# Globals.pm

The Globals.pm is responsible for the following functions:

- It defines the global NavBar menu of the plugin
- It globally defines the location of the configuration files.
- It holds all default settings of all S4L parts, and automatically merges them with the settings in stats4lox.json
- It contains the import mapping of Loxone statistic element types to outputs and the default mapping
- It contains the BLACKLIST of Loxone element types that should not be parsed from the LoxPlan xml to Loxplan json, if these types to not contain information for statistics
- It contains the **`whoami`** function that returns the function call stack to be used in logging.

# NavBar

The %navbar is filled with the menu, before lbheader is called in your script.

It also automatically sets active=1 on the page that is currently loaded.

# Configuration files

- $statsconfig (access with $Globals::statsconfig)
- $stats4loxconfig (access with $Globals::stats4loxconfig )
- $stats4loxcredentials (access with $Globals::stats4loxcredentials )

are globally defined and cannot be changed by stats4lox.json.

# Default settings and merging

The following hashes are defined as **hashref** in Globals:

- `$Globals::grafana`
- `$Globals::influx`
- `$Globals::loxberry`
- `$Globals::loxone`
- `$Globals::miniserver`
- `$Globals::stats4lox`
- `$Globals::telegraf`

In your code, access these hashes like objects, e.g.:

`$Globals::stats4lox→{s4ltmp}`

Every new default setting needs to be placed in these hashes, so if stats4.lox.json is lost, the default settings are used.

**On every inclusion** of Globals.pm, the `stats4lox.json` is automatically merged with these hashes (a stats4lox.json settings wins against a Globals setting).

Example:

$Globals::grafana→{port} defines 3000.

stats4lox.json grafana.port defines 5000.

→ 5000 is used. If port is missing in stats4lox.json, 3000 is used.

To see the current merged configuration,
run `/opt/loxberry/bin/plugins/stats4lox/debugging/showconfig.pl` that dumps the merged configuration.

### Reload the config

Whenever your code changes stats4lox.json during runtime, force to re-read the config:

`Globals::merge_config( force => 1 );`

This will update the config variables.

# Import Mappings

Example

```
$ImportMapping->{ENERGY} = [
    { statpos => "0", lxlabel => "Default" },
    { statpos => "0", lxlabel => "AQ" },
    { statpos => "1", lxlabel => "AQp" }
];
```

$ImportMapping is a hashref containing import mappings for Loxone element types (e.g. ENERGY → "Verbrauchszähler").

The type **must be** in capital letters and represents the type property of LoxPlan XML respectively the Type property of the LoxPlan json.

The content of every mapping is an array holding the statistic value and the element output label:

- `statpos` represents the position on the statistic file (zero-based index: 0 … First value, 1 … second value,…)

- `lxlabel` is the name of the output label of the element, fetched by the http call (it is not available in the Loxplan file)

The same `statpos` can be mapped to multiple outputs. Usually, this is the case for the "Default" output, that also maps to a specific output.

If the mapping defines a lxlabel that does not exist at the element, the import will silently ignore this lxlabel.

### Default mapping

If $ImportMapping contains no specific mapping for an element type, a fallback to the default mapping is done:

```
# DEFAULT MAPPING
$ImportMapping->{Default} = [
    { statpos => "0", lxlabel => "Default" },
    { statpos => "0", lxlabel => "AQ"}
];
```

The import falls back to the element `'Default'` that's name is hardcoded and not written in capital letters.

As with the normal mappings, if an lxlabel does not exist on that element, that lxlabel is silently ignored.

# Blacklist

The LoxPlan.xml contains many elements that are for organisation of the tree and for visualization in LoxConfig and the App, but are not active elements with values (e.g. pages, labels for room and category,...), furthermore some Loxone elements return no data, or make no sense to collect (no-one would collect the Loxone time 'Year').

To not need to update S4L when new element types arriving from Loxone, this is a BLACKLIST not a WHITELIST. New Loxone elements by default will be shown, and - if required - blacklisted on demand.

The blacklist is an array of strings with the element types from LoxPlan.xml, written in CAPITAL LETTERS.

Blacklisted elements do not arrive in the parsed loxplan.json.

Before of the actual variable definition is a list of *commented*, known element types that are not blacklisted.

# whoami

`my $me = whoami();`

The function whoami reads the current call-stack and returns a variable that can be used in logging.

```
sub getLoxplan
{
    my %args = @_;
    my $me = whoami();

$log->INF("$me Local file: $localfile");
```

results in

`getLoxplan--> Local file:`

in the log. It makes it easier to identify the location in code on analyzing the log, especially with deep function call-stacks and reoccuring/recursive calls.