

Odysseus Cheat Sheet

Full Grammar of PQL

```
QUERY          = (STREAM | VIEW | SOURCE)+
STREAM         = STREAM "=" OPERATOR
VIEW          = VIEWNAME ":"=" OPERATOR
SOURCE        = SOURCENAME "::-=" OPERATOR
OPERATOR       = QUERY | [OUTPUTPORT ":"] OPERATOR
               = (" (PARAMETERLIST [ "," OPERATORLIST ]
                  | OPERATORLIST) ")"
OPERATORLIST  = [ OPERATOR ("," OPERATOR)* ]
PARAMETERLIST = "{" PARAMETER ("," PARAMETER)* "}"
PARAMETER     = NAME "=" PARAMETERVALUE
PARAMETERVALUE = LONG | DOUBLE | STRING | PREDICATE |
               LIST | MAP
LIST          = "[" [PARAMETERVALUE (","
                  PARAMETERVALUE)*] "]"
MAP           = "[" [MAPENTRY ("," MAPENTRY)*] "]"
MAPENTRY      = PARAMETERVALUE "=" PARAMETERVALUE
STRING        = "'" [~']* "'"
PREDICATE     = PREDICATETYPE "(" STRING ")"
```

Advanced Operators

ASSOCIATIVESTORAGE

This operator stores streaming data in an associative storage

```
SIZES
INDEX
SUPPRESSPUNCTUATIONS If set to true, no punctuations will be
                      delivered from this operator. Default is
                      false
VALUE
DEBUG                 Flag, that this operator should be de-
                      bugged.
```

```
STORAGENAME
HIERARCHY
```

BUFFEREDFILTER

This operator can be used to reduce data rate. It buffers incoming elements on port 0 (left) for bufferTime and evaluates a predicate over the elements on port 1 (right). If the predicate for the current element e evaluates to true, all elements from port 0 that are younger than e.startTimeStamp()-bufferTime will be enriched with e and delivered for deliverTime. Each time the predicate evaluates to true, the deliverTime will be increased.

```
DELIVERTIME
PREDICATE
BUFFERTIME
SUPPRESSPUNCTUATIONS If set to true, no punctuations will be
                      delivered from this operator. Default is
                      false
DEBUG                 Flag, that this operator should be de-
                      bugged.
```

COALESCE

This Operator can be used to combine sequent elements, e.g. by a set of grouping attributes or with a predicates. In the attributes case, the elements are merged with also given aggregations functions, as long as the grouping attributes (e.g. a sensorid) are the same. When a new group is opened (e.g. a measurement from a new sensor) the old aggregates values and the grouping attributes are created as a result. In the predicate case, the elements are merged as long as the predicate evaluates to false, i.e. a new tuple is created when the predicates evaluates to true.

```
DRAINATDONE         If set to true (default), elements are
                      not yet written will be written at
                      done.
FASTGROUPING        Use hash code instead of tuple com-
                      pare to create group. Potentially un-
                      safe!
```

```
OUTPUTPA
CREATEONHEARTBEAT
PREDICATE           Do not use. Use StartPredicate and
                      EndPredicate instead.
```

```
MAXELEMENTSPERGROUP
ENDPREDICATE
DEBUG              Flag, that this operator should be
                      debuged.
```

```
HEARTBEATRATE
STARTPREDICATE
USEROUNDRBINALLOCATION Enables RoundRobin allocation.
                      This is used in multithreaded exe-
                      cution for selecting the specific
                      thread
```

```
ATTR
NUMBEROFTHREADS    Use multiple threads for execution
                      (only possible if grouping attributes
                      are set)
```

```
DUMPATVALUECOUNT
AGGREGATIONS
SUPPRESSPUNCTUATIONS If set to true, no punctuations will
                      be delivered from this operator. De-
                      fault is false
```

```
DRAINATCLOSE       If set to true (default is false), el-
                      ements are not yet written will be
                      written at close.
```

```
DRAIN              If set to true (default), elements are
                      not yet written will be written at
                      done.
```

```
MAXBUFFERSIZE      Defines the size of the buffers used
                      in multithreaded execution
```

CONVOLUTION

This operator applies a convolution filter, which is often used in electronic signal processing or in image processing to clean up wrong values like outliers. The idea behind the convolution is to correct the current value by looking at its neighbours. The number of neighbours is the size of the filter. If, for example, SIZE=3, the filter uses the three values before the current and three values after the current value to correct the

current value. Therefore, the filter does not deliver any results for the first SIZE values, because it also needs additionally SIZE further values after the current one!

```
ATTRIBUTES
GROUP_BY
SIZE
SUPPRESSPUNCTUATIONS If set to true, no punctuations will be
                      delivered from this operator. Default is
                      false
```

```
OPTIONS
FUNCTION
DEBUG              Flag, that this operator should be de-
                      bugged.
```

FASTMEDIAN

Calculate the median for one attribute in the input tuples

```
PERCENTILES
GROUP_BY
ATTRIBUTE
ROUNDINGFACTOR
HISTOGRAM
NUMERICAL
SUPPRESSPUNCTUATIONS If set to true, no punctuations will be
                      delivered from this operator. Default is
                      false
DEBUG              Flag, that this operator should be de-
                      bugged.
```

```
APPENDGLOBALMEDIAN If a GROUP_BY element is given, the
                    global median (i.e. median without re-
                    specting groups) will be annotated to
                    each element.
```

GENERATOR

Generates missing values in a stream

```
EXPRESSIONS
GROUP_BY
FREQUENCY
PREDICATE
SUPPRESSPUNCTUATIONS If set to true, no punctuations will be
                      delivered from this operator. Default is
                      false
DEBUG              Flag, that this operator should be de-
                      bugged.
MULTI
```

TOPK

Calculate the top k elements of the input	
FASTGROUPING	Use hash code instead of tuple compare to create group. Potentially unsafe!
GROUP_BY	
TRIGGERONLYBYPUNCTUATION	If set to true, output is only generated when punctuation arrives.
RECALCSCORE	Sometime the score for an elements depends on state information. Set recalcScore to true to update for each (!) stored element every time a new output is triggered.
K	The number of elements to sort
SCORINGFUNCTION	The scoring function for ordering
TEARDOWNFUNCTION	This function is called for every input element after calculating the score.
CLEANUPPREDICATE	This (optional) predicate is used to clean up the state after processing the input.
TRIGGERBYPUNCTUATION	If set to true, output is only generated when punctuation arrives.
DEBUG	Flag, that this operator should be debuged.
SETUPFUNCTION	This function is called for every input element before calculating the score.
TIEWITHTIMESTAMP	If two elements have the same score, this value can be used to define an order by time stamps. (Default is false)
DESCENDING	Sort descending (default is true)
PRESCOREFUNCTION	This function be will called on the input before each element is scored. Typically used in case where recalcScore is set to true.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
SUPPRESSDUPLICATES	If set to true (default), output is only generated when a new top k set is available
UNIQUEATTRIBUTES	
ADDScore	If set to true, the score value will be added to each output element in the top k list. Default is false.

TUPLEAGGREGATE

Select from all elements of a window on with the given method	
ATTRIBUTE	Attribute on which the method is evaluated
METHOD	Method to use (MIN, MAX, LAST, FIRST)
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

UDO

Calls a user defined operator	
ATTRIBUTES	
INIT	
CLASS	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

Base Operators

AGGREGATE

Aggetations on attributes e.g Min, Max, Count, Avg, Sum and grouping.	
DRAINATDONE	If set to true (default), elements are not yet written will be written at done.
FASTGROUPING	Use hash code instead of tuple compare to create group. Potentially unsafe!
GROUP_BY	
OUTPUTPA	
DEBUG	Flag, that this operator should be debuged.
USEROUNDRBINALLOCATION	Enables RoundRobin allocation. This is used in multithreaded execution for selecting the specific thread
NUMBEROFTHREADS	Use multiple threads for execution (only possible if grouping attributes are set)
DUMPATVALUECOUNT	
AGGREGATIONS	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DRAINATCLOSE	If set to true (default is false), elements are not yet written will be written at close.
DRAIN	If set to true (default), elements are not yet written will be written at done.
MAXBUFFERSIZE	Defines the size of the buffers used in multithreaded execution

AGGREGATION

Aggetations on inputAttributeIndices e.g Min, Max, Count, Avg, Sum and grouping.	
EVAL_BEFORE_REMOVE_OUTDATING	
GROUP_BY	
EVAL_AT_DONE	
AGGREGATIONS	
EVAL_AT_NEW_ELEMENT	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
OUTPUT_ONLY_CHANGES	
DEBUG	Flag, that this operator should be debuged.
EVAL_AT_OUTDATING	
COMMAND	
This operator executes commands on other operators or services.	
COMMANDEXPRESSION	Expression for the commands, e.g. an attribute or a string
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

DIFFERENCE

This operator calculates the difference between two input sets.	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

DISTINCT

This operator removes duplicates.	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

DUPLICATEELIMINATION

Removes duplicates (Depending on the time model!)	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

EXISTENCE

This operator tests an existence predicate and can be used with the type EXISTS (semi join) and NOT_EXISTS (anti semi join). The predicates can be evaluated against the element from the first input and the second input. Semi join: All elements in the first input for which there are elements in the second input that fulfills the predicate are sent. Semi anti	
---	--

join: All elements in the first input for which there is no element in the second input that fulfills the predicate are sent.
PREDICATE

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

TYPE

FILTER

Filters elements of the input stream. If predicate evaluates to true, element will be sent to port 0 else to port 1.

PREDICATEISUPDATEABLE If set to true, the predicate of the select can be updated with punctuations.

PREDICATE

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

HEARTBEATRATE

INTERSECTION

This operator calculates the intersection between two input sets.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

JOIN

Operator to combine two datastreams based on the predicate

ASSUREORDER If set to false, the operator will not guarantee order in output. Default is true

PREDICATE Predicate to filter combinations

SWEEPAREANAME Overwrite the sweep area

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

CARD Type of input streams. For optimization purposes: ONE_ONE, ONE_MANY, MANY_ONE, MANY_MANY

LEFTJOIN

Operator to combine two datastreams based on the predicate. All attributes from the first (left) source remain. If an element from the first source has no join partner, it will also be part of

the output stream and the output schema contains null values

for the missing fields.

ASSUREORDER If set to false, the operator will not guarantee order in output. Default is true

PREDICATE Predicate to filter combinations

SWEEPAREANAME Overwrite the sweep area

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

CARD Type of input streams. For optimization purposes: ONE_ONE, ONE_MANY, MANY_ONE, MANY_MANY

MAP

Performs a mapping of incoming attributes to out-coming attributes using map functions. Odysseus also provides a wide range of mapping functions. Hint: Map is stateless. To used

Map in a statebased fashion see: StateMap

ALLOWNULL If set to true (default) and an error occurs in calculation a null value is added to the element. Else the element is skipped and no output is produced. Default is true.

THREADS Number of threads used to calculate the result.

EVALUATEONPUNCTUATION If set to true, map will also create an output (with the last read element) when it receives a punctuation.

SUPPRESSERRORS If set to true calculation errors will not appear in log or console. Could be helpful in scenarios where null values are allowed.

DEBUG Flag, that this operator should be debuged.

EXPRESSIONS A list of expressions.

REMOVEATTRIBUTES If keepInput is set to true, you can here provides attributes that should not be part of the output.

EXPRESSIONSUPDATEABLE If set to true, the expressions can be updated with punctuations Does not work in threaded mode.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

KEEPINPUT If set to true, all attributes of the input are also part of the output, so there is no need to repeat all attributes.

MERGE

Merge different input streams into one stream with "first comes first served" semantics.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

PROJECT

Make a projection on the input object (i.e. filter attributes)

ATTRIBUTES A list of attributes that should be used.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

RENAME

Renames the attributes

ISNOOP A flag to avoid removing this operator even if nothing in the schema is changed.

NOOP A flag to avoid removing this operator even if nothing in the schema is changed.

ALIASES The new list of attributes. Must be exactly the same length as in the input schema.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

TYPE The new type name of the output schema.

PAIRS If set to true, aliases will be interpreted as pairs oldAttribute, new Attribute.

SELECT

The select operator filters the incoming data stream according to the given predicate.

PREDICATEISUPDATEABLE If set to true, the predicate of the select can be updated with punctuations.

PREDICATE

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

HEARTBEATRATE

SETSYSTEMTIME

The SetSystemTime operator sets the system time to the timestamp of incoming elements when the difference is too big.

THRESHOLD Max allowed difference between system time and element time stamp before system time is set

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debugged.

SORT

Sort operator

ATTRIBUTES A list of attributes that should be used.

ASCENDING The sort of each attribute

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debugged.

STATEMAP

Performs a mapping of incoming attributes to out-coming attributes using map functions. Odysseus also provides a wide range of mapping functions. Hint: StateMap can use history information. To access the last n.th version of an attribute use

”_last.n.” Mind the two ”_.” at the beginning!

ALLOWNULL If set to true (default) and an error occurs in calculation a null value is added to the element. Else the element is skipped and no output is produced. Default is true.

GROUP_BY

THREADS Number of threads used to calculate the result.

EVALUATEONPUNCTUATION If set to true, map will also create an output (with the last read element) when it receives a punctuation.

SUPPRESSERRORS If set to true calculation errors will not appear in log or console. Could be helpful in scenarios where null values are allowed.

DEBUG Flag, that this operator should be debugged.

ALLOWNULLINOUTPUT

EXPRESSIONS A list of expressions.

REMOVEATTRIBUTES If keepInput is set to true, you can here provides attributes that should not be part of the output.

EXPRESSIONSUPDATEABLE If set to true, the expressions can be updated with punctuations Does not work in threaded mode.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

KEEPINPUT If set to true, all attributes of the input are also part of the output, so there is no need to repeat all attributes.

SYNCHRONIZE

Synchronizes different input streams

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DRAINATCLOSE If set to true (default is false), this buffer be emptied when calling close. Remark: Could lead to longer termination time!

DEBUG Flag, that this operator should be debugged.

TIMESTAMP

This Operator can be used to update the timestamp information in the meta data part. Be careful because this

may lead undefined semantics

DATEFORMAT If using a string for date information, use this format to parse the date (in Java syntax).

MONTH The name of the attribute for the month part of the start timestamp for application time

HOURL The name of the attribute for the hour part of the start timestamp for application time

FACTOR A multiplication factor for a single attributed timestamp to calc milliseconds (e.g. if input is seconds, use 1000 here)

DEBUG Flag, that this operator should be debugged.

CLEAREND If set to true, the end timestamp will be set to infinity

LOCALE Interpret the date string with this locale

YEAR The name of the attribute for the year part of the start timestamp for application time

SYSTEMTIME If set to true, system time instead of application time will be used

OFFSET An offset in milliseconds that will be added to the timestamp

START The name of the attribute for the start timestamp for application time

END The name of the attribute for the end timestamp for application time

MINUTE The name of the attribute for the minute part of the start timestamp for application time

SECOND The name of the attribute for the second part of the start timestamp for application time

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

MILLISECOND The name of the attribute for the millisecond part of the start timestamp for application time

TIMEZONE The timezone in Java syntax.

DAY The name of the attribute for the day part of the start timestamp for application time

UNION

Merges different input streams. (Typically preserves input order. Depending on the processing model)

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DRAINATCLOSE If set to true (default is false), this buffer be emptied when calling close. Remark: Could lead to longer termination time!

DEBUG Flag, that this operator should be debuged.

Benchmark Operators

CALCLATENCY

Odysseus has some features to measure the latency of single stream elements. This latency information is modeled as an interval. An operator in Odysseus can modify the start point of this interval. This operator sets the endpoint and determines the place in the query plan, where the latency measurement finds place. There can be multiple operators in the plan, to measure latency at different places.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

CLOSESTREAM

This operator allow to stop stream processing based on a predicate.

PREDICATE

COUNT

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

LATENCYTOPAYLOAD *(Deprecated)*

Deprecated: You Latency.start, Latency.end, Latency.latency etc. directly as attributes! Adds attributes with the current latency information (start,end,latency,max_start,max.latency) to each tuple.

SMALL

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

APPEND

Enrich Operators

ENRICH

This operator enriches tuples with data that is cached, e.g. to enrich a stream with a list of categories. The first input stream, therefore, should be only stream limited data to avoid buffer overflows. The second input is the data stream that

should be enriched.

MINIMUMSIZE

Blocks all until there are at least minimumSize elements in the cache

PREDICATE

Predicate to filter combinations

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

Order Operators

ASSUREORDER *(Deprecated)*

Deprecatd. Use ReOrder.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

REORDER

Operator which ensures the order of tuples based on punctuations. Requires heartbeats.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

Pattern Operators

CHANGECORRELATE

Operator used in DEBS Grand Challenge 2012

RIGHTLOWPREDICATE

LEFTLOWPREDICATE

LEFTHIGHPREDICATE

RIGHTHIGHPREDICATE

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

CHANGEDETECT

This operator can reduce traffic. It lets an event pass if its different than the last event, if specified, numeric values can have a tolerance band (relative or absolute defined) e.i. only if the new values lies outside this band, it is send (aka known as

deadband or histerese band)

GROUP_BY

SUPPRESSCOUNTATTRIBUTE

USEBASEVALUE If this is set to true, the actual value is compared to the base value instead to the last value. Default is false. Does not work with 'useWindow'.

TOLERANCE

DEBUG Flag, that this operator should be debuged.

SENDLASTOFSAMEOBJECTS If set to false (default), in a group of same objects, the first is send. If set to true, the last one is send.

HEARTBEATRATE

ATTR

BASEVALUE

If 'useBaseValue' is true, the actual value is compared to the base value instead to the last value.

DELIVERFIRSTELEMEN

USEWINDOW

If this is set to true, the operator compares not to the last value (or base value), but instead to the elements in the window. Therefore the difference to the minimum and maximum value to the new value is calculated. Default is false.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

RELATIVETOLERANCE

PATTERN

This generic operator allows the definition of different kinds of pattern (e.g. all, any). For sequence based patterns see SASE

operator

OUTPUTMODE

ATTRIBUTE

ASSERTIONS

SIZE

INPUTPORT

TIME

DEBUG Flag, that this operator should be debuged.

RETURN

TIMEUNIT

COUNT

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

TYPE

EVENTTYPES

SASE

This operator can parse a query in SASE+ syntax.

SCHEMA	
QUERY	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
ONEMATCHPERINSTANCE	
DEBUG	Flag, that this operator should be debuged.
TYPE	
HEARTBEATRATE	

Plan Operators

PLANMODIFICATIONACTION

Executes plan modifications based on receiving tuple data

COMMANDEXPRESSION	Expression for the plan modification commands, e.g. an attribute or a string
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
QUERYIDEXPRESSION	Expression to calculate the query id to execute the commands on
DEBUG	Flag, that this operator should be debuged.

Processing Operators

ADWIN (*Deprecated*)

Change detection window operator.

ATTRIBUTE	
DELTA	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

ASSUREORDER (*Deprecated*)

Deprecatd. Use ReOrder.

SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

ASSUREHEARTBEAT (*Deprecated*)

Deprecated. Use Heartbeat instead!

ALLOWOUTOFORDERCREATION	Deprecated. Value will be ignored!
APPLICATIONTIMEDELAY	
RESTARTTIMERFOREVERYINPUT	
STARTATCURRENTTIME	
SENDALWAYSHEARTBEAT	
REALTIMEDELAY	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.
STARTTIMERAFTERFIRSTELEMENT	

BLOOMFILTER

Filter incoming streams using a Bloom filter

ATTRIBUTES	
FPP	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

INSERTIONS

BUFFER

Typically, Odysseus provides a buffer placement strategy to place buffers in the query plan. This operator allows adding buffers by hand. Buffers receives data stream elements and stores them in an internal elementbuffer. The scheduler stops the execution here for now. Later, the scheduler resumes to execution (e.g. with an another thread).

THREADED	If set to true, this buffer will not be scheduled by the scheduler, but uses an own thread. Handle with care!
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DRAINATCLOSE	If set to true (default is false), this buffer be emptied when calling close. Remark: Could lead to longer termination time!
MAXBUFFERSIZE	
DEBUG	Flag, that this operator should be debuged.
TYPE	

CACHE

This operator can can some stream elements. At runtime, every time a new operator is connected it will get the cached elements. This can be usefull when reading from a csv file and

multiple parts of a query need this information.

MAXELEMENTS	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

COMBINE

Takes values of attributes from the input operators and combines them in one tuple

SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
BUFFERNEWINPUTELEMENTS	If WaitForAllChanged is set, specifies, if new Input should be buffered or overrides older Input that hast not been transfered yet
DEBUG	Flag, that this operator should be debuged.
WAITFORALLCHANGED	If true, there is only output when there has been input on all ports

CREATENEWFILENAMEPUNCTUATION

Depending on a predicate and a name: Create

NewFilenamePunctuations	
PREDICATE	If expression evaluates to true, a New-FileNamePunctuation is created from the filename attribute value
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
FILENAME	The expression to create the output filename.
DEBUG	Flag, that this operator should be debuged.

CREATEUPDATEEXPRESSIONSPUNCTUATION

Creates a punctuation with which the expressions can be updated if the receiving operator support it.

SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
EXPRESSIONTEMPLATES	A list of expressionTemplates. A template can include an attribute name in <>-brackets (<attribute>) which is replaced by the value of the attribute.
DEBUG	Flag, that this operator should be debuged.
TARGETOPERATORNAMES	A list of operators for which these punctuations are for.

CREATEUPDATEPREDICATEPUNCTUATIONREPLICATIONMERGE

Creates a punctuation with which a predicate can be updated if the receiving operator support it.

PREDICATETEMPLATE	The new predicate.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.
TARGETOPERATORNAMES	A list of operators for which these punctuations are for.

HEARTBEAT

This operator assures that every n time elements there will be a heartbeat on the guarantees, that no element (heartbeat or streamobject) is send, that is older than the last send heartbeat (i.e. the generated heartbeats are in order and indicate time progress). Heartbeats can be send periodically (sendAlwaysHeartbeats = true) or only if no other stream elements indicate time progress (e.g. in out of order scenarios) independent if a new element has been received or not.

ALLOWOUTOFORDERCREATION	Deprecated. Value will be ignored!
APPLICATIONTIMEDELAY	
RESTARTTIMERFOREVERYINPUT	
STARTATCURRENTTIME	
SENDALWAYSHEARTBEAT	
REALTIMEDELAY	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.
STARTTIMERAFTERFIRSTELEMENT	

METADATA

Change the current meta data

SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
METAATTRIBUTE	This overwrites the current set meta data. Existing values will not be overwritten.
DEBUG	Flag, that this operator should be debuged.

REORDER

Operator which ensures the order of tuples based on punctuations. Requires heartbeats.

SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

Merge input from semantically equal queries.

SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

ROUTE

This operator can be used to route the elements in the stream to different further processing operators, depending on the predicate.

SENDINGHEARTBEATS	If an element is routed to an output, heartbeats will be send to all other outputs
PREDICATES	
OVERLAPPINGPREDICATES	Evaluate all (true) or only until first true predicate (false), i.e. deliver to all ports where predicate is true or only to first
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

REPLACEMENT

This operator can be used if a value is expected but was not delivered timely. Different methods to determine the missing value are available.

INTERVAL	Size of the intervals
REPLACEMENTMETHOD	The replacement method for missing value.
VALUEATTRIBUTE	The attribute with the value attribute.
TIMESTAMPATTRIBUTE	The attribute with the timestamp attribute that should be updated.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.
QUALITYATTRIBUTE	The attribute with the quality attribute that should be updated.

SAMPLE

This operator can reduce load by throwing away tuples.

SAMPLERATE	
TIMEVALUE	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

SETTIMEPROGRESSMARKER

This operator updates the time order marker flag for each tuple. It can be used to state that an input stream should not

be used to determine time progress.

SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
VALUE	
DEBUG	Flag, that this operator should be debuged.

SYNCWITHSYSTEMTIME

This operator tries to delay elements so that they are not faster than realtime.

APPLICATIONTIMEFACTOR	Factor to calculate milliseconds from application time
APPLICATIONTIMEUNIT	Unit of application timestamps
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

TIMESHIFT

Shifts the timestamp(s) a given time

SHIFT	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

TIMESTAMPORDERVALIDATE

Assure that all elements are ordered by start timestamp and eliminate out of order elements.

DEBUGMODE	Set output mode: 0 = minimal, 1 = medium, 2 = maximum
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

WATERMARK

Sends a watermark (heartbeat) with a certain delay. The watermark then lags behind a certain timespan.

TIMESPANVALUE	How long the watermark lags behind the data stream timestamps.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

Probabilistic Operators

DISTRIBUTION *(Deprecated)*

Assign a distribution to the given attributes

ATTRIBUTES The attributes holding the expected value.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

VARIANCE The attribute holding the variance of the distribution.

CONTINUOUS The distribution is continuous or discrete.

DEBUG Flag, that this operator should be debuged.

EM *(Deprecated)*

Estimate the distribution of the given attributes using a Gaussian mixture model

MIXTURES The number of mixture components.

ATTRIBUTES The attributes to fit a distribution to

THRESHOLD The threshold for the loglikelihood to terminate the fitting process (default: 10E-5).

ITERATIONS The number of iterations (default: 1000).

PREDICATE The predicate to run a new fitting process.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

INCREMENTAL Reuse the existing model in each fitting process.

EXISTENCETOPAYLOAD *(Deprecated)*

The input object gets one new field with tuple existence.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

KALMAN *(Deprecated)*

Kalman filter operator

ATTRIBUTES

PROCESSNOISE

DEBUG Flag, that this operator should be debuged.

INITIALERROR

TRANSITION

MEASUREMENTNOISE

VARIABLES

MEASUREMENT

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

INITIALSTATE

CONTROL

KDE *(Deprecated)*

Estimate the distribution of the given attributes using a Gaussian mixture model

ATTRIBUTES The attributes to fit a distribution to

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

PROBABILISTIC *(Deprecated)*

This Operator can be used to update the existence uncertainty information in the meta data part.

ATTRIBUTE The name of the attribute for the existence uncertainty.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

SAMPLEFROM *(Deprecated)*

Create samples from a given distribution

ATTRIBUTES The distribution to sample from.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

SAMPLES The number of samples to create.

DEBUG Flag, that this operator should be debuged.

Set Operators

DIFFERENCE

This operator calculates the difference between two input sets.

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

EXISTENCE

This operator tests an existence predicate and can be used with the type EXISTS (semi join) and NOT_EXISTS (anti semi join). The predicates can be evaluated against the element from the first input and the second input. Semi join: All elements in the first input for which there are elements in the second input that fulfills the predicate are sent. Semi anti join: All elements in the first input for which there is no element in the second input that fulfills the predicate are sent.

PREDICATE

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DEBUG Flag, that this operator should be debuged.

TYPE

SYNCHRONIZE

Synchronizes different input streams

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DRAINATCLOSE If set to true (default is false), this buffer be emptied when calling close. Remark: Could lead to longer termination time!

DEBUG Flag, that this operator should be debuged.

UNION

Merges different input streams. (Typically preserves input order. Depending on the processing model)

SUPPRESSPUNCTUATIONS If set to true, no punctuations will be delivered from this operator. Default is false

DRAINATCLOSE If set to true (default is false), this buffer be emptied when calling close. Remark: Could lead to longer termination time!

DEBUG Flag, that this operator should be debuged.

Sink Operators

CSVFILESINK

Allows to write tp a csv based file

CSV.NUMBERFORMATTER	Formatter for integer numbers.
TEXTDELIMITER	Delimiter for Strings. No default.
NULLVALUETEXT	Text to output for 'null'. Default is empty string.
OPTIONS	Additional options.
DEBUG	Flag, that this operator should be debuged.
WRITEMETADATA	Write metadata.
DELIMITER	Default delimiter is ','
SINK	The name of the sink.
CSV.WRITEMETADATA	Write metadata.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
CSV.FLOATINGFORMATTER	Formatter for floating numbers.
FILENAME	

CONSOLESINK

Print input to standard out.

PRINTPORT	Set to true, if input port should be printed. Default is false
DUMPPUNCTUATION	Set to true, if punctuations should be printed. Default is false
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

FILESINK *(Deprecated)*

The operator can be used to dump the results of an operator to a file.

CACHESIZE	
FLOATINGFORMATTER	
FILETYPE	
DEBUG	Flag, that this operator should be debuged.
APPEND	
DUMPMETADATA	
NUMBERFORMATTER	
LINENUMBERS	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
FILENAME	

GROUPSPLITFILEWRITER

GroupSplitFileWriter	
PATH	Outputfolder
GROUPATTRIBUTES	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DATAHANDLER	The name of the datahandler to use, e.g. Tuple or Document.
DEBUG	Flag, that this operator should be debuged.

MEMSTOREWRITE

This operator writes all elements to a given store. If the store does not exists, it will be created.

CLEARSTORE	The store is cleaned every time the query is started new. If set to false, the elements will be appended without cleaning the store.
STORE	The name of the memory store to write to
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

SENDER

This operator can be used to publish processing results to multiple endpoints using different transport and application protocols.

PROTOCOL	
OPTIONS	Additional options for different handler.
DEBUG	Flag, that this operator should be debuged.
WRITEMETADATA	Write metadata.
TRANSPORT	
SINK	The name of the sink.
CSV.WRITEMETADATA	Write metadata.
WRAPPER	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false

DATAHANDLER

SINK

Represents a view for s sink.

SINK	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

Source Operators

ACCESS

Generic operator to connect to an input.

DATEFORMAT	The date format used.
SCHEMA2	The output schema for port 2.
SCHEMA1	The output schema for port 1.
OVERWRITESCHEMASOURCENAME	Output schema typically contains source name in attributes. Sometime this is not wanted. Set to false to avoid overwriting.
SCHEMA3	The output schema for port 3.
REALTIMEDELAY	For out of order. How long should be waited for new elements.
PROTOCOL	The name of the protocol handler to use, e.g. Csv or SizeByte-Buffer.
OPTIONS	Additional options.
METAATTRIBUTE	If set, this value overwrites the meta data created from this source.
DEBUG	Flag, that this operator should be debuged.
OUTOFORDER	The system needs to know if the input is ordered by timestamps. Set to true if this is not the case!
TRANSPORT	The name of the transport handler to use, e.g. File or TcpServer.
MAXTIMETOWAITFORNEWEVENTMS	For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
READMETADATA	If the source provides meta data, use this flag to enable reading of meta data.
SCHEMA	The output schema.
APPLICATIONTIMEDELAY	For out of order. After waiting some realTimeDelay, what time should be added to application time.
WRAPPER	The name of the wrapper to use, e.g. GenericPush or GenericPull.
INPUTSCHEMA	A list of data types describing the input format. Must be compatible with output schema!
SOURCE	The name of the sourcetype to create.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DATAHANDLER	The name of the datahandler to use, e.g. Tuple or Document.

CSVFILESOURCE

Allows to read input from a csv based file

TRIM	If set to true, for each element leading and trailing whitespaces are removed. Default false.
METAATTRIBUTE	If set, this value overwrites the meta data created from this source.
OUTOFORDER	The system needs to know if the input is ordered by timestamps. Set to true if this is not the case! Default delimiter is ','
DELIMITER	Default delimiter is ','
READFIRSTLINE	If fist line contains header information, set to false. Default true.
MAXTIMETOWAITFORNEWEVENTMS	For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
SCHEMA	The output schema.
APPLICATIONTIMEDELAY	For out of order. After waiting some realTimeDelay, what time should be added to application time.
FILENAME	
DATEFORMAT	The date format used.
SCHEMA2	The output schema for port 2.
SCHEMA1	The output schema for port 1.
OVERWRITESCHEMASOURCENAME	Output schema typically contains source name in attributes. Sometime this is not wanted. Set to false to avoid overwriting.
SCHEMA3	The output schema for port 3.
TEXTDELIMITER	Delimiter for Strings. No default.
REALTIMEDELAY	For out of order. How long should be waited for new elements.
OPTIONS	Additional options.
DEBUG	Flag, that this operator should be debuged.
READMETADATA	If the source provides meta data, use this flag to enable reading of meta data.
INPUTSCHEMA	A list of data types describing the input format. Must be compatible with output schema!
SOURCE	The name of the sourcetype to create.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false

MEMSTORESOURCE

This operator provides all elements of the given memory store as stream.

SCHEMA	The output schema.
STORE	The name of the memory store to read from.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.
METAATTRIBUTE	If set, this value overwrites the meta data created from this source.

QUERYSOURCE

Attach a named query as source

OPERATOR	The name of the query that should deliver data or a tuple with queryname and operatorname
PORT	The number of the output port of the operator in the query that should be connect to.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

RECEIVE

Generic operator to connect to an input that sends data (i.e. pushed from source).

DATEFORMAT	The date format used.
SCHEMA2	The output schema for port 2.
SCHEMA1	The output schema for port 1.
OVERWRITESCHEMASOURCENAME	Output schema typically contains source name in attributes. Sometime this is not wanted. Set to false to avoid overwriting.
SCHEMA3	The output schema for port 3.
REALTIMEDELAY	For out of order. How long should be waited for new elements.
PROTOCOL	The name of the protocol handler to use, e.g. Csv or SizeByte-Buffer.
OPTIONS	Additional options.
METAATTRIBUTE	If set, this value overwrites the meta data created from this source.
DEBUG	Flag, that this operator should be debuged.
OUTOFORDER	The system needs to know if the input is ordered by timestamps. Set to true if this is not the case!
TRANSPORT	The name of the transport handler to use, e.g. File or TcpServer.
MAXTIMETOWAITFORNEWEVENTMS	For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
READMETADATA	If the source provides meta data, use this flag to enable reading of meta data.
SCHEMA	The output schema.
APPLICATIONTIMEDELAY	For out of order. After waiting some realTimeDelay, what time should be added to application time.
INPUTSCHEMA	A list of data types describing the input format. Must be compatible with output schema!
SOURCE	The name of the sourcetype to create.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DATAHANDLER	The name of the datahandler to use, e.g. Tuple or Document.

RETRIEVE

Generic operator to connect to an input which input must be retrieved (i.e. pulled from source).

DATEFORMAT	The date format used.
SCHEMA2	The output schema for port 2.
SCHEMA1	The output schema for port 1.
OVERWRITESCHEMASOURCENAME	Output schema typically contains source name in attributes. Sometime this is not wanted. Set to false to avoid overwriting.
SCHEMA3	The output schema for port 3.
REALTIMEDELAY	For out of order. How long should be waited for new elements.
PROTOCOL	The name of the protocol handler to use, e.g. Csv or SizeByte-Buffer.
OPTIONS	Additional options.
METAATTRIBUTE	If set, this value overwrites the meta data created from this source.
DEBUG	Flag, that this operator should be debuged.
OUTOFORDER	The system needs to know if the input is ordered by timestamps. Set to true if this is not the case!
TRANSPORT	The name of the transport handler to use, e.g. File or TcpServer.
MAXTIMETOWAITFORNEWEVENTMS	For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
READMETADATA	If the source provides meta data, use this flag to enable reading of meta data.
SCHEMA	The output schema.
APPLICATIONTIMEDELAY	For out of order. After waiting some realTimeDelay, what time should be added to application time.
INPUTSCHEMA	A list of data types describing the input format. Must be compatible with output schema!
SOURCE	The name of the sourcetype to create.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DATAHANDLER	The name of the datahandler to use, e.g. Tuple or Document.

STREAM

Integrate a view.	
SOURCENAME	
SCHEMA	The output schema.
NODE	
SOURCE	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DATAHANDLER	The name of the datahandler to use, e.g. Tuple or Document.
DEBUG	Flag, that this operator should be debuged.

TIMER

A trigger with time events	
DATEFORMAT	The date format used.
SCHEMA2	The output schema for port 2.
SCHEMA1	The output schema for port 1.
OVERWRITESCHEMASOURCENAME	Output schema typically contains source name in attributes. Sometime this is not wanted. Set to false to avoid overwriting.
SCHEMA3	The output schema for port 3.
REALTIMEDELAY	For out of order. How long should be waited for new elements.
OPTIONS	Additional options.
METAATTRIBUTE	If set, this value overwrites the meta data created from this source.
DEBUG	Flag, that this operator should be debuged.
OUTOFORDER	The system needs to know if the input is ordered by timestamps. Set to true if this is not the case!
MAXTIMETOWAITFORNEWEVENTMS	For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
READMETADATA	If the source provides meta data, use this flag to enable reading of meta data.
SCHEMA	The output schema.
APPLICATIONTIMEDELAY	For out of order. After waiting some realTimeDelay, what time should be added to application time.
INPUTSCHEMA	A list of data types describing the input format. Must be compatible with output schema!
PERIOD	The timer period in ms
TIMEFROMSTART	Start from 0. If set to false, start from Jan 1th 1970.
SOURCE	The name of the sourcetype to create.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false

Test Operators

COMPARE

Compares to input streams	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

Transform Operators

CONVERTER

This operator can be used to transform element with other protocol handler, e.g. read a complete document from a server and then parse this document with csv or xml

DATEFORMAT	Format used if schema contains (Start End)TimestampString
PROTOCOL	Protocol handler to use.
OPTIONS	Additional options. See help doc for further information
DEBUG	Flag, that this operator should be debuged.
OUTPUTDATAHANDLER	Datahandler to use for creation of elements.
SCHEMA	The output schema of this operator
SOURCE	Overwrite source name
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
INPUTDATAHANDLER	Datahandler to use as input (e.g. format deliefered from preceeding operator)
UPDATEMETA	If set to false, existing meta data will not be touched.

KVUNNEST

Creates from one key value object a set of key value objects	
ATTRIBUTE	The input attribute that should be unnested. Must be a multi value attribute!
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

TIMESTAMPTOPAYLOAD (Deprecated)

Depracted: Use Map and TimeInterval.Start and TimeInterval.End directly. This operator is needed before data is send to another system (e.g. via a socket sink) to keep the time meta information (i.e. start and end time stamp). The input object gets two new fields with start and end timestamp. If this output is read again by (another) Odysseus instance, the following needs to be attached to the schema: ['start', 'StartTimestamp'], ['end', 'EndTimestamp']

ATTRIBUTES	Names of the attributes for the start and endtimestamp (default meta_valid_start and meta_valid_end.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

TOKEYVALUE

Converts an input object a key-value/JSON object

TEMPLATE	Template for the JSON object. Variables have to be in <brackets> and their names have to match the tuples attribute names.
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

TOTUPLE

Translates objects to a tuple

DATEFORMAT	If using a string for date information, use this format to parse the date (in Java syntax).
SCHEMA	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

TYPE

UNNEST

The UnNest operator unpacks incoming tuple with a multi value attribute to create multiple tuples

RECALCULATE	
ATTRIBUTE	
SUPPRESSPUNCTUATIONS	If set to true, no punctuations will be delivered from this operator. Default is false
DEBUG	Flag, that this operator should be debuged.

Aggregates

AMEDIAN	MAX
AMEDIAN2	MEDIAN
AVG	MIN
COMPLETENESS	NEST
CORR	NTH
COUNT	RATE
COV	STDDEV
DISTINCTNEST	SUM
FIRST	VAR
LAST	

Functions

Array

elementAt(List, Number)	→ Object
elementAt(List, Number)	→ Object

Bit

adler(BitVector)	→ BitVector
crc(BitVector)	→ BitVector
subset(BitVector, Integer, Integer)	→ BitVector

toBinary(UnsignedInt16)	→ BitVector
toBinary(Floating Number)	→ BitVector
toBinary(Byte)	→ BitVector
toBinary(String)	→ BitVector
toLong(BitVector)	→ Long

Bool

xor(Boolean, Boolean)	→ Boolean
-----------------------	-----------

Command

addQuery(String, String)	→ Command
fullQuery(Object)	→ Command
partialQuery(Object)	→ Command
removeQuery(Object)	→ Command
resumeQuery(Object)	→ Command
setPeriod(Object, Number)	→ Command
startQuery(Object)	→ Command
stopQuery(Object)	→ Command
suspendQuery(Object)	→ Command
updateProtocolOption(Object, String, String)	→ Command
updateTimeWindow(Object, Discrete Number, Discrete Number)	→ Command
updateTransportOption(Object, String, String)	→ Command

Compare

strlike(String, String)	→ Boolean
-------------------------	-----------

Crypt

DSA(Number)	→ List_String
EC(Number)	→ List_String
MD2withRSASign(Simple Type, String)	→ String
MD2withRSAVerify(Simple Type, String, String)	→ Boolean
MD5(String)	→ String
MD5withRSASign(Simple Type, String)	→ String
MD5withRSAVerify(Simple Type, String, String)	→ Boolean
NONEwithDSASign(Simple Type, String)	→ String
NONEwithDSAVerify(Simple Type, String, String)	→ Boolean
NONEwithECDSASign(Simple Type, String)	→ String
NONEwithECDSAVerify(Simple Type, String, String)	→ Boolean
NONEwithRSASign(Simple Type, String)	→ String
NONEwithRSAVerify(Simple Type, String, String)	→ Boolean
RSA(Number)	→ List_String
SHA1(String)	→ String
SHA1withDSASign(Simple Type, String)	→ String
SHA1withDSAVerify(Simple Type, String, String)	→ Boolean
SHA1withECDSASign(Simple Type, String)	→ String
SHA1withECDSAVerify(Simple Type, String, String)	→ Boolean
SHA1withRSASign(Simple Type, String)	→ String

SHA1withRSAVerify(*Simple Type, String, String*) → Boolean
 SHA244(*String*) → String
 SHA256(*String*) → String
 SHA256withECDSASign(*Simple Type, String*) → String
 SHA256withECDSAVerify(*Simple Type, String, String*) → Boolean
 SHA256withRSASign(*Simple Type, String*) → String
 SHA256withRSAVerify(*Simple Type, String, String*) → Boolean
 SHA384(*String*) → String
 SHA384withECDSASign(*Simple Type, String*) → String
 SHA384withECDSAVerify(*Simple Type, String, String*) → Boolean
 SHA384withRSASign(*Simple Type, String*) → String
 SHA384withRSAVerify(*Simple Type, String, String*) → Boolean
 SHA512(*String*) → String
 SHA512withECDSASign(*Simple Type, String*) → String
 SHA512withECDSAVerify(*Simple Type, String, String*) → Boolean
 SHA512withRSASign(*Simple Type, String*) → String
 SHA512withRSAVerify(*Simple Type, String, String*) → Boolean

Dstring

strcontains(*DString, String*) → Boolean
 indexof(*DString, String*) → Integer
 length(*DString*) → Integer
 lower(*DString*) → String
 regex(*DString, String*) → Boolean
 startsWith(*DString, String*) → Boolean
 strcontains(*DString, String*) → Boolean
 substring(*DString, Number, Number*) → String
 substring(*DString, Number*) → String
 upper(*DString*) → String

Function

difference(*IntervalByte, IntervalByte*) → IntervalDouble
 DolToEur(*Number*) → Double
 filterQueryIDs(*List, String*) → List
 getSharedOpsCount(*Integer*) → Integer
 getSourceCount(*Integer*) → Integer
 intersection(*IntervalByte, IntervalByte*) → IntervalDouble
 IsACQuery(*Integer*) → Boolean
 kvread(*String, String*) → Object
 kvremove(*String, String*) → Object
 kvwrite(*String, String, Simple Type*) → Object
 MaxSheddingFactor(*Integer*) → Integer
 QueryBasePriority(*Integer*) → Long
 QueryLastStateChangeTS(*Integer*) → Long
 QueryName(*Integer*) → String
 QueryPriority(*Integer*) → Long
 QuerySheddingFactor(*Integer*) → Integer
 QueryStartTS(*Integer*) → Long
 QueryState(*Integer*) → String

retrieveQueryIDs(*String*) → List
 toInterval(*Number, Number*) → IntervalDouble
 union(*IntervalByte, IntervalByte*) → IntervalDouble

Functions

burn(*Double*) → Double
 busyWait(*Double*) → Double
 counter() → Long
 counter(*Byte, Byte*) → Long
 counter(*Boolean*) → Long
 eif(*Boolean, Object, Object*) → Object
 eval(*String*) → Object
 hash(*Simple Type*) → Integer
 isNull(*Object*) → Boolean
 isNaN(*Number*) → Boolean
 isNull(*Object*) → Boolean
 load() → Double
 MDAAddDim(*String, List_Double*) → Object
 MDAAddDim(*String, Integer, List_Double*) → Object
 MDADim(*Double, Double, Integer*) → List_Double
 MDADrop(*String*) → Object
 MDAExchangeDim(*String, Integer, List_Double*) → Object
 MDAIndex(*String, Double*) → Integer
 MDAIndices(*String, List_Double*) → List_Integer
 MDAInit(*String, List*) → Object
 MDARemoveDim(*String, Integer*) → Object
 mem() → Long
 random(*Byte, Integer*) → Integer
 read(*String*) → String
 rnd() → Double
 sleep(*Double*) → Double
 SMAX(*Object, Double*) → Double (*Deprecated*)
 SMIN(*Object, Double*) → Double (*Deprecated*)
 Split(*String, String, Long*) → List_String (*Deprecated*)
 Split(*String, String*) → List_String
 storedLine(*String, Matrix, Matrix*) → Matrix
 storedValue(*String, Matrix, Matrix*) → Double
 uptime() → Long
 uuid() → String

Hex

toHex(*Discrete Number*) → HexString
 toHex(*Double*) → HexString
 toHex(*String*) → HexString

Kvstore

kvread(*String*) → Object
 kvremove(*String*) → Object
 kvwrite(*String, Simple Type*) → Boolean

List

All(*List, String*) → List
 Any(*List, String*) → List
 asList(*Object*) → List
 avg(*List*) → Double
 contains(*String, List*) → Boolean
 contains(*Number, List*) → Boolean

fill(*List, Discrete Number, Object*) → List
 filter(*List, String*) → List
 first(*List*) → Object
 foreach(*List, String*) → List
 foreachpair(*List, String*) → List
 IndexOf(*List, Simple Type*) → Integer
 IsEmpty(*List*) → Boolean
 last(*List*) → Object
 ListProject(*List, String*) → List
 ListTupleProject(*List, String*) → List
 max(*List_Tuple, Discrete Number*) → Tuple
 max_(*List*) → Object
 min(*List_Tuple, Discrete Number*) → Tuple
 min_(*List*) → Object
 removeDuplicates(*List*) → List
 rest(*List*) → List
 rnd(*List*) → Object
 search(*List, Simple Type, Boolean*) → Object
 size(*List*) → Integer
 split(*String, String*) → List
 split(*String, String, String*) → List
 sublist(*List, Discrete Number, Discrete Number*) → List
 sublist(*List, Discrete Number*) → List
 sum(*List*) → Double
 toList(*Object, Object, Object, Object, Object*) → List
 toList(*Object, Object, Object, Object*) → List
 toList(*Object, Object, Object, Object, Object, Object, Object, Object, Object*) → List
 toList(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → List
 toList(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → List
 toList(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → List
 toList(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → List
 toList(*Object, Object*) → List
 toList(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → List
 toList(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → List
 toList(*Object*) → List

Math

abs(*Number*) → Double
 acos(*Number*) → Double
 as2DVector(*ProbabilisticDouble, ProbabilisticDouble*) → VectorProbabilisticDouble
 as3DVector(*ProbabilisticDouble, ProbabilisticDouble, ProbabilisticDouble*) → VectorProbabilisticDouble
 asin(*Number*) → Double
 atan(*Number*) → Double
 atan2(*Number | Object, Number | Object*) → Double
 binomialTest(*Integer, Integer, Double, String, Double*) → Timestamp
 ceil(*Number*) → Double
 cos(*Number*) → Double
 cosh(*Number*) → Double
 distance(*ProbabilisticDouble, Number*) → Double

distance(*VectorProbabilisticDouble*, *MatrixBoolean*) → Double
distance(*Matrix*, *Matrix*) → Double
distance(*Number*, *Number*) → Double
e() → Double
exp(*Number*) → Double
floor(*Number*) → Double
inf() → Double
int(*ProbabilisticDouble*, *Number*, *Number*) → Double
kl(*VectorProbabilisticDouble*,
VectorProbabilisticDouble) → Double
kl(*ProbabilisticDouble*, *ProbabilisticDouble*) → Double
log(*Number*) → Double
log10(*Number*) → Double
loglikelihood(*Vector*, *ProbabilisticDouble*) → Double
MAX(*Number* | *Object*, *Number* | *Object*) → Double
max(*Number* | *Object*, *Number* | *Object*) → Double
min(*Number* | *Object*, *Number* | *Object*) → Double
nan() → Double
pi() → Double
round(*Number*, *Integer*) → Double
sign(*Number*) → Double
similarity(*ProbabilisticDouble*, *ProbabilisticDouble*) → Double
similarity(*VectorProbabilisticDouble*, *MatrixBoolean*) → Double
sin(*Number*) → Double
sinh(*Number*) → Double
sqrt(*Number*) → Double
tan(*Number*) → Double
tanh(*Number*) → Double
ToDegrees(*Number*) → Double
ToRadians(*Number*) → Double
UnaryMinus(*Discrete Number*) → Long
UnaryMinus(*Floating Number*) → Double

Matrix

AVG(*Matrix*) → Double
AVG(*Vector*) → Double
Count(*Matrix*) → Integer
Count(*Vector*) → Integer
det(*Matrix*) → Double
dotProduct(*Vector*, *Vector*) → Double
dotProduct(*Matrix*, *Matrix*) → Double
eig(*Matrix*) → Vector
get(*Matrix*, *Number*, *Number*) → Double
get(*Vector*, *Number*) → Double
identity(*Number*) → Matrix
ieig(*Matrix*) → Vector
inv(*Matrix*) → Matrix
Max(*Vector*) → Double
Max(*Matrix*) → Double
Median(*Vector*) → Double
Median(*Matrix*) → Double
Min(*Vector*) → Double
Min(*Matrix*) → Double
ones(*Number*, *Number*) → Matrix

perm(*Matrix*) → Double
perms(*Vector*) → Matrix
readMatrix(*String*) → Matrix
readVector(*String*, *Number*) → Vector
readVector(*String*) → Vector
StdDev(*Vector*) → Double
StdDev(*Matrix*) → Double
sub(*Matrix*, *Number*, *Number*, *Number*, *Number*) → Matrix
sub(*Vector*, *Number*, *Number*) → Vector
Sum(*Vector*) → Double
Sum(*Matrix*) → Double
svd(*Matrix*) → Vector
toMatrix(*Vector*) → Matrix
toString(*Vector*) → String
toString(*Matrix*) → String
toVector(*Matrix*) → Vector
tr(*Matrix*) → Double
trans(*Matrix*) → Matrix
Var(*Vector*) → Double
Var(*Matrix*) → Double
vectorFromString(*String*, *String*, *Discrete Number*,
Discrete Number) → Vector
vectorFromString(*String*, *String*) → Vector
zeros(*Number*, *Number*) → Matrix

Mep

assureNumber(*Number*) → Double
get(*KeyValueObject*, *String*) → Object
getElement(*KeyValueObject*, *String*) → Object
getElements(*KeyValueObject*, *String*) → Object
path(*KeyValueObject*, *String*) → List_KeyValueObject
toKeyValue(*String*) → KeyValueObject

String

concat(*Object*, *Object*) → String
strcontains(*String*, *String*) → Boolean
indexof(*String*, *String*) → Integer
length(*String*) → Integer
lower(*String*) → String
regex(*String*, *String*) → Boolean
replace(*String*, *String*, *String*) → String
replaceAll(*String*, *List.String*, *List.String*) → String
replaceAll(*String*, *String*, *String*) → String
replaceFirst(*String*, *String*, *String*) → String
startsWith(*String*, *String*) → Boolean
strcontains(*String*, *String*) → Boolean
substring(*String*, *Number*, *Number*) → String
substring(*String*, *Number*) → String
upper(*String*) → String

Time

businessDays(*Date*, *Date*) → Integer
curdate() → Date
dateInMillis(*Date*) → Long
day(*Date*) → Integer
day(*String*, *String*) → Integer
dayofmonth(*String*, *String*) → Integer

dayofmonth(*Date*) → Integer
days(*Date*, *Date*) → Integer
format(*Date*, *String*) → String
hour(*String*, *String*) → Integer
hour(*Date*) → Integer
hours(*Date*, *Date*) → Integer
millisecond(*String*, *String*) → Long
millisecond(*Date*) → Long
milliseconds(*Date*, *Date*) → Long
milliTime() → Long
minute(*String*, *String*) → Integer
minute(*Date*) → Integer
minuteOfDay(*Date*) → Integer
minutes(*Date*, *Date*) → Integer
month(*String*, *String*) → Integer
month(*Date*) → Integer
months(*Date*, *Date*) → Integer
nanoTime() → Long
second(*Date*) → Integer
second(*String*, *String*) → Integer
seconds(*Date*, *Date*) → Integer
sysdate() → Date (*Deprecated*)
toDate(*String*, *String*) → Date
toDate(*Number*) → Date
toLong(*Date*) → Long
toString(*Date*, *String*, *String*) → String
toString(*Date*, *String*) → String
toTimestamp(*Discrete Number*) → Timestamp
week(*String*, *String*) → Integer
week(*Date*) → Integer
weekday(*String*, *String*) → Integer
weekday(*Date*) → Integer
year(*String*, *String*) → Integer
year(*Date*) → Integer
years(*Date*, *Date*) → Integer

Transform

doubleToBoolean(*Double*) → Boolean (*Deprecated*)
doubleToByte(*Double*) → Byte (*Deprecated*)
doubleToChar(*Double*) → Char (*Deprecated*)
doubleToFloat(*Double*) → Float (*Deprecated*)
doubleToInteger(*Double*) → Integer (*Deprecated*)
doubleToLong(*Double*) → Long (*Deprecated*)
doubleToShort(*Double*) → Short (*Deprecated*)
toString(*Object*) → String
toBoolean(*Number*) → Boolean
toBoolean(*String*) → Boolean
toByte(*String*) → Byte
toByte(*Number*) → Byte
toByte(*BitVector*) → Byte
toByte(*Boolean*) → Byte
toChar(*Number*) → Char
toChar(*Boolean*) → Char
toChar(*String*) → Char
toDouble(*String*) → Double
toDouble(*Boolean*) → Double
toDouble(*Number*) → Double

toFloat(*UnsignedInt16, UnsignedInt16, Boolean*) → Float
 toFloat(*UnsignedInt16, UnsignedInt16*) → Float
 toFloat(*Boolean*) → Float
 toFloat(*String*) → Float
 toFloat(*Number*) → Float
 toInteger(*String*) → Integer
 toInteger(*Boolean*) → Integer
 toInteger(*Number*) → Integer
 toInteger(*BitVector*) → Integer
 toLong(*Boolean*) → Long
 toLong(*Number*) → Long
 toLong(*String*) → Long
 toNumber(*Object*) → Double
 toProbabilisticContinuousDouble(*MatrixBoolean, MatrixBoolean*) → ProbabilisticDouble
 toProbabilisticDiscreteDouble(*MatrixBoolean, MatrixBoolean*) → ProbabilisticDouble
 toShort(*String*) → Short
 toShort(*Number*) → Short
 toShort(*Boolean*) → Short
 toString(*Object*) → String
 toUnsignedInt16(*String*) → UnsignedInt16
 toUnsignedInt16(*Boolean*) → UnsignedInt16
 toUnsignedInt16(*Number*) → UnsignedInt16

Tuple

asTuple(*Object*) → Tuple
 elementAt(*Tuple, Number*) → Object
 toTuple(*Object, Object, Object, Object, Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object, Object, Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → Tuple
 toTuple(*Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object*) → Tuple
 elementAt(*Tuple, Number*) → Object

Symbols

!(*ProbabilisticResult*) → ProbabilisticResult
 !(*Boolean*) → Boolean
 !=(*DString, DString*) → Boolean
 !=(*String, String*) → Boolean
 !=(*Number | Object, Number | Object*) → Boolean
 %(*Number | Object, Number | Object*) → Double
 &(*BitVector, BitVector*) → BitVector
 &(*Number | Object, Number | Object*) → Long
 &&(*Boolean, Boolean*) → Boolean
 &&(*ProbabilisticResult, ProbabilisticResult*) → ProbabilisticResult

*(*Number, Matrix*) → Matrix
 *(*Number, Vector*) → Vector
 *(*Matrix, Number*) → Matrix
 *(*String, String*) → String
 *(*IntervalByte, IntervalByte*) → IntervalDouble
 *(*ProbabilisticDouble, ProbabilisticDouble*) → ProbabilisticDouble
 *(*Number | Object, Number | Object*) → Double
 *(*ProbabilisticDouble, Number*) → ProbabilisticDouble
 *(*Vector, Vector*) → Matrix
 *(*Number, ProbabilisticDouble*) → ProbabilisticDouble
 *(*Vector, Number*) → Vector
 *(*Matrix, Matrix*) → Matrix
 +(*String, String*) → String
 +(*List, List*) → List
 +(*Vector, Vector*) → Vector
 +(*Number, ProbabilisticDouble*) → ProbabilisticDouble
 +(*Vector, Number*) → Vector
 +(*Date, Number*) → Date
 +(*ProbabilisticDouble, Number*) → ProbabilisticDouble
 +(*ProbabilisticDouble, ProbabilisticDouble*) → ProbabilisticDouble
 +(*Number | Object, Number | Object*) → Double
 +(*Matrix, Matrix*) → Matrix
 +(*Number, Matrix*) → Matrix
 +(*Number, Vector*) → Vector
 +(*Date, Date*) → Date
 +(*Matrix, Number*) → Matrix
 +(*DString, String*) → DString
 +(*IntervalByte, IntervalByte*) → IntervalDouble
 -(*ProbabilisticDouble, ProbabilisticDouble*) → ProbabilisticDouble
 -(*List, Simple Type*) → List
 -(*DString, String*) → String
 -(*Matrix, Number*) → Matrix
 -(*Vector, Number*) → Vector
 -(*Matrix, Matrix*) → Matrix
 -(*Date, Number*) → Date
 -(*IntervalByte, IntervalByte*) → IntervalDouble
 -(*Date, Date*) → Date
 -(*String, String*) → String
 -(*Number | Object, Number | Object*) → Double
 -(*Number, ProbabilisticDouble*) → ProbabilisticDouble
 -(*List, List*) → List
 -(*Vector, Vector*) → Vector
 -(*ProbabilisticDouble, Number*) → ProbabilisticDouble
 /(*Number | Object, Number | Object*) → Double
 /(*Vector, Number*) → Vector
 /(*Matrix, Number*) → Matrix
 /(*IntervalByte, IntervalByte*) → IntervalDouble
 /(*String, String*) → Integer
 /(*ProbabilisticDouble, ProbabilisticDouble*) → ProbabilisticDouble
 /(*Number, ProbabilisticDouble*) → ProbabilisticDouble
 /(*ProbabilisticDouble, Number*) → ProbabilisticDouble
 <(*Number | Object, Number | Object*) → Boolean
 <(*ProbabilisticDouble, Number*) → ProbabilisticResult

<(*VectorProbabilisticDouble, MatrixBoolean*) → ProbabilisticResult
 <<(*Number | Object, Number | Object*) → Long
 <=(*VectorProbabilisticDouble, MatrixBoolean*) → ProbabilisticResult
 <=(*Number | Object, Number | Object*) → Boolean
 <=(*ProbabilisticDouble, Number*) → ProbabilisticResult
 !=(*DString, DString*) → Boolean
 !=(*Number | Object, Number | Object*) → Boolean
 !=(*String, String*) → Boolean
 =(*Boolean, Boolean*) → Boolean
 =(*String, String*) → Boolean
 =(*Number | Object, Number | Object*) → Boolean
 =(*DString, DString*) → Boolean
 =(*String, String*) → Boolean
 =(*Number | Object, Number | Object*) → Boolean
 ==(*ProbabilisticDouble, Number*) → ProbabilisticResult
 ==(*Vector, Vector*) → Boolean
 ==(*Boolean, Boolean*) → Boolean
 ==(*DString, DString*) → Boolean
 ==(*VectorProbabilisticDouble, MatrixBoolean*) → ProbabilisticResult
 ==(*Matrix, Matrix*) → Boolean
 >(*ProbabilisticDouble, Number*) → ProbabilisticResult
 >(*Number | Object, Number | Object*) → Boolean
 >(*VectorProbabilisticDouble, MatrixBoolean*) → ProbabilisticResult
 >=(*VectorProbabilisticDouble, MatrixBoolean*) → ProbabilisticResult
 >=(*ProbabilisticDouble, Number*) → ProbabilisticResult
 >=(*Number | Object, Number | Object*) → Boolean
 >>(*Number | Object, Number | Object*) → Long
 [](*Matrix, Vector*) → Double
 [](*BitVector, Integer*) → Boolean
 [](*Vector, Number*) → Double
 [](*Matrix, Number*) → Vector
 ~(*IntervalByte, Byte*) → IntervalDouble
 ~(*Number | Object, Number | Object*) → Double
 ~(*Matrix, Number*) → Matrix
 |(*BitVector, BitVector*) → BitVector
 |(*Number | Object, Number | Object*) → Long
 |(*ProbabilisticResult, ProbabilisticResult*) → ProbabilisticResult
 |(*Boolean, Boolean*) → Boolean
 ~(*BitVector*) → BitVector
 ~(*Number*) → Long

Handlers

Data Handlers

AVGSUMPARTIALAGGREGATE	LIST_LIST
BITVECTOR	LIST_LONG
BOOLEAN	LIST_SHORT
BYTE	LIST_STRING
BYTEBUFFER	LIST_TUPLE
COUNTPARTIALAGGREGATE	LONG
DATE	MATRIX
DOCUMENT	MULTI_VALUE
DOUBLE	MV
DSTRING	NTUPLE
ENDTIMESTAMP	OBJECT
FLOAT	PROBABILISTICDOUBLE
INTEGER	PROBABILISTICtuple
INTERVALDOUBLE	RELATIONALELEMENTPARTIALAGGREGATE
KEYVALUEOBJECT	SHORT
LIST	STARTTIMESTAMP
LIST_BOOLEAN	STARTTIMESTAMPSTRING
LIST_BYTE	STRING
LIST_CHAR	TIMESTAMP
LIST_DATE	TUPLE
LIST_DOUBLE	UNSIGNEDINT16
LIST_FLOAT	VECTOR
LIST_INTEGER	

Protocol Handlers

BSON	ODYSSEUS
CSV	ODYSSEUSMARKER
DOCUMENT	SIMPLEBYTEBUFFER
JSON	SIMPLECSV
LINE	SIZEBYTEBUFFER
MARKERBYTEBUFFER	SVM
NONE	TEXT

Transport Handlers

DIRECTORY	TCPCLIENT1
FILE	TCPSERVER
NONBLOCKINGTCP	TCPSERVER1
PLANMODIFICATIONWATCHER	TCPSERVER2
SIMPLEUDPRECEIVE	TIMER
TCP	UDPCLIENT
TCPCLIENT	UDPSERVER

Odysseus Script

Commands

#INCLUDE	NO_METADATA
#INPUT	ODYSSEUS_PARAM
ACQUERY	OPTIMIZE_PREDICATES
ACTIVATEREWRIERULE	PARSER
ADDQUERY	PARTIALQUERY
BEGIN	PLANGENERATIONMETHOD
BUFFERPLACEMENT	PRETRANSFORM
CONFIG	PRINT
CREATE_KV_STORE	PROCEDURE
DEACTIVATEREWRIERULE	QNAME
DEFINE	QPARAM
DOADAPT	QPRIORITY
DODISTRIBUTE	QUERY
DOQUERYSHARING	RECOVERYCONFIGURATION
DOREWRITE	RELOADFROMLOG
DROPALLQUERIES	REMOVEQUERY
DROPALLSINKS	REQUIRED
DROPALLSOURCES	RESETUPDATESITE
DROPPROCEDURE	RESUMEONERROR
DROP_KV_STORE	RESUMEQUERY
ELSE	RUNCOMMAND
END	RUNQUERY
ENDIF	SCHEDULER
ENDLOOP	SLEEP
EVAL	STARTQUERIES
EXECUTE	STARTQUERY
IF	STARTSCHEDULER
IFDEF	STOPQUERIES
IFNDEF	STOPQUERY
IFSRCDEF	STOPSCHEDULER
IFSRCNDEF	SUSPENDQUERY
LOGIN	TRAFOOPTION
LOGOUT	TRANSCFG
LOOP	UNDEF
MAXSHEDDINGFACTOR	UPDATE
MDASTORE_DROP	UPDATESITE
MDASTORE_INIT	UPTO
METADATA	WAITFORQUERY

Constants

AWT.TOOLKIT
CHEATSHEET
ECLIPSE.APPLICATION
ECLIPSE.COMMANDS
ECLIPSE.HOME.LOCATION
ECLIPSE.P2.DATA.AREA
ECLIPSE.P2.PROFILE
ECLIPSE.PRODUCT
ECLIPSE.STARTTIME
ECLIPSE.STATESAVEDDELAYINTERVAL
EQUINOX.USE.DS
FILE.ENCODING
FILE.ENCODING.PKG
FILE.SEPARATOR

GOSH.ARGS
JAVA.AWT.GRAPHICSENV
JAVA.AWT.PRINTERJOB
JAVA.CLASS.PATH
JAVA.CLASS.VERSION
JAVA.ENDORSED.DIRS
JAVA.EXT.DIRS
JAVA.HOME
JAVA.IO.TMPDIR
JAVA.LIBRARY.PATH
JAVA.RUNTIME.NAME
JAVA.RUNTIME.VERSION
JAVA.SPECIFICATION.NAME
JAVA.SPECIFICATION.VENDOR
JAVA.SPECIFICATION.VERSION
JAVA.VENDOR
JAVA.VENDOR.URL
JAVA.VENDOR.URL.BUG
JAVA.VERSION
JAVA.VM.INFO
JAVA.VM.NAME
JAVA.VM.SPECIFICATION.NAME
JAVA.VM.SPECIFICATION.VENDOR
JAVA.VM.SPECIFICATION.VERSION
JAVA.VM.VENDOR
JAVA.VM.VERSION
LINE.SEPARATOR
LOG4J.CONFIGURATION
ORG.ECLIPSE.UPDATE.RECONCILE
ORG.OSGI.FRAMEWORK.EXECUTIONENVIRONMENT
ORG.OSGI.FRAMEWORK.LANGUAGE
ORG.OSGI.FRAMEWORK.OS.NAME
ORG.OSGI.FRAMEWORK.OS.VERSION
ORG.OSGI.FRAMEWORK.PROCESSOR
ORG.OSGI.FRAMEWORK.SYSTEM.CAPABILITIES
ORG.OSGI.FRAMEWORK.SYSTEM.PACKAGES
ORG.OSGI.FRAMEWORK.UUID
ORG.OSGI.FRAMEWORK.VENDOR
ORG.OSGI.FRAMEWORK.VERSION
ORG.OSGI.SUPPORTS.FRAMEWORK.EXTENSION
ORG.OSGI.SUPPORTS.FRAMEWORK.FRAGMENT
ORG.OSGI.SUPPORTS.FRAMEWORK.REQUIREBUNDLE
OS.ARCH
OS.NAME
OS.VERSION
OSGI.ARCH
OSGI.BUNDLES
OSGI.BUNDLES.DEFAULTSTARTLEVEL
OSGI.COMPATIBILITY.BOOTDELEGATION
OSGI.CONFIGURATION.AREA
OSGI.FRAMEWORK
OSGI.FRAMEWORK.EXTENSIONS
OSGI.FRAMEWORK.SHAPE
OSGI.FRAMEWORK.USESYSTEMPROPERTIES
OSGI.FRAMEWORKCLASSPATH
OSGI.INSTALL.AREA
OSGI.LOGFILE


```
OSGI.NL
OSGI.OS
OSGI.REQUIREDJAVAVERSION
OSGI.SYSPATH
OSGI.WS
PATH.SEPARATOR
SUN.ARCH.DATA.MODEL
SUN.BOOT.CLASS.PATH
SUN.BOOT.LIBRARY.PATH
SUN.CPU.ENDIAN
SUN.CPU.ISALIST
SUN.IO.UNICODE.ENCODING
SUN.JAVA.COMMAND
SUN.JAVA.LAUNCHER
```

```
SUN.JNU.ENCODING
SUN.MANAGEMENT.COMPILER
SUN.OS.PATCH.LEVEL
USER.COUNTRY
USER.DIR
USER.HOME
USER.LANGUAGE
USER.NAME
USER.TIMEZONE
```

Sample Odysseus query

```
#PARSER PQL
#ADDQUERY
```

```
input = ACCESS({source='source',
```

```
    wrapper='GenericPull',
    transport='File',
    protocol='CSV',
    dataHandler='Tuple',
    metaattribute=['TimeInterval'],
    options=[['filename','example.csv']],
    schema=[['value','Double']]
})
output = MAP({expressions = ['value + 3']}, input)
```

Copyright © 2017 ODYSSEUS Team
http://odysseus.informatik.uni-oldenburg.de
Wiki: <http://wiki.odysseus.informatik.uni-oldenburg.de>
Forum: <http://forum.odysseus.informatik.uni-oldenburg.de>