QVD Writer – an EVL Microservice

QVD Writer is an EVL microservice which enables writing to QVD files without using Qlik Sense or QlikView. Very useful when you need to load data from various sources immediately into a QVD file and you don't want to overload your Qlik server by creating many, or large, QVD files.

QVD Writer advantages

- Write speed
- Available on Linux, Windows, Mac, and as Docker image
- Low implementation and operating costs
- Automation and batch processing

EVL Microservices are built on top of the core EVL software and retain its flexibility, robustness, high productivity, and ability to read data from various sources; including csv and Excel files, databases – Oracle, Teradata, SQL Server, etc – and Hadoop streaming data like Kafka.

Example Converting CSV to QVD

Example input CSV file with header and semicolon separated fields:

ID	Name	Code	Price	Created
1	item_11	11	15,85	05/04/2016
2	item_12	12	21,25	11/11/2019
3	item_21	21	12,99	10/05/2019

Run:

```
csv2qvd < table.csv > table.qvd
```

`csv2qvd` will use the field names present in the header, and data types derived from the values, to produce an output file named `table.qvd`.

Assigned data types can be reviewed by checking `STDERR`. Our example created this EVD (an EVL data type schema):

```
ID      int              null="" sep=";"
Name    string           null="" sep=";"
Code    int              null="" sep=";"
Price   decimal(8,2)     null="" sep=";"
Created date("%d/%m/%Y") null="" sep="\n"
```

Assumed data types may not always be correct. In our example the date format should be `MM/DD/YYYY`, not `DD/MM/YYYY`. Also, the `Code` field should interpreted as a string, not an integer, to preserve the leading zeros. We can manually modify the generated output and save it as an EVD file. In our example, `table.evd`:

```
ID      int              null="" sep=";"
Name    string           null="" sep=";"
Code    string           null="" sep=";"
Price   decimal(8,2)     null="" sep=";"
Created date("%m/%d/%Y") null="" sep="\n"
```

We can then use the saved schema, `table.evd`, to process the CSV file:

```
csv2qvd table.evd < table.csv > table.qvd
```

Now the QVD output will have the correct data types.