Access operator

The access operator can be used to integrate new sources into Odysseus. Further information can be found in the Documentation to the Access Operator Framework.

Remark: There is no need to define an access operator as view (:=) or source (::=). Each access operator is automatically a source with name source. For most cases the assignment is only for parsing purposes (see example below).

Parameter

- `source:` The name of the access operator. **Remark**: This name must be different to all source names and all view or stream definitions! A new source will be added to the data dictionary automatically.
- `wrapper:` In Odysseus the default wrappers are *GenericPush* and *GenericPull*
- `transport:` The transport defines the transport protocol to use.
- `protocol:` The protocol parameter defines the application protocol to transform the processing results.
- `datahandler:` This parameter defines the transform of the single attributes of the processing results.
- `options:` Transport protocol and application protocol depending options
- `schema:` The output schema of the access operator (may depend on the protocol handler)
- `metaattribute`: What kind of meta data should the source provide.
- `readMetaData`: If you receiving data that already contains meta data in an Odysseus based format (e.g. if you are sending from another Odysseus node) this value must be set to 'true' (default is false), to read this meta data. **Important:** In this case, the schema should not contain Timestamp attributes, else the meta data may be overwritten.
- `dateformat`: If you are using STARTTIMESTAMPSTRING or ENDTIMESTAMPSTRING, this option is define the format of the timestamp string. See https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html for an explanation of the needed parameters. Hint: If you timestamp does not contain a timezone and this containing timezone is different that the local timezone you will need to use the Timestamp operator for processing.

There are some options that are not used by transformation:

- BaseTimeUnit: Timeunit used for long based input start/endtimestamps (be careful, the option is case sensitive). The TimeUnits are case sensitive as well, e.g. SECONDS (https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/TimeUnit.html)

Example

PQL

Access Operator

```
input = ACCESS({source='Source',
wrapper='GenericPush',
transport='TCPClient',
protocol='CSV',
dataHandler='Tuple',
METAATTRIBUTE = ['TimeInterval','Latency','Datarate'], /// or: METAATTRIBUTE = 'TimeIntervalLatencyDatarate'
options=[['host', 'example.com'],['port', '8080'],['read', '10240'],['write', '10240']],
schema=[
['id', 'Double'],
['data', 'String']]
})
```

Here a complexer example about what could be possible:

```
in = ACCESS({
          transport = 'FILE',
          datahandler = 'Tuple',
          source = 'AVKVIN1',
          wrapper = 'GenericPull',
          protocol = 'simplecsv',
          dateformat = 'dd.MM.yyyy HH:mm:ss',
          options = [
            ['filename','${FILEPATH}'],
            ['skipFirstLines','5'],
            ['delimiter',';'],
            ['decimalseparator',','],
            ['csv.floatingformatter','#,##'],
            ['debug','true'],
            ['maxLines','323944'] /// there are empty lines at the end of the file
          ],
          schema = [
            ['ts','STARTTIMESTAMPSTRING'],
            ['1SMA10CW001/XQ12.OS','Double',[['Unit','t/h'],['Info','This is explaining text']]],
            ['1LBA10CF901/GW/MA1.U','Double',[['Unit','t/h'],['Info','Other Text']]],
            ['1LBA10CP001/GW/MA1.U','Double',[['Unit','°C'],['Info','...']]],
            ['1HAH25CT001/XQ01/MA2.U','Double',[['Unit','°C'],['Info','...']]],
            ['1HAH25CT002/XQ01/MA2.U','Double',[['Unit','°C'],['Info','...']]],
            ['1HAH35CT001/XQ01/MA2.U','Double',[['Unit','°C'],['Info','..']]]
          ]
        }
      )
```

There are some special logical Operators, that allow an easier handling of the ACCESS-Operation (but are semantically equal)

- CSVFileSource
- OdysseusSource