Dump_InFluxDB: User's Guide

v1.00.00, Stefan Eichenberger, 2020-05-23

1 Table of Contents

2 Introduction

This script dumps an InFlux database into an Excel file for further analysis, adds some calculated columns and creates charts, all controlled from a config file. (Charts are considered an experimental feature at this time)

We will use a sample InFlux database as created by SolarAnzeige (https:solaranzeige.de), configured to log data from a Kostal Plenticore Plus 10 Inverter. This InFlux database contains tables named

- `PV, AC, Batterie, Summen` which are written synchronously – that is, they contain entries for the same time stamps
- other tables `Info, Service, Statistik` are not aligned on time stamps. Whilst they can be dumped in a similar way, this Users Guide does not discuss them further.

3 Concept

The script described here performs the following tasks:

- Connect to InFlux DB and dump tables into an Excel .xlsx file. The dumped time window is selectable from the command line.
- Tables can be joined (aligned) by time stamp
- Calculated columns can be added as needed
- Charts can be created and are stored in the same Excel file

Table joins, calculated columns and charts are defined in a config file, so that operations can easily be repeated.

The script can run either on the Raspberry where SolarAnzeige is installed (slow, easy to install) or on a connected PC (fast, but some installation to do).

4 Command Line Options

4.1 Basics

The following output can be generated with `Dump_InFluxDB.pl -h`:

```
Dump_InFluxDB.pl [<time_period_options> -f -o -T <list> -c <CfgFile> -v]
   dumps data from InFlux DB into an .xlsx file

   Time period of dump is defined through <time_period_options>:
       Default: current month from yyyy-mm-01 to YYYY-MM-DD (DD = last day of month)
       Parts can be overwritten:
       -d <d>     overwrites start date day
                  If <d> is negative, start date is set to <today> - <d> days
                  Example: -d -1 dumps today only, -d -10 dumps last 10 days incl. today
       -m <m>, -y <y> overrites month mm and year  yy of start date (neglected if <d> < 0)
       -D <d>, -M <m>, -Y <y> overwrites day DD, month MM, year YYYY of end date
       Time is from 00:00 to 23:59 UTC of the respective start and end dates

   For long periods, memory consumption may become huge (and performance slow). Hence,
   period cannot be longer than 32 days unless -f forces a protection overwrite.
   This is done at the users risk and may crash Influx and/or the Raspi due to memory
   consumption

   -o  allow overwriting if output file already exists (default: no overwrite)
   -O  <OutFile> creates output file <file_yyyy-mm-dd.xlsx> where yyyy-mm-dd is the start
               day of the dumped time period. Default 'SolarAnzeige'
   -T  <list>    <list> is a comma-separated list of tabs to be created
               Default: all tabs
   -c  <CfgFile> Config file; default: config.ini
   -v  verbose output (for chart debugging)

Dump_InFluxDB.pl -h
   --> shows this help text
```

4.2 Remarks

- Depending on the environment, the script is called:
  - Raspberry: `./Dump_InFluxDB.pl`
    or `perl Dump_InFluxDB.pl`
  - Windows: `perl Dump_InFluxDB.pl`
- If config file is named `config.ini` and resides in the same directory as `Dump_InFluxDB.pl`, it is found automatically (option -c not needed)
- On Raspberry of SolarAnzeige, a basic version operates without a `config.ini` file. It will dump InFluxDB tables `PV`, `AC`, `Batterie` and `Summen` into one Excel tab `01_Data`. The tables will be aligned by date stamp.
- Time window is 00:00 of the start date `yyyy-mm-dd` to 23:59 of the end date `YYYY-MM-DD`. Time zone used is always UTC.
- The Excel sheet cannot be over-written if open in Excel already
- The script has not been tested for long time periods to be dumped. Hence the `-f` option.

4.3 Config File Content

This section describes the content of the config file. The exact syntax is described in the chapter Config File Syntax. Hence, in this chapter, we'll more focus on the high level functional view.

In the following examples, config values are shown in gray for illustration only.

4.3.1 Section [GENERAL]

```
influx      : C:/Exec/Portable/InfluxDB/influxdb-1.7.10-1/influx.exe
host        : 192.168.178.89
db          : solaranzeige
time_zone   : Europe/Berlin
xlsx        : SolarAnzeige
tables      : { 01_Data => PV, AC ;
              02_Sums => Summen ;
            }
```

`influx` `influx` executable. Default is '`influx`' which assumes that the executable is available in the path (as is the case in the default installation of SolarAnzeige).

`host` host where InFlux database resides. Default is localhost (which assumes that the script is executed on the SolarAnzeige Raspberry)

`db` name of the InFlux database to be dumped. Default is '`solaranzeige`'

Keys `host` and `db` can be moved to table specific sections if different tables reside on different hosts and/or different databases (as is the case in multi-controller setups).

`time_zone` specifies the time zone used for column `DateTime` in Excel. Default is UTC. Valid time zone names are specified here:

https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

Note that the time zone for the period selection on the command line is always UTC.

`tables` Tables (or Excel tabs) to be generated. Since multiple tables can be generated, this is a map, specifying a table (tab) name (such as `01_Data`) and InFlux table(s) to be dumped into that tab. Tabs are sorted alphabetically.

If multiple InFlux tables are dumped into one Excel tab (as in the first case: `PV, AC`) it is assumed that these tables can be aligned by the time stamp.

Default is `{ 01_Data => PV, AC, Batterie, Summen; }`. This allows to perform a basic dump without any config file.

`xlsx` can be used to overwrite the default of the command line option `-O` ('`SolarAnzeige`') for the output file name.

4.3.2 Table specific sections (eg. [01_Data])

Each table defined above can be further specified in its own section as follows:

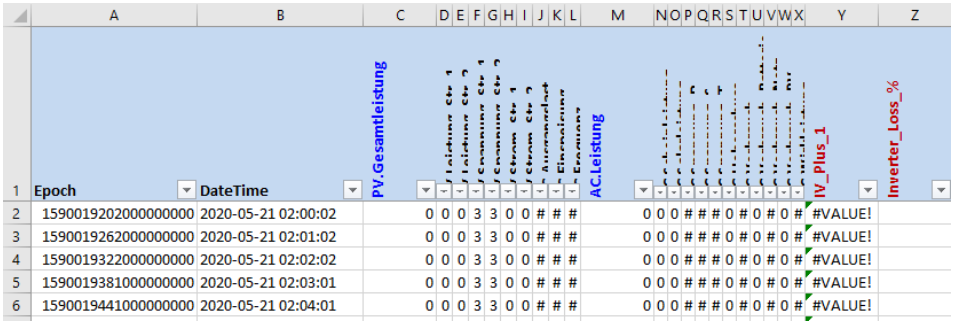
```
[01_Data]
drop       : { PV.Leistung_Str_3 ;
               PV.Spannung_Str_3 ;
               PV.Strom_Str_3    ;
             }

calculate  : { Inverter_Loss_% =>
                 if('PV.Gesamtleistung'=0, "",
                 ('PV.Gesamtleistung' - 'AC.Leistung')/'PV.Gesamtleistung') ;

               IV_Plus_1       => 'Inverter_Loss_%'+1 }

charts     : { eff_loss => add_chart( type => 'scatter',
                                      name => 'Inverter' )
                           add_series( categories => 'PV.Gesamtleistung',
                                       values     => 'Inverter_Loss_%',
                                       name       => 'InvLoss' ) ;

               PV_Power => add_chart( type => 'line',
                                      name => 'PV_power')
                           add_series(categories => 'DateTime',
                                      values     => 'PV.Gesamtleistung')
             }
```

`drop` — InFlux database fields to be dropped. Database field names (eg. `Leistung_Str_3`) are pre-pended with the table name (eg. `PV`) and separated with a dot (`.`)

`calculate` — Multiple Calculated columns can be added. The map key is the name of the calculated column and the map value must be a valid Excel expression except that instead of referring to cell values, reference is to column names (quoted in single quotes (`'`). Columns are added in alphabetical order

Explanation:

Assume the resulting Excel table looks something like this:



	A	B	C	M	Y	Z
1	Epoch	DateTime	PV.Gesamtleistung	AC.Leistung	IV_Plus_1	Inverter_Loss_%
2	1590019202000000000	2020-05-21 02:00:02	0	0	#VALUE!	
3	1590019262000000000	2020-05-21 02:01:02	0	0	#VALUE!	
4	1590019322000000000	2020-05-21 02:02:02	0	0	#VALUE!	
5	1590019381000000000	2020-05-21 02:03:01	0	0	#VALUE!	
6	1590019441000000000	2020-05-21 02:04:01	0	0	#VALUE!	

The column `Inverter_Loss_%` is calculated from values in column C (`PV.Gesamtleistung`) and column M (`AC.Leistung`). Hence, we can calculate a column as shown above and the resulting formula in eg. row 2 will be `=IF(C2=0, "", (C2 - M2)/C2)`

Likewise, column `IV_Plus_1` will contain the trivial formula `=Z2+1` (which Excel will evaluate to `#VALUE!` if the referenced cell in column Z contains "").

charts

Multiple charts (eg. `eff_loss` and `PV_Power`) can be created, based on the underlying table data.

Charts are created by calls to methods of the perl module

Excel::Writer::XLSX::Chart

with documentation here: https://metacpan.org/pod/Excel::Writer::XLSX::Chart

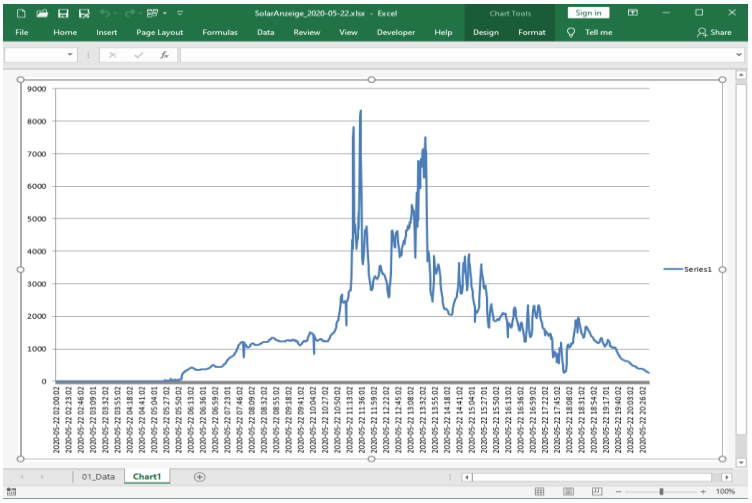
Any legal sequence of method calls (except `combine()`) should result in a corresponding chart.

A minimalistic chart definition can look like this:

```
PV_Power => add_chart( type => 'line' )   # <-- no separator here!
            add_series(categories => 'DateTime',
                       values     => 'PV.Gesamtleistung' );
```

`add_chart()` is mandatory and describes the chart `type`.

At least one call do `add_series()` is required, specifying the columns to be plotted. This creates an Excel tab `Chart1` (default name) where `PV.Gesamtleistung` is plotted as a function of time. This may look like this:



Note the whole value section for the key `PV_Power` (see Map Values) consists of a sequence of method calls *without a separator between them*! The last semicolon (`;`) terminates the sequence of method calls

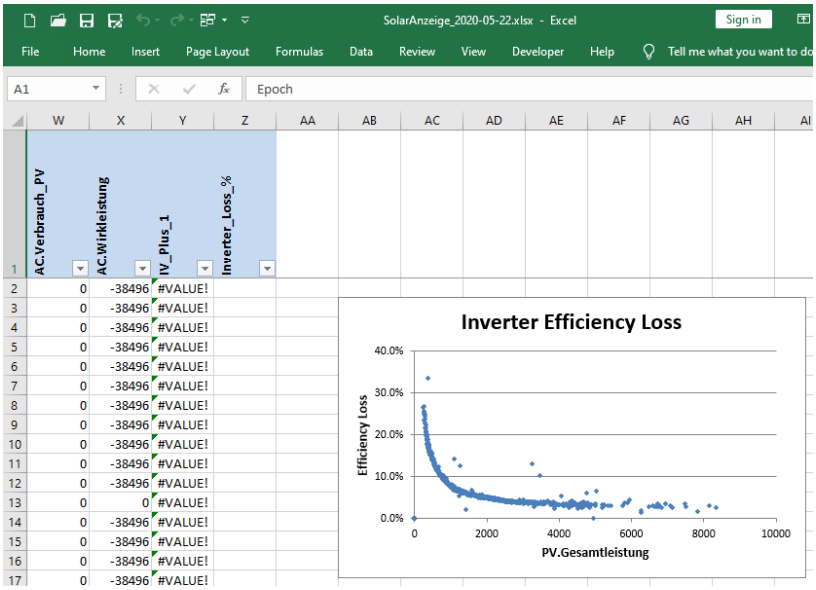
```
method_1 (parameter_1, parameter_2, ...)
method_2 (parameter_1, ...);
```

A more elaborate chart definition can look like this:

```
Eff_Loss => add_chart( type => 'scatter',
                       name => 'Inverter',
                       embedded => 1 )
            add_series( categories => 'PV.Gesamtleistung',
                        values     => 'Inverter_Loss_%',
                        name       => 'Inverter Efficiency Loss',
                        marker     => { type => 'diamond',
                                        size => 3 } )
            set_x_axis( name       => 'PV.Gesamtleistung',
                        name_font  => { name => 'Calibri (Body)',
                                        size => 10.5, bold => 1 },
                        num_font   => { name => 'Calibri (Body)',
                                        size =>  9,   bold => 0 },
                        min        =>     0,
                        max        => 10000 )
            set_y_axis( name       => 'Efficiency Loss',
                        name_font  => { name => 'Calibri (Body)',
                                        size => 10.5, bold => 1 },
                        num_font   => { name => 'Calibri (Body)',
                                        size =>  9,   bold => 0 },
                        num_format => '0.0%',
                        major_unit =>  0.1,
                        min        =>    0,
                        max        =>  0.4 )
            set_legend( none => 1 )
            insert_chart('AB3', $chart);
```

For the details of the methods, refer to Excel::Writer::XLSX::Charts. This creates an *embedded* chart (`add_chart (…, embedded => 1)`) on the table in which this chart definition occurs (eg. `[01_Data]`) placed on cell AB3 (`insert_chart('AB3', $chart)`). The placeholder `$chart`, referring to the current chart, must be added un-altered.

This may look like this:



5 Appendix A

5.1 Config File Syntax

5.1.1 Basic Format

The config file is a text file, consisting of sections and key-value pairs:

```
[GENERAL]
db     : solaranzeige              # comment
[01_Data]
drop   : { PV.Leistung_Str_3; }
```

The section `GENERAL` is mandatory and can very well be the only one present. The key values are described throughout this manual and are case sensitive.

A comment is anything from a hash `#` to the end of the line. There are *no* multi-line comments. Values can be scalars lists and maps.

Unfortunately, the config file parser is not too good in providing useful error messages – hence, if the program acts strangely, this is very likely because of a syntax error in the config file.

The only place where space is relevant is inside values (but not leading or trailing spaces to values). There is no escape character mechanism. Hence, if a value needs contain eg. the comment character `#`, this is not possible.

Sequence of key-value pairs within a section (and sequence of sections) is irrelevant.

5.1.2 List Values

```
drop        : { PV.Leistung_Str_3 ;
                PV.Spannung_Str_3 ;
                PV.Strom_Str_3
              }
```

In the above example, the key `drop` consists of a list of three values `PV.Leistung_Str_3`, `PV.Spannung_Str_3` and `PV.Strom_Str_3`. Note that the list is encapsulated in curly brackets (`{ }`) and the list elements separated by semicolons (`;`).

5.1.3 Map Values

```
calculate   : { Inverter_Loss_% => if(...) ;
                IV_Plus_1       => 'Inverter_Loss_%'+1 ;
              }
```

In the above example, the key `calculate` consists of map entries. As an example, the map source `Inverter_Loss_%` maps to target `if(...)`. Note that the map is encapsulated in curly brackets (`{ }`) and the map elements separated by semicolons (`;`). Source and target are separated by an arrow (`=>`). Legal target values cannot end with '`}`'

5.1.4 Sections

As stated above, the config file in any case must contain the mandatory `GENERAL` section. Other sections are optional and further detail what is to be done with data in a tab (table):

```
[GENERAL]
tables : { 01_Data => PV, AC ;
           02_PV   => PV
         }

[01_Data]
drop   : { PV.Leistung_Str_3; }
```

This will create two Excel tabs `01_Data` and `02_PV`. The field `PV.Leistung_Str_3` will only be dropped in table `01_Data`.

5.2 Installation

The script can be executed on the Raspberry where SolarAnzeige is installed, or on a Windows PC connected to the Raspberry. The former is easier to install but (depending on the PC used) considerably slower – can be a factor of 10x and more.

5.2.1 Installation on Raspberry (SolarAnzeige)

Current (v4.6x) SolarAnzeige image comes with perl v 5.28.1 pre-installed. Hence, all we need do is:

1. install two additional perl packages:

```
pi%solaranzeige: su                                          (supply password)
root@solaranzeige:/home/pi# cpan -i DateTime                 (...this takes long!)
root@solaranzeige:/home/pi# cpan -i Excel::Writer::XLSX
root@solaranzeige:/home/pi# exit
pi%solaranzeige:
```

2. create a sub-directory and put script files in there:

```
./Dump_InFluxDB.pl                                           (main program)
./Dump_InFluxDB/Dump.pm
./Dump_InFluxDB /J750/Config.pm
./config.ini
```

3. for a first trial, rename `config.ini` to a different name (so that it's not automatically picked up) and run

```
pi%solaranzeige: perl Dump_InFluxDB.pl
```

   As the Raspberry is relatively slow, this may take a ~4-5 minutes
4. copy the resulting `Solaranzeige_yyyy-mm-dd.xlsx` to a place where Excel is available for inspection.
5. Follow this Users Guide to configure to your needs.

5.2.2 Installation on Windows

1. Install perl on Windows. Most easy is Strawberry perl from http://strawberryperl.com/. Portable installation may be more convenient.
2. Install extra modules needed: From `cmd` window, run

```
cpan -i DateTime
cpan -i Excel::Writer::XLSX
```

3. create sub-directory and put script files in there:

```
./Dump_InFluxDB.pl                                    (main program)
./Dump_InFluxDB/Dump.pm
./Dump_InFluxDB /J750/Config.pm
./config.ini
```

4. Download and install `influx` from https://www.influxdata.com/products/influxdb-overview/. It should be sufficient to download the 1.x version .zip file and put `influx.exe` into the directory where `Dump_InFluxDB.pl` resides.
5. Edit config.ini (specially the entries `influx` and `host` in the `[GENERAL]` section) to your needs
6. Run the script with

```
perl Dump_InFluxDB.pl
```

   and inspect the resulting `SolarAnzeige_yyyy-mm-dd.xlsx` file
7. Follow this Users Guide to configure to your needs

5.2.3 Source code structure

- The source code is a perl script, consisting of
  - `./Dump_InFluxDB.pl (main program)`
  - `./Dump_InFluxDB/Dump.pm`
  - `./Dump_InFluxDB /J750/Config.pm`

  The script has been developed with perl 5.26.1, but should run under most perl 5.x versions. For database access, influx version 1.7.10 was used.

  Modules required which are not part of the standard perl distribution:

  - `Excel::Writer::XLSX`
  - `DateTime`

5.2.4 Compilation

The executable can be created with PAR::Packer under Strawberry perl 5.26

```
pp -x -o Dump_InFluxDB.exe DumpInFluxDB.pl
```

The packer reports

```
# Use of runtime loader module Module::Runtime detected.  Results of static
scanning may be incomplete.
# Use of runtime loader module Module::Implementation detected.  Results of
static scanning may be incomplete.
```

if run without the -x option. Hence, whether the resulting Dump_InFluxDB.exe actually works may depend on the config file used during compilation: Does it find all necessary modules to create all desired chart features?

5.3 Potential Improvements

In absence of a good use-case, combining multiple chart types in one (using `Excel::Writer::XLSX::Chart->combine()`) is currently not supported

It is also not possible to add series from multiple tables to one chart. This maybe useful for multi-controller setups of SolarAnzeige.

Number formatting could be made selectable per column. Currently, Excel 'General' format is used.

5.4 Known Bugs and Annoyances

5.4.1 Error and Warning Messages

Error checking (specifically for calculated columns and charts) is probably incomplete, leading to difficult to interpret output in case of incorrect `config.ini` entries.

5.4.2 Bugs and Annoyances

- No known bugs.
- Charts are considered an experimental feature at this time.
- Charts have fixed size in Excel chart tabs and don't re-size with Excel window. This is a problem with Excel::Writer::XLSX:Chart. Work-around: Once in Excel, move chart to another sheet with *Chart tools / Design / Move Chart*.

5.5 Revision History

Version	Date	Comment
1.00.00	2020-05-21	Initial, documented release