

CSV in FileMaker with MBS FileMaker Plugin

We like to show you a few functions related to CSV and what you can do with combining various functions of our [MBS FileMaker Plugin](#). Usually you get a CSV file as an export from some other database, e.g. a list of sales.

Read Textfile

If you like to read a text file, you can simply use our [Text.ReadTextFile](#) function. Pass the path and the text encoding to use and you get back the text. The encoding should be right for umlauts or accents. Usually everyone uses UTF-8 nowadays, but we also support various older encodings.

If you have the file in a container, please check the [Text.ReadTextFromContainer](#) function, which takes a container value as reference.

Parse CSV

Next you like to parse the CSV and split it into a two-dimensional array. For this we use the [Matrix.CSVSplit](#) function. It creates a new matrix object for us with the right dimensions and reads in the values. Our plugin can usually detect whether to use tab, comma or semicolon as separated, but you can also pass an explicit separator.

Column Names

Most CSV files have the column names in the first row. You can get the list of names with the [Matrix.GetRow](#) function by passing row = 0. Once you got the list of names, you can remove the row for further processing, so call [Matrix.RemoveRow](#) to remove it. If you like, you can store the them with the matrix using the [Matrix.SetColumnNames](#) function (new in version 12.5).

Comma vs. dot

If you have numbers in the CSV, they may not have the right decimal separator. We can use [Matrix.GetColumn](#) function to get all the values in a row as a list. Next you can do a simple call to `Substitute()` to replace dot with comma for example. Then put the values back with [Matrix.SetColumn](#) function. That allows us to convert decimal separator to comma for a German database file, where FileMaker uses comma. Otherwise the functions for numeric values won't work.

Math Operations

Let us say you like to check a column for record ID numbers for the smallest or biggest value. For that you use [Matrix.Min](#) and [Matrix.Max](#) functions. So you know what is the first and the last ID.

For a column with sale prices, you could use [Matrix.Sum](#) to know the sum of all sales. [Matrix.Min](#), [Matrix.Max](#) and [Matrix.Avg](#) can tell you the minimum, maximum and average sale price.

When you have a column with percentages, they may either be stored as number from 0 to 100 or more mathematically as fraction from 0.0 to 1.0. Usually you need the other way, so you use our [Matrix.Multiply](#) function to multiply them either by 100 or by 0.01 to divide by 100.

Add or Remove rows and columns

You may not need all rows, so you can remove some with [Matrix.RemoveRow](#) function. Or add new rows with [Matrix.AddRow](#) or [Matrix.AddRows](#) functions and fill them with [Matrix.SetRow](#) function.

If you later need a new column with some value, like an Import ID, you can add columns with [Matrix.AddColumn](#) function. Or remove a column you don't need with [Matrix.RemoveColumn](#) function. Fill in the values of a new column with the [Matrix.SetColumn](#) function.

JSON

Now if you like to pass this matrix to a web service (like Data API) as JSON, you can query records in this format using the [Matrix.JSONRecords](#) function. Since you pass the field names as value list, you can have different names for them. The easiest may be to pass the list of column names from above.

If you only need one record as JSON, check the [Matrix.JSONRecord](#) function.

When you use our [MongoDB](#) functions, you can pass the JSON for one record to the [MongoDB.InsertOne](#) function. Or you pass the records as block to [MongoDB.InsertMany](#) function and avoid the loop in the FileMaker script.

Data Types

The matrix preserves the FileMaker data types. But the CSV parse just creates text values and you may need them as date or number. So we recently added the [Matrix.ConvertDataType](#) function to convert types. This allows you to convert the values of a text column to a number. And that avoids trouble later with SQL complaining about wrong data types.

Insert to FileMaker

Finally we talk about the import to a FileMaker table. The [Matrix.InsertRecords](#) function creates records for you. Pass the name of the database file, the name of the table and the list of field names. That can be the column names from above or your own list of column names. This allows you to change the order of the field names and have completely different names. Since the last assignment wins, you could also specify a field name several times in the list to ignore some fields.

If you connect to an external SQL database with our [SQL](#) functions, you can use [Matrix.InsertRecordsToSQL](#) function. This creates a record in your external database. Like above you can pass a table name and the list of field names to use, so they may be different than your column names.

Please don't hesitate to contact us if you have questions.