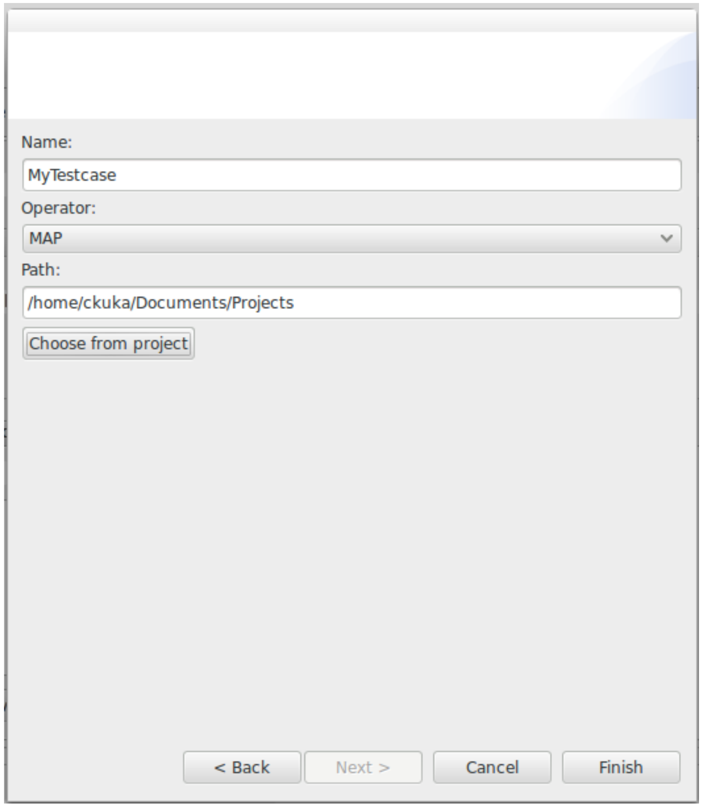
Testcase Generator

The Testcase Generator simplifies the creation of integration tests for operators. To use the Testcase Generator, the developer feature has to be installed in the current Odysseus setup.

Testcase Wizard

A new Testcase for an operator can be created by issuing the New Test Case wizard using File -> New… (Strg+N) in the menubar.



Name: MyTestcase

Operator: MAP

Path: /home/ckuka/Documents/Projects

Applying the settings by pressing the “Finish” button will open the Testcase editor for editing.

Testcase Editor

The Testcase Editor has the following structure:

In the name, operator type and output path is defined. The name is used to identify the generated files later. Restrictions on the name may depend on the current operating system. The operator type can be selected from the operator type drop down list. The content of the operator type drop down list depends on the current available operators in the Odysseus setup. If an operator is missing in the list of operators it is an indicator that the bundle with the operator is currently not loaded. The output directory field is used to specify the output path for the generated query and input/output files. The output path can be a relative or an absolute path.

In the input schema for each port can be specified. Each tab defines the schema for exactly one input port of an operator. The number of ports is limited to 10 input ports, e.g., the Union operator for example can only be tested with a maximum of 10 input streams. For each attribute of the schema it is possible to define the name, the data type, the min and max value and if the attribute can be null.

- **Name**: The name of the attribute in the schema of the given input port.
- **Type**: The data type of the attribute. The available set of data type depends on the available data types in the current Odysseus setup.
- **Min**: The minimum value defines the lowest value that should be used as an instantiation of an attribute.
- **Max**: The maximum value defines the greatest value that should be used as an instantiation of an attribute.
- **Null**: A Flag indicating if an instantiation of the attribute can be *NULL*.

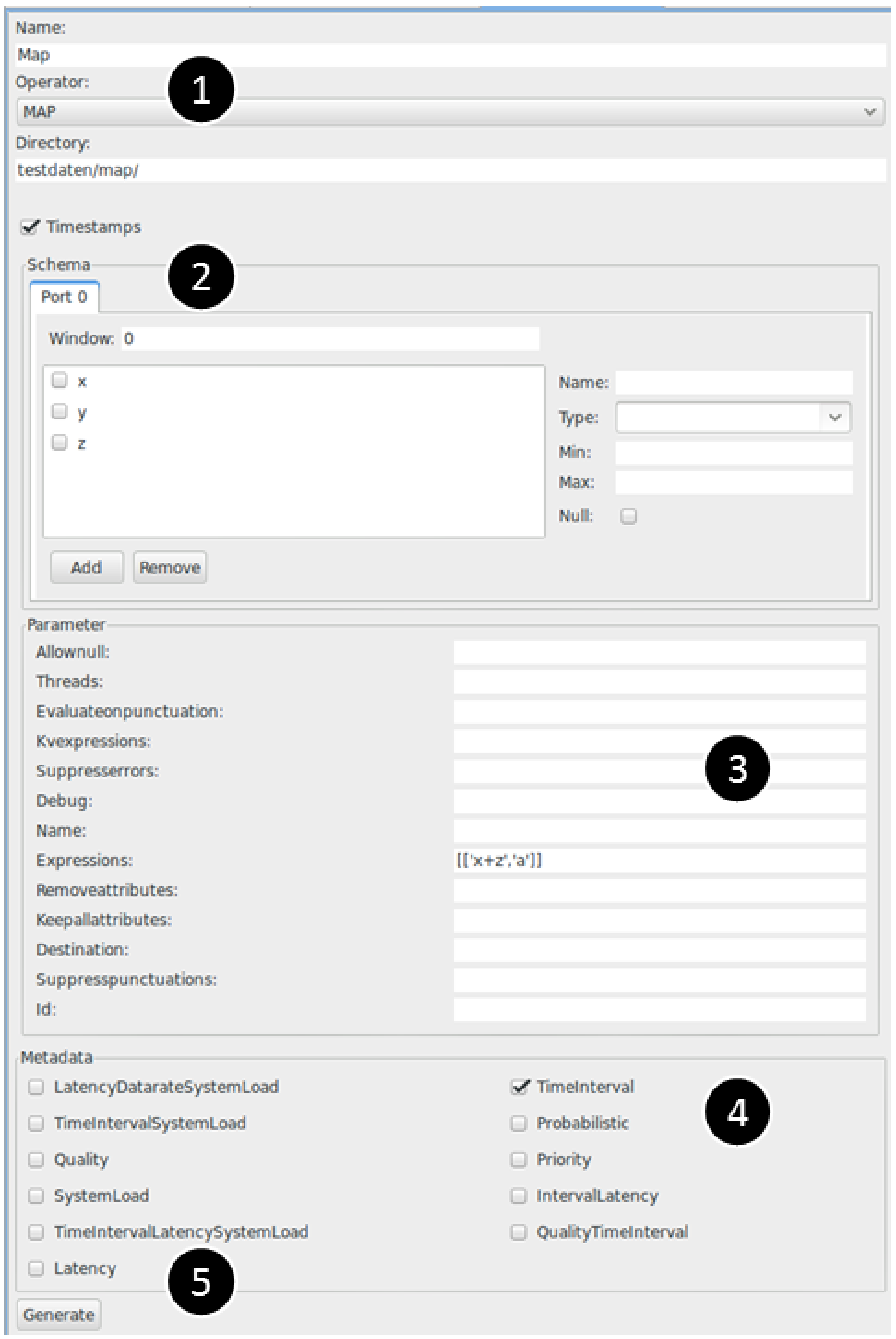
Based on this information the Testcase Generator will generate an input stream. To do so, a permutation of the minimum value, maximum value, zero value, average value, a null value (if selected) of each all input attributes will be generated. The values of zero and average may depend on the selected data type. In case of a double value, the zero value is 0 and the average value is the average between the min and max value.

In addition, the Testcase Generator can add a timestamp to each entry in the input stream by adding an additional attribute called *timestamp*. The value is an increasing numeric sequence starting by 0.

To configure the operator under test, the operator specific parameters can be defined in . The syntax for the parameters are the same as in PQL. Consult the operator specific documentation in the Odysseus Wiki for mandatory and optional parameter and accepted parameter values.

Next, the metadata combinations that should be used to test the operator can be selected in . For each selected metadata or metadata combination, the generator will create a test query file. Be aware, that not every operator can be used with every metadata combination. Consult the operator specific documentation in the Odysseus Wiki for operator support.

At the input/output and query file can be generated. The generated input and output files are text based CSV files. The generated test case is a PQL query file as shown in the following example. In the first lines of the file is the header. The header of the generated query file contains the name of the operator under test, the generation date and the system user name. Further, it lists all operator specific parameters with its defined values. Next, a set of Odysseus Script commands follows to make sure, the query is running without any interferences of other queries. The main query consists of a declaration of the access operator with the used metadata combination to read the generated input streams and the operator under test with its specific parameters.



Name: Map

Operator: MAP

Directory: testdaten/map/

✓ Timestamps

Schema

Port 0

Window: 0

☐ x
☐ y
☐ z

Name:
Type:
Min:
Max:
Null: ☐

Parameter

Allownull:	
Threads:	
Evaluateonpunctuation:	
Kvexpressions:	
Suppresserrors:	
Debug:	
Name:	
Expressions:	[['x+z','a']]
Removeattributes:	
Keepallattributes:	
Destination:	
Suppresspunctuations:	
Id:	

Metadata

☐ LatencyDatarateSystemLoad	✓ TimeInterval
☐ TimeIntervalSystemLoad	☐ Probabilistic
☐ Quality	☐ Priority
☐ SystemLoad	☐ IntervalLatency
☐ TimeIntervalLatencySystemLoad	☐ QualityTimeInterval
☐ Latency	

```
/// Odysseus Testcase: Map
/// Operator: MAP
/// Date: Sat Aug 15 13:43:54 CST 2015
/// User: ckuka
/// Parameter:
///  ALLOWNULL:
///  THREADS:
///  EVALUATEONPUNCTUATION:
///  KVEXPRESSIONS:
///  SUPPRESSERRORS:
///  DEBUG:
///  NAME:
///  EXPRESSIONS: [['x+z','y']]
///  REMOVEATTRIBUTES:
///  KEEPALLATTRIBUTES:
///  DESTINATION:
///  SUPPRESSPUNCTUATIONS:
///  ID:
#PARSER PQL
#DROPALLQUERIES
#DROPALLSINKS
#DROPALLSOURCES
#ADDQUERY
input0 = ACCESS({
    source='source0_TimeInterval',
    wrapper='GenericPull',
    transport='file',
    protocol='SimpleCSV',
    dataHandler='Tuple',
    metaattribute=[
                  'TimeInterval'
                  ],
    options=[
        ['filename', '${BUNDLE-ROOT}/testdaten/map/input0.csv'],
        ['csv.delimiter', ';'],
        ['csv.trim', 'true']
        ],
    schema=[['timestamp', 'STARTTIMESTAMP'], ['x', 'Double'], ['y', 'Double'], ['z', 'Double']]})
output = MAP({EXPRESSIONS=[['x+z','y']]}, input0)
```