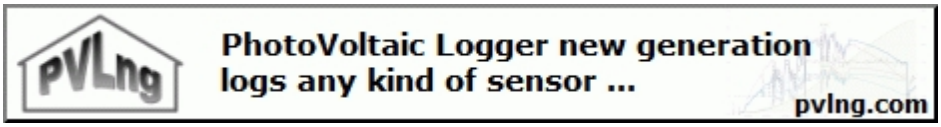
PVLng - PhotoVoltaic Logger new generation

Table of contents

Start



PVLng - PhotoVoltaic Logger new generation - 1.0.0

A logging and data warehouse system for measuring data of sensors and photovoltaic plants.

View the Project on GitHub	Download	Download
K-Ko/PVLng	ZIP file	TAR ball

Get support:	Google group	Issues tracker
	PVLng	Browse issues

---

last changed: 4/28/2013 5:17 PM

Introduction

A logging and data warehouse system for measuring data of sensors and photovoltaic plants.

The system acts as a data warehouse.

All data will be stored in an very simple and efficient internal structure (readings), optimized for data readout of any kind with a well defined interface.

This system addresses all the user interested in data analysis, which will have a deeper look into their logged data.

The system will accept data from any external sources and can deliver them for any purpose. A web front end for graphing is also integrated.

To use this system best, a deeper knowledge in server administration, shell scripting and probably PHP is recommended but not required.

The system is build using a channel logic which will described later in detail.

Each channel will handled by a specific model.

Depending of the installed base of our plant, the following components are still implemented:

1. Data storage

- Inverter data (feed-in power, DC power voltage, Current etc.) from a SMA Webbox
- Irradiation and module temperature from a SMA Sensorbox
- Outside and inverter temperatures via USB -> 1-Wire -> Temperature sensor
- Power consumption from watt meter via S0 impulses

2. Data readout

- Model for the PV-Log URL interface in JSON
- Script to push data to PVOutput
- Script to update a Twitter account

- [planned] Extractor witch creates Solar-Log compatible JS files, for example for Solar-Log homemade or SolarAnalyzer

Quick start

To get a 1st impression of the system, 2 fake channels are installed by default.

They deliver random data on each request.

To get a feeling with charts, you can use them to play around with the presentation settings.

RANDOM Temperature sensor

	RANDOM Temperature sensor	Random	15 ... 25, ±0.5

This sensor simulates a temperature sensor in °C with a valid range of 15 ... 25 °C.

Each "next" reading differs max. ± 0.5 °C from last one.

Parameter	Parameter value		Hint
Name	RANDOM Temperature sensor	*	Unique channel name
Description	15 ... 25, ±0.5		Long description
Serial number			Unique sensor serial number
Channel			Channel name for multi sensors
Resolution	1	*	Resolution for data readout
Unit	°C		Channel unit
Meter	○ Yes ◉ No *		Meter channels stores raising values
Numeric values	◉ Yes ○ No *		Channels have numeric or alphanumeric data?
Cost	0		Cost per unit, for meter channels only
Threshold	0.5		A reading is only accepted, if the value is +- threshold from last reading.
Valid from	15		Readings are only valid if they are greater or equal this value.
Valid to	25		Readings are only valid if they are lower or equal this value.

* Required

RANDOM Energy meter

	RANDOM Energy meter	Random	0 ... ∞, +0.2

This sensor simulates a energy meter with kilo watt hours readings, starting by 0 with (nearly :-) open end.

Each "next" reading differs max. + 0.2 kilo watt hour from last one.

The consumption cost 0.20 per kilo watt hour, because of the unit watt hours is it set to 0.0002

Please note: The readings here are in **kilo watt hours**, to get **watt hours** out, the **resolution** is set to 1000.

Parameter	Parameter value		Hint
Name	RANDOM Energy meter	*	Unique channel name
Description	0 ... ∞, +0.2		Long description
Serial number			Unique sensor serial number
Channel			Channel name for multi sensors
Resolution	1000	*	Resolution for data readout
Unit	Wh		Channel unit
Meter	(•) Yes ( ) No *		Meter channels stores raising values
Numeric values	(•) Yes ( ) No *		Channels have numeric or alphanumeric data?
Cost	0.0002		Cost per unit, for meter channels only
Threshold	0.2		A reading is only accepted, if the value is +- threshold from last reading.
Valid from	0		Readings are only valid if they are greater or equal this value.
Valid to	10000000000		Readings are only valid if they are lower or equal this value.

* Required

Installation

Please check the software and hardware requirements first.

At 1st find the document root of your web server, usually `/var/www/htdocs`, `/var/htdocs`, `/srv/www/htdocs` ...

If you want to run only PVLng on your server, just install PVLng direct into the document root.

If you have more running on your server, setup a own document root for PVLng.

Install with Git

If your system have Git installed, it will be simplest way to clone the Github repository.

Clone the repository directly **into the selected** directory:

```
$ git clone git@github.com:K-Ko/PVLng.git .
```

If you find than there a file `README.md`, your done.

Move forward to setup.

Install from ZIP file or TAR ball

FTP

Download an archive on your desk, extract it and upload the contents of directory `PVLng-master` into your document root.

Console

Just download your preferred archive and extract it direct on your server.

```
$ wget https://github.com/K-Ko/PVLng/zipball/master -O master.zip
$ unzip master.zip
$ mv PVLng-master/* .
$ rm -r master.zip PVLng-master
```

If you find than there a file README.md, your done.

Setup database

Create a database for PVLng if you want. I recommend this, but it is not required.

All PVLng specific tables have a `pvlng_` as prefix.

Import the SQL file `sql/install.sql` into your database.

From command line

```
$ mysql -u[username] -p[password] [database] <sql/pvlng.sql
```

This will install all the required tables, a predefined set of channel types, a view and some functions.

Web frontend

PVLng have Adminer installed by default.

> "Adminer (formerly phpMinAdmin) is a full-featured database management tool written in PHP. Conversely to phpMyAdmin, it consist of a single file ready to deploy to the target server."

```
http://your.domain.here/adminer/
```

Choose `SQL-Query > File upload > Execute`

Configuration

Database

Prepare the configuration file:

```
$ cp config/config.php.dist config/config.php
```

Fill in your database credentials.

Privacy

You have to protect your installation with an administration account.

Open

```
http://your.domain.here/adminpass
```

and update your `config/config.php` afterwards.

Scripts

Prepare the configuration file:

```
$ cp bin/PVLng.conf.dist bin/PVLng.conf
```

Fill in at least your API key, you can find it on the **Information** page.

Requirements

Software

- Operating system with support of periodic job scheduling (for example *nix with cron)
- MySQL ≥ 5.1.* (supports table partitions, views, trigger, functions, procedures and events)
- PHP ≥ 5.3.*
- Web server with PHP support

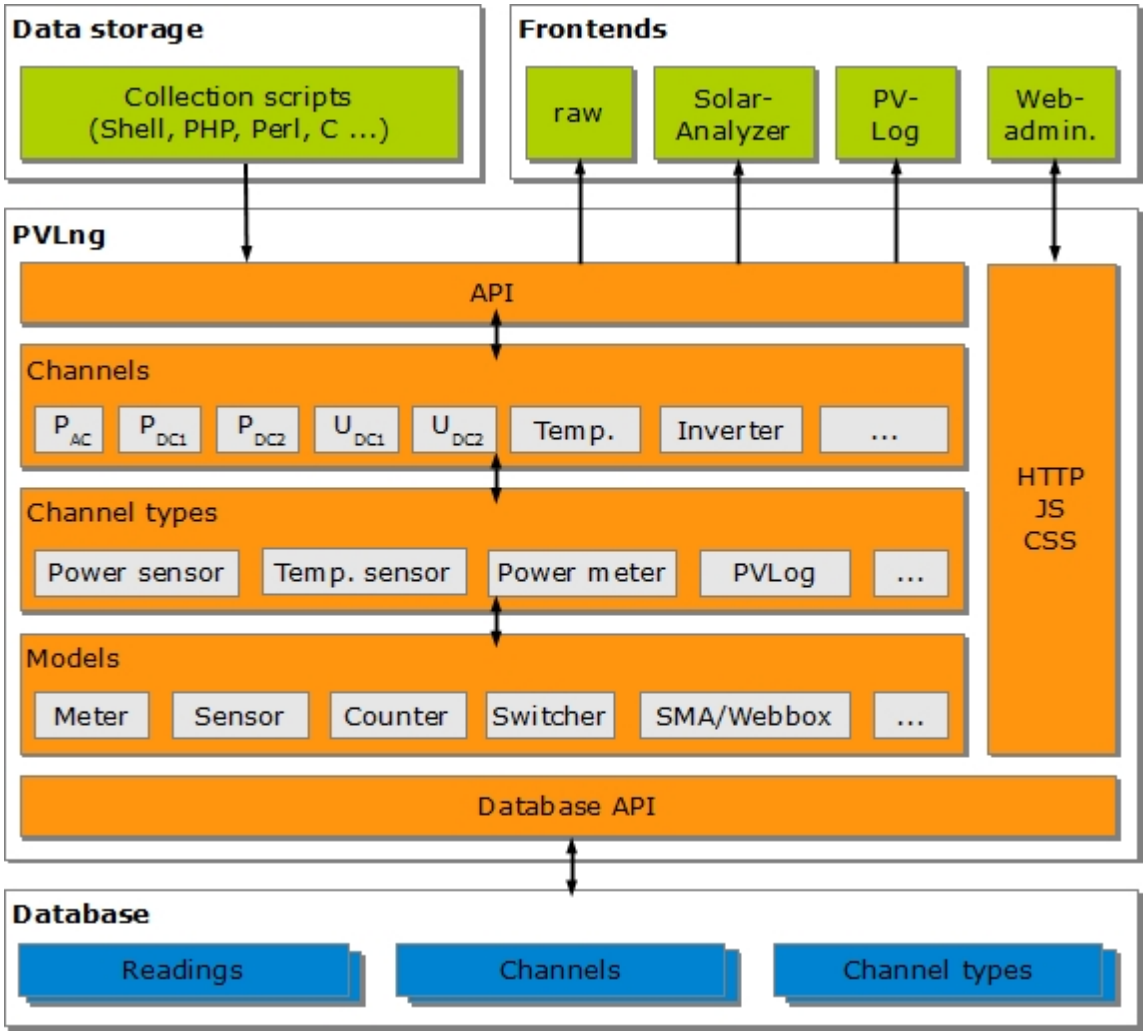
Hardware

- A server, which fits the software requirements :-)
- Some sensors, like 1-Wire sensors, watt meters
- Solar inverter which supports either
    - live query of data or
    - write data in short intervals (quasi-live) somewhere

Design & Concept

Software layers

The whole system consists of 3 essential layers:



- External systems, which handles
  - the data acquisition or
  - the data analysis
- PVLng as core component
- A MySQL database for data storage

External systems

The most data storage work must be done by external scripts. They obtain the measuring data from e.g. sensors and push them via API into the system.

In these scripts are simple to use functions defined, which pushes the data and handles errors.

Scripts to push your data to other sides are also available, for photovoltaic plants I recommend PV-Log or PVOutput.

The web front end handles all channel definitions, their attributes and the organization of channels in the channel tree and a charting module for simple visualizations.

PVLng layers

The system is build (bottom-up) by

- Models handles a very special kind of data/sensor
- Channel types implements a set set of predefined sensor types
- Channels represent concrete sensors
- Channel groups handles special needs about grouping or calculation/transformation

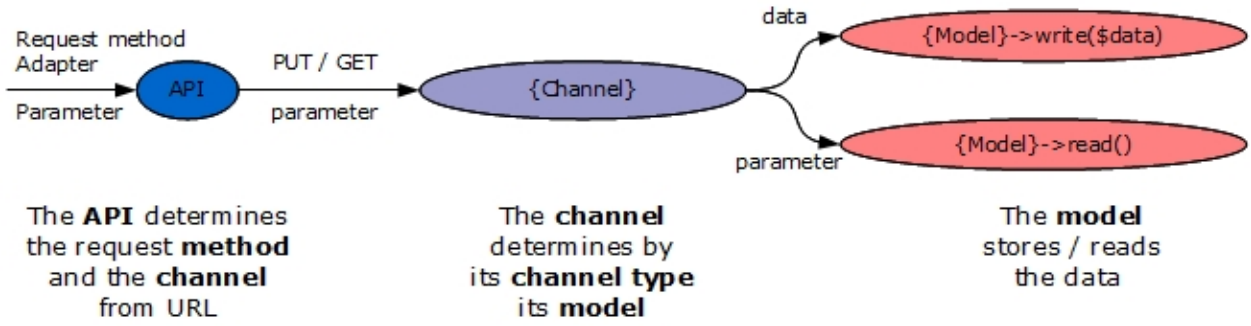
The following documentation will mainly describe the characteristics of PVLng and the other layers only if required.

Database

The database have 3 tables for channel organization and 2 tables for readings.

Data flow

The general information and data flow is like this:



Model

A model implements the direct interface to the database.

They are PHP classes below `core/Channel` directory.

There are in general 3 different model types:

- **Direct** channels which stores their data directly and can't have sub channels (e.g. Sensor, Counter, Switcher)
- **Proxy** channels with exact 1 sub channel for readout (e.g. SensorToMeter)
- **Grouping** channels which merges always sub channel data (e.g. Accumulator)
  (Sub channels can be all kind of type again)

A model implements in their sources the following characteristics and interface methods.

General characteristics

Read

All direct and proxy channels are readable to query their readings.

All grouping models are readable (except SMA/Webbox model)

(This attribute is held redundant in database for better performance on feature requests.)

Write

All direct channels are writable to store readings.

No grouping model is writable (except SMA/Webbox model)

(This attribute is held redundant in database for better performance on feature requests.)

Direct models

Direct models implement the **direct** data access. They don't care about child channels on data storing or readout.

Sensor

A sensor is the generic model for universal use, it stores any value as is.

Counter

A counter stores impulses from sensors, for example S0 impulses from watt meters.

The real "data" will be during readout calculated by the time difference between the readings and the defined resolution.

Switcher

A switcher model will store only readings, if the actual value is different from the last stored reading.

Switching sensors are for example an inverter states or valve states.

Proxy models

SensorToMeter

This model type acts as a proxy and requires exact one sub channel.

It accept only channels, which are based on models "Sensor" or "Counter"!

Grouping models

Grouping models delegate **all** data access during storing or readout to their sub channels.

Accumulator

An accumulator model accepts only childs of the **same channel type**

It sums up all data from the sub channels for the same timestamp.

It is used for example to combine all single string powers ($P_{DC?}$) into one P_{DC}

Differentiator

An differentiator model accepts only childs of the **same channel type**

It subtract from the 1st sub channel all data from the other sub channels for the same timestamp.

It skips all timestamps, where at least one reading is missing.

It is used for example to calculate the difference between outside temperature and modules temperature.

DifferentiatorFull

This model is similar to the Differentiator**,** but it **ignores** missing readings and handle them as 0 (zero), for example a power balance.

Refer to this example for more details.

InternalConsumption

> "Internal consumption refers to the amount of PV energy consumed where it is generated. Therefore the internal consumption rate equals the portion of the total PV energy generated that is used for internal consumption." -- SMA

This channel type requires exact **2 sub channels** with:

- **1st** sub channel: Power **consumption meter**
- **2nd** sub channel: Power **production meter**

The model checks the difference between consumption and production and works as this:

- If the Consumption is higher than production => full or partly energy import
- If the production is higher than consumption => no energy import, partly energy export

See this example for channels and charts definition.

Ratio

A ratio calculator calculates a ratio between **exact 2** sub channels.

It skips all timestamps, where one reading is missing.

It is used for example to calculate a inverter efficiency by using the P_{AC} and the (summarized) P_{DC}.

Specials

For special usage are the following models are implemented:

- SMA/Webbox
- PV-Log
- Kaco1
- Kaco2

SMA/Webbox

A special model to split Webbox RPC content into single channel values.

This model is in contrast to other grouping models writable but not readable!

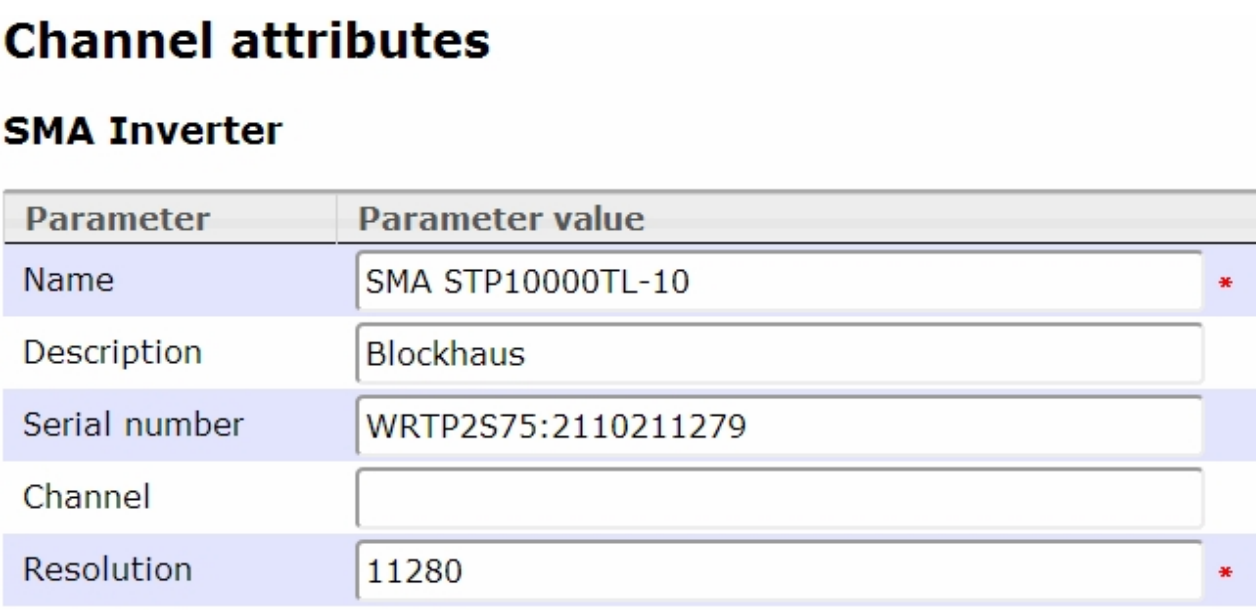
Operation mode

The SMA Webbox model handles JSON outputs from Webbox RPC requests.

It scans the data and search its **direct** sub channels for matching channel definitions!

Setup

Create a channel for the Inverter



Please note:

- The **serial number** is used from external upload script to ask the SMA Webbox for the correct equipment.
- When the serial number **changes** after **firmware updates, don't** forget to update the serial number here!

Special use for this channel type is the resolution. By default, the resolution is only used for reading data.

But as you know, the SMA inverter is a not readable group... That is correct, BUT the attributes are always readable!

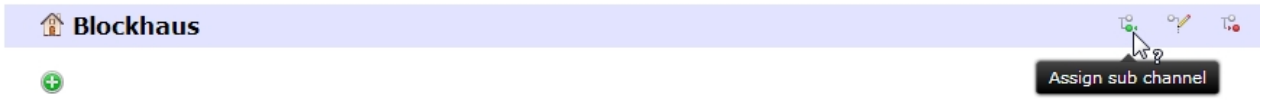
In this case, the resolution should held the **installed power in watt peak** on this inverter.

So it is possible to calculate the relative yield of the inverter (watt or watt hours per watt peak) by using a sub channel and this resolution.

(This is done for example by the Twitter update script)

Insert the inverter into the channel tree

If you have more than 1 inverter, it could make sense for a better overview to group them, e.g. by building or location.



Result:

Define the inverter channels according to your inverter

Create all further needed channels, e.g. P_{AC} for a STP series inverter:

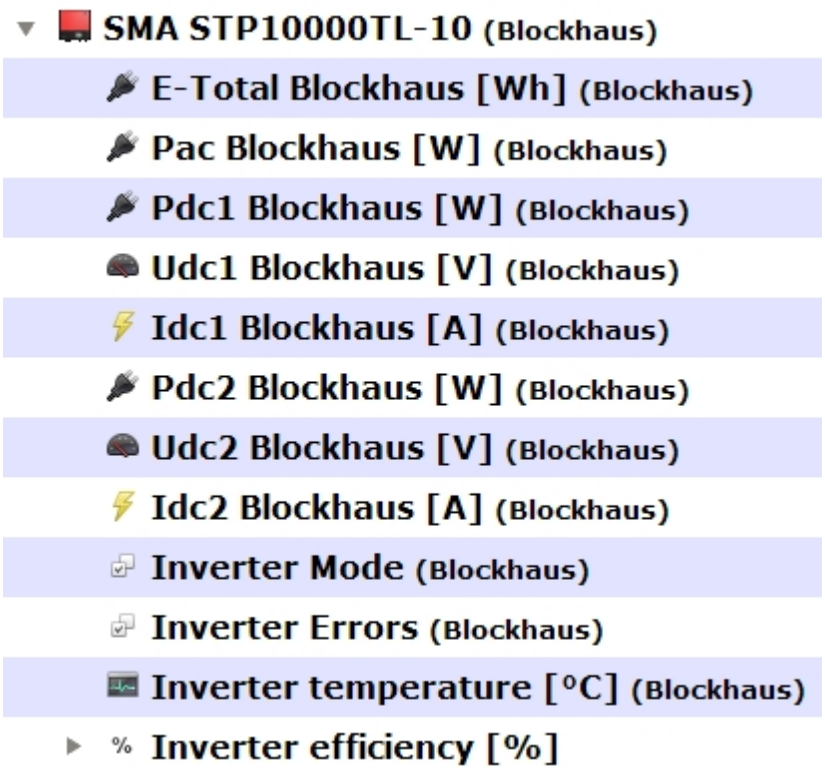
Channel attributes

Power sensor (W)

Parameter	Parameter value
Name	Pac Blockhaus *
Description	Blockhaus
Serial number	
Channel	Pac
Resolution	1 *
Unit	W
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *
Cost	0
Threshold	0

Please note: Use the exact channel name as your inverter delivers. This is needed to find the correct channel to store readings for!

Insert all the channels as sub channels of the inverter in the channel tree



A special case here is the Inverter efficiency.

STP channels

These channels are provided by the SMA Tripower inverter series STP*.

- A.Ms.Amp
- A.Ms.Vol
- A.Ms.Watt
- A1.Ms.Amp
- A2.Ms.Amp
- A3.Ms.Amp
- A4.Ms.Amp
- A5.Ms.Amp
- B.Ms.Amp
- B.Ms.Vol
- B.Ms.Watt
- B1.Ms.Amp
- Error
- E-Total
- GridMs.Hz
- GridMs.PhV.phsA
- GridMs.PhV.phsB
- GridMs.PhV.phsC
- GridMs.TotPFPrc

- `Inv.TmpLimStt`
- `InvCtl.Stt`
- `Mode`
- `Mt.TotOpTmh`
- `Mt.TotTmh`
- `Op.EvtCntUsr`
- `Op.EvtNo`
- `Op.GriSwStt`
- `Op.TmsRmg`
- `Pac`
- `PlntCtl.Stt`
- `Serial Number`

PV-Log

These models provide data for import into PV-Log in JSON via URL.

PV-Log supports "installations" with multiple inverters.

Please note: If you have more than one inverter in your plant with different modules, strings, direction or roof pitches, you should split them into multiple "installations".

The system supports both scenarios and can deliver either whole plant data or single inverter data.

It is handled by 2 models:

- `PVLog\Plant`
- `PVLog\Inverter`

The plant model acts as a super group combining all its inverter data.

URL for the HTTP GET requests:

```
http://your.domain.here/api/r1/[GUID]/YYYY-MM-DD.json
http://your.domain.here/api/r1/[GUID].json
```

Use either your "plant" or the "inverter" GUID. If no date is provided, today is used by default.

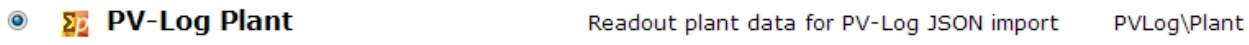
Please note: Don't forget the required `.json`!

Setup channel tree

Start the definition with the plant, which will combine all your inverters.

Plant

Create a new "PV-Log plant" channel by selecting channel type "PV-Log plant"



and maintain it like this:

Channel attributes

PV-Log Plant

Parameter	Parameter value
Name	KKoPV *
Description	PV-Log

Inverter

Create next an inverter channel the same way using channel type "PV-Log Inverter" and assign it afterwards as sub channel to the plant.

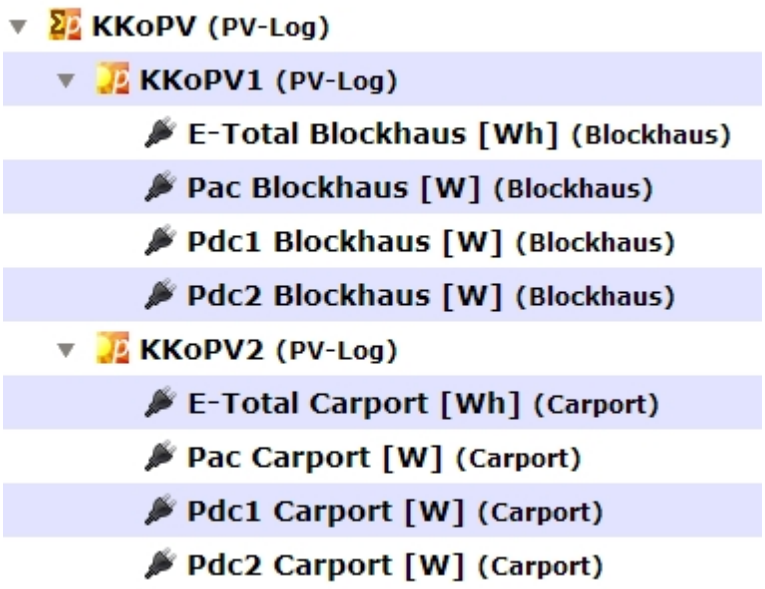
Data channels

Now the inverter channel must have **at least** 2 sub channels with following conventions:

- **1st sub channel**: Total or today production of the inverter (must have "consumption" parameter set)
- **2nd sub channel**: PAC power of the inverter
- **3rd and following**: PDC string powers (optional)

Please note: These requirements can't really checked by the model during data readout, a wrong definition it will result in wrong results!

The whole definition for our installation (2 inverters with 2 strings each) look like this:

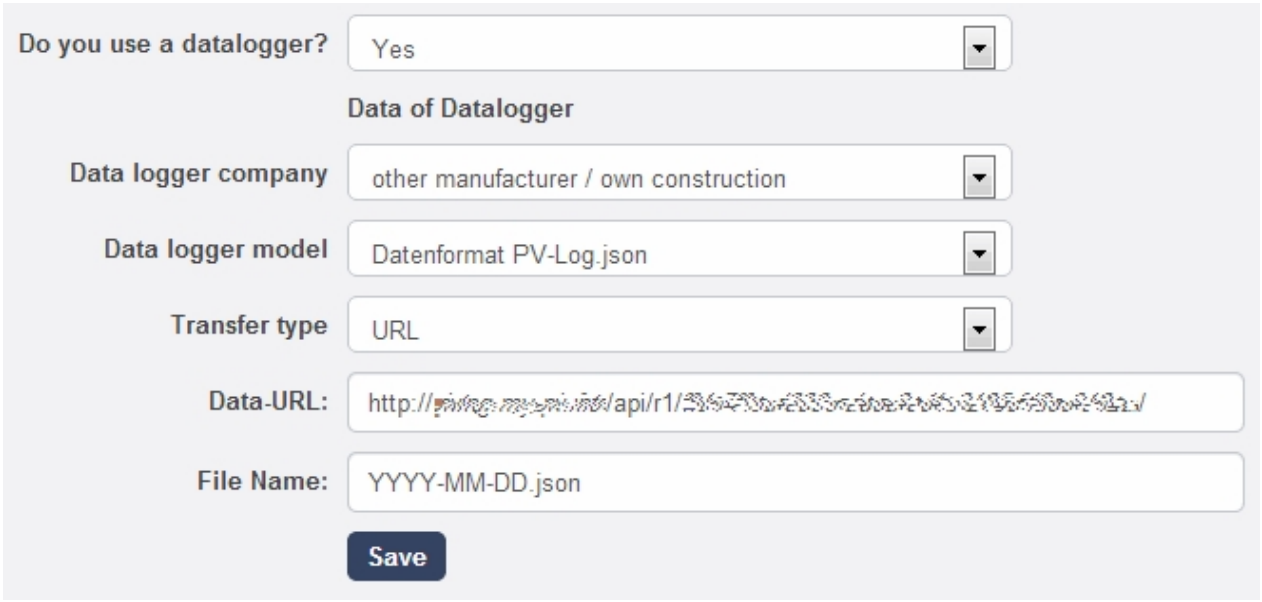


The corresponding "plants" on PV-Log:

- http://www.pv-log.com/photovoltaikanlage-kkopv
- http://www.pv-log.com/photovoltaikanlage-kkopv1
- http://www.pv-log.com/photovoltaikanlage-kkopv2

Setup PV-Log

- Open your profile page
- Create a plant with "Add new PV installation"
- In the plant section open "Yield data: Automatic data import / Data logger"
- Make the following settings (with **your domain** and **GUID**):



Examples for fields and formats on PV-Log

http://www.pv-log.com/Resources/Public/Upload/1-2012-04-20-pvlog.json

Check your definitions

Please check your PV-Log definition (Data-URL + File Name) by using a service like http://jsoneditoronline.org/, just use `Open > Open URL`

Fetch your URL and check for completeness and correctness!

If you use Chrome, I recommend POSTman - REST client or Advanced Rest Client

Kaco 1 & 2

Kaco 1

ToDo - NOT YET implemented

The Kaco inverter with a RS-232 interface provide a semicolon separated string with the operating data.

```
<Date>;<Time>;<Operating mode>;<Generator voltage>;<Generator current>;<Generator power>;<Mains voltage>;<Mains current>;<Feed-in power>;<Temperature>
```

Define the inverter with sub channels as this:

- Date and time will be ignored
- Fill the "Channel" attribute for a sub channel with the Id (zero based) of the value Id.

For example, the "Generator power" is the 6th value and have therefor Id 5

Power sensor (W)

Parameter	Parameter value
Name	Pac *
Description	Source: Kaco 1
Serial number	
Channel	5
Resolution	1 *
Unit	W
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *

Kaco 2

ToDo - NOT YET implemented

The Kaco inverter with a RS-485 interface provide a semicolon separated string with the operating data.

```
<Inverter Id>;<Operating mode>;<Generator voltage>;<Generator
current>;<Generator power>;<Mains voltage>;<Mains current>;<Feed-in
power>;<Temperature>;<Diurnal energy>;<Inverter type>;<Check sum>
```

The inverter Id here must be defined as the serial number in the master data of the inverter!

The channel definition is the same as for Kaco 1:

For example, the "Generator power" is the 5th value and have therefor Id 4

Power sensor (W)

Parameter	Parameter value
Name	Pac *
Description	Source: Kaco 2
Serial number	<Inverter Id>
Channel	4
Resolution	1 *
Unit	W
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *

Channel type

A channel type uses a model to handle the write / read requests.

Many channel types can be based on the same model.

Channel types are defined in database table `pvlng_type`.

General characteristics

The **read** and **write** characteristics are primarily defined by the model but they are doubled into database for performance.

Name, Description

Each channel type is uniquely characterized by a name and a description.

Model

The model which will handle the read / write requests.

Unit

This is the unit which will be proposed on channel definition for orientation

Childs

Defines how many childs this type will accept.

- Channel type based on single models will not accept childs at all: `childs = 0`.
- Channel type which accepts only a specific number of childs, will have a positive number, e.g. `childs = 1`
- Channel types, which accept any number of childs must have `childs = -1`

Graph

This flag define, if the channel is available for the internal graphing module.

By default, all readable channels can be used for graphs.

But there are also channel groups, which are readable, but can't be used for graphing: PV-Log

Icon

An icon file name to mark the channels of this type.

The icons are located in `images/ico`

Channel

A channel is the interface of PVLng to the outer world.

Many channels can be based on the same channel type.

Channels are defined in database table `pvlng_channel`.

You can **really** delete a channel only, if it is **not assigned** in channel hierarchy!

ATTENTION

*If you delete a channel, **ALL** readings will be cleared from database forever (unless you have a backup :-)!*

Characteristics

GUID

All channels are accessed by its unique GUID in form of `e107-c8b1-d4b4-aaa6-5e56-e003-ea14-4705`

Direct channels

The GUID is assigned during channel creation on channel level.

Such a channel can be assigned to more than one group in during channel grouping and must be found uniquely independent from its position.

For example each inverter have a "Actual power" channel (based on the "Power sensor" channel type).

Grouping channels

The GUID is assign during channel grouping.

Such a channel can be used multiple in channel tree, but it must be able to find its sub channels according to its position in the channel tree.

For example a "Inverter" channel (based on the "Inverter" grouping channel type) can be used multiple in the

channel tree, but with different sub channels.

Please refer to the grouping examples for details.

Name, Description

Each channel is uniquely characterized by a name and a description.

The combination of name and description must be unique for each channel.

Serial

If the equipment have a serial number, store it here for reference.

For SMA inverter and sensorbox and 1-Wire sensors is this field required!

Channel

Define here the channel name for equipments with more than one channel.

See SMA inverter and sensorbox and 1-Wire sensors.

For 1-Wire sensors is this field required!

Numeric

The system can handle numeric and alphanumeric readings.

Alphanumeric readings are for example the inverter mode and inverter error channels.

Resolution

(For numeric channels only)

Multiply the reading during readout with this resolution, default 1.

Unit

(For numeric channels only)

Unit for display in internal graphing module.

Meter

All numeric channel types can be handled as meters.

A meter stores in every case raising values, this means a new reading to be stored can't be smaller than the last stored reading, for example watt meters.

Some grouping channels support only meter channels for correct working.

Cost

Cost per unit.

Please refer to this example.

Threshold

Defines a max. threshold between reading values.

If the difference between the actual and the last reading is grater than the threshold, the reading will be ignored to avoid wrong peaks.

(For example a temperature should change not more than ? °C per minute.)

Valid from, Valid to

Beside the threshold also a valid value range can be defined. All readings have to be inside the range incl. boundaries.

(For example my irradiation sensor reports at night sometimes 1 W/m² "moon power", so I set the range of 2 ... 5000)

1-Wire sensor

1-Wire sensors need a special setup.

For a 1-Wire sensor reading **2 information** needed:

- Sensor serial
- Sensor channel

For correct handling from script in `bin/owfs` setup your sensor like this:

Temperature sensor (°C)

Parameter	Parameter value
Name	Outside temperature *
Description	
Serial number	28CA2307040000E0
Channel	temperature
Resolution	1 *
Unit	°C

Channel grouping

Channels can be grouped for several usages.

General

One type of grouping is for general view. So you can group things by location or building.

Sensors with more than channel should be grouped also.

Assume a solar inverter as a sensor with many channels...

Calculation

The other kind of grouping is for calculations, like summarized data.

The calculation will be done recursive according to the grouping definition.

To avoid memory overflow, all this is done via temp. files.

So it is for example possible to extract all data at once, independent from amount of readings!

See the examples for details.

Operating instructions

Some hints about usage the web front end.

Language

The initial language is detected by the preferred browser language, but you overwrite this with

At the moment English and German are implemented.

If you are interested in translating into another language, give me a hint, but there are about 220 texts to translate...

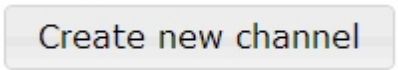
Overview

In the overview you build your the channel hierarchy.

Before you can add a channel to the hierarchy, you to define them on the Channels view.

- Add channels on top level with
- Add sub channels to existing grouping channels with
- Delete single channels with and grouping channels incl. childs with
  This removes these channels **only** from hierarchy!
- Channels are always appended at bottom.
  Rearrange them with
- To get an better overview, you can expand/collapse the whole hierarchy by
- If you collapse (/) certain knots, the state will be saved and restored on next view.
- If you need the channel GUID for external use, just click on the **channel name**

- You can create new channels on Channels view with



Channels

Here you have a sortable list of all your defined channels, initially sorted by channel type.

You can:

- Edit a channel with

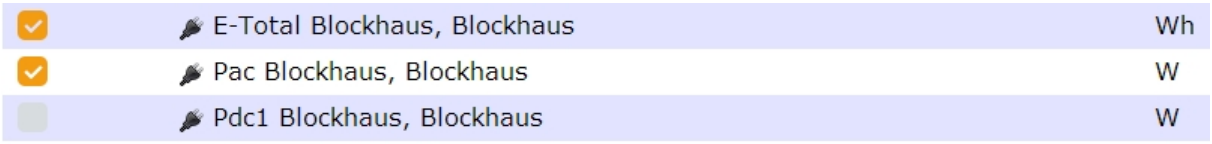
- Clone a channel with , it presents the "Add channel" form filled out with the attributes of the template.
- Delete a channel with
  This only works, if the channel is **not assigned** in channel hierarchy!

> ***ATTENTION***
>
> *If you delete a channel, **ALL** readings will be cleared from database forever (unless you have a backup :-)!*

Charts

Charts view starts plain until you mark channels for charting.

- Mark some channels



  and consider the units you combine!

- Customize the series with

  Please note: Put ***different units*** to ***different axis***!

  You need so much axes, as you have different units! (see Examples)

- If you're done click "Refresh"

- Move the displayed time range backward / forward with the arrows

  The inputs are linked, a click will change them both. The chart will be refreshed automatic!

- To collapse all not selected channels, click

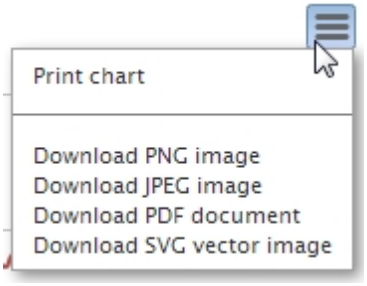
- If you're fine with your selection and settings, save this chart variant for later

- You can **Load** it back, if you made changes or **Delete** it, if it is obsolete.

- For fast access to a defined variant you can save a bookmark from 

  This button will always link to the selected variant in the variant drop down!

- If you like to use your chart somewhere else, you can get / extract it



Information

The most important information is your **API key** for data storing requests.

Keep it **secret** to avoid unwanted data manipulation!

If you afraid, your key was compromised, you generate a new one on here.

Please note: If you generate a new API key, **don't forget to update** any place, where the API key is used for data storage, e.g. in your script configuration in `bin/PVLng.conf`

On this view you can see also some stats about your stored data.

Application Programming Interface (API)

> "An application programming interface (API) is a protocol intended to be used as an interface by software components to communicate with each other." -- Wikipedia

The API is implemented as a RESTful API, so it uses HTTP requests to store and read data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.

The following request methods are supported by PVLng:

- `PUT` = create
- `GET` = read
- Update (`POST`) and deletion (`DELETE`) of data is **not** supported by this API!

Data storage

For data storage we have two possibilities:

- Web front end for the master data, like channels and their relationships
- HTTP REST API for the operational channel data with HTTP PUT requests

The storage API accepts only one parameter "data" and routes it to the requested channel.

By default (load live data) system date and time will be used. The time will be rounded down to the next full minute to align the readings.

The model behind the channel represents the interface to the database.

Data readout

The data readout from the system must be done with HTTP GET requests.

The readout API analyzes the requests, identifies the channel and returns the extracted data.

The model behind the channel provides a defined interface to query data.

Store data

- Supports only `PUT` request method for data storage
- The **API key** of your installation is **required** for security reasons for every storage request.
  The key have to be send by HTTP PUT requests as so called **X-Header** with the name `X-PVLng-key` (see your information site in the web front end)
- Floating point numbers must not contain a thousand separator and a dot as decimal separator.
- Each request must contain the channel GUID in the URL.

URL

```
http://your.domain.here/api/r1/[GUID]
```

Data

The data to store have to be provided as parameter `data`

```
data="123.45"
```

For concrete build of the requests see the examples.

Batch

The data to store have to be provided as parameter `batch`

This parameter consists of up to 100 data sets (this should be enough in most cases :-)

- A data set contains **comma** separated values for
  - **date** (YYYY-MM-DD / YYYYMMDD)
  - **time** (HH:mm / HHmm)
  - **value** (with decimal **dot**)
- Data tuples must be separated by **semicolon**.

To store 2 reading values:

```
batch="1012-01-01,00:00,123.45;2012-01-01,00:05,123.67"
```

The count of inserted readings is returned in the response body on success.

Scripts

There are working samples which handles data from

- 1-Wire sensors
- Energy meters with S0 interface
- SMA Webbox data

All scripts lcated in `bin/`

OWFS

Scripts located in `bin/owfs/`

Installation

The script needs a correct installed OWFS on your system.

Please refer to the specific installation guides for your system.

Setup

Define your channels as described.

```
$ cp bin/owfs/owfs.conf.dist bin/owfs/owfs.conf
```

Edit config file and fill in all your channel GUIDs.

Test your config with

```
bin/owfs/owfs.sh -t
```

To make the test more verbose use `-tv`

Debian

For Debian is no stable package for Squeeze available, so you have to install it from source.

You can use the included `bin/owfs/owfs-init.sh` on system start.

Copy the file into `/etc/init.d` and run `update-rc.d owfs.init.sh defaults`

S0 reader

Scripts located in `bin/S0/`

To read S0 impulses from e.g. energy meters, a program and a script to store this into PVLng is included.

Please refer 1st to the hardware requirements sections.

Please put the device where your energy meter is connected to into the "channel" attribute like this:

Power sensor (W)

Parameter	Parameter value
Name	Power consumption *
Description	Blockhaus
Serial number	B-Watt Swissnox / S0
Channel	/dev/usb-ftdi-1
Resolution	1 *
Unit	W
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *

HowTo: Define a "named" device

Mode of operation

An energy meter with S0 interface sends depending of the actual power usage an amount of impulses per kilo watt hour.

A common frequency is 1000 or 2000 per kilo watt hour, or 800 as for our energy meter.

The S0 binary listen the device and detects the time between these impulses. It converts them to power usage in watt.

Installation

Go to `bin/S0` and run

```
$ ./configure
```

Make and install the S0 binary into `/usr/local/bin` with

```
$ make && sudo make install
```

Configuration

Please refer to `S0.conf.dist` to configure your environment regarding interfaces and channels.

```
$ cp bin/S0/S0.conf.dist cp bin/S0/S0.conf
```

Test

To test your configuration, run

```
$ ./S0.sh -t
```

Run with cron

Edit your crontab with

```
$ crontab -e
```

and add the following

```
# run each 2 minutes
*/2 * * * * /path/to/bin/S0.sh
```


Hardware requirements

Energy meter

You need a energy meter with S0 interface, best for 3 phases.

B-Watt Swissnox 3x5(80A)

Serial port

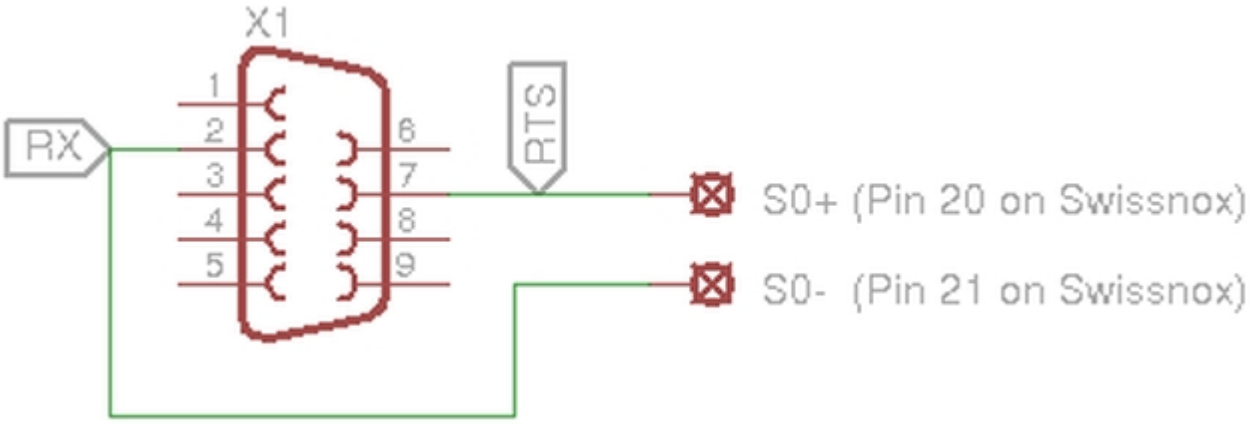
If your computer not more have a serial port (like mine), you need a USB serial adapter.

The best choice is an adapter with a FTDI / **FT232RL** chip set.

Digitus DA-70156 USB to Serial Adapter, USB 2.0

Connect

Connect your S0 port like this to your serial interface:



During prototyping is a gender changer (female/female) recommended.

Or you recycle an old serial mouse cable...

SMA Webbox

Scripts located in `bin/SMA/`

Setup

```
$ cp #bin/SMA/Webbox.conf.dist bin/SMA/Webbox.conf.dist
```


Data readout

- Support for request method **`GET`**
- Each request must contain the **channel GUID** in the URL.

URL

```
http://your.domain.here/api/r1/<GUID>.<format>
```

Format

Supported formats are:

- `json`
- `csv` (Semicolon separated values)
- `tsv` (TAB separated values)
- `xml`

If one of the following "Accept" HTTP header is provided with the call, the format in the URL is optional and will be overwritten by this!

- `application/json`
- `text/json`
- `application/csv`
- `text/csv`
- `text/comma-separated-values`
- `application/tsv`
- `text/tsv`
- `text/tab-separated-values`
- `application/xml`
- `text/xml`

Parameters

The following parameters describes the requested data.

Parameter	Name	default	Format	since
start	Starting time stamp	00:00	string|integer	r1
end	End time stamp	24:00	string|integer	r1
period	Aggregation period		(last|minute|hour|day|week|month|quarter|year)	r1
full	Return full data, see below		(0|1)	r1

The selection range for the data is build by: `start ≤ timestamp < end`

start, end

The "start" and the "end" can be provided as

- A string "YYYY-MM-DD hh:mm:ss"

- Relative from now (see below)
- In seconds since 1.1.1970 00:00:00 UTC

The definition can also be done with PHPs Relative Formats syntax. Using this, the time is set always to midnight!

Example: Query the data from last month

```
start=first day of last month&end=first day of this month
```

period

Special periods are

- [blank] = raw data, all lines between `start` and `end` (default)
- `last` = one (last) line of the whole result

To multiply the period, add a number:

- `12hour` = half day
- `30minute` = half hour
- `10year` = decade

Please note the difference between aggregation of "hour" and "minute":

- Hour always covers the beginning of an hour (??:00)
- Minute always work exact from "end" parameter time stamp backwards

What does it mean?

Let me show an example. The following results looks like the same: ("end" == now == 12:19)

But the left one (hour) is aligned to **`??:00 & ??:30`** and the right one (minute) exactly to **`??:50 & ??:20`**

(To get the readable date time, we request the `full` result.)

```
http://.../<GUID>.csv?
period=0.5hour&full=1
```

```
2013-03-28
10:00:00;1364461200;398.06666667
;384;412;15;1680
2013-03-28
10:30:00;1364463000;480.2;411;56
8;15;1680
2013-03-28
11:00:00;1364464800;634.73333333
;520;745;15;1680
2013-03-28
11:30:00;1364466600;789.8;482;91
```

```
http://.../<GUID>.csv?
period=30minute&full=1
```

```
2013-03-28
09:50:00;1364460600;378.73333333;
324;401;15;1680
2013-03-28
10:20:00;1364462400;435.33333333;
407;502;15;1680
2013-03-28
10:50:00;1364464200;578.2;518;745
;15;1680
2013-03-28
11:20:00;1364466000;776.66666667;
```

2;15;1680

649;846;15;1680

2013-03-28 **12:00:00**;1364468400;834;749;886;10;1080

2013-03-28 **11:50:00**;1364467800;807.06666667;482;912;15;1680

full

See below at the result description.

Result

By default, the 1st line / array element will contain the channel attributes and further rows the following fields.

Data are not rounded at all, they are delivered as stored or calculated.

By default, only the **Timestamp** and the **Data** column are returned. If you set the request parameter `full=1`, **all** the following data are returned.

Date time

A readable timestamp. Mostly used for debugging.

Timestamp

UTC Timestamp, good for direct use in graphing.

Data

Reading value for this timestamp.

Min, Max

Minimal / maximal value during aggregation period.

Count

Count of aggregated values between returned rows.

Time range

Time range of the aggregation in seconds.

JSON

Default

Request

```
http://your.domain.here/api/r1/{GUID}.json
```

Result


```
[
    {
        "counter": 0,
        "start": 1365372000,
        "end": 1365458400,
        "id": 135,
        "entity": 19,
        "guid": "{GUID}",
        "name": "Outside temperature",
        "serial": "28CA2307040000E0",
        "channel": "temperature",
        "description": "",
        "resolution": 1,
        "cost": 0,
        "meter": 0,
        "numeric": 1,
        "unit": "°C",
        "threshold": 5,
        "type": "Temperature sensor",
        "model": "Sensor",
        "childs": 0,
        "read": 1,
        "write": 1,
        "icon": "application_monitor.png",
        "consumption": 0,
        "costs": 0
    },
    {
        "timestamp": 1365372000,
        "data": 2
    },
    {
        "timestamp": 1365372120,
        "data": 1.8125
    },
    ...
]
```


Full

Request

```
http://your.domain.here/api/r1/{GUID}.json?full=1
```

Result

```
[
    {
        ...
    }
    {
        "datetime": "2013-04-08 00:00:00",
        "timestamp": 1365372000,
        "data": 2,
        "min": 2,
        "max": 2,
        "count": 1,
        "timediff": 0,
        "consumption": 0
    },
    {
        "datetime": "2013-04-08 00:02:00",
        "timestamp": 1365372120,
        "data": 1.8125,
        "min": 1.8125,
        "max": 1.8125,
        "count": 1,
        "timediff": 0,
        "consumption": 0
    },
    ...
]
```


CSV / TSV

Default

Request

```
http://your.domain.here/api/r1/{GUID}.csv
http://your.domain.here/api/r1/{GUID}.tsv
```

CSV Result

```
0;1365372000;1365458400;135;19;{GUID};Outside
temperature;28CA2307040000E0;temperature;;1;0;0;1;°
C;5;Temperature
sensor;Sensor;0;1;1;application_monitor.png;0;0
1365372000;2
```

```
1365372120;1.8125
...
```

Full

Request

```
http://your.domain.here/api/r1/{GUID}.csv?full=1
http://your.domain.here/api/r1/{GUID}.tsv?full=1
```

CSV Result

```
0;1365372000;1365458400;135;19;{GUID};Outside
temperature;28CA2307040000E0;temperature;;1;0;0;1;°
C;5;Temperature
sensor;Sensor;0;1;1;application_monitor.png;0;0
2013-04-08 00:00:00;1365372000;2;2;2;1;0;0
2013-04-08 00:02:00;1365372120;1.8125;1.8125;1.8125;1;0;0
...
```


XML

Default

Request

```
http://your.domain.here/api/r1/{GUID}.xml
```

Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<data>
    <attributes>
        <counter>0</counter>
        <start>1365372000</start>
        <end>1365458400</end>
        <id>135</id>
        <entity>19</entity>
        <guid>{GUID}</guid>
        <name>Outside temperature</name>
        <serial>28CA2307040000E0</serial>
```

```
        <channel>temperature</channel>
        <description></description>
        <resolution>1</resolution>
        <cost>0</cost>
        <meter>0</meter>
        <numeric>1</numeric>
        <unit>°C</unit>
        <threshold>5</threshold>
        <type>Temperature sensor</type>
        <model>Sensor</model>
        <childs>0</childs>
        <read>1</read>
        <write>1</write>
        <icon>application_monitor.png</icon>
        <consumption>0</consumption>
        <costs>0</costs>
    </attributes>
    <node>
        <timestamp>1365372000</timestamp>
        <data>2</data>
    </node>
    <node>
        <timestamp>1365372120</timestamp>
        <data>1.8125</data>
    </node>
    ...
</data>
```

Full

Request

```
http://your.domain.here/api/r1/{GUID}.xml?full=1
```

Result

```
<?xml version="1.0" encoding="UTF-8" ?>
<data>
    <attributes>
        ...
    </attributes>
    <node>
        <datetime>2013-04-08 00:00:00</datetime>
        <timestamp>1365372000</timestamp>
        <data>2</data>
        <min>2</min>
        <max>2</max>
        <count>1</count>
        <timediff>0</timediff>
```

```
        <consumption>0</consumption>
    </node>
    <node>
        <datetime>2013-04-08 00:02:00</datetime>
        <timestamp>1365372120</timestamp>
        <data>1.8125</data>
        <min>1.8125</min>
        <max>1.8125</max>
        <count>1</count>
        <timediff>0</timediff>
        <consumption>0</consumption>
    </node>
    ...
</data>
```


Return codes

The status code will be doubled into the response header and the response body.

This offers the possibility to analyze either the header or the body to check for success or errors.

For every request the response body will contain further information, especially in case of an error.

The following HTTP status codes are used by the API.

200 - OK

- Read of data was successful
- Data was error free, but not saved.
    - Insert data within 1 minute interval will be ignored.
    - Not changed Switcher states will be ignored

201 - Created

- Data was successful saved.

On batch loads the response body contains the count of inserted data sets.

400 - Bad Request

- There was a problem reading the data

See response body for details.

401 - Unauthorized

- Missing or wrong API key for PUT requests

405 - Method Not Allowed

- Only PUT / GET are allowed

500 - Internal Server Error

See response body for details.

Examples

All examples assume a *nix system with installed curl.

- Data storage
- Data readout
- Channel grouping
  - Inverter efficiency
  - Energy import
- Charts

Data storage

As example we will store a temperature.

Parameter	Value	since
API key	a10d3cca-370c-121e-b79c-2f0a58030a01	r1
Channel GUID	87fd-cc15-cb33-71a8-6b78-3855-4c11-a79e	r1
Temperature	23.45 °C	r1

Insert data

To store data we send a PUT request from a console with curl.

Interesting is the HTTP code returned and the response body for error analysis in case of HTTP code is not 200 or 201.

```
$ set $(curl --request PUT \
         --write-out %{http_code} \
         --output /tmp/pvlng.curl \
         --header "X-PVLng-key: a10d3cca-370c-121e-b79c-2f0a58030a01"
\
         --data data=23.45 \
         http://your.domain.here/api/r1/87fd-cc15-cb33-71a8-6b78-
3855-4c11-a79e)
```

- `--request` - PUT / insert data
- `--write-out` - Fetch only HTTP code from response headers
- `--output` - Put response body into file for error analysis

- `$1` will contain afterwards the HTTP code

Simple check

```
$ test $1 -eq 200 -o $1 -eq 201 || echo "Error ($1): $(cat /tmp/
pvlng.curl)" && exit 1
```


Data readout

If the queried time period (end – start) is less than the aggregation period, the result can be only one line by design.

(Remember the defaults `start: 00:00, end: 24:00`)

Time declarations

Here are some useful examples for "start" and "end" declarations

What	Request as "start" or "end"
1-Jan this year	`1/1` `first day of this year`
1-Jan of last year	`1/1-1year` `first day of last year`
1st day of this month	`first day of this month`
1st day of last month	`first day of last month`

Return content type

Get all feed-in power values from today as CSV, so many lines as data records exists

```
http://your.domain.here/api/r1/{GUID}.csv
```

As above, but as JSON array of objects

```
http://your.domain.here/api/r1/{GUID}.json
```

Get only one CSV line of the last stored data record today

```
http://your.domain.here/api/r1/{GUID}.csv?period=last
```

Get CSV lines of data on daily basis.

Because of the queried time period (00:00 – now), the result will be one line...

```
http://your.domain.here/api/r1/{GUID}.csv?period=day
```

Get CSV lines on daily basis, start at begin of the year

This will return one line for each day in the time period.

```
http://your.domain.here/api/r1/{GUID}.csv?start=1/1&period=day
```

Get data as above, but on a weekly basis

```
http://your.domain.here/api/r1/{GUID}.csv?start=1/1&period=week
```

Get all data for last 7 days

```
http://your.domain.here/api/r1/{GUID}.csv?start=-7days
```

Get the monthly data since commissioning

```
http://your.domain.here/api/r1/{GUID}.csv?start=0&period=month
```

Get CSV lines of data on yearly basis

Because of the queried time period ("1-Jan this year" ... "now" is lower than aggregation period), the result will be one line...

```
http://your.domain.here/api/r1/{GUID}.csv?start=1/1&period=year
```


Inverter efficiency

"By efficiency, we are really saying, what percentage of the power that goes into the inverter comes out as usable AC current (nothing is ever 100% efficient, there will always be some losses in the system).
This efficiency figure will vary according to how much power is being used at the time, with the efficiency generally being greater when more power is used." -- Solar-

facts.com

The Inverter efficiency is calculated by:

$$\frac{P_{AC}}{\sum_{1}^{n} P_{DC}} * 100$$

For this we need the special channel group Ratio

Ratio calculator (%)

Parameter	Parameter value
Name	Inverter efficiency *
Description	
Serial number	
Channel	
Resolution	100 *
Unit	%
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *
Cost	0
Threshold	0
Valid from	95
Valid to	100

* Required

- The Ratio calculator would return in the required channel combination values between 0 ... 1
  To get readout values as percentage (0 ... 100), set **resolution** to 100.
- The **valid range** 0 ... 100 avoid values values over 100%, which is possible by inaccurate readings.
  So the result will be aligned into the range.

Please note: The performance ratio model accepts **only exact 2 sub channels** for correct operation!

If your inverter have more than one MPP tracker (as mine), we need another special group to sum all $P_{DC?}$

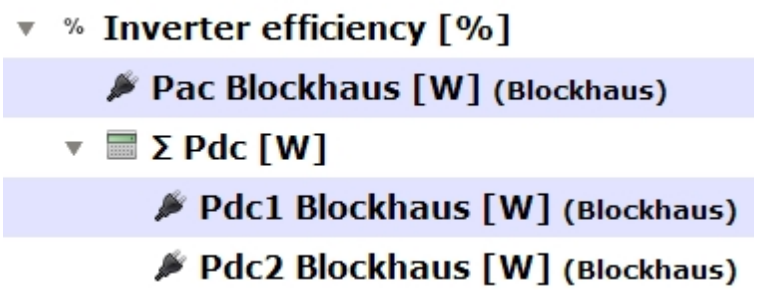
Accumulator

Parameter	Parameter value
Name	Σ Pdc *
Description	
Serial number	
Channel	
Resolution	1 *
Unit	W
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *
Cost	0
Threshold	0
Valid from	
Valid to	

* Required

HTML entities are fine here, e.g. for the name.

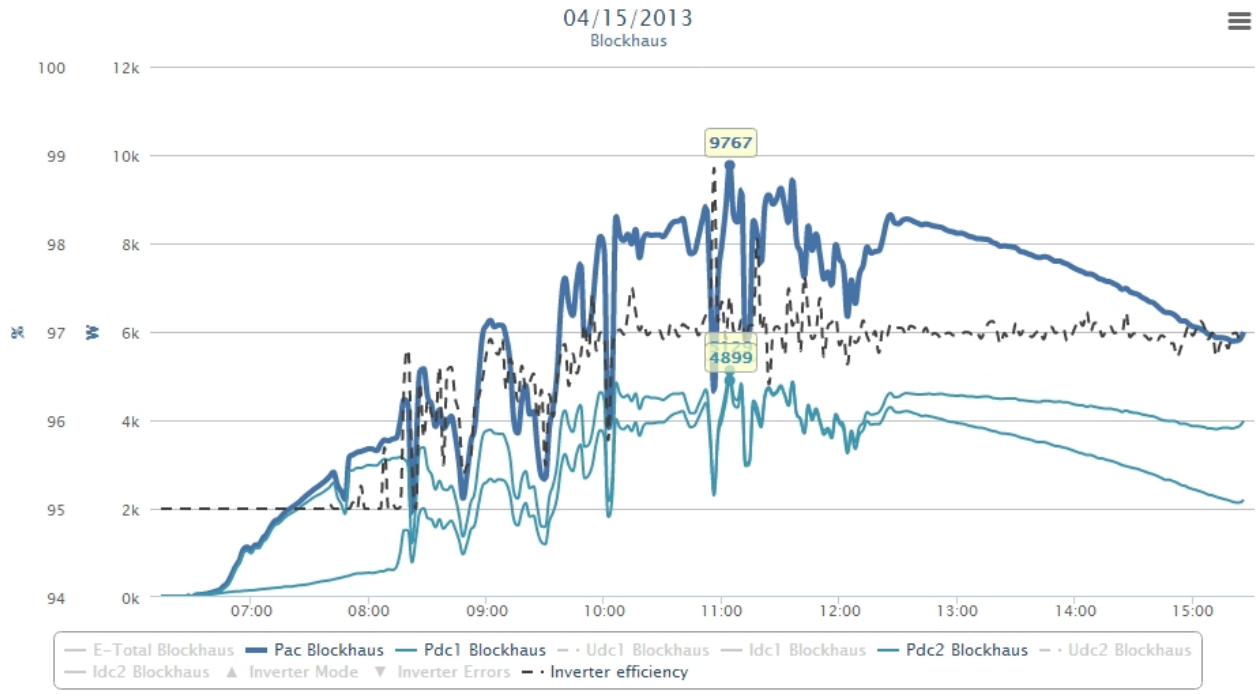
Define your required channels, put this all together and you should have the following definition:



Please note:

- The real **sensor channels** for P_{AC}, P_{DC1}, P_{DC2} etc. must be defined for **each inverter**!
- The **grouping channels** "Inverter efficiency [%]" and "Σ Pdc [W]" must be defined only **once**! They **find out** on request their correct sub channels from their **position in the channel tree**.

If you want to have a more detailed result for efficiency, set the "valid from" for example to **95**:



There is very good to see, that the efficiency (black dashed) is after noon about 97%!

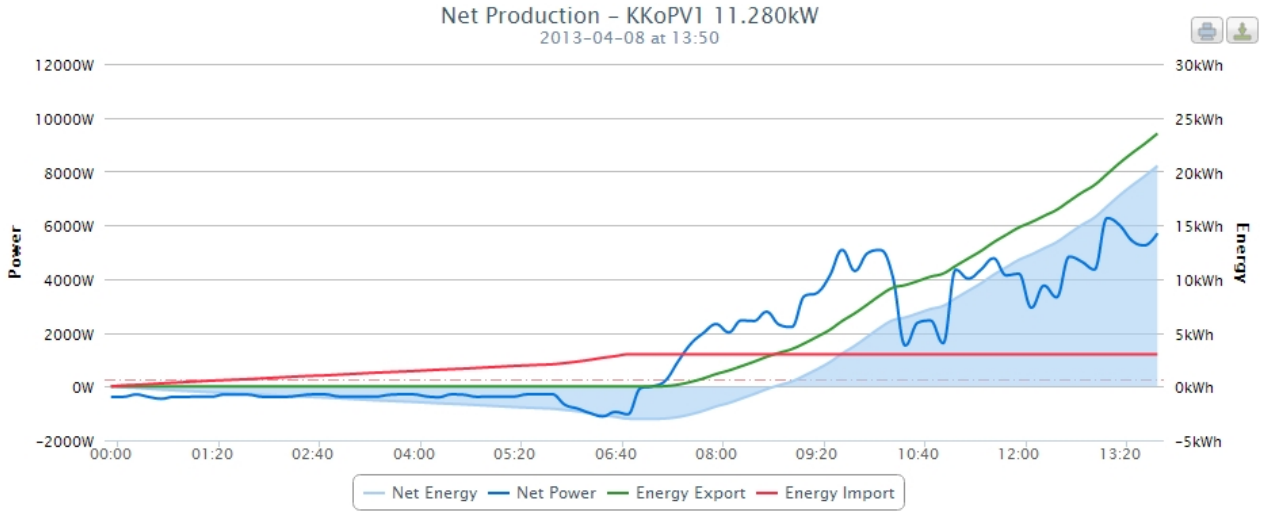
All values below 95% are "raised" to 95%, so just ignore them until 8:00

Energy import

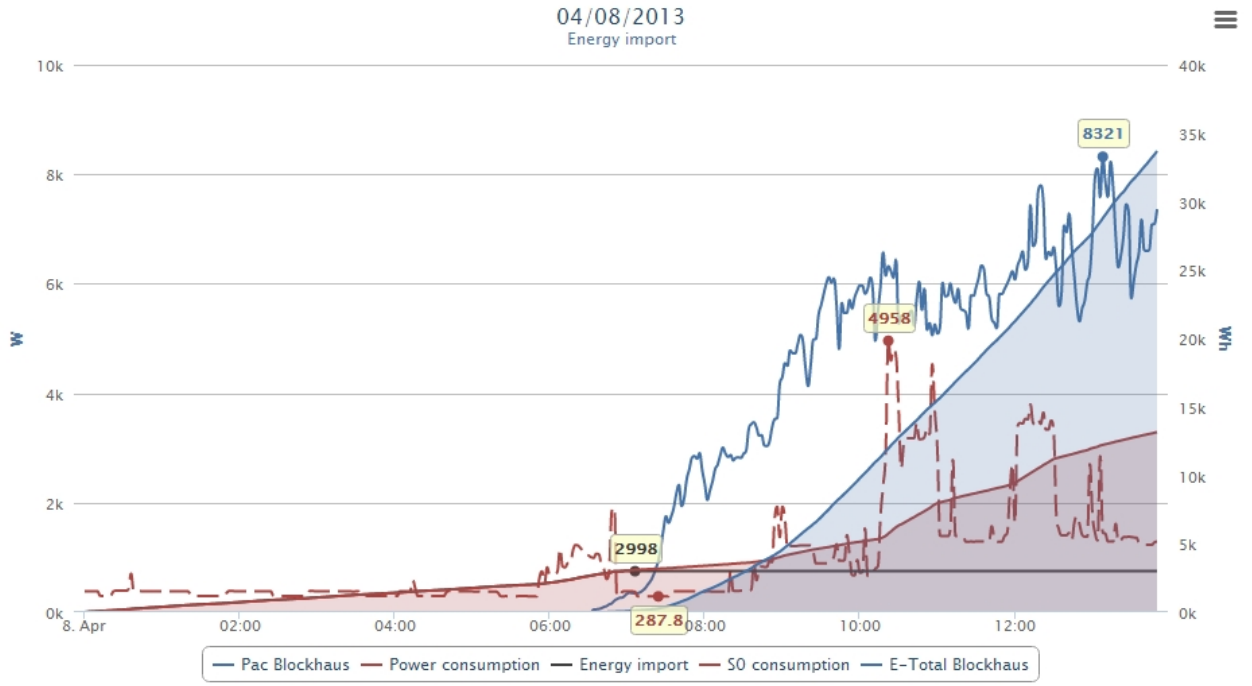
The so called "Energy import" is inspired by PVOutput energy import chart.

If the generated power (Net power) is greater than the consumption, the energy import is no more raising.

PVOutput chart



PVLng adoption



First there are 2 areas, the overall production (E-Total, blue) and overall consumption (Power consumption, red) over time.

Then the actual inverter yield (P_{AC} Blockhaus, blue) and actual consumption (S0 consumption, dashed red).

The interesting part is the black Energy import spline, which stays by 2998 Wh since 07:08 here. This means, since 07:08 there is no energy import, only export.

Please note: Arrange the channels according their **units** to the **axes**!

Definition

Let's start with the meter channels for the areas.

I decided to put all "watt" channels on the left and all "watt hour" channels on the right axis.

E-Total

Definition

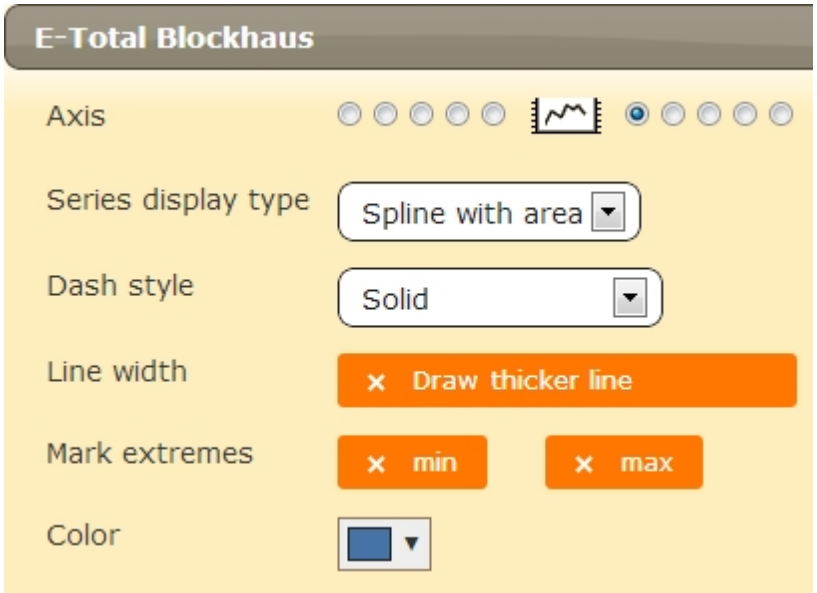
Energy meter, absolute (Wh)

Parameter	Parameter value
Name	E-Total Blockhaus *
Description	Blockhaus
Serial number	
Channel	E-Total
Resolution	1000 *
Unit	Wh
Meter	Yes No *
Numeric values	Yes No *
Cost	0.0002874

We define here a **positive** cost per watt hour to see later the income amount generated.

(We get here 28.74 cent per kilo watt hour == 0.0002874 Euro per watt hour)

Chart settings



S0 consumption and Power consumption

Definition

Here we need a special channel type, the Sensor to Meter proxy.

We have to convert the actual power consumption (Power sensor in watt) into an meter channel (in watt

hours) which also sums up the data.

The consumption is a normal power sensor.

Power sensor (W)

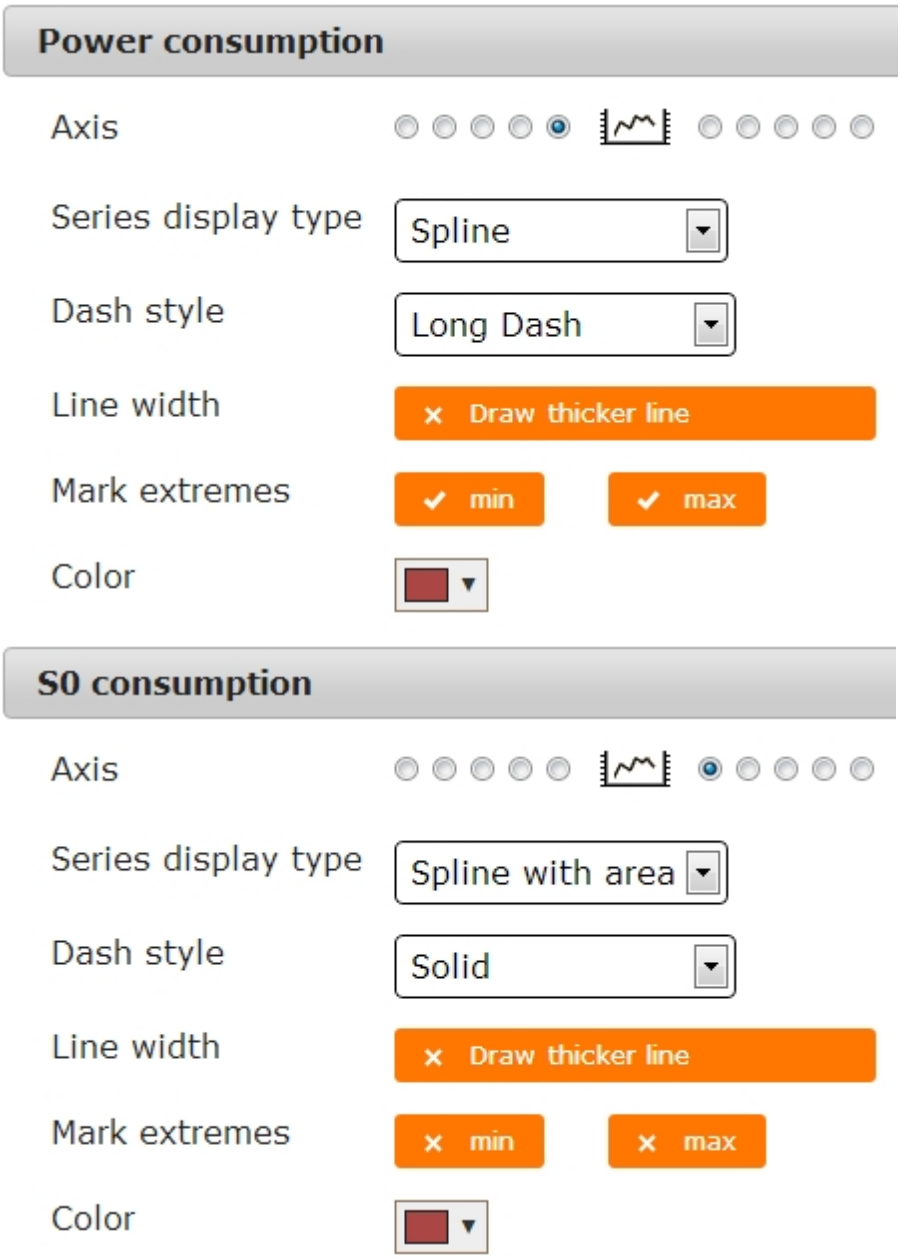
Parameter	Parameter value
Name	Power consumption *
Description	Blockhaus
Serial number	B-Watt Swissnox / S0
Channel	/dev/usb-ftdi-1
Resolution	1 *
Unit	W
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *

Here comes the proxy which convert the sensor to a meter channel!

Sensor to meter

Parameter	Parameter value
Name	S0 consumption *
Description	Blockhaus
Serial number	
Channel	
Resolution	1 *
Unit	Wh
Meter	◉ Yes ○ No *
Numeric values	◉ Yes ○ No *

Chart settings



For "Power consumtion" we are also intersted in peeak powers used.

PAC

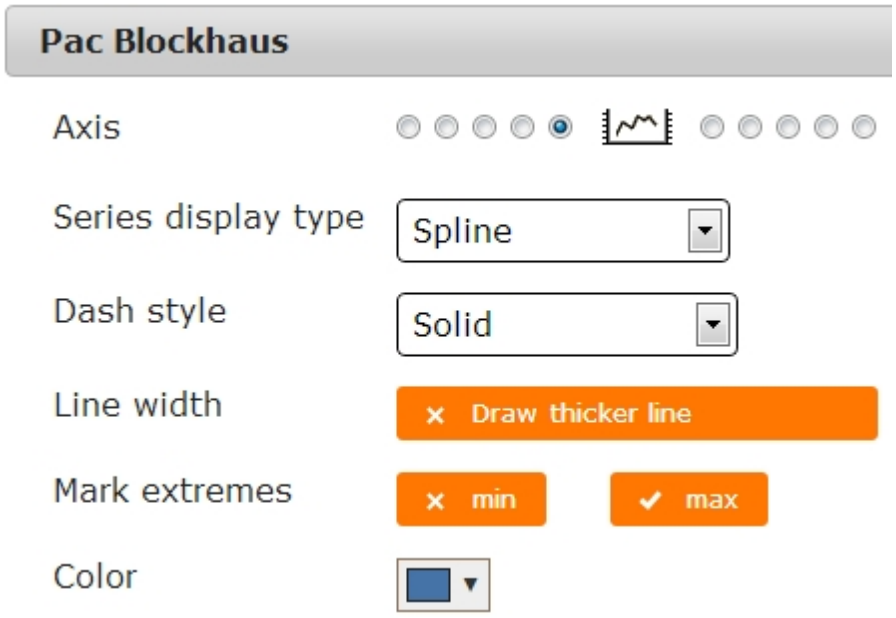
Definition

Simply the actual generated inverter power:

Power sensor (W)

Parameter	Parameter value
Name	Pac Blockhaus *
Description	Blockhaus
Serial number	
Channel	Pac
Resolution	1 *
Unit	W
Meter	○ Yes ◉ No *
Numeric values	◉ Yes ○ No *

Chart settings



We are also interested in the max. value.

Energy Import

Definition

There we need the channel type "Internal consumption" based on the named model.

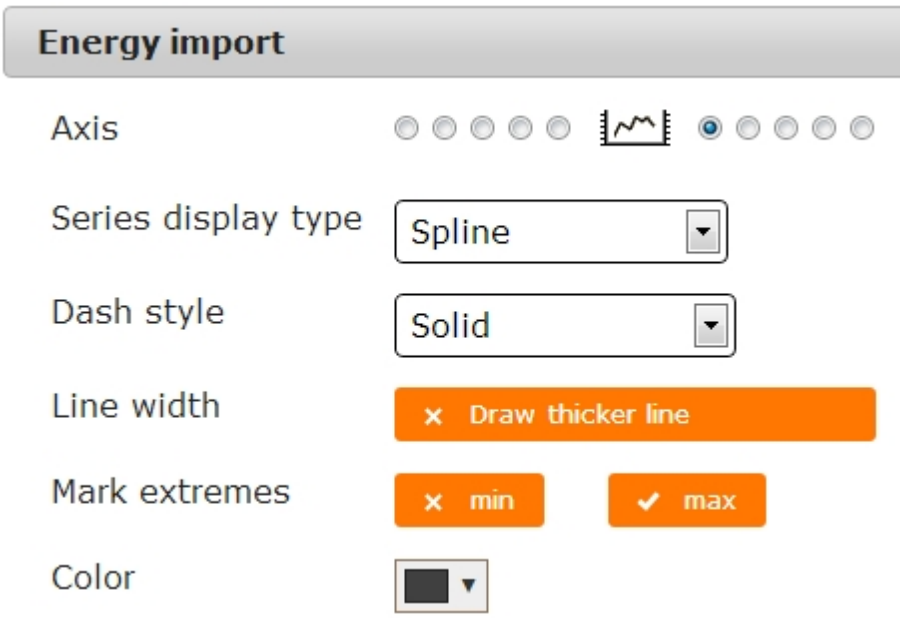
Internal consumption

Parameter	Parameter value
Name	Energy import *
Description	Blockhaus
Serial number	
Channel	
Resolution	1 *
Unit	Wh
Meter	Yes No *
Numeric values	Yes No *
Cost	-0.0002344

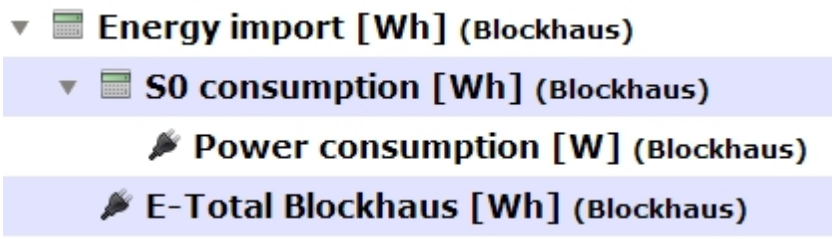
We define here a **negative** cost per Wh to see later the amount consumed.

(We pay here 23.44 cent per kilo watt hour == 0.0002344 Euro per watt hour)

Chart settings



Now we can put this all together:



In the Charts view we will have this:

Channel	Amount	Unit	Cost
Pac Blockhaus, Blockhaus		W	
Power consumption, Blockhaus		W	
Energy import, Blockhaus	2,795.77	Wh	-0.66
S0 consumption, Blockhaus	4,602.16	Wh	
E-Total Blockhaus, Blockhaus	18,968.00	Wh	5.45
			4.79

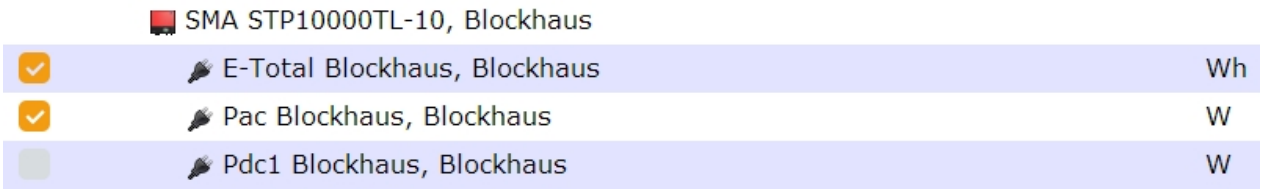
So we have **consumed** 4.6 kWh, but **bought** only 2.8 kWh and have a **win** of € 4.79 at the moment!

The **Amount** column will be filled only for channels with meter flag.

The **Cost** column will be filled only for meter channels with cost defined.

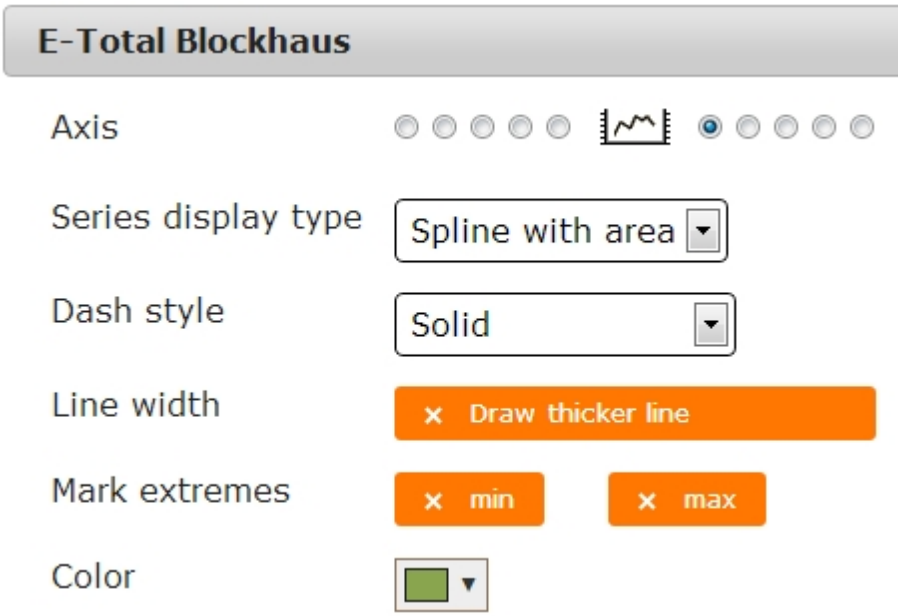
Charts

Simple Spline and Area spline



Please note: Put channels with **same unit** onto the **same axis**!

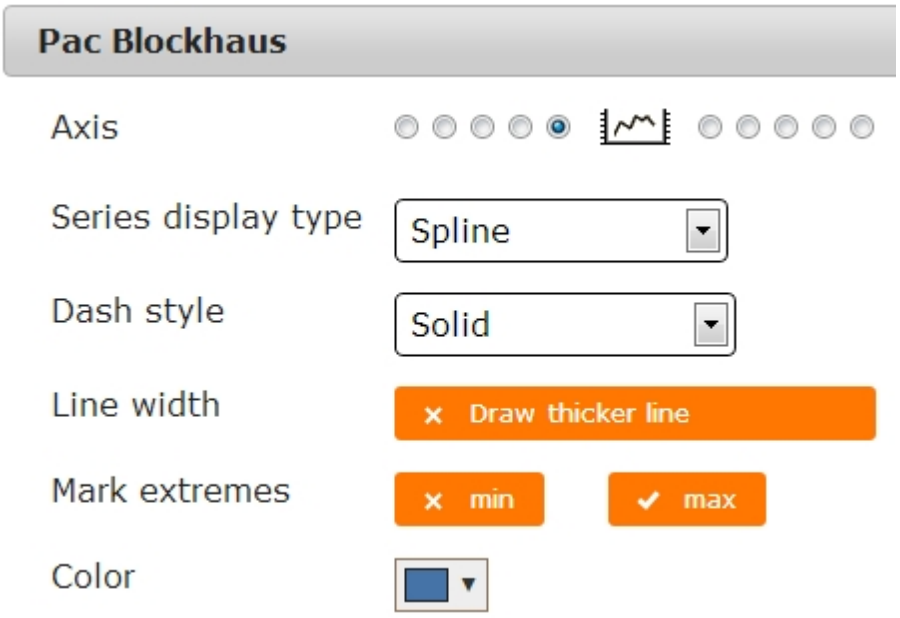
Spline with area



E-Total is a **meter channel**, it looks good with area spline.

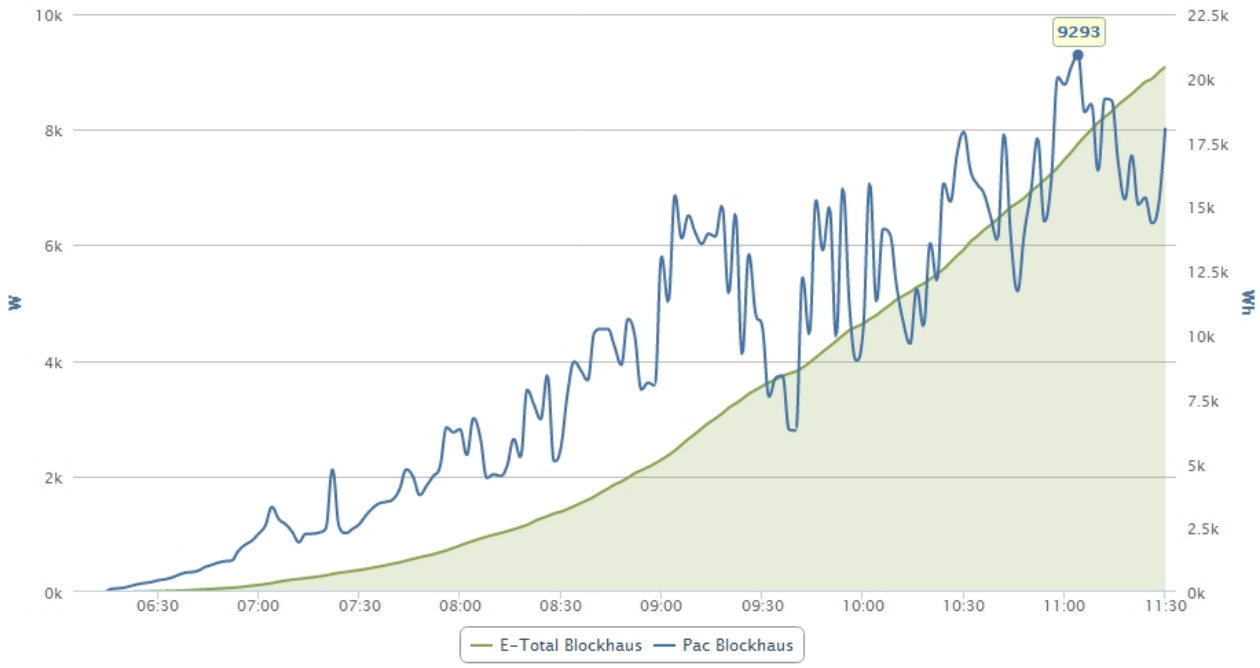
min and max makes no sense here, a meter raises from 0 to its max. value...

Spline



This is a **normal sensor**, so the max value could be interesting.

Result



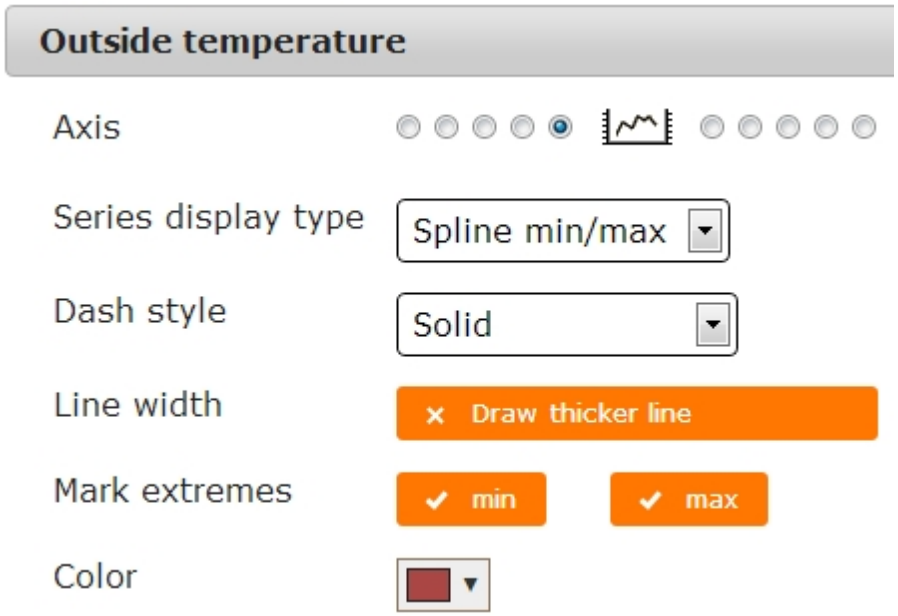
If you hover the chart, the single values for the data points will be displayed.

Spline with ranges

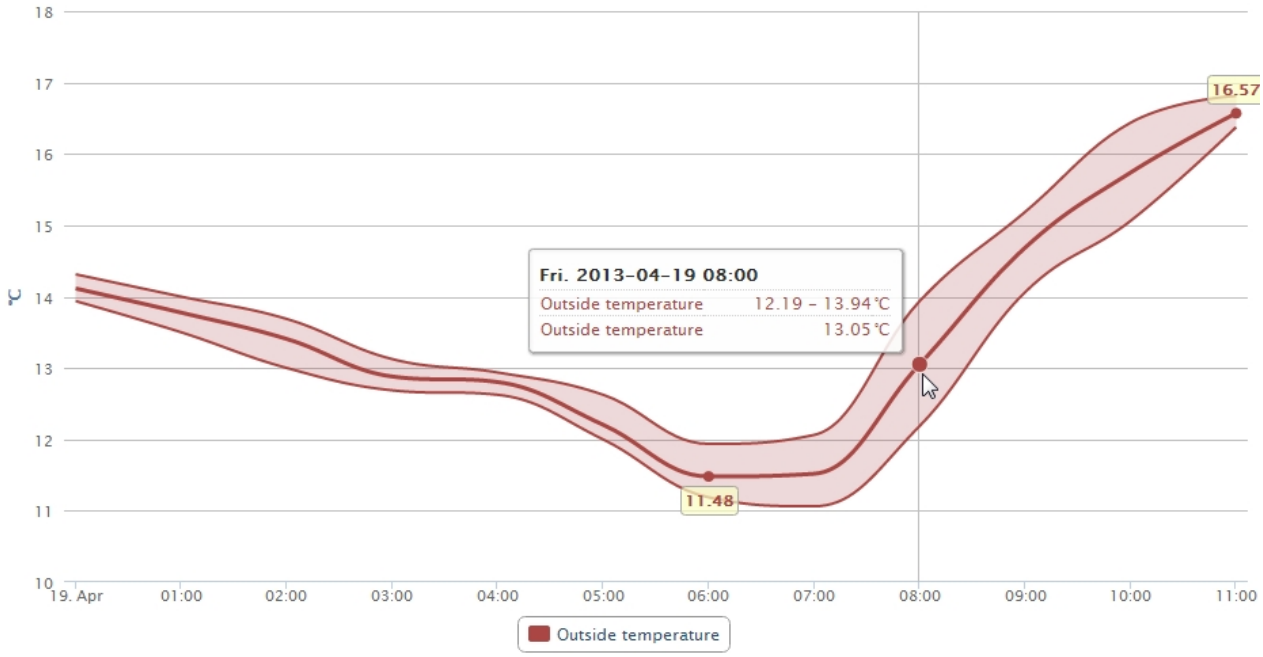
Let's show a temperature average by hour with min/max values

- Set a aggregation period

- Set the presentation



And Refresh  :-)



The time 08:00 in the hint means the time slot from 08:00 + 1hour = 09:00, it always shows the **min. timestamp** of the aggregation period.

So the last 11:00 means 11:00 till now...

If you show with the same settings, but **without** aggregation, the series degrades safely to a **normal spline**:

Alphanumeric channels

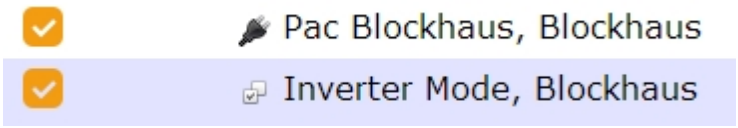
To show alphanumeric channels data, define the presentation as this

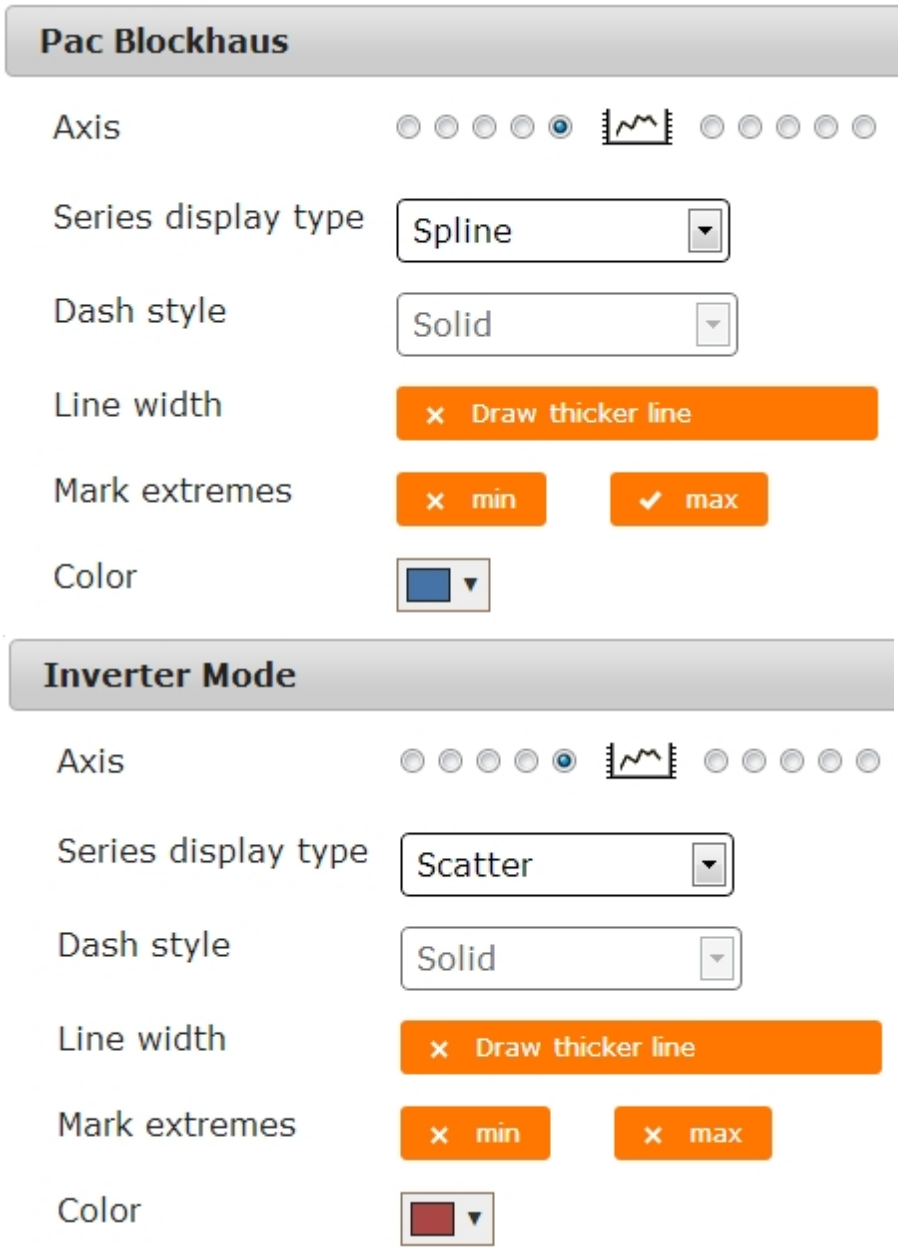
- Select Series display type: **Scatter**

This trigger the display of data labels for each data point

Lets build an example.

Inverter working hours





The inverter starts in the morning with state "Warten" (waiting) and if there is enough power, production starts with "MPP".

In the evening the same, but in reverse order...

Axis handling

Recommendations

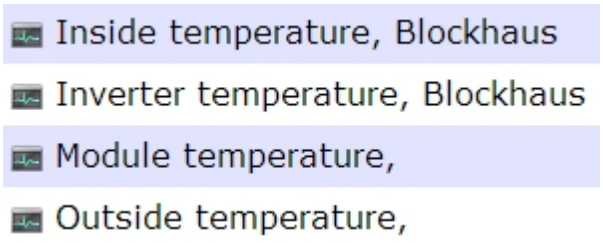
Assign channels with different units to different axes

If the value ranges is mostly the same, collect channels with same unit on the same axis

Sometimes you have channels with same unit, but different value ranges. Then it can make sense, to have different axes with same unit.

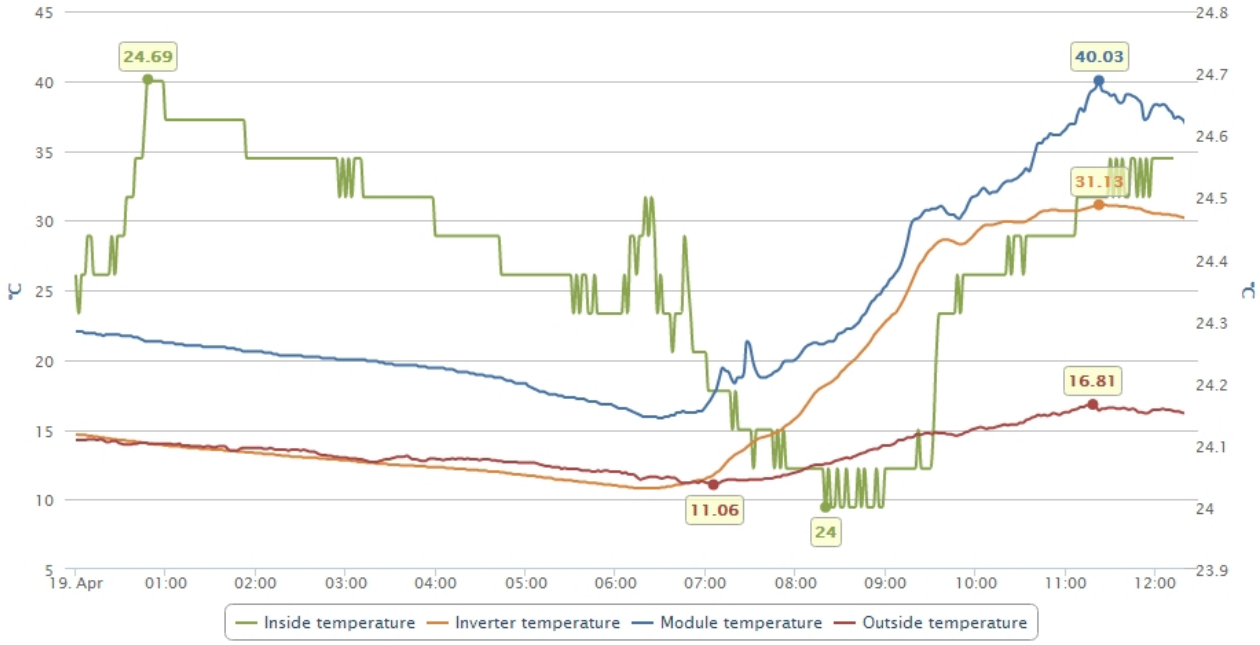
Example

Here we have 4 temperatures:



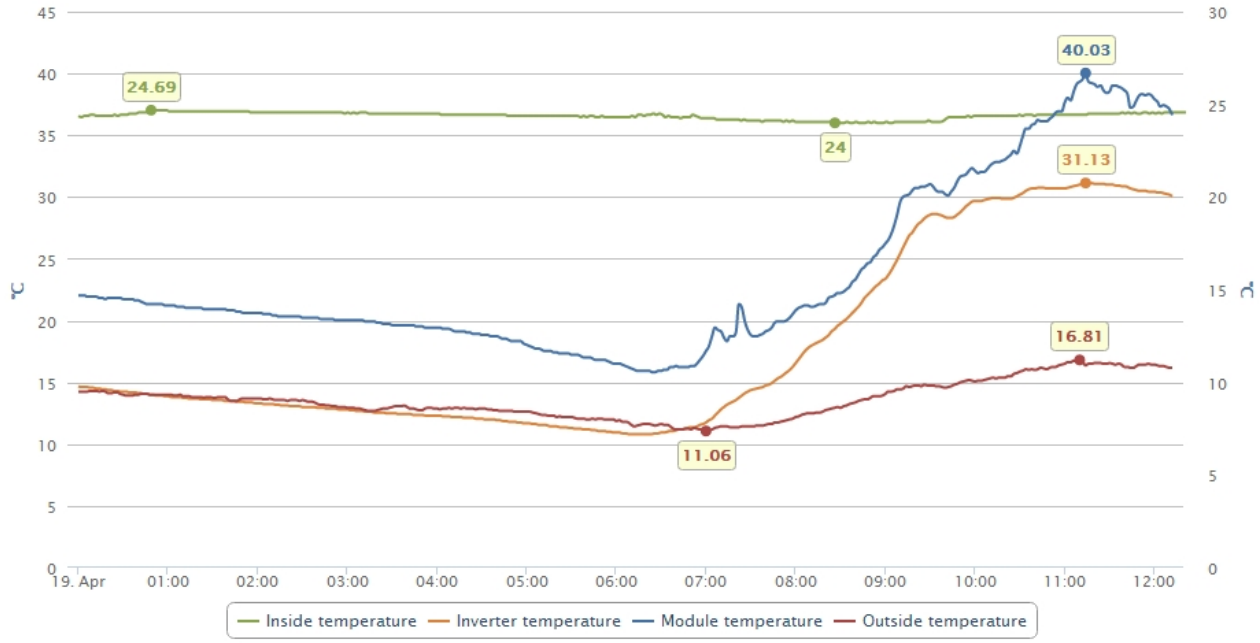
But of the inside temperature, the other correlate more or less to "outside" located.

So we put the last 3 on the left and the inside temp to the right axis.



But there is an option to align the axis if required...

Just check ☑ Set Y axis min. to 0 and you will get this:



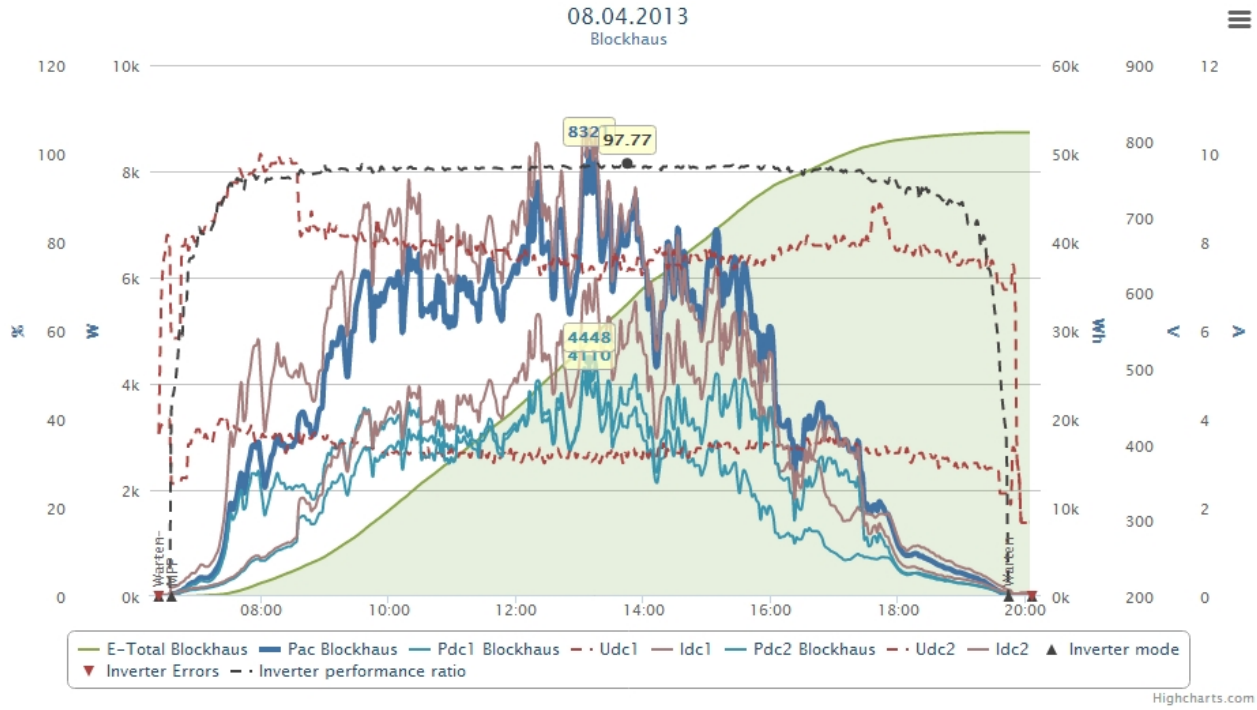
Limits

What's possible

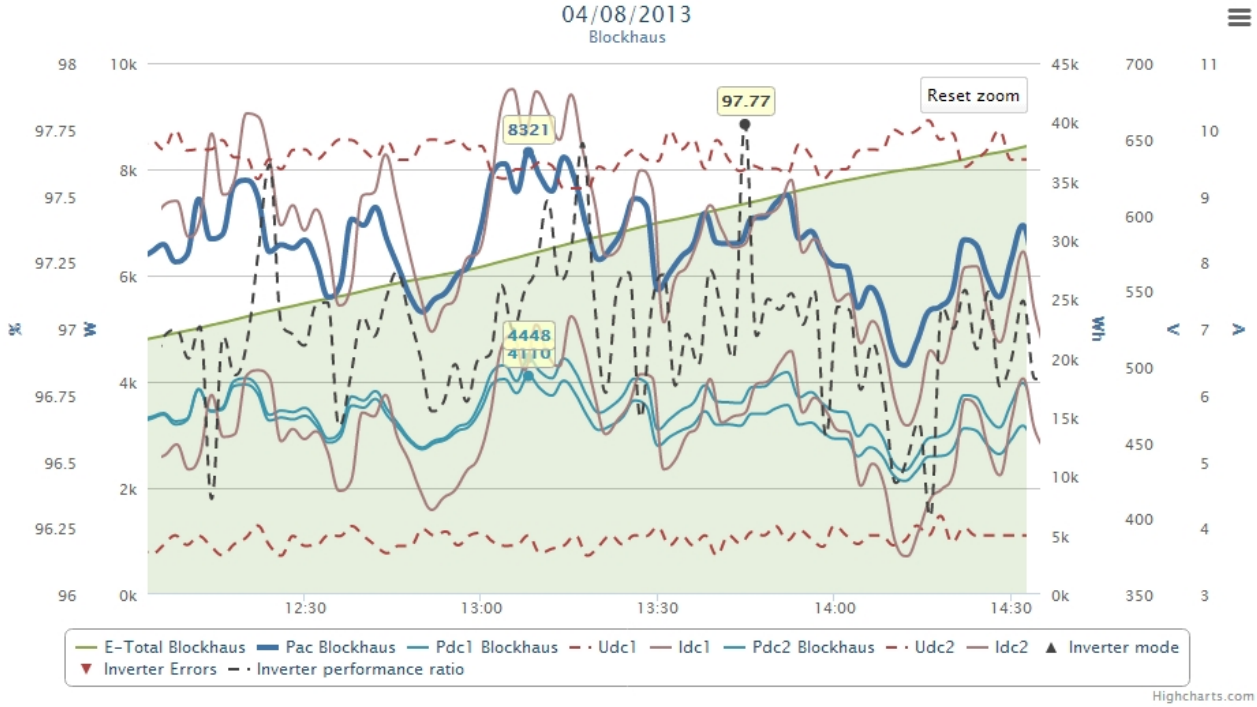
You have on each side of the chart 5 axes available.

If you need more than 5 axes on each side, just take a look at the end of `frontend/Chart/tpl/content.tpl` ;-)

My most comprehensive chart renders up to 11 channels on 5 axes:

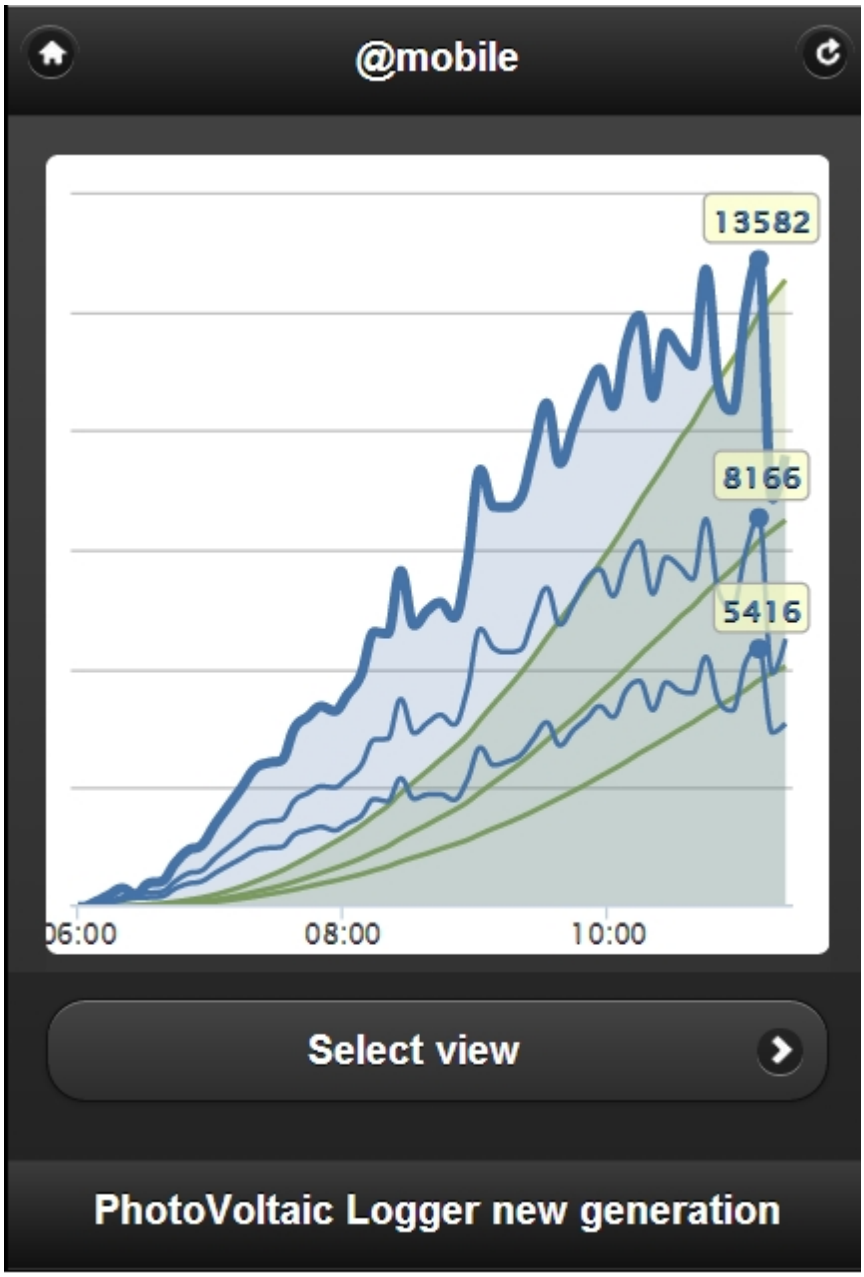


To get a better view on details, you can zoom the x axis by click and drag.



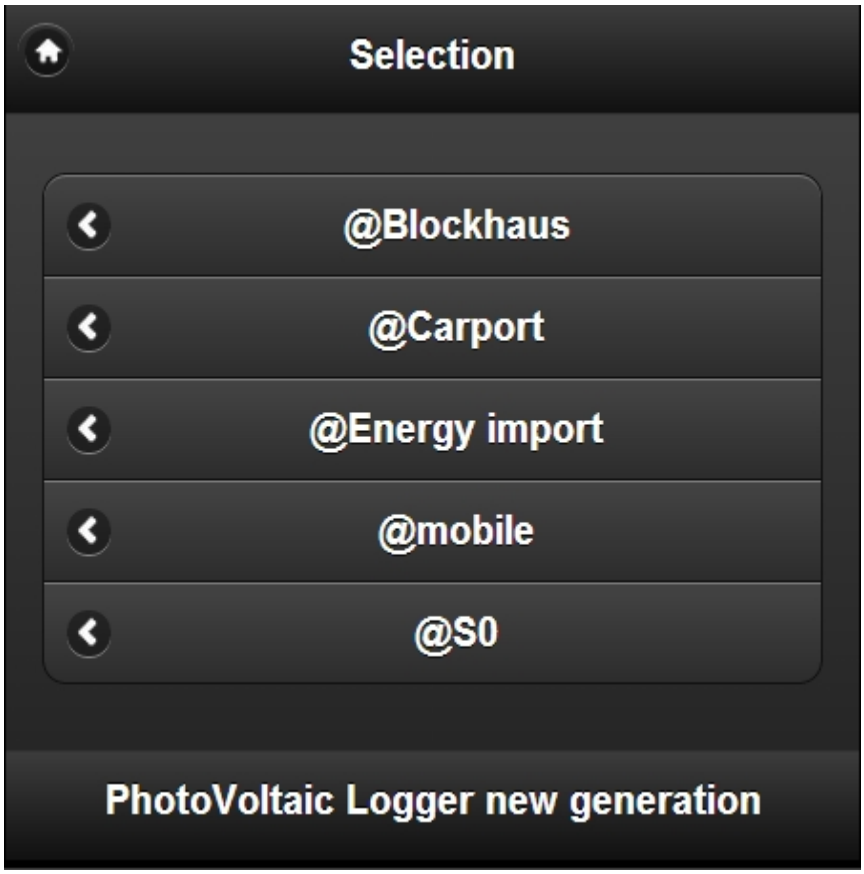
Mobile

Home



- The "Home" button loads always @mobile
- Ther "Refresh" button reloads graphs only

Select View



HowTo

Some instructions from lessons learned.

- Fixed name for USB device
- Backup

Fixed name for USB device

If you have no more a real serial port on your system, you will have to use in most cases a USB-serial adapter.

If you have more than adapter, the device (`/dev/ttyUSB?`) assigned to the adapter during boot time is not in every cases the same.

But it is required to have for the PVLng scripts to have always the same device...

No problem with `udev`.

> "udev is a device manager for the Linux kernel. Primarily, it manages device nodes in /dev. It is the successor of devfs and hotplug, which means that it handles the /dev directory and all user space actions when adding/removing devices, including firmware load." -- Wikipedia

At 1st find out information about your device:

```
$ sudo udevadm info --query=all --name=/dev/ttyUSB0
```

Look for these lines:

```
P: /devices/pci0000:00/0000:00:13.1/usb3/3-2/3-2:1.0/ttyUSB0/tty/
ttyUSB0
...
N: ttyUSB0
...
S: serial/by-id/usb-FTDI_USB_Serial_Converter_FTGCYLSS-if00-port0
...
E: SUBSYSTEM=tty
...
E: ID_SERIAL_SHORT=FTGCYLSS
...
```

Choose a (for you :-) meaningful device name.

I use `usb-ftdi-1` because its my only serial adapter, all others are for 1-Wire.

I also played with the idea of `consumption-watt-meter-s0` ...

Create a rules file

```
$ echo 'SUBSYSTEM=="tty", ENV{ID_SERIAL_SHORT}=="FTGCYLSS", SYMLINK
```

```
+="usb-ftdi-1"' >/etc/udev/rules.d/99-usb-S0.rules
```

Restart udev

```
$ udevadm trigger
```

Check

```
$ ls -al /dev/u*
lrwxrwxrwx 1 root root    7  9. Feb 21:47 /dev/usb-ftdi-1 -> ttyUSB0
```

Here you can see, that the adapter is at he moment also `/dev/ttyUSB0`

Put this device name into the channels attribute "Channel" of your Power sensor.

Backup

Database backup

A full database backup is quite simple.

```
$ mysqldump -u [username] -p[password] -h [host] [databaseName]
>PVLng-$(date +"Y-m-d").sql
```

It just dumps all data into a date stamped SQL file.

Backup with API

If you are interested in CSV/TSV data as backup, you can do this by requesting all data for a channel via the API

```
$ wget -o PVLng-[GUID]-$(date +"Y-m-d").csv http://your.domain.here/
api/r1/[GUID].csv?start=0&full=1
```

- `start=0` means all data since 1970...
- `full` extract also the readable date+time column

Database structure

pvlng_tree

The channel tree is organized in a nested set structure.

> "The nested set model is a particular technique for representing nested sets (also known as trees or hierarchies) in relational databases." -- Wikipedia

More theory

Column	Type	Comment
id	int(10) unsigned Auto-Increment	
lft	int(10) unsigned	
rgt	int(10) unsigned	
moved	tinyint(1) unsigned	
entity	int(10) unsigned	pvlng_channel -> id
guid	varchar(39) *NULL*	

pvlng_channel

The channels are organized in the tree and based on a channel type.

Column	Type	Comment
id	int(10) unsigned Auto-Increment	Unique Id
guid	varchar(39) NULL	Unique GUID
name	varchar(255)	Unique identifier
description	varchar(255)	Longer description
serial	varchar(30)	
channel	varchar(30)	
type	int(10) unsigned	pvlng_type -> id
resolution	double [1]	
unit	varchar(10)	
meter	tinyint(1) unsigned	
numeric	tinyint(1) unsigned [1]	0 for alphanumeric data
cost	double	per unit or unit * h
threshold	double unsigned	
valid_from	double NULL	numeric min. acceptable value

valid_to	double NULL	numeric max. acceptable value

pvlng_type

Channel types describe common attributes and will be handled by a model.

Column	Type	Comment
id	int(10) unsigned	Unique Id
name	varchar(60)	
description	varchar(255)	
model	varchar(30)	
unit	varchar(10)	
childs	tinyint(1)	
read	tinyint(1) unsigned	
write	tinyint(1) unsigned	
graph	tinyint(1) unsigned	
icon	varchar(30)	

pvlng_reading_num

The reading tables hold the data for each channel and timestamp.

The timestamp column is plain integer for better indexes.

Column	Type	Comment
id	int(10) unsigned	pvlng_channel -> id
timestamp	int(10) unsigned	
data	decimal(13,4)	

pvlng_reading_str

Column	Type	Comment
id	int(10) unsigned	pvlng_channel -> id
timestamp	int(10) unsigned	
data	varchar(50)	

Contributions

PHP

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

http://php.net/

MySQL

The world's most popular open source database

http://www.mysql.com/

Highcharts

Highcharts is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to your web site or web application.

http://www.highcharts.com

jQuery

jQuery is a fast, small, and feature-rich JavaScript library.

http://jquery.com/

jQuery UI

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library.

http://jqueryui.com/

jQuery Mobile

A unified, HTML5-based user interface system for all popular mobile device platforms, built on the rock-solid jQuery and jQuery UI foundation.

http://jquerymobile.com/

DataTables

DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, based upon the foundations of progressive enhancement, which will add advanced interaction controls to any HTML table.

http://datatables.net/

jQuery treetable

jQuery treetable is a plugin for jQuery. With this plugin you can display a tree in an HTML table, i.e. a directory structure or a nested list.

http://ludo.cubicphuse.nl/jquery-treetable/

jQuery Pines Notify (pnotify) Plugin

Pines Notify is a JavaScript notification plugin, developed by Hunter Perrin as part of Pines.

http://pinesframework.org/pnotify/

iCheck plugin

Super customized checkboxes and radio buttons with jQuery

https://github.com/damirfoy/iCheck

Fugue Icon set

http://findicons.com/pack/1156/fugue

Author

KKoPV

Knut Kohl

Twitter: @KKoPV