Odysseus Cheat Sheet

Full Grammar of PQL

QUERY = (STREAM | VIEW | SOURCE)+
STREAM = STREAM "=" OPERATOR
VIEW = VIEWNAME ";=" OPERATOR
SOURCE = SOURCENAME ";=" OPERATOR
OPERATOR = QUERY | [OUTPUTPORT ";:" OPERATOR OR OPERATOR | OPERATORLIST)
OPERATORLIST = [OPERATOR | OPERATOR OR OPERATOR | OPERATORLIST)
PARAMETERLIST = (PARAMETER | PARAMETER | PARAMETERLIST)
PARAMETERVALUE = LONG | DOUBLE | STRING | PREDICATE |
PARAMETERLIST = {PARAMETER | PARAMETER | PARAMETERLIST)
OPERATORLIST = [OPERATOR | OPERATOR OR OPERATOR | OPERATORLIST)
LIST = "[" [PARAMETERVALUE | PARAMETERVALUE | PARAMETERLIST)
MAP = "[" MAPENTRY | MAPENTRY | MAPENTRY)
MAPENTRY = PARAMETERVALUE "=" PARAMETERVALUE
STRING = "" [STRING | STRING | STRING)
PREDICATE = PREDICATETYPE "(" STRING ")"

Operators

ACCESS

Generic operator to connect to an input.
SCHEMA
INPUTSCHEMA
The output schema.
A list of data types describing the input format. Must be compatible with output schema!
The name of the transport handler to use, e.g. File or TcpServer.
The name of the sourcetype to create. For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end.
The name of the datahandler to use, e.g. Tuples or Document.
The name of the wrapper to use, e.g. GenericPush or GenericPull.
The name of the protocol handler to use, e.g. Csv or SizeByteBuffer.
Additional options.
The date format used.

ADWIN

Change detection window operator.
DELTA
ATTRIBUTE

AGGREGATE

Aggregations on attributes e.g Min, Max, Count, Avg, Sum and grouping.
AGGREGATIONS
GROUP BY
FASTGROUPING
Use hash code instead of tuple compare to create group. Potentially unsafe!
DRAINADATONE
If set to true (default), elements are not yet written will be written at done.
OUTPUTPA
DRAINATCLOSE
If set to true (default), 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.
DUMPVALUECOUNT
ASSOCIATIVESTORAGE
This operator stores streaming data in an associative storage
INDEX
HIERARCHY
VALUE
STORAGENAME
SIZES
ASSUREORDER
Operator which ensures the order of tuples
AUDIENCEENGAGEMENT
This operator can reduce traffic. It lets an event pass if its different than
ALLTOPICS
THRESHOLDVALUE
COUNTOFALLTOPICS
INCOMINGTEXT
CONCRETE TOPICS
APPENDTO
Attach a subplan to another operator with a specific id
APPENDTO
ASSUREHEARTBEAT
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 order of out of scenario) independent if a new element has been received or not.
SENDALWAYSHEARTBEAT
ALLOWOUTOFORDER
REALTIMEDELAY
STARTATCURRENTTIME
APPLICATIONTIMEDELAY
STARTTIMERAFTERFIRSTELEMENT
BUFFERR
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 another thread).
MAXBUFFERSIZE
THREADED
BUFFEDFILTER
This operator can be used to reduce data rate. It buffers incoming elements on port 0 (left) for deliverTime 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.
BUFFERTIME
DELIVERTIME
PREDICATE
CACHE
This operator can also stream elements. At runtime, every new operator is connected it will get the cached elements. This can be useful when reading from a csv file and multiple parts of a query need this information.
MAKEELEMENTS
CALCULATE
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.
CHANGECORRELATE
Operator used in DEBS Grand Challenge 2012
LEFTTOPHEDICATE
RIGHTTOPHEDICATE
RIGHTLOWPREDICATE
CHANGEDTECT
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)
TOLERANCE
GROUP BY
RELATIVER TOLERANCE
DELIVERFIRSTELEMENT
ATTR
HEARTBEATRATE
SUPPRESSCOUNTATTRIBUTE
evaluates to false, i.e., a new tuple is created when the predicates
the predicate case, the elements are merged as long as the predicates
aggregates values and the grouping attributes are created as a result. In
the grouping attributes (e.g., a sensorid) are the same. When a new
elements are merged with also given aggregations functions, as long as
the attributes case, the elements are not yet written will be written at close.
CREATIONHEARTBEAT
DRAIN
If set to true (default), elements are not yet written will be written at done.
AGGREGATIONS
MAKEELEMENTSGROUP –
ENDPREDICATE –
OUTPUTPDA –
STARTPREDICATE –
PREDICATE –
ATTR
HEARTBEATRATE –
DUMPATVALUECOUNT –
CONTEXTENRIC
This operator enriches tuples with information from the context store. Further Information can be found here. There is also an DBENRICH
operator for fetching data from a database or a simple ENRICH that
caches incoming streams.
OUTER –
ATTRIBUTES –
STORE –
CONVERSATIONREACH

Allows to calculate the Conversation Reach of a topic.

ALLOW –
THRESHOLDVALUE –
USERIDS –
INCOMINGTEXT –
CONCRETETOPIC –
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 csp or xml
SOURCE
OUTPUTDATANZLDER
DATanlader to use for creation of elements.
SCHEMA
The output schema of this operator
PROTOCOL
Protocol handler to use.
INPUTDATANZLDER
Datahandler to use as input (e.g., format delimetered from preceding operator)
DATEFORMAT
Format used if schema contains (Start)EndTimestampString
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!
FUNCTION –
GROUP_BY –
ATTRIBUTES –
SIZE –
OPTIONS –
CSVFILESINK

Allows to write tp a csv based file

CSV.FLOATINGFORMATTER
Formatter for floating numbers.
FILENAME –
TEXTDELMITER
Delimiter for Strings. No default.
SINK
The name of the sink.
CSV.NUMBERFORMATTER
Formatter for integer numbers.
OPTIONS
Additional options.
DELIMITER
Default delimiter is ;

CSVFILESOURCE

Allows to read input from a csv based file

SCHEMA
The output schema.
INPUTSCHEMA
A list of data types describing the input format. Must be compatible with output schema!
FILENAME –
TRIM
If set to true, for each element leading and trailing whitespaces are removed. Default false.
SOURCE
The name of the sourcetype to create. Check with help or the UI.
MAXTIMETOWAITFORNEWEVENTMS
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
TEXTDELMITER
Delimiter for Strings. No default.
READFIRSTLINE
If fist line contains header informa-
tion, set to false. Default true.
OPTIONS
Additional options.
DELIMITER
Default delimiter is ;

DATEFORMAT
The date format used.

COMMENTS

This operator is used to create a classifier. Therefore, the result is a
stream of classifiers (this is an own datatype)
CLASS –
CLASSIFIER –
NAME OF CLASSIFIER Use only one classifier at once
CLASSNAME
The name of the classification result

This operator classifies a tuple by using a classifier. The operator needs
two inputs: A stream of tuples that should be classified and a stream of
classifiers (that normally comes from a CLASSIFICATION_LEARN
operator). It appends a new attribute called “clazz” which contains the
nominal class value or continuous value from a regression. For the
classify operator, the type of the classifier (tree, list, bayes... )
doesn’t matter. You may even mixup them to classify the same tuple
with different classifiers (see Ensembles). The left port is the input for
t he tuples that should be classified and the right input is the one with
the classifiers.
CLASSIFIER –
ATTRIBUTES

This operator clusters a set of tuples.

ATTRIBUTES –
ALGORITHM –
LEARNER –
OPTIONS –

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 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 predicates
evaluates to false, i.e., a new tuple is created when the predicates
DATABASESINK
This operator can write data to a relational database.
TABLESCHEMA The types of the target database that should be used to create the target table. Order must be the same as the output schema.
CONNECTION –
DROP Drop table at start
DB –
LAZY_CONNECTION_CHECK –
BATCHSIZE How many elements should be buffered before storing to database.
BATCHTIMEOUT If batchsize is set, write tuple after some time (in ms) after last write even if batch is not full.
TRUNCATE Empty table at start
USER –
JDBC –
HOST –
TABLE Name of store table
PORT –
PASSWORD –
TYPE –

DATABASESOURCE
This operator can read data from a relational database.
WAITACHE This is an element based window.
PREDICATE –
ATTRIBUTE –
GROUP_BY –
UNIT –
SLIDE –
SIZE –
EXISTENCETOPAYLOAD The input object gets one new field with tuple existence.
DIFFERENCE This operator calculates the difference between two input sets.
DISTINCT This operator removes duplicates.
DISTRIBUTION Assign a distribution to the given attributes
VARIANCE The attribute holding the variance of the distribution.
CONTINUOUS The distribution is continuous or discrete.
ATTRIBUTES The attributes holding the expected value.
DUPLICATEELIMINATION Removes duplicates (Depending on the time model!)
DATABASE
Calculates the datarate and inserts the results into metadata
UPDATE RATE Element count after recalculating the datarate. Zero means no measurements.
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
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 –
TYPE –
ELEMENTWINDOW This is an element based window.
ADVANCE –
UNIT –
PARTITION –
SLIDE –
SIZE –
EXISTENCETOPAYLOAD The input object gets one new field with tuple existence.
FEATUREEXTRACTION Feature Extraction is used to extract the most important information from an input stream, e.g. calculating the orientation angle from given coordinates.
FILESINK
The operator can be used to dump the results of an operator to a file.
LINENUMBERS –
APPEND –
NUMBERFORMATTER –
FLOATINGFORMATTER –
DUMPMETADATA –
FILENAME –
FILETYPE –
CACHE SIZE –
FILTER Filters elements of the input stream. If predicate evaluates to true, element will be sent to port 0 else to port 1.
PREDICATE –
HEARTBEAT RATE –
FREQUENTPATTERN This operator create frequent item sets from a given stream. The result stream creates a tuple with 3 attributes: id: the number (a simple counter) of the pattern, set: the frequent pattern, which is a list of tuples (a nested attribute ~ NF^2), support: the support of the pattern
SUPPORT –
TRANSACTIONS –
ALGORITHM –
LEARNER –
OPTIONS –
FASTMEDIAN Calculate the median for one attribute in the input tuples
APPENDGLOBALMEDIAN If a GROUP_BY element is given, the global median (i.e. median without respecting groups) will be annotated to each element.
HISTOGRAM –
NUMERICAL –
GROUP_BY –
PERCENTILES –
ROUNDDINGFACTOR –
ATTRIBUTE –
GENERATERULES This operator uses a list of tuples and creates rules like “x => y”. A rule is a special datatype called “AssociationRule”, which is principally a tuple of two patterns (one for the premise and one for the consequence of the rule)
ITEMSET –
SUPPORT –
CONFIDENCE –
GENERATOR Generates missing values in a stream
FREQUENCY –
GROUP_BY –
MULTI –
EXPRESSIONS –
PREDICATE –

DTECTFACES
Detects faces in the images from the Kinect Camera
GROUPSPLITFILEWRITER

GroupSplitFileWriter

**DATASOURCE**
The name of the datahandler to use, e.g. Tuple or Document.

**PATH**
Outputfolder

**GROUPATTRIBUTES**

HDFSOURCE

Allows to read input from a nsc a hdf(5) based file

**MAXTIMETOWAITFORNEWEVENTMS**
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end

**SOURCE**
The name of the sourcetype to create.

**SCHEMA**
The output schema.

**INPUTSCHEMA**
A list of data types describing the input format. Must be compatible with output schema!

**FILENAME**

**PATHS**

**OPTIONS**
Additional options.

**DATEFORMAT**
The date format used.

HTTPSTREAMACCESS

Connect to a http stream

**MAXTIMETOWAITFORNEWEVENTMS**
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end

**SOURCE**
The name of the sourcetype to create.

**SCHEMA**
The output schema.

**DATASOURCE**
The name of the datahandler to use, e.g. Tuple or Document.

**INPUTSCHEMA**
A list of data types describing the input format. Must be compatible with output schema!

**PROTOCOL**
The name of the protocol handler to use, e.g. Csv or SizeByteBuffer.

**OPTIONS**
Additional options.

**DATEFORMAT**
The date format used.

INTERSECTION

This operator does not exist anymore.

IVEFNMEACONVERTER

This operator is used to convert Ief messages into Nmea messages and vice versa.

**CONVERSIONTYPE**
The conversion type between Maritime messages: AIS_To_IVEF, IVEF_To_AIS, TTM_To_IVEF, IVEF_To_TTM

**IVEFVERSION**
The version of IVEF elements: v015 (0.1.5), v025 (0.2.5)

**POSITIONSTATICRATIO**
The number of position messages the operator should wait iteratively before generating a new Static&Voyage message.

JOIN

Operator to combine two datastreams based on the predicate

**SWEEPAREANA**
Overwrite the sweep area

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

**PREDICATE**
Predicate to filter combinations

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

KALMAN

Kalman filter operator

**MEASUREMENT**

**TRANSITION**

**ATTRIBUTES**

**INITIALSTATE**

**CONTROL**

**INITIALERROR**

**PROCESSNOISE**

**MEASUREMENTNOISE**

**VARIABLES**

KEYPERFORMANCEINDICATORS

Allows KeyPerformanceIndicators for social media on input streams.

**TOTALQUANTITYOFTERMS**

**USERNAMES**

**THRESHOLDVALUE**

**SUBSETTERMS**

**INCOMINGTEXT**

**KPINAME**

KEYVALUETOPROBABILISTIC TUPLE

Translates a key-value/json object to a tuple

**SCHEMA**

**KEEPINPUT**

**TYPE**

KEYVALUE TUPLE

Translates a key-value/json object to a tuple

**SCHEMA**

**KEEPINPUT**

**TYPE**

LATENCY TO PAYLOAD

Adds attributes with the current latency information

**WRITE_BITVECTOR**

**WRITE_REGISTERS**

**WRITE_FUNCTION_CODE**

**WRITE_BOOLEAN**

**WRITE_BITVECTOR**

**WRITE_REGISTERS**

**WRITE_FUNCTION_CODE**

**WRITE_BOOLEAN**

**FUNCTION_CODE**

**SLAVE**

**UNITID**

**APPEND**

**SMALL**

LEFT JOIN

Left join: CURRENTLY NOT WORKING CORRECTLY.

**SWEEPAREA NAME**
Overwrite the sweep area

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

**PREDICATE**
Predicate to filter combinations

**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

**THREADS**
Number of threads used to calculate the result.

**EXPRESSIONS**

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

MERGE

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

MODBUSTCP SOURCE

Allows to read from a Modbus TCP connections.

**SLAVE**

**WRITE_BOOLEAN**

**FUNCTION_CODE**

**SCHEMA**

**WRITE_REGISTERS**

**WRITE_FUNCTION_CODE**

**WRITE_BOOLEAN**

**MEMBER**

**ACCESS**

For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
## OPCDASOURCE

Allows to read input from a OPC-DA connections.

**SCHEMA**
The output schema.

**PROGID**
-

**INPUTSCHEMA**
A list of data types describing the input format. Must be compatible with output schema!

**PATHS**
-

**CLSID**
-

**HOST**
-

**SOURCE**
The name of the sourcetype to create.

**MAXTIMETOWAITFORNEWEVENTS**
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end

**USERNAME**
-

**PASSWORD**
-

**DOMAIN**
-

**OPTIONS**
Additional options.

**DATEFORMAT**
The date format used.

## PATTERN

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

**TIME**
-

**INPUTPORT**
-

**COUNT**
-

**EVENTTYPES**
-

**OUTPUTMODE**
-

**SIZE**
-

**TIMEUNIT**
-

**TYPE**
-

**ASSERTIONS**
-

**RETURN**
-

**ATTRIBUTE**
-

## PREDICATEWINDOW

This is an predicat based window, set start and end condition with predicates.

**START**
-

**UNIT**
-

**END**
-

**SAMESTARTTIME**
-

**SIZE**
-

## PROJECT

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

**ATTRIBUTES**
A list of attributes that should be used.

**PATHS**
a list of attribute to use with keyvalue objects

## PROBABILISTIC

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

**ATTRIBUTE**
The name of the attribute for the existance uncertainty.

## PROBABILITY

Updates the existance probability of the input element.

**ATTRIBUTE**
The attribute holding the existen value

## PUBLISH

This Operator provides the publish functionality in publish/Subscribe systems.

**ROUTING**
if routing topology is selected, a routing algorithm must be added

**TOPICS**
advertise, which topics the processed objects match

**TOPOLOGYTYPEx**
the used topology type

**DOMAIN**
domain, where published objects will be processed

## QUALITY

Appending quality information to the incoming stream object.

**ATTRIBUTES**
-

**PROPERTIES**
-

## QUALITYINDICATOR

Store quality information in the metadata.

**FREQUENCY**
-

**COMPLETENESS**
-

**CONSISTENCY**
-

## RECEIVE

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

**MAXTIMETOWAITFORNEWEVENTS**
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end

**SOURCE**
The name of the sourcetype to create.

**SCHEMA**
The output schema.

**DATAMODEL**
The name of the datahandler to use, e.g. Tuple or Document.

**INPUTSCHEMA**
A list of data types describing the input format. Must be compatible with output schema!

**TRANSPORT**
The name of the transport handler to use, e.g. File or TcpServer.

**SOURCE**
The name of the sourcetype to create.

**DATAMODEL**
The output schema.

**INPUTSCHEMA**
A list of data types describing the input format. Must be compatible with output schema!

**DATEFORMAT**
The date format used.

## RECOMMENDATION

This operator learns a recommendation model. The result is a stream of recommendation models.

**ITEM**
The attribute with the item IDs.

**LEARNER**
The name of the leaner that should be used.

**RATING**
The attribute with the rating IDs.

**OPTIONS**
-

**TYPE**
The new type name of the output schema.

## RENAME

Renames the attributes

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

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

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

## RETRIEVE

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

**MAXTIMETOWAITFORNEWEVENTS**
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end

**TRANSPORT**
The name of the transport handler to use, e.g. File or TcpServer.

**SOURCE**
The name of the sourcetype to create.

**DATAMODEL**
The name of the datahandler to use, e.g. Tuple or Document.

**INPUTSCHEMA**
A list of data types describing the input format. Must be compatible with output schema!

**PROTOCOL**
The name of the protocol handler to use, e.g. Csv or SizeByteBuffer.

**DATEFORMAT**
The date format used.

## ROUTE

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

**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

**SENDINGHEARTBEATS**
If an element is routed to an output, heartbeats will be send to all other outputs

**PRECEDENCES**
-

## RPIGPIOSINK

Sink for Raspberry Pi GPIO-Port

**PINSTATE**
GPIO Pin state ('high' or 'low')

**PIN**
GPIO Pin Number
**RPBGIOsource**
Source for Raspberry Pi GPIO-Port

**MaxTimeToWaitForNewEvents**
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end.

**source**
The name of the sourcetype to create.

**Schema**
The output schema.

**Inputschema**
A list of data types describing the input format. Must be compatible with output schema!

**PIN**
GPIO Pin Number

**Options**
Additional options.

**Dateformat**
The date format used.

**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.

**QualityAttribute**
The attribute with the quality attribute that should be updated.

**TimestampAttribute**
The attribute with the value attribute.

**TimestampAttribute**
The attribute with the timestamp attribute that should be updated.

**ReplacementMethod**
The replacement method for missing value.

**Sample**
This operator can reduce load by throwing away tuples.

**TimeValue**

**Samplerate**

**SampleFrom**
Create samples from a given distribution

**Samples**
The number of samples to create.

**Attributes**
The distribution to sample from.

**Sase**
This operator can parse a query in SASE+ syntax.

**Query**

**Schema**

**OnematchPerInstance**

**Heartbeatrate**

**Type**

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

**Predicate**

**Heartbeatrate**

**Sentimentanalysis**
Allows sentiment detection on input streams.

**Thresholdvalue**

**Nominals**

**Classifier**

**AttributeTrainsettext**

**MaxTrainSize**

**Texttubeclassified**

**AttributeTrainsettrueDecision**

**SentimentDetection**
Allows sentiment detection on input streams.

**NGram**

**Trainsettext**

**EnrichAttribute**

**TrainsettrueDecision**

**Language**

**SplitDecision**

**StemmWords**

**Maxbuffersize**

**TestsettrueDecision**

**RemoveStopWords**

**DebugClassifier**

**Ngramupto**

**Classifier**

**Domain**

**Testsettext**

**Texttubeclassified**

**Trainsetminsize**

**Shareofvoice**
Allows to calculate the SoV.

**ThresholdValue**

**Owncompany**

**Incomingtext**

**Allcompanies**

**ShiprouteConverter**
This operator is used to convert ship route messages into IEC messages and vice versa.

**ConversionType**
The conversion type between shipRoute messages: JSON_TO_IEC, JSON_NMEA_TO_IVEF, IEC_TO_JSON_ROUTE, IEC_TO_JSON_MANOEUVRE, IEC_TO_JSON_PREDICTION, IEC_NMEA_TO_IVEF, IVEF_TO_JSON_ROUTE, IVEF_TO_JSON_MANOEUVRE, IVEF_TO_JSON_PREDICTION

**IVEFVersion**
The version of IVEF elements: v015 (0.1.5), v025 (0.2.5)

**SliceImage**
Slices images from the Kinect Camera

**Slice**

**Socketsink**
This operator can be used to send/provide data from Odysseus via a tcp socket connection. (Remark: This operator will potentially change in future)

**Host**

**Connecttoserver**

**Logineeeded**

**Sinktype**

**Sinkport**

**Datatransfer**

**SinkName**

**WITHMETADATA**

**Sinksink**
Stores the inertia cube stream to a file.

**Path**

**Storeinertia**
Stores the inertia cube stream to a file.
STOREKINECT
Stores the kinect stream to a file.

PATH –

STOREURG
Stores the urg stream to a file.

PATH –

STREAM
Integrate a view.

SOURCE –

SCHEMA The output schema.

NODE –

DATAHANDLER The name of the datahandler to use, e.g. Tuple or Document.

SOURCE –

SUBSCRIBE
This Operator provides the subscribe functionality in publish/Subscribe systems.

SOURCE –

SCHEMA The output schema.

INPUTSCHEMA A list of data types describing the input format. Must be compatible with output schema!

OPTIONS Additional options.

TEMPNUMBER The number of the temperature sensor

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

APPLICATIONTIMEFACTOR Factor to calculate milliseconds from application time

APPLICATIONTIMEUNIT Unit of application timestamps

TEXTPROCESSING
Allows preprocessing of incoming text.

DINGRAM –

DOSTEMMING –

INPUTTEXT –

DOREMOVESTOPWORDS –

NGRAMSIZE –

THROUGHPUT
Measure the current throughput

EACH –

ACTIVE –

DUMP –

TIMESHIFT
Shifts the timestamp(s) a given time

SHIFT –

TIMEWINDOW
The window sets the validity of the tuple. The default time granularity is in milliseconds. So, if you have another time granularity, you may use the unit-parameter (e.g. use 5 for size and SECONDS for the unit parameter) or you have to adjust the arity (e.g. use 5000 for size without the unit parameter)

ADVANCE –

UNIT –

SLIDE –

SIZE –

TEMPER1ACCESS
Returns the value of a temperature sensor of the type TEMPer1.

MAXTIMETOMAINTFNOREVENTMS For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end

SOURCE –

SCHEMA –

INPUTSCHEMA –

OPTIONS –

DATEFORMAT –

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

TUPLEAGGREGATE
Select from all elements of a window on with the given method

METHOD (MIN, MAX, LAST, FIRST)

ATTRIBUTE Attribute on which the method is evaluated

TUPLETOKKEYVALUE
Converts a tuple to a key-value/JSON object

TYPE type of key value object the tuples will be transformed to

TIMESTAMP
This Operator can be used to update the timestamp information in the meta data part. Be careful because this may lead undefined semantics

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

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

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

TIMEZONE The timezone in Java syntax.

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

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

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

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

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

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

THROUGHPUT
Measure the current throughput

EACH –

ACTIVE –

DUMP –

TIMESHIFT
Shifts the timestamp(s) a given time

SHIFT –
TWITTERSOURCE
Allows to read input from twitter.
SCHEMA
The output schema.
CONSUMERKEY
Twitter consumer key. See documentation.
ACCESSSTOKENSECRET
Twitter access token secret. See documentation.
INPUTSCHEMA
A list of data types describing the input format. Must be compatible with output schema!
ACCESSSTOKEN
Twitter access token. See documentation.
SEARCHKEYS
Twitter search keys. See documentation.
SOURCE
The name of the source type to create.
MAX TIMETOwaitFORNEWEVENTMS
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
CONSUMERSECRET
Twitter consumer secret. See documentation.
OPTIONS
Additional options.
DATEFORMAT
The date format used.
UDO
Calls a user defined operator
CLASS
ATTRIBUTES
INIT
UNION
Merges different input streams. (Typically preserves input order. Depending on the processing model)
UNNEST
The UnNest operator unpacks incoming tuple with a multi value attribute to create multiple tuples
RECALCULATE
ATTRIBUTE
VECTORQUANTIZATION
Process the incoming feature vector, from the Feature Extraction operator to determine the cluster id. Distinguish autonomous the incoming data, e.g. orientation, velocity, coordinates, to determine the correct method to work with
NUMCLUSTER
VKINECTSINK
Zeigt ein Fenster mit den Bildern der Kinect an.
WINDOW
use TimeWindow, ElementWindow or PredicateWindow instead
ADVANCE
UNIT
SLIDE
SIZE
TYPE
WEBCRAWLER
Crawl your website with custom depth and fetch.
SITE
MAX TIMETOwaitFORNEWEVENTMS
For access. Max time to wait for a new element before calling done. Typically used when the input stream has an end
SOURCE
The name of the source type to create.
SCHEMA
The output schema.
FETCH
INPUTSCHEMA
A list of data types describing the input format. Must be compatible with output schema!
DEPTH
OPTIONS
DATEFORMAT
The date format used.
Aggregates
AMEDIAN
NPV
AMEDIAN2
NTH
AVG
PKURT
COMPLETENESS
PSKEW
CURR
PSDEV
COUNT
RATE
CUW
REPRESSION
DPO
SKEW
DTW
SKURT
F FT
SPECTRALCENTROID
FIRST
SSKEW
JARQUE
SSTDEV
KURT
STDDEV
LAST
SUM
MAX
TEST
MEDIAN
UNIONGEOMETRY
MIN
VAR
NEST
WSEN RICH
Enrich tuples with data from external web services.
OUTERJOIN
URLSUFFIX
–
REMOVALSTRATEGY
–
WSDLLOCATION
–
ARGUMENTS
–
UNIQUEKEYS
–
CACHESIZE
–
PARSINGMETHOD
–
CACHING
–
DATAFIELDS
–
OPERATION
–
CHARSET
–
MULTITUPLEOUTPUT
–
SERVICEMETHOD
–
KEYVALUEOUTPUT
–
METHOD
–
URL
–
EXPIRATIONTIME
–
Functions
Bit
subset(BitVector, Integer, Integer) → BitVector
toBinary(Byte) → BitVector
toBinary(String) → BitVector
toBinary(Floating Number) → BitVector
toBinary(UnsignedInt16) → BitVector
tolong(BitVector) → Long
Bool
toBoolean(Object) → Boolean
toByte(ByteVector) → Byte
toInteger(ByteVector) → Integer
xor(Boolean, Boolean) → Boolean
Compare
disjoint(String, String) → Boolean
Crypt
DSA(Number) → List_String
EC(Number) → List_String
MD2withRSASign(Simple Type, String) → String
MD2withRSASign(Simple Type, String, String) → Boolean
MD5withRSAVerify(Simple Type, String, String) → Boolean
MD5withRSAVerify(Simple Type, String) → Boolean
NONEwithDSASign(Simple Type, String) → String
NONEwithDSASign(Simple Type, String, String) → Boolean
NONEwithECDSAVerify(Simple Type, String, String) → Boolean
NONEwithECDSASign(Simple Type, String, String) → Boolean
NONEwithECDSAVerify(Simple Type, String) → Boolean
NONEwithECDSASign(Simple Type, String) → Boolean
SHA1withRSAVerify(Simple Type, String, String) → Boolean
SHA1withRSAVerify(Simple Type, String) → Boolean
SHA1withECDSAVerify(Simple Type, String, String) → Boolean
SHA1withECDSAVerify(Simple Type, String) → Boolean
SHA1withECDSASign(Simple Type, String, String) → Boolean
SHA1withECDSASign(Simple Type, String) → Boolean
SHA1withECDSAVerify(Simple Type, String, String) → Boolean
SHA1withECDSAVerify(Simple Type, String) → Boolean
SHA1withECDSASign(Simple Type, String, String) → Boolean
SHA1withECDSASign(Simple Type, String) → Boolean
SHA1withDSASign(Simple Type, String, String) → Boolean
SHA1withDSASign(Simple Type, String) → Boolean
SHA224withRSAVerify(Simple Type, String, String) → Boolean
SHA224withRSAVerify(Simple Type, String) → Boolean
SHA256withRSAVerify(Simple Type, String, String) → Boolean
SHA256withRSAVerify(Simple Type, String) → Boolean
SHA256withECDSAVerify(Simple Type, String, String) → Boolean
SHA256withECDSAVerify(Simple Type, String) → Boolean
SHA256withECDSASign(Simple Type, String, String) → Boolean
SHA256withECDSASign(Simple Type, String) → Boolean
SHA256withDSASign(Simple Type, String, String) → Boolean
SHA256withDSASign(Simple Type, String) → Boolean
SHA384withRSAVerify(Simple Type, String, String) → Boolean
SHA384withRSAVerify(Simple Type, String) → Boolean
SHA384withECDSAVerify(Simple Type, String, String) → Boolean
SHA384withECDSAVerify(Simple Type, String) → Boolean
SHA384withECDSASign(Simple Type, String, String) → Boolean
SHA384withECDSASign(Simple Type, String) → Boolean
SHACrypto
toImage(Number, Number) → Image
toMatrix(Image) → Matrix

Interval
after(Inteval_Double, Interval_Double) → Boolean
before(Interval_Double, Interval_Double) → Boolean
contains(Interval_Double, Interval_Double) → Boolean
difference(Interval_Double, Interval_Double) → Interval_Double
duration(Interval_Double, Interval_Double) → Boolean
equals(Interval_Double, Interval_Double) → Boolean
finishes(Interval_Double, Interval_Double) → Boolean
inf(Interval_Double) → Double
intersection(Interval_Double, Interval_Double) → Interval_Double
meets(Interval_Double, Interval_Double) → Boolean
overlaps(Interval_Double, Interval_Double) → Boolean
starts(Interval_Double, Interval_Double) → Boolean
sup(Interval_Double) → Double
union(Interval_Double, Interval_Double) → Interval_Double

List
contains(Simple Type, List) → Boolean
IndexOf(List, Simple Type) → Integer
isEmpty(List) → Boolean
size(List) → Integer
toList(Object) → List

Math
abs(Number) → Double
acos(Number) → Double
AIC(Vector, ProbalisticDouble) → Double
AICc(Vector, ProbalisticDouble) → Double
as2DVector(ProbalisticDouble, ProbalisticDouble) → VectorProbalisticDouble
as3DVector(ProbalisticDouble, ProbalisticDouble, ProbalisticDouble) → VectorProbalisticDouble
asin(Number) → Double
atan(Number) → Double
atan2(Number, Number) → Double
BIC(Vector, ProbalisticDouble) → Double
cos(Number) → Double
cosh(Number) → Double
distance(ProbalisticDouble, Number) → Double
distance(VectorProbalisticDouble, MatrixBoolean) → Double
e() → Double
exp(Number) → Double
floor(Number) → Double
HQIC(Vector, ProbalisticDouble) → Double
inf() → Double
int(ProbalisticDouble, Number, Number) → Double
kl(VectorProbalisticDouble, ProbalisticDouble) → Double
kl(ProbalisticDouble, ProbalisticDouble) → Double
log(Number) → Double
log10(Number) → Double
loglikelihood(Vector, ProbalisticDouble) → Double
nan() → Double
pi() → Double
round(Number, Integer) → Double
sign(Number) → Number
similaritity(VectorProbalisticDouble, ProbalisticDouble) → Double
similarity(ProbalisticDouble, ProbalisticDouble) → Double
sin(Number) → Double
sinh(Number) → Double
sqrt(Number) → Double
tan(Number) → Double
tanh(Number) → Double
toDegrees(Number) → Double
toRadians(Number) → Double
UnaryMinus(Number) → Double

Matrix
det(Matrix) → Double
dotProduct(Vector, Vector) → Double
dotProduct(Matrix, Matrix) → Double
get(Vector, Number) → Number
get(Matrix, Number, Number) → Number
identity(Number) → Matrix
inv(Matrix) → Matrix
ones(Number, Number) → Matrix
perm(Matrix) → Double
perms(Matrix) → Matrix
readMatrix(String) → Matrix
readVector(String, Number) → Vector
readVector(String) → Vector
sAVG(Vector) → Double
sAVG(Matrix) → Double
sCount(Vector) → Integer
sCount(Matrix) → Integer
sMax(Vector) → Double
sMax(Matrix) → Double
sMedian(Matrix) → Double
sMedian(Vector) → Double
sMin(Matrix) → Double
sMin(Vector) → Double
sSum(Vector) → Double
sSum(Matrix) → Double
sub(Matrix, Number, Number) → Vector
sub(Vector, Number, Number) → Vector
toMatrix(Vector) → Matrix
toString(Vector) → String
toString(Matrix) → String
toVector(Matrix) → Vector
tr(Matrix) → Matrix
transpose(Matrix) → Matrix
vectorFromVector(String, Vector) → Vector
zeros(Number, Number) → Matrix

Polynomial
 assuresNumber(Number) → Double

Miscellaneous
c2f(Number) → Double
f2c(Number) → Double
f2k(Number) → Double
k2f(Number) → Double
km2mph(Number) → Double
km2mmps(Number) → Double
mph2kmph(Number) → Double
mps2kmph(Number) → Double
speedOfLight() → Double
speedOfSound(Number) → Double

Signal
imaginary(Complex) → Double
real(Complex) → Double

Store
ContextStore(String) → Tuple

String
concat(Object, Object) → String
length(String) → Integer
lower(String) → String
startsWith(String, String) → Boolean
strContains(String, String) → Boolean
substring(String, String, String) → String
substring(String, Number, String) → String
substring(String, Number, Number) → String
upper(String) → String

Text
collapseSpaces(String) → String
levenstein(String, String) → Integer
metaphone(String) → String
soundex(String) → String

Time
businessDays(Date, Date) → Integer
curdate() → Date
dateInMillis(Date) → Long
day(Date) → Integer
day(Date, String) → Long
dayOfWeek(String, String) → Integer
dayOfYear(Date) → Integer
days(Date, Date) → Integer
hours(String, String) → Integer
hour(Date) → Integer
hours(Date, Date) → Integer
millisecond(Date) → Long
milliseconds(String, String) → Long
millisTime() → Long
doubleToShort(
doubleToLong(
doubleToFloat(
doubleToByte(
toFloat(
toChar(
toByte(
doubleToInteger(
toInteger(
toInteger(
toNumber(
toInterval(
toString(
toObject(
toString(Complex, Complex) → Complex

Symbols
!) → Boolean
!(String, String) → Boolean
!(Number | Object, Number | Object) → Double
!& (Number | Object, Number | Object) → Double
!(Boolean | Object, Number | Object) → Double

Transform
doubleToBoolean(Double) → Boolean
doubleToByte(Double) → Byte
doubleToChar(Double) → Char
doubleToFloat(Double) → Float
doubleToInteger(Double) → Integer
doubleToShort(Double) → Short
toByte(Object) → Byte
toChar(String) → Char
toChar(Discrete Number) → Char
toComplex(Number, Number) → Complex
toDouble(Double) → Object
toDouble(Object) → Double
toDouble(Decimal) → Double
toDouble(UnsignedInt16, UnsignedInt16, Boolean) → Float
toDouble(UnsignedInt16, UnsignedInt16) → Float
toInteger(Boolean) → Integer
toInteger(Number) → Integer
toInteger(String) → Integer
toInteger(interval, Number) → Interval_Double
toLong(Object) → Long
ToPolynomial(Vector) → Polynomial
ToProportionalContinuousDouble(MatrixBoolean, MatrixBoolean) → ProportionalContinuousDouble

toProportionalDouble(MatrixBoolean, MatrixBoolean) → ProportionalDouble
toShort(Object) → Short
toSpatialGrid(Number, Number) → Grid
toSpatialGrid(Matrix, Number, Number, Number) → Grid
toString(Polynomial) → Polynomial
toString(Interval_Double) → String
toString(Object) → String
toString(Complex) → String

- (Number | Object, Number | Object) → Double
- (Polynomial, Polynomial) → Polynomial
- (ProportionalDouble, Number) → ProportionalDouble
- (Complex, Complex) → Complex
- (Number, ProportionalDouble) → ProportionalDouble
- (Date, Number) → Date
- (Interval_Double, Interval_Double) → Interval_Double
- (Matrix, Number) → Matrix
- (Vector, Vector) → Vector
- (Vector, Number) → Vector
- (Matrix, Matrix) → Matrix
- (Number | Object, Number | Object) → Double
- (Boolean, String) → Boolean
- (Date, Date) → Date
| Matrix, Number | → | Vector |
| List, Number | → | Object |
| Vector, Number | → | Double |
| Matrix, Vector | → | Double |
| (Interval, Double, Number) | → | Interval_Double |
| (Matrix, Number) | → | Matrix |
| (Number, Object, Number, Object) | → | Double |
| (Number, Object, Number, Object) | → | Long |
| (BitVector, BitVector) | → | BitVector |
| (Boolean, Boolean) | → | Boolean |
| (ProbabilisticResult, ProbabilisticResult) | → | ProbabilisticResult |
| (BitVector) | → | BitVector |
| (Number) | → | Long |

### Handlers

<table>
<thead>
<tr>
<th>Data Handlers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG</td>
</tr>
<tr>
<td>BIT</td>
</tr>
<tr>
<td>BOOLEAN</td>
</tr>
<tr>
<td>BYTE</td>
</tr>
<tr>
<td>COLOR MAP</td>
</tr>
<tr>
<td>COUNTER</td>
</tr>
<tr>
<td>DATE</td>
</tr>
<tr>
<td>DEPTH</td>
</tr>
<tr>
<td>DETECTED</td>
</tr>
<tr>
<td>DOCUMENT</td>
</tr>
<tr>
<td>DOUBLE</td>
</tr>
<tr>
<td>END</td>
</tr>
<tr>
<td>FLOAT</td>
</tr>
<tr>
<td>IMAGE</td>
</tr>
<tr>
<td>IMAGE</td>
</tr>
<tr>
<td>INTEGER</td>
</tr>
<tr>
<td>INTERVAL</td>
</tr>
<tr>
<td>INTERVAL</td>
</tr>
<tr>
<td>KEY</td>
</tr>
<tr>
<td>LIST</td>
</tr>
<tr>
<td>LONG</td>
</tr>
<tr>
<td>MATRIX</td>
</tr>
<tr>
<td>MULTI</td>
</tr>
<tr>
<td>MY</td>
</tr>
<tr>
<td>NESTED</td>
</tr>
<tr>
<td>NTUPLE</td>
</tr>
<tr>
<td>OPC</td>
</tr>
</tbody>
</table>

### Protocol Handlers

<table>
<thead>
<tr>
<th>Protocol Handlers</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSON</td>
</tr>
<tr>
<td>CSV</td>
</tr>
<tr>
<td>DOCUMENT</td>
</tr>
<tr>
<td>FACEBOOK</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>HTML</td>
</tr>
<tr>
<td>INERTIACUBE</td>
</tr>
<tr>
<td>IVEF_0</td>
</tr>
<tr>
<td>IVEF_0</td>
</tr>
<tr>
<td>JASPER</td>
</tr>
<tr>
<td>JSON</td>
</tr>
<tr>
<td>KINETIC</td>
</tr>
<tr>
<td>LINE</td>
</tr>
<tr>
<td>LMS</td>
</tr>
<tr>
<td>MARKER</td>
</tr>
<tr>
<td>NMEA</td>
</tr>
<tr>
<td>NONE</td>
</tr>
<tr>
<td>PLUG</td>
</tr>
<tr>
<td>SASIZE</td>
</tr>
<tr>
<td>SHIP</td>
</tr>
<tr>
<td>SHIP</td>
</tr>
<tr>
<td>SIMPLE</td>
</tr>
<tr>
<td>SIMPLE</td>
</tr>
<tr>
<td>SIZER</td>
</tr>
<tr>
<td>STRING</td>
</tr>
<tr>
<td>SUN</td>
</tr>
<tr>
<td>SVM</td>
</tr>
<tr>
<td>TEXT</td>
</tr>
<tr>
<td>TIKA</td>
</tr>
<tr>
<td>UMG</td>
</tr>
<tr>
<td>WAV</td>
</tr>
<tr>
<td>XLS</td>
</tr>
<tr>
<td>XML</td>
</tr>
</tbody>
</table>

### Transport Handlers

<table>
<thead>
<tr>
<th>Transport Handlers</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPEND</td>
</tr>
<tr>
<td>AUDIO</td>
</tr>
<tr>
<td>DIRECTORY</td>
</tr>
<tr>
<td>FACEBOOK</td>
</tr>
<tr>
<td>FILE</td>
</tr>
<tr>
<td>HTTP</td>
</tr>
<tr>
<td>HTTP</td>
</tr>
<tr>
<td>IMAP</td>
</tr>
<tr>
<td>INERTIACUBE</td>
</tr>
<tr>
<td>KINETIC</td>
</tr>
<tr>
<td>MODBUS</td>
</tr>
<tr>
<td>MODBUS</td>
</tr>
<tr>
<td>NCS</td>
</tr>
<tr>
<td>NON</td>
</tr>
<tr>
<td>NUMERIC</td>
</tr>
<tr>
<td>OPC</td>
</tr>
<tr>
<td>PING</td>
</tr>
<tr>
<td>POP</td>
</tr>
<tr>
<td>PRINTER</td>
</tr>
<tr>
<td>PRO</td>
</tr>
<tr>
<td>PRO</td>
</tr>
<tr>
<td>PRO</td>
</tr>
<tr>
<td>TEMPE</td>
</tr>
<tr>
<td>TIMER</td>
</tr>
<tr>
<td>TWITTER</td>
</tr>
<tr>
<td>UMB</td>
</tr>
<tr>
<td>UDP</td>
</tr>
<tr>
<td>UDP</td>
</tr>
<tr>
<td>UMG</td>
</tr>
<tr>
<td>WEB</td>
</tr>
<tr>
<td>YAHOO</td>
</tr>
<tr>
<td>YAH</td>
</tr>
<tr>
<td>ZER</td>
</tr>
</tbody>
</table>

### Odysseus Script

<table>
<thead>
<tr>
<th>Odysseus Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
</tr>
<tr>
<td>#</td>
</tr>
<tr>
<td>ACTIVATE</td>
</tr>
<tr>
<td>ADD</td>
</tr>
<tr>
<td>BEGIN</td>
</tr>
<tr>
<td>BUFFER</td>
</tr>
<tr>
<td>CONFIG</td>
</tr>
<tr>
<td>DATA</td>
</tr>
<tr>
<td>DEACTIVATE</td>
</tr>
<tr>
<td>DEFINE</td>
</tr>
<tr>
<td>DO</td>
</tr>
<tr>
<td>DO</td>
</tr>
<tr>
<td>DO</td>
</tr>
<tr>
<td>DO</td>
</tr>
<tr>
<td>DO</td>
</tr>
<tr>
<td>DROP</td>
</tr>
<tr>
<td>DROP</td>
</tr>
<tr>
<td>DROP</td>
</tr>
<tr>
<td>DROP</td>
</tr>
<tr>
<td>DROP</td>
</tr>
<tr>
<td>ELSE</td>
</tr>
<tr>
<td>END</td>
</tr>
<tr>
<td>END</td>
</tr>
<tr>
<td>END</td>
</tr>
<tr>
<td>EVAL</td>
</tr>
<tr>
<td>EXECUTE</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>LOGIN</td>
</tr>
<tr>
<td>LOGOUT</td>
</tr>
<tr>
<td>LOOP</td>
</tr>
<tr>
<td>METADATA</td>
</tr>
<tr>
<td>ODYSSEUS</td>
</tr>
<tr>
<td>PARSER</td>
</tr>
<tr>
<td>PAR</td>
</tr>
<tr>
<td>PLA</td>
</tr>
<tr>
<td>PRE</td>
</tr>
<tr>
<td>PRINT</td>
</tr>
<tr>
<td>PROCEDURE</td>
</tr>
<tr>
<td>QUERY</td>
</tr>
<tr>
<td>QUEUE</td>
</tr>
<tr>
<td>REQUIRED</td>
</tr>
<tr>
<td>RESUME</td>
</tr>
<tr>
<td>RESUME</td>
</tr>
<tr>
<td>RUN</td>
</tr>
<tr>
<td>SCHEDULE</td>
</tr>
<tr>
<td>SCHEDULE</td>
</tr>
<tr>
<td>SCHEDULE</td>
</tr>
<tr>
<td>SCHEDULE</td>
</tr>
<tr>
<td>SCHEDULE</td>
</tr>
<tr>
<td>SCHEDULE</td>
</tr>
<tr>
<td>TEMP</td>
</tr>
<tr>
<td>TEMP</td>
</tr>
<tr>
<td>TRAFFIC</td>
</tr>
<tr>
<td>TRANS</td>
</tr>
<tr>
<td>UNDEF</td>
</tr>
<tr>
<td>UPDATE</td>
</tr>
<tr>
<td>UPTO</td>
</tr>
<tr>
<td>WAIT</td>
</tr>
</tbody>
</table>

### Constants

<table>
<thead>
<tr>
<th>Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWT</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>ECLIPSE</td>
</tr>
<tr>
<td>EQUINOX</td>
</tr>
<tr>
<td>FILE</td>
</tr>
<tr>
<td>FILE</td>
</tr>
<tr>
<td>FILE</td>
</tr>
<tr>
<td>JAVA</td>
</tr>
<tr>
<td>JAVA</td>
</tr>
<tr>
<td>JAVA</td>
</tr>
<tr>
<td>JAVA</td>
</tr>
<tr>
<td>JAVA</td>
</tr>
</tbody>
</table>
Sample Odysseus query

```python
#PARSER PQL
#ADDQUERY

ing = ACCESS(source='source',
            wrapper='GenericPush',
            transport='File',
            protocol='CSV',
            dataHandler='Tuple',
            options=['filename','example.csv'],
            schema=[['value','Double']])

output = MAP(expressions = ['value + 3'], input)
```

Copyright © 2014 ODYSSEUS Team

[http://odysseus.informatik.uni-oldenburg.de/](http://odysseus.informatik.uni-oldenburg.de/)