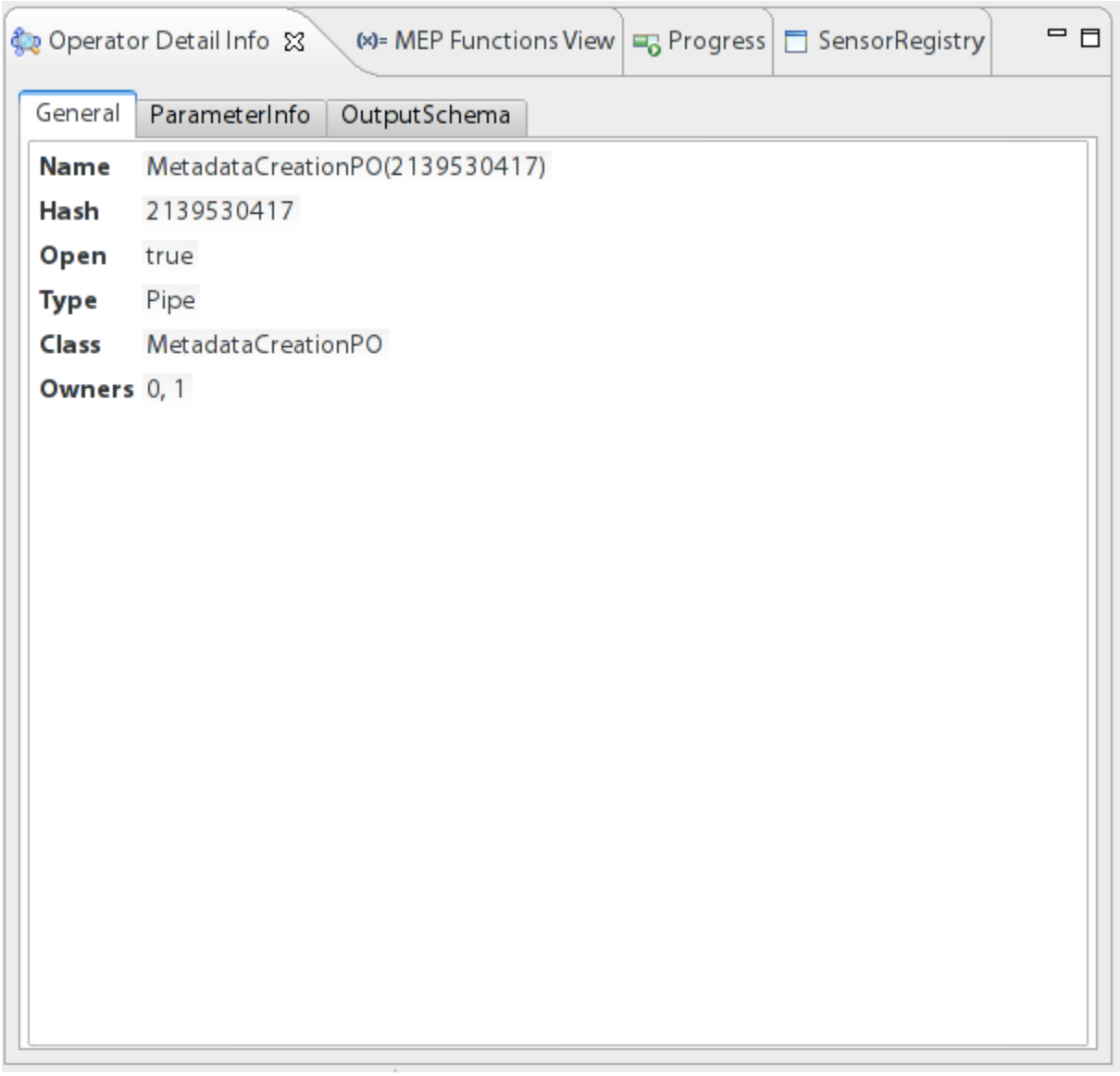
The MEP Functions View provides a convenient way to look up available functions to be used in filter and mapping operators. The list includes the symbol, the return type, the arity, and the possible parameter for the function.The first column is the symbol to be used in a query. As you can see, the symbol can be overwritten allowing you to use the function on different parameter settings with slightly different semantics, i.e., the *similarity* function can be used on two different parameter types and performs different calculations internally. The second column defines the return type of the function. The third column is the number of parameter the function takes. Finally, the fourth column includes a list of accepted parameter types, i.e., the *sin* function takes an integer, long, double, or float value for the first parameter and the *round* takes the same for the first and the second parameter. You can also restrict the list of MEP functions by providing a search string in the upper text field. Here, you can search either by wildcards or using regular expressions to filter for a particular function symbol.

## Users View




The users view holds a list of users and their privileges. In the example above we see the *queryexecutor* user that has all permissions for registering, starting, stopping, and dropping a query.

## Operator Detail Info View




The Operator Detail Info View allows a deeper look to the settings of an operator and is thus a perfect debugging tool for your processing queries.

An deeper Example can be found in Run Nexmark Example

## Configuration

Odysseus Studio can be configured by editing the `odysseusRCP.conf` file located in the Odysseus home directory. The following configuration options are available.

| Key | Possible values /options | Description |
|---|---|---|
| node.layout | • `horizontal` (default)<br>• `vertical` | Sets the layout direction of the operator graph. The **horizontal** option draws the graph from left (sources) to the right (sinks), the **vertical** option draws the graph from bottom (sources) to top (sinks). |
| layout. horizontal. distance.x | positive integer (default: 50) | Sets the distance between two nodes on the X axis for the horizontal layout. |

| | | |
|---|---|---|
| layout.horizontal.distance.y | positive integer (default: 50) | Sets the distance between two nodes on the Y axis for the horizontal layout. |
| layout.vertical.distance.x | positive integer (default: 100) | Sets the distance between two nodes on the X axis for the vertical layout. |
| layout.vertical.distance.y | positive integer (default: 75) | Sets the distance between two nodes on the Y axis for the vertical layout. |
| viewer.config | String | Sets the file_name of the configuration file for the operator graph visualization style. Odysseus Studio provides several default themes that can be set as a value:<br>• `symbol.xml`<br>• `symbol_light.xml` (default)<br>• `symbol_dark.xml`<br><br>Additionaly it is possible to set configuration files user defined visualization styles. The configuration files must be located in the Odysseus Home Directory and its path must be set relative to it.<br><br>More information on how to customize the operator graph visualization can be found here. |