CSV protocol handler

The CSV protocol handler parses delimiter separated text lines.

For performance reasons: If your input does not contains strings with a delimitter (i.e. there are escaped strings) you should use SimpleCSV protocol handler, which is much faster!

Important: Each CSV-line must end with a newline! Especially important when not using a file for input but e.g. TCPServer transport handler or AMQP /RabbitMQ transport handler.

Options

- **delay**: Delay of reading in milliseconds (default 0). DEPRECATED: use `scheduler.delay` instead if not used together with delayeach
- **nanodelay**: Delay of reading in nanoseconds (default 0).
- **delayeach**: The number of lines between a delay is used (default 0).
- **readfirstline**: Should the first line of the file be ignored (e.g. because of header information) (default: true)
- **debug**: If set to true, some additional thinks are available: (default false)
- **dumpEachLine**: Dumps lines to the console. if set to 1 each line will be dumped
- **measureEachLine**: Measures the processing time between n elements that are dumped
- **maxLines**: Stop processing after n elements are read
- **csv.delimiter**: The delimiter for splitting the input (Default: ,).
- **csv.textDelimiter**: The delimiter for strings (Default: "). Inside of strings, csv.delimiter is ignored.
- **addlinenumber**: Adds the line number (starting with 0) to the beginning of the line. Remember to add a proper attribute to the schema! (default false).
- **csv.floatingFormatter**:If used for writing, **each double/float** value will be formatted using this formatter (default null). See https://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html
    - This option can also be used for reading.
    - This option can be used together with:
        - decimalseparator: How is the fraction of the number separated. E.g. for reading german formatted doubles you must use something like: ['csv.floatingformatter','#,##'], ['decimalseparator',',']
        - exponentseparator: typically E
        - groupingseparator: Typically ",". In german this is "."
- **csv.numberFormatter**: If used for writing, each number **other than double/float** value will be formatted using this formatter (default null). See https://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html
- **csv.trim**: Removes leading and trailing whitespaces in each element (default false)
- **nullvaluetext**: The representation of null values, e.g. '<NULL>' (Default: '')
- **csv.writeheading**: Writes the schema as header of the csv, if set to true (Default: false)

Example

PQL

CSV Protocol Handler

```
input = ACCESS({
                          source='fridgeVibrationCSV',
                          wrapper='GenericPull',
                          transport='File',
                          protocol='CSV',
                          datahandler='Tuple',
                          options=[
                            ['delimiter',','],
                            ['textDelimiter',"'"],
                            ['readfirstline','true'],
                            ['delay','100'],
                            ['filename', 'path to file']
                          ],
                          schema=[
                            ['description','Double']
                          ]
                        }
                      )
```

CQL

CSV Protocol Handler

```
CREATE STREAM csv (symbol String, points Double)
    WRAPPER 'GenericPush'
    PROTOCOL 'CSV'
    TRANSPORT 'File'
    DATAHANDLER 'Tuple'
    OPTIONS ( 'delimiter' ',', 'textDelimiter' "'", 'readfirstline' 'true', 'delay' '100')
```