

26. October 2026

© Symcon GmbH 2026
www.symcon.de

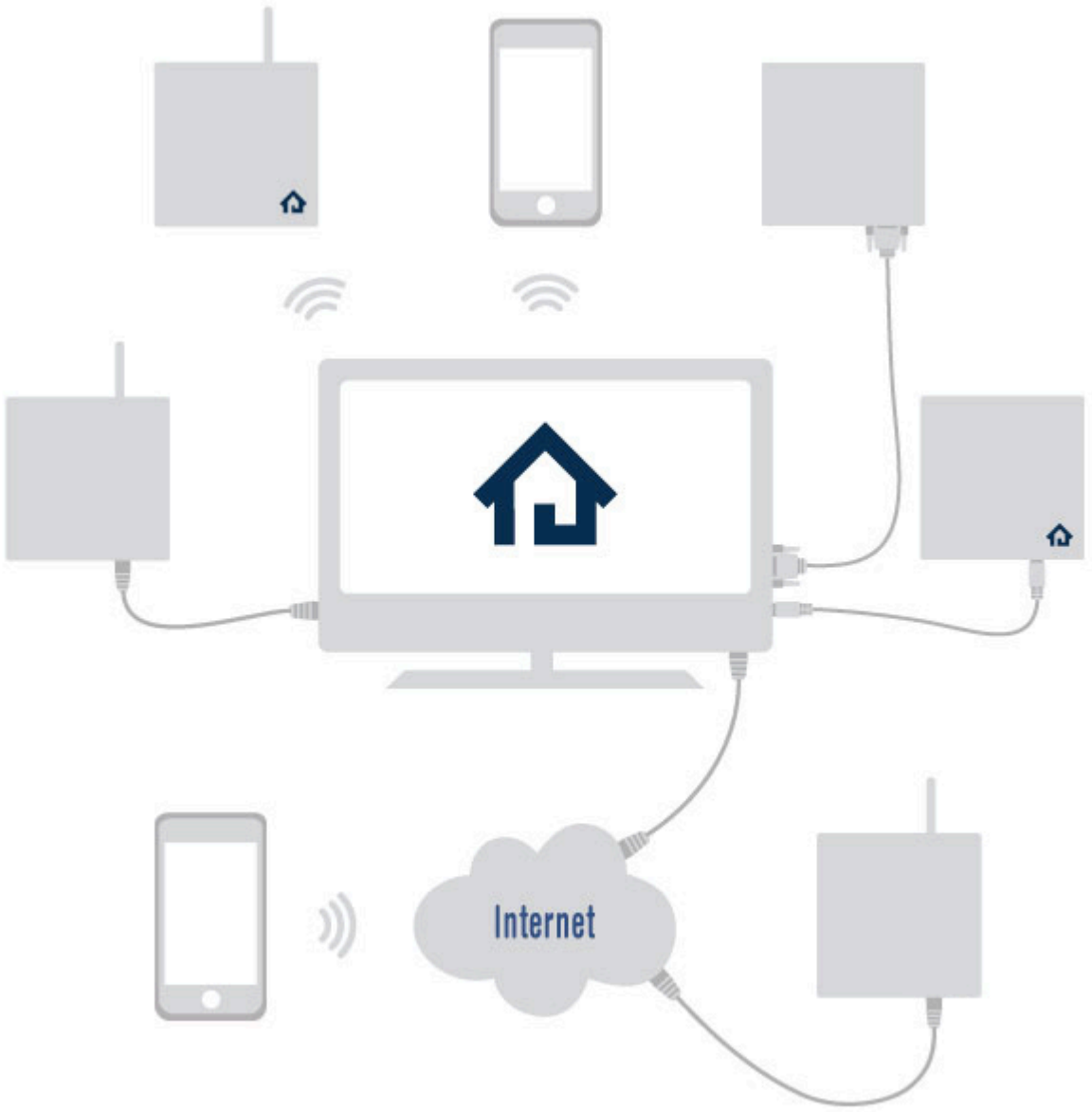
Introduction

This is the documentation of IP-Symcon.
On the one hand it provides technical backgrounds, basic knowledge, and functionalities. On the other hand it supports the development of individual scripts and automation. Examples allow an easy access to the content. The document is available at all times online and offline.
The basic features of IP-Symcon are described on the product page .

HOW DOES IP-SYMCON WORK?

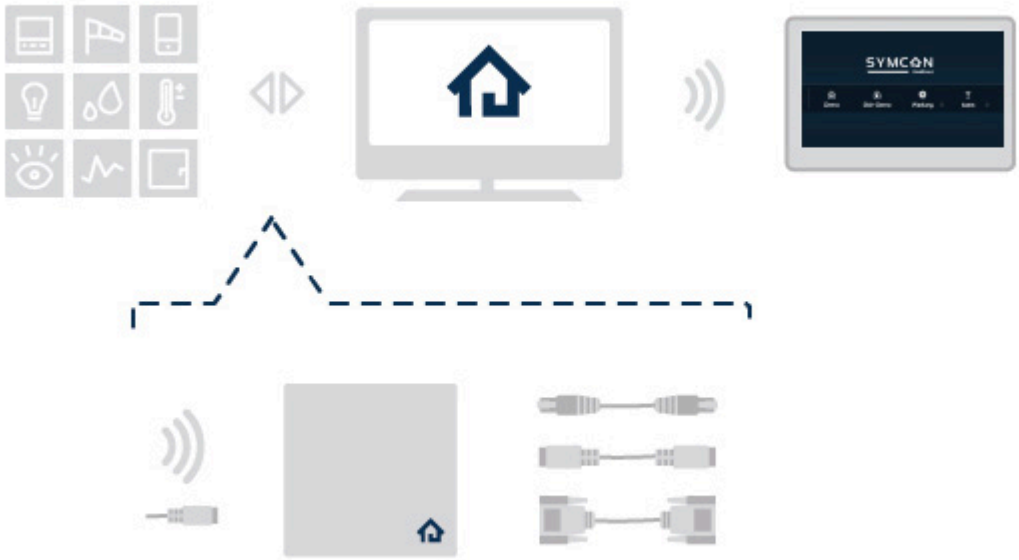
VENDOR INDEPENDENCE

IP-Symcon links diverse devices from different manufacturers within one application. The softwares unified pattern enables access to all devices.



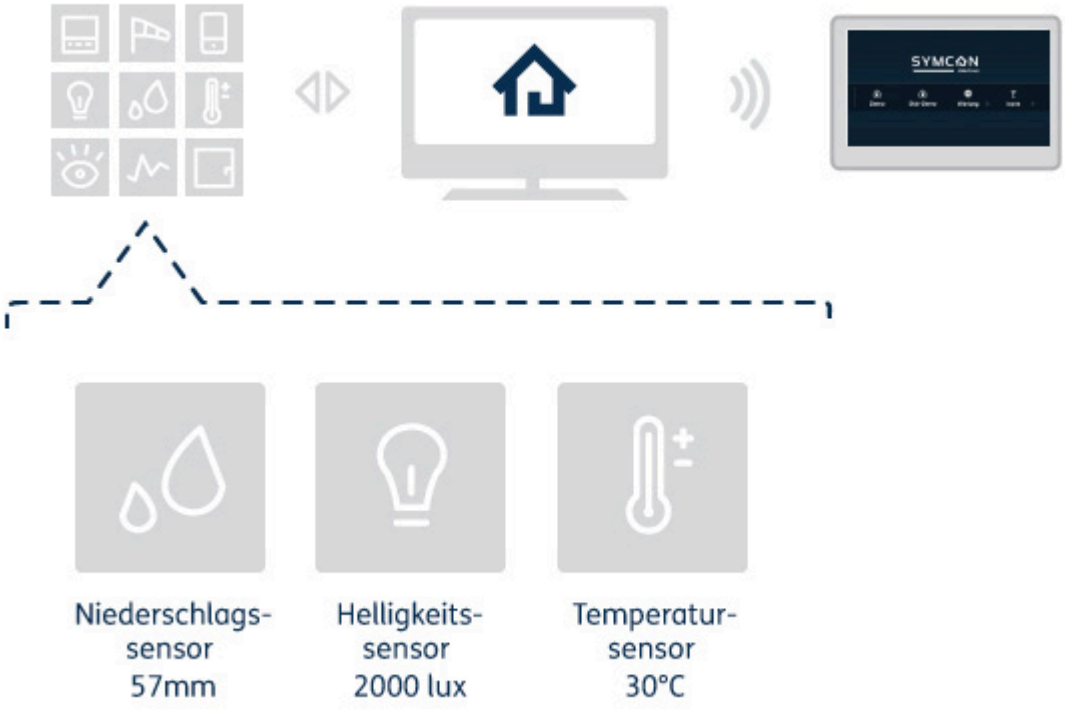
CONNECTION

IP-Symcon uses its own configurators and the manufacturers protocols to connect to the devices using a simple modular structure. It does not matter whether the manufacturers system is based on radio or cable. The software always interprets the connection with the same pattern: Device instance -> Gateway -> I/O



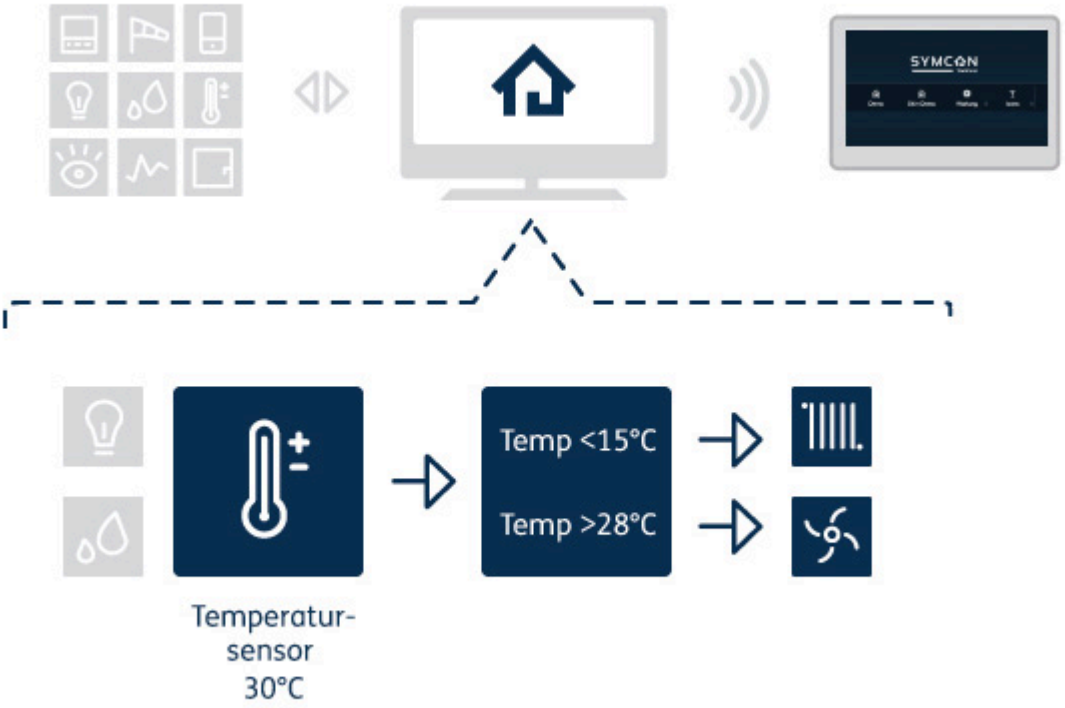
DEVICE STATUS

When the device is connected IP-Symcon can read and write its variables and states. The device can be controlled.



MANAGEMENT

The management console is the technical nerve center. It controls the devices that are configured and connected to the server. The intelligence and the explicite automation is configured and realized via scripts in the management console.



VISUALIZATION

In a next step, IP-Symcom provides the visualization of the configured devices. The visualization clearly shows all functions and data. Controlling via the visualization is no problem at all.



FIRST STEPS

The Quick Access explains most basics.

System Requirements

HARDWARE

A system with one of the listed operating systems is required to use IP-Symcon. In addition, a LAN connection is vital for most gateways. Energy saving components are recommended as the system is permanently active. Our own SymBox is also suitable for most projects.

SOFTWARE

IP-Symcon can be configured as server on most operating systems. A special app for mobile operating systems (iOS, Android) presents the visualization that is provided by the IP-Symcon server.
A complete list of the compatibility between IP-Symcon versions and operating systems is available in the version overview.

LIMITATIONS OF DIFFERENT OPERATING SYSTEMS

LIMITATIONS TO SYMBOX, RASPBERRY PI, LINUX, MACOS, AND NAS SYSTEMS

The compatibility functions are deactivated by default. See special switches for reactivation, if you need compatibility with IP-Symcon 3.4 or older.
No MediaPlayer (currently only in Windows)
Text to Speech: The TTS module can currently only be started within a Windows environment. Alternatively, the module "Text To Speech (AWS Polly)" from the Module Store can be used. It works for all operating systems.
System Information: The function Sys_GetSpooler is only available in Windows.

Version Overview

Version Policy
Version 9.0
Version 8.1
Version 8.0
Version 7.2
Version 7.1
Version 7.0
Version 6.4
Version 6.3
Version 6.2
Version 6.1
Version 6.0
Version 5.5
Version 5.4
Version 5.3
Version 5.2
Version 5.1
Version 5.0
Version 4.4
Version 4.3
Version 4.2
Version 4.1
Version 4.0
Version 3.4

VERSION POLICY

OPERATING SYSTEM	PLANNED SUPPORT
SymOS	Always
Windows Desktop	All with the required PHP support
Windows Server	All with the required PHP support
MacOS	Latest two versions at development start
Linux (Ubuntu)	Latest LTS (Long Term Support) version at development start
Raspberry Pi	Current release at development start
Docker	Docker compatible container
QNAP	Docker compatible container
Synology	Docker compatible container

VERSION 9.0

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	9.0	operating system for the SymBox
Windows Desktop	yes	Windows 8 and above	64bit version only
Windows Server	yes	Windows Server 2012 and above	64bit version only
MacOS	yes	Version 13 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 24.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bookworm or newer)	32bit/64bit (no Pi 1, Zero 1)
Docker	yes	Intel/AMD 64-Bit, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 13 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 8.1

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	8.1	operating system for the SymBox
Windows Desktop	yes	Windows 8 and above	64bit version only
Windows Server	yes	Windows Server 2012 and above	64bit version only
MacOS	yes	Version 13 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 24.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bookworm or newer)	32bit/64bit (no Pi 1, Zero 1)
Docker	yes	Intel/AMD 64-Bit, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
iOS	control only	iOS 13 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 8.0

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	8.0	operating system for the SymBox
Windows Desktop	yes	Windows 8 and above	64bit version only
Windows Server	yes	Windows Server 2012 and above	64bit version only
MacOS	yes	Version 13 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 24.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bullseye or newer)	32bit/64bit (no Pi 1, Zero 1)
Docker	yes	Intel/AMD 64-Bit, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 12 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 7.2

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	7.2	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 13 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 24.04	64bit version only

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
Raspberry Pi	yes	Raspberry Pi OS (Bullseye or newer)	32bit/64bit (no Pi 1, Zero 1)
Docker	yes	Intel/AMD 64-Bit, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 11 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 7.1

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	7.1	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 12 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 22.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bullseye or newer)	32bit/64bit
Docker	yes	Intel/AMD 64-Bit, ARM6, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 11 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 7.0

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	7.0	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 12 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 22.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bullseye or newer)	32bit/64bit
Docker	yes	Intel/AMD 64-Bit, ARM6, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 11 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 6.4

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	6.4	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.15 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 20.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bullseye or newer)	32bit/64bit
Docker	yes	Intel/AMD 64-Bit, ARM6, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 6.3

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	6.3	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.15 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 20.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bullseye or newer)	32bit/64bit
Docker	yes	Intel/AMD 64-Bit, ARM6, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 6.2

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	6.2	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
MacOS	yes	Version 10.15 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 20.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Bullseye or newer)	32bit/64bit
Docker	yes	Intel/AMD 64-Bit, ARM6, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 6.1

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	6.1	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.15 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 20.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Buster or newer)	32bit/64bit
Docker	yes	Intel/AMD 64-Bit, ARM6, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 6.0

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	6.0	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.15 and above	Intel/Apple M1
Linux (Ubuntu)	yes	Ubuntu 20.04	64bit version only
Raspberry Pi	yes	Raspberry Pi OS (Buster or newer)	32bit/64bit
Docker	yes	Intel/AMD 64-Bit, ARM6, ARM7, ARM64 CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 5.5

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	5.5	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.14 and above	
Linux (Ubuntu)	yes	Ubuntu 20.04	64bit version only
Raspberry Pi	yes	Raspbian Buster	
Docker	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Docker

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 5.4

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.8	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.13 and above	
Linux (Ubuntu)	yes	Ubuntu 18.04	64bit version only
Raspberry Pi	yes	Raspbian Buster	
Docker	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 5.3

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.7	operating system for the SymBox

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.13 and above	
Linux (Ubuntu)	yes	Ubuntu 18.04	64bit version only
Raspberry Pi	yes	Raspbian Buster	
Docker	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 5.2

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.6	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.12 and above	
Linux (Ubuntu)	yes	Ubuntu 18.04	64bit version only
Raspberry Pi	yes	Raspbian Buster	
Docker	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 5.1

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.5	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.11 and above	
Linux (Ubuntu)	yes	Ubuntu 18.04	64bit version only
Raspberry Pi	yes	Raspbian Stretch	
Docker	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 5.0

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.4	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	64bit version only
Windows Server	yes	Windows Server 2008 R2 and above	64bit version only
MacOS	yes	Version 10.11 and above	

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
Linux (Ubuntu)	yes	Ubuntu 18.04	64bit version only
Raspberry Pi	yes	Raspbian Stretch	no Raspbian Jessie
Docker	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Docker
QNAP	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see QNAP
Synology	yes	Intel/AMD 64-Bit CPU	specific processor architecture required; see Synology
iOS	control only	iOS 9.3 and above	visualization via app
Android	control only	Android 5.0 and above	visualization via app

VERSION 4.4

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.3	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	
Windows Server	yes	Windows Server 2008 R2 and above	
MacOS	yes	Version 10.11 and above	
Linux (Ubuntu)	yes	Ubuntu 16.04	64bit version only
Raspberry Pi	yes	Raspbian Stretch	no Raspbian Jessie
iOS	control only		visualization via app
Android	control only		visualization via app

VERSION 4.3

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.2	operating system for the SymBox
Windows Desktop	yes	Windows 7 and above	
Windows Server	yes	Windows Server 2008 R2 and above	
MacOS	yes	Version 10.10 and above	
Linux (Ubuntu)	yes	Ubuntu 16.04	64bit version only

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
Raspberry Pi	yes	Raspbian Jessie	no Raspbian Stretch
iOS	control only		visualization via app
Android	control only		visualization via app

VERSION 4.2

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.1	operating system for the SymBox
Windows Desktop	yes	Windows Vista and above	
Windows Server	yes	Windows Server 2008 and above	
MacOS	yes	Version 10.9 and above	
Linux (Ubuntu)	yes	Ubuntu 16.04	64bit version only
Raspberry Pi	yes	Raspbian Jessie	no Raspbian Stretch
iOS	control only		visualization via app
Android	control only		visualization via app

VERSION 4.1

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.1	operating system for the SymBox
Windows Desktop	yes	Windows Vista and above	
Windows Server	yes	Windows Server 2008 and above	
MacOS	yes	Version 10.9 and above	
Linux (Ubuntu)	yes	Ubuntu 16.04	64bit version only
Raspberry Pi	yes	Raspbian Jessie	no Raspbian Wheezy
iOS	control only		visualization via app
Android	control only		visualization via app

VERSION 4.0

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
SymOS	yes	1.0	operating system for the SymBox
Windows Desktop	yes	Windows Vista and above	
Windows Server	yes	Windows Server 2008 and above	
MacOS	yes	Version 10.8 and above	
Linux (Ubuntu)	yes	Ubuntu 14.04	64bit version only
Raspberry Pi	yes	Raspbian Wheezy	no Raspbian Jessie
iOS	control only		visualization via app
Android	control only		visualization via app

VERSION 3.4

OPERATING SYSTEM	SUPPORTED	VERSION	REMARK
Windows Desktop	yes	Windows 2000 and above	
Windows Server	yes	Windows Server 2003 and above	
iOS	control only		visualization via app
Android	control only		visualization via app

Terms and Conditions



Our "General Terms and Conditions of the Software Purchase Agreement" can be downloaded at any time: Download



A detailed explanation of the rights of use can be found here: Usage rights.

Usage rights

The legally binding, general terms and conditions of the software purchase agreement can be downloaded from Terms and Conditions

An IP-Symcon license entitles you to install IP-Symcon as often as desired within a private household. For commercial installations, a separate IP-Symcon license is required for each server/computer in operation. Only one productive installation per license can be in operation at any one time, which communicates with the server for push notifications or the Connect-Service. Any further installations can therefore only be used for programming and testing purposes. An IP-Symcon license is valid for all terminal devices (operating systems) on which IP-Symcon is available. A valid subscription is required for an online installation. If an installation of IP-Symcon is desired after the subscription has expired, a self-made backup is required.

The IP-Symcon Pro console can be downloaded from the IP-Symcon server and automatically updated using the integrated update function without an IP-Symcon license.

Subscription

A subscription provides the listed services during its duration. These are required by some modules for operation (e.g. Connect Service for Alexa Smart Home Skill)

SERVICES

The following services require an active subscription.

INSTALLATION/UPDATE

An active subscription is required for new installations and updates of IP-Symcon.

PUSH NOTIFICATION

Push notifications are sent to the mobile apps via our server.

CONNECT SERVICE

Encrypted access is provided via our Connect Service server enabling the system to be securely accessed from anywhere.

PERIOD OF VALIDITY

The duration (validity) of the subscription can be determined from one of the following three points:

In the last email of a license or subscription order: "... valid until: DD.MM.YY "
In our forum under "License Management". To do this, one has to register there.
Inquire by email or using the contact form
One month before the subscription expires, an email will be sent to inform about its soon expiration



A subscription is **not** self-renewing.
When the subscription expires, it can be purchased/renewed in the IP-Symcon Shop .
One month before the expiration date, our system will automatically send a notification email.
When ordering, the new time contingent is added to any remaining duration (similar to a prepaid card).

There is more information about the license used in the Management Console via the information icon in the top right corner.

Quick Start

SYMCON

Symcon is a server software which aims to combine, link and automate different automation systems in one server. This means that systems that cannot communicate with one another are brought together in a uniform, clear configuration. Furthermore, Symcon offers a uniform Visualization and control, both for the browser and via Mobile Apps for iOS and Android. A comprehensive description is available on the product page .

SUPPORTED SYSTEMS

Symcon supports a wide variety of well-known manufacturers and protocols. An overview of the native systems can be seen in the Module Reference. Additionally, the Module Store is offered. This is a platform on which modules are submitted, quality controlled and made available by the community to the user free of charge.

SYMCON EDITIONS

There are six different editions: Community, Professional, Expert, Basic, Industrial and Enterprise. These are valid indefinitely and offer the same system support. The editions differ in the number of variables, Visualizations and the included Extensions. A detailed list of all functions can be found in the Function overview. An existing license can be upgraded to a higher version at any time. An overwiev of the upgrades can be viewed in the shop.

SUBSCRIPTION

A Subscription activates features, which are offered by our server. This includes, for example, push notifications, updates, Connect Control (especially important for Amazon Alexa and Google Assistant). An expired subscription does not affect the running server, only the features associated with the subscription will be deactivated. A subscription can be extended at any time via the Shop.

FREE VERSION

It is possible to request a free license. For personal, non-commercial use, there is a permanently free Community Edition. To test the scope of a Professional or Enterprise Edition, there are demo versions available for one month.

HARDWARE REQUIREMENTS

The Symcon server must run permanently. A SymBox , Windows, MacOS, Linux Ubuntu, Raspberry Pi or Docker (also QNAP, Synology) system is required for this. Which version of the respective operating system is required can be looked up in the Version Overview in the System Requirements area. Furthermore, so-called gateways are required as an interface for the built-in systems. Which gateways are supported can be looked up in the respective

Module Reference and their device list. Some gateways are offered in the Shop or even as a SymBox extension .

SYMBOX FUNCTIONALITIES

The SymBox is the in-house server. This can be bought in a DIN rail housing for the electric cabinet or in an aluminum enclosure. In addition to the very low power consumption (max. 3W) the SymBox with SymOS offers a specially adapted operating system and tailored web interface for Symcon. In addition to updates and settings, the Data Backup can be conveniently managed with this. If a Symbox is in use, the SymOS can be reached in the local network via "symbox.local ".

INSTALLATION OF SYMCON

The installation is described for the respective system under Installation. After a successful installation, the Symcon Service starts automatically and is ready for configuration.

TRAY FUNCTIONALITIES

After installation, the Tray Application is now available on Windows and MacOS. Depending on the operating system, the tray application is located as a Symcon symbol in the respective system tray. Via the application The service can be started, stopped and updated. Under "Information" there is the possibility to set up Remote Access.

CONFIGURATION

Management Console is used for configuration. This can be called up from anywhere in the network via a browser. To open it, "[IPServer]:3777/console/" just needs to be entered in the address line. If a SymBox is in use, this is symbox.local:3777/console/ instead.

BASICS

The basic concepts and their handling are described in the area Basics. The next step are the Components, which explain the functionalities for control, automation and visualization.

VISUALIZATION AND CONTROL

The Tile Visualization of the instances set up is created automatically and can be further adapted to one’s personal needs. Control takes place via the visualization. Another option to control the system are the Free Mobile Apps for iOS and Android.
All necessary information can be found under Tile Visualization.

SECURITY

In order to cover the important aspect of security in home automation, external access is not possible by default factory settings. Further information on all security aspects can be looked

up in the section Security. In addition, a security widget checks the configuration and provides an overview of any passwords that may need to be set.

DATA BACKUP

For Data Backup a backup can be created. This can be installed onto a different or a new system. Exact steps for this can be found under load backup. Such a backup is fully compatible even with a Change of Platform.

CONNECT CONTROL

The Connect Control is used for encrypted external communication. It enables secure control and Visualization outside of your local network. Certain functions and modules (e.g. Amazon Alexa, Push Notifications) require a permanent internet connection. This is also guaranteed by the encrypted Connect Control.

REMOTE ACCESS

The Remote Access offers the possibility to configure Symcon from the outside. Only after assigning a password the access from the outside, additionally to the local network access, will be activated. It is thus possible to carry out full remote maintenance. Connect Control is required for this.

MORE INFORMATION AND VIDEOS

Further information and examples can be found on the Symcon YouTube Channel and under Procedures. In addition to the Documentation, the Forum also offers a large amount of information, script collections and ideas, as well as a broad base of professionals and newcomers.

Video Tutorials

For a quick introduction to IP-Symcon, here are some useful video tutorials:



All videos can be found on our YouTube channel.

- Our SymBox
  - Part 1 - Figures, data, facts
  - Part 2: Commissioning + Recovery Tool
  - Part 3: KNX Demo & Pro Console
  - Part 4: Tailscale VPN (Module)
- Basics
  - Part 1: Basics
  - Part 2: Installations on Windows & Raspberry Pi
  - Part 3: Symcon with Docker
  - Part 4: macOS & Synology Installation
  - Part 5: Update with Portainer
  - Part 6: Integrating & configuring devices
  - Part 7: User Interface & Navigation
  - Part 8: Variable display
  - Part 9: Location Control Module
- Logic plans
  - Part 1: Introduction and Basics
  - Part 2: Sublogic plans
  - Part 3: Switch-off delay - Sublogic diagram
  - Part 4: Watchdog - Sublogic plan
  - Part 5: Doorbell trigger - Sublogic diagram
  - Part 6: min/max values logging - Sublogic plan
- Visualization
  - WebFront functions and the WYSIWYG editor
  - Symcon Connect- The secure connection to IP-Symcon and WebFront
  - IP-Symcon Webinar: Preview for visualization
  - IP-Symcon Webinar: Public beta of the new visualization
- Developer Webinars
  - Getting started for developers
  - Dynamic modules
  - Automatic testing
  - Configuration forms and their elements
  - Configurator element and data flow
  - Create actions
  - Git Basics
  - Changes to the PHP SDK for 7.0
  - IP-Symcon Developer Webinar: HTML-SDK
  - Presentation in modules
- Basics - Webinars
  - IP-Symcon Basics Webinar

IP-Smcon Basics Webinar - Part 2
IP-Symcon Developer Webinar: Automated Module Testing
IP-Symcon Webinar Basics Visualization
IP-Symcon Webinar: Tabs & Widgets
Releases
Version 8.0 Preview
Version 7.2
Version 7.1
Version 7.0
Version 6.4
IPSView & IPSWorkflows
Energy Management with IP-Symcon
Version 6.3
Version 6.2
Version 6.1
Version 6.0
Version 5.5
Version 5.4



Missed the event? Check it out!

Set up system
Set up KNX devices incl. feedback addresses
Set up SymBox Pro with KNX Data Secure
KNX: Set up IP-Secure
xComfort
dS-Setup
Simens OZW setup
XML export using ETS6
Setting up a BACnet device in IP-Symcon
Setting up an MQTT client in IP-Symcon
Setting up an MQTT server in IP-Symcon
Setting up a KNX system with configurator in IP-Symcon
XML export using the ETS
Setting up a Z-Wave system in IP-Symcon
Setting up an LCN system in IP-Symcon
Symbox - First setup of the IP-Symcon control center
OPC- EXport using the ETS(from v5.6.5)
Set up an Alexa controller in IP-Symcon in under 10 minutes
Set up a HomeMatic system in IP-Symcon
HomeMatic firewall settings for a functional feedback channel
Setting up a 1-wire system in IP-Symcon
Finding out the IP address in the ETS
Setting up a KNX system in IP-Symcon
OPC export using the ETS(up to v 5.6.4)

Installation

Installation on Windows
Installation on MacOS
Installation on Linux
Installation on Raspberry Pi
Installation on Docker
Installation on QNAP
Installation on Synology
Installation on SymBox
Installation für Catan

INSTALLATION

The initial installation and activation on the various platforms. For this purpose IP-Symcon uses installation steps adapted to the respective system or platform.
Detailed information can be found on the respective system's page.



In order to grant IP-Symcon access to certain functions, e.g. updates, from outside, it may be necessary to set them up in the firewall. Further information can be found under Firewall

UPDATE

IP-Symcon can be updated with a valid subscription.
How to initiate an update can be read on the respective system's page.
For an update, the IP-Symcon server must be contacted to request the latest packages. If this fails, there are several possible causes:

- The subscription has expired. A new one should be purchased from the shop.
- The proxy settings are incorrect.
- The Internet connection is not established.

DATA BACKUP

To avoid data loss there are several ways to backup IP Symcon data.
These are described under Backup.

CHANGE OF PLATFORM

IP-Symcon can move from one platform to another without a new setup.
The procedure is explained under Change of Platform.

UNINSTALLING

How to uninstall IP-Symcon can be read on the respective system's page.

CHANGE LICENSE

Should the license change, it can be changed within IP-Symcon via the management console.
The necessary steps can be found under Change License.

MIGRATION

Migration from an old version to a new version is possible without data loss.
In the left menu all previous migrations can be found.
Individual migrations should be performed from version to version.

Windows

The installation is only possible with a valid subscription. However, an existing backup can be used to restore the software even without subscription. See Change of Platform

A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.

CONTENT

Installation
Update
Uninstallation

INSTALLATION

VERSION REQUIREMENTS

The version overview shows which version of IP-Symcon is compatible with which Windows version.

INSTALLATION

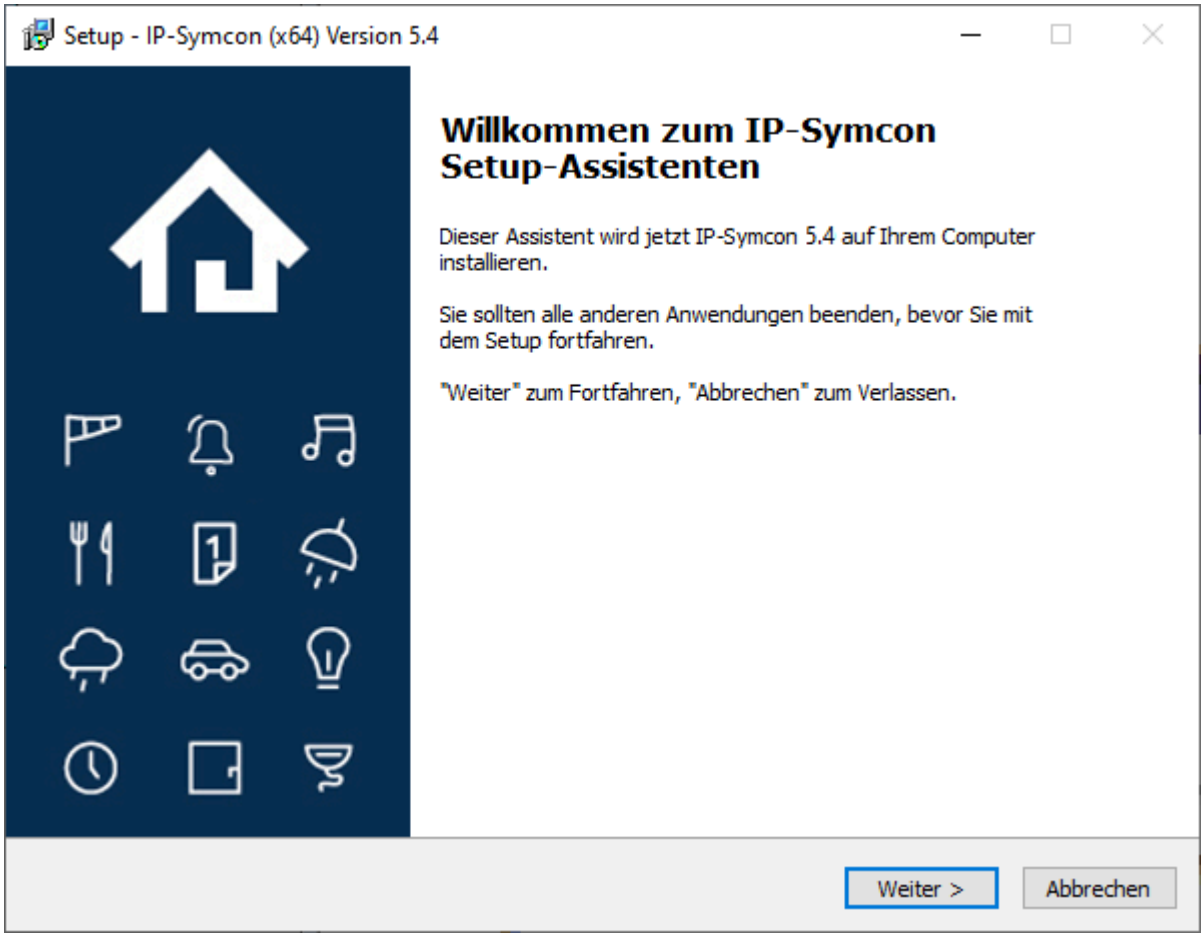
Download: Setup (64-Bit)
IP-Symcon is one of the few programs that enables a clean installation and uninstalls without residue as it does not create a huge load of entries in the Windows registry. It merely stores the license key.

REMARK

An internet connection is required to install IP-Symcon. If no connection is available on the target computer, IP-Symcon needs to be installed on another computer and copied to the target computer afterwards.

SETUP

The following window should appear after starting the setup:



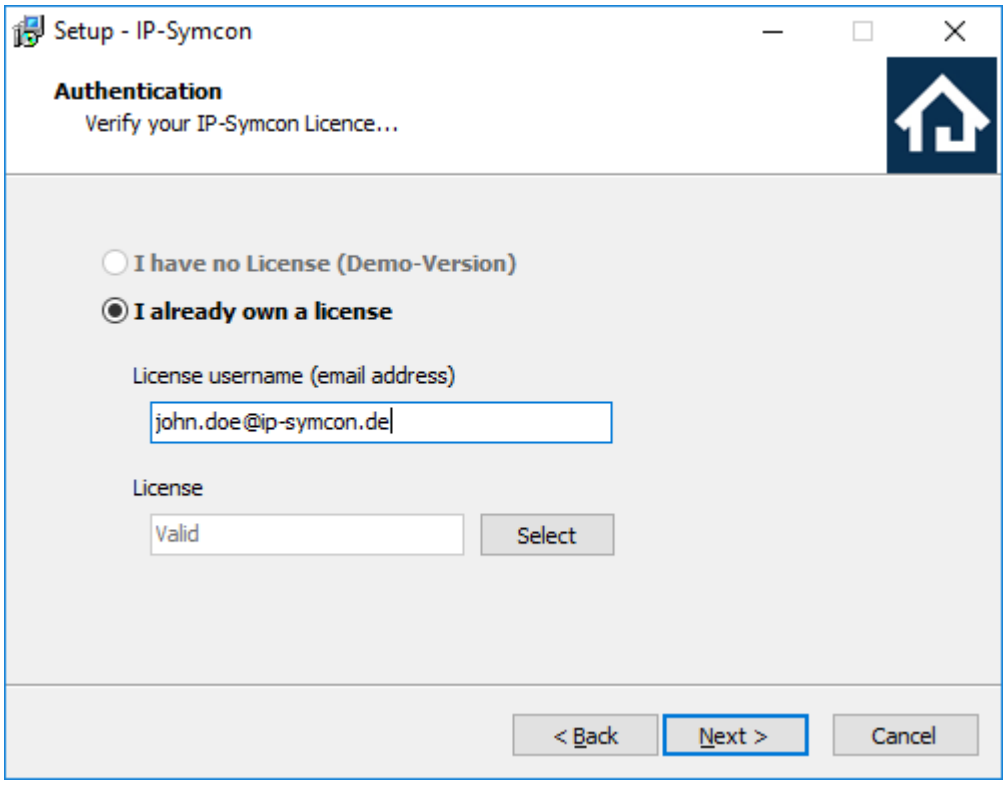
The next step is reached after comfirming the dialog with "Next".

PROVIDE LIZENSE DATA

It is possible to either use a demo version or a complete license by entering a user name and the license file. Afterwards, "Next" finishes the dialog.



The user name usually corresponds to the e-mail address used in the ordering process.



The subscription is checked now.
It is only possible to install IP-Symcon with a valid subscription unless you install the demo version.

When the subscription has expired, an extension can be ordered in the shop: Shop

If no connection to the server can be established, please check to internet connection and the proxy settings.

After accepting the terms of license, a fitting path needs to be chosen (e.g., C:\IP-Symcon) and confirmed with "Next".

During installation the folder C:\ProgramData\Symcon is automatically created. It includes all server data, the individual data of the local installation. If wanted, it is possible to move the server data since version 6.0. Move Server Data

The button "Install" starts the copying of files.
After a click on "OK", the installation on the system is done. Depending on the activated options, the IP-Symcon service and the IP-Symcon tray application are started automatically.
The IP-Symcon WebFront should be reachable at http://ipadresse:3777/.
The IP-Symcon web based Management Console can be called via http://ipadresse:3777/console/. Alternativly, the Pro Console can be used.

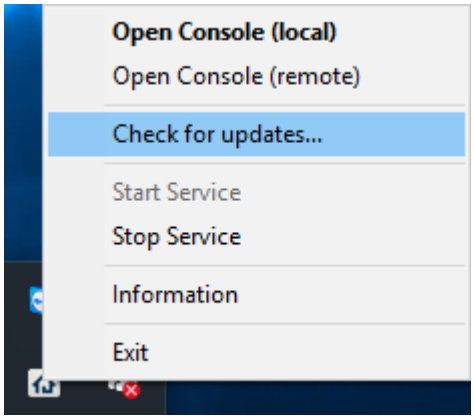
SET REMOTE ACCESS PASSWORD

Remote Access provides a tutorial for configuring remote access.

UPDATE

The newest version can be downloaded and installed by right clicking on the tray icon and choosing "Check for Updates -> Download". The service needs to terminate once during the

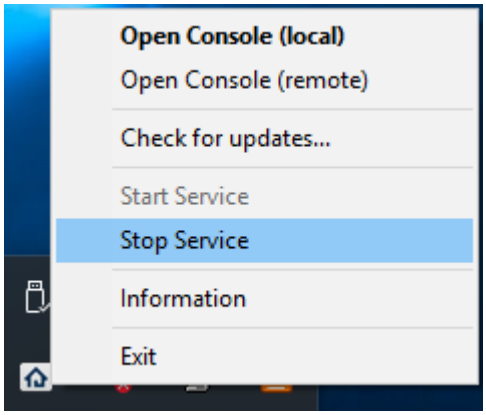
update and is automatically rebooted afterwards.



UNINSTALLATION

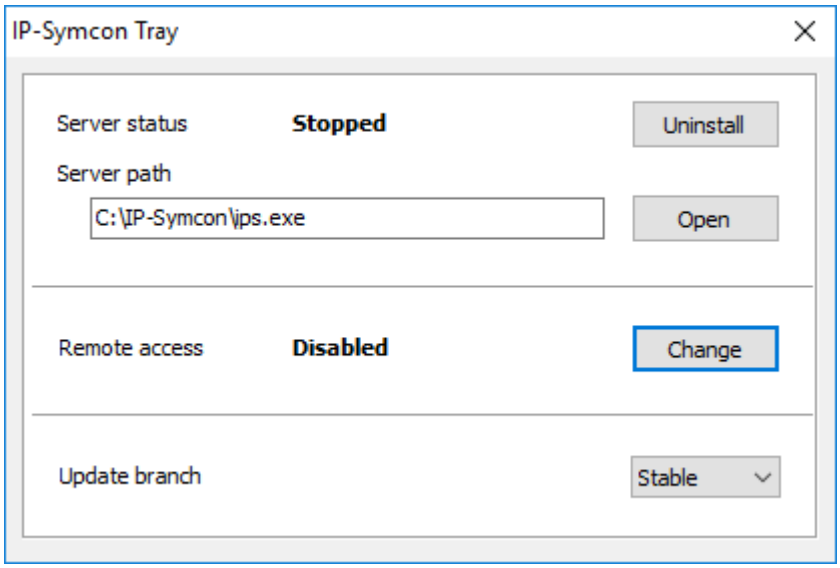
STOP SERVICE

The service can be stopped by clicking "Stop Service" in the context menu of the tray application.



UNINSTALL SERVICE/SERVER

The dialog "Information" can be by the context menu. It should look like this:
Windows



"Uninstall" removes the service from the system.

DELETE IP-SYMCON

Finalize the uninstallation by deleting the complete IP-Symcon folder.

MOVE SERVER DATA

The default folder should not be moved. However, in certain constellations it may be useful, e.g. for automatical backups.

The following steps are possible since version 6.0.
Stop the IP-Symcon service
Enter "regedit" in the Start Menu to open the registry editor
Go to the path HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\IP-Symcon
Create a new string here and name it "DataPath"
Enter the new full path into the string "DataPath"
Move the files from C:\ProgramData\Symcon to the new folder
Verify that the folder C:\ProgramData\Symcon is either empty or does not exist any more
Start the IP-Symcon service
From that time on, IP-Symcon starts from the new path.

Extremely important: The folder C:\ProgramData\Symcon is required to be either not exist or be completely empty

MacOS

The installation is only possible with a valid subscription. However, an existing backup can be used to restore the software even without subscription. See Change of Platform

A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.

CONTENT

Installation
Update
Uninstallation

INSTALLATION

SYSTEM REQUIREMENTS

The version overview shows which version of IP-Symcon is compatible with which MacOS version.

INSTALLATION

Download: Setup
Extract the ZIP file and copy its contents to the programs folder.
Start the program. Install service respectively and start.
The IP-Symcon WebFront should be reachable at http://ipadresse:3777/.
The IP-Symcon web based Management Console can be called via http://ipadresse:3777/console/. Alternativly, the Pro Console can be used.

WHERE IS WHAT?

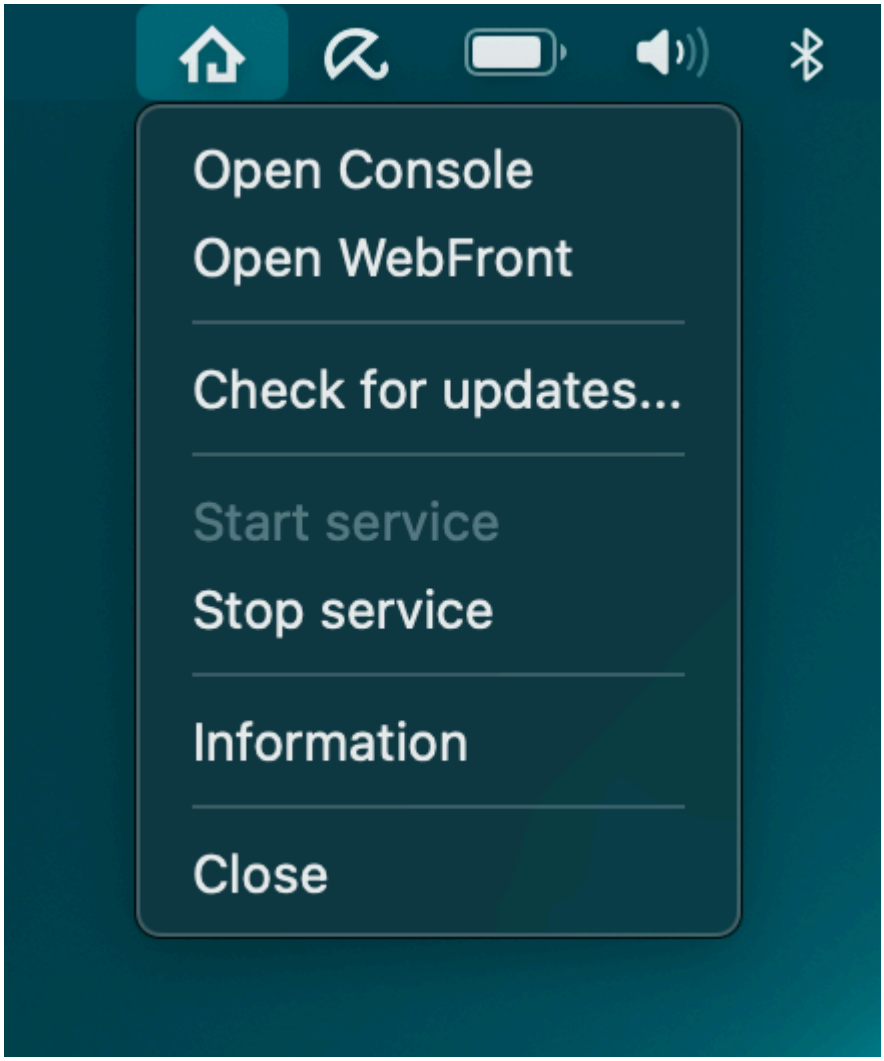
/Library/Application Support/Symcon (Settings, scripts, media...)
/Library/Logs/Symcon (Logfiles...)

SET REMOTE ACCESS PASSWORD

Remote Access provides a tutorial for configuring remote access.

UPDATE

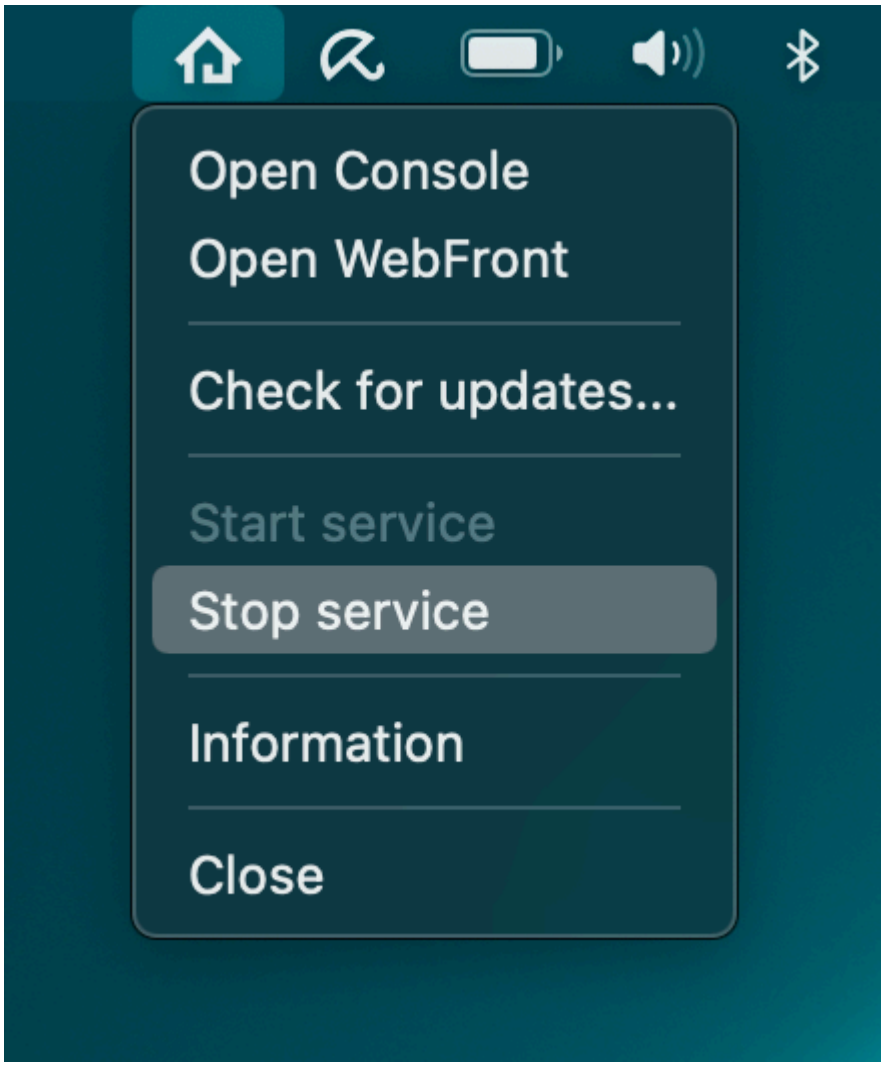
The latest version can be downloaded and installed by clicking on the tray icon and choosing "Check for Updates -> Install Update". Once the installation is complete, the service needs to be stopped and restarted via the tray icon.



UNINSTALLATION

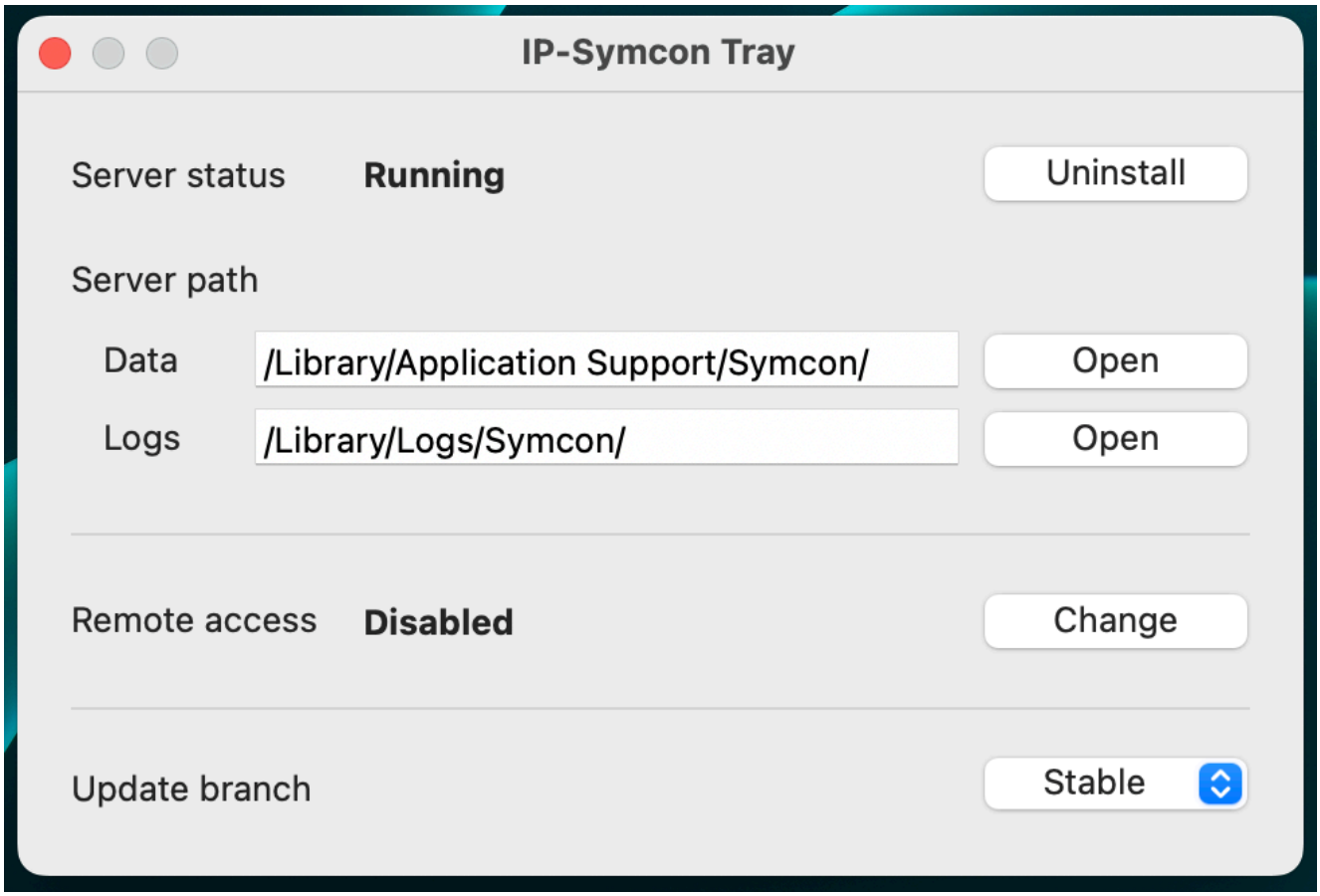
STOPPING SERVICE/SERVER

The service can be stopped by clicking "Stop Service" in the context menu of the tray application.



UNINSTALL SERVICE/SERVER

Via the "information" dialog the context menu can be opened. It should look like this:
MacOS



"Uninstall" removes the service from the system.

DELETE IP-SYMCON

To finalize the uninstallation, the complete IP-Symcon folder must be deleted or moved to the bin.

Linux



The installation is only possible with a valid subscription. However, an existing backup can be used to restore the software even without subscription. See Change of Platform



A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.

CONTENT

Installation
Update
Uninstallation

INSTALLATION

VERSION REQUIREMENTS

The version overview shows which version of IP-Symcon is compatible with which Linux version.

PREPARATION

Start with executing the following command and verify the correct time or set the time zone if needed

```
date
```

If the time zone needs to be set

```
sudo dpkg-reconfigure tzdata
```

INSTALLATION

```
wget -qO- https://apt.symcon.de/install.sh | bash /dev/stdin
```

Execute the following commands:

```
sudo apt-get update
sudo apt-get install symcon
```

The IP-Symcon WebFront should be reachable at http://ipadresse:3777/.
The IP-Symcon web based Management Console can be called via http://ipadresse:3777/console/. Alternativly, the Pro Console can be used.

UPDATE

Update to the most current version:

```
sudo apt-get update
sudo apt-get upgrade
```

HOW CAN I START AND STOP THE IP-SYMCON SERVICE?

```
sudo /etc/init.d/symcon start
sudo /etc/init.d/symcon stop
sudo /etc/init.d/symcon restart
```

WHERE IS WHAT?

/usr/bin/symcon - Executable
/usr/share/symcon/ - Static Data (IP-Symcon installation)
/var/lib/symcon/ - Variable Data (Settings, scripts, media...)
/var/log/symcon/ - Log Files (Logfiles...)

HOW CAN I CHECK IF THE SERVICE RUNS CORRECTLY?

```
sudo ps x | grep symcon
```

HOW CAN I VIEW/FOLLOW THE LOG FILE?

tail -f /var/log/symcon/logfile.log

SET REMOTE ACCESS PASSWORD

Remote Access provides a tutorial for configuring remote access.

UPDATE

Update to the most current version:

```
sudo apt-get update
sudo apt-get upgrade
```

HOW CAN I START AND STOP THE IP-SYMCON SERVICE?

```
sudo /etc/init.d/symcon start
sudo /etc/init.d/symcon stop
sudo /etc/init.d/symcon restart
```

WHERE IS WHAT?

/usr/bin/symcon - Executable
/usr/share/symcon/ - Static Data (IP-Symcon installation)
/var/lib/symcon/ - Variable Data (Settings, scripts, media...)
/var/log/symcon/ - Log Files (Logfiles...)

HOW CAN I CHECK IF THE SERVICE RUNS CORRECTLY?

```
sudo ps x | grep symcon
```

HOW CAN I VIEW/FOLLOW THE LOG FILE?

```
tail -f /var/log/symcon/logfile.log
```

SET REMOTE ACCESS PASSWORD

Remote Access provides a tutorial for configuring remote access.

UPDATE

The following commands need to be used to update IP-Symcon to the newest version.

```
sudo apt-get update
sudo apt-get upgrade
```



When updating from a version prior to 9.0, the new signing key needs to be downloaded once:

wget -qO- https://apt.symcon.de/symcon.key | gpg --dearmor | sudo tee /usr/share/keyrings/symcon.gpg > /dev/null
sudo chmod 644 /usr/share/keyrings/symcon.gpg

The service is terminated and rebooted automatically afterwards.

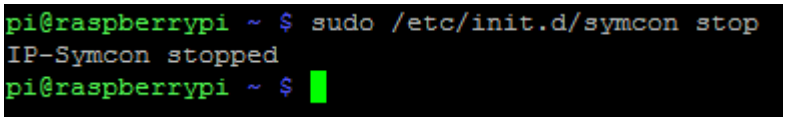
UNINSTALLATION

The IP-Symcon service needs to be terminated before uninstalling.

TERMINATE SERVICE/SERVER

The command to terminate IP-Symcon server needs to be entered via console. This is identical for Linux and Raspberry Pi.

```
sudo /etc/init.d/symcon stop
```



UNINSTALL SERVICE/SERVER

The shown command needs to be entered via console. This is identical for Linux and Raspberry Pi.

```
sudo apt-get purge symcon
```

```
pi@raspberrypi ~ $ sudo apt-get purge symcon
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete wurden automatisch installiert und werden nicht mehr benöti
gt:
  libfontconfig1-dev libpthread-stubs0 libpthread-stubs0-dev libx11-dev
  libxau-dev libxcb1-dev libxdmcp-dev libxpm-dev x11proto-core-dev
  x11proto-input-dev x11proto-kb-dev xorg-sgml-doctools xtrans-dev
Verwenden Sie »apt-get autoremove«, um sie zu entfernen.
Die folgenden Pakete werden ENTFERNT:
  symcon*
0 aktualisiert, 0 neu installiert, 1 zu entfernen und 0 nicht aktualisiert.
Nach dieser Operation werden 0 B Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren [J/n]? j
(Lese Datenbank ... 104352 Dateien und Verzeichnisse sind derzeit installiert.)
Entfernen von symcon ...
IP-Symcon is not running.
dpkg: Warnung: Während Entfernens von symcon ist Verzeichnis »/etc/symcon/forms«
 nicht leer, wird daher nicht gelöscht
dpkg: Warnung: Während Entfernens von symcon ist Verzeichnis »/etc/symcon/webfro
nt« nicht leer, wird daher nicht gelöscht
Trigger für man-db werden verarbeitet ...
pi@raspberrypi ~ $
```

DELETE IP-SYMCON

Finalize the uninstallation by deleting the complete IP-Symcon-Data folder (/var/lib/symcon).

Raspberry Pi



The installation is only possible with a valid subscription. However, an existing backup can be used to restore the software even without subscription. See Change of Platform



A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.

CONTENT

Installation
Update
Uninstallation

INSTALLATION

VERSION REQUIREMENTS

The version overview shows which version of IP-Symcon is compatible with which Raspberry Pi version.

PREPARATION

Start with executing the following command and verify the correct time or set the time zone if needed

```
date
```

If the time zone needs to be set

```
sudo raspi-config
```

INSTALLATION

```
wget -qO- https://apt.symcon.de/install.sh | bash /dev/stdin
```

Execute the following commands on the shell:

```
sudo apt-get update
sudo apt-get install symcon
```

The IP-Symcon WebFront should be reachable at http://ipadresse:3777/.
The IP-Symcon web based Management Console can be called via http://ipadresse:3777/console/. Alternativly, the Pro Console can be used.

UPDATE

Update to the most current version

```
sudo apt-get update
sudo apt-get upgrade
```

HOW CAN I START AND STOP THE IP-SYMCON SERVICE?

```
sudo /etc/init.d/symcon start
sudo /etc/init.d/symcon stop
sudo /etc/init.d/symcon restart
```

WHERE IS WHAT?

/usr/bin/symcon - Executable
/usr/share/symcon/ - Static Data (IP-Symcon installation)
/var/lib/symcon/ - Variable Data (Settings, scripts, media...)
/var/log/symcon/ - Log Files (Logfiles...)

HOW CAN I CHECK IF THE SERVICE RUNS CORRECTLY?

```
sudo ps x | grep symcon
```

HOW CAN I VIEW/FOLLOW THE LOG FILE?

```
tail -f /var/log/symcon/logfile.log
```

SET REMOTE ACCESS PASSWORD

Remote Access provides a tutorial for configuring remote access.

UPDATE

The following commands need to be used via terminal to update IP-Symcon to the newest version.

```
sudo apt-get update
sudo apt-get upgrade
```



When updating from a version prior to 9.0, the new signing key needs to be downloaded once:

wget -qO- https://apt.symcon.de/symcon.key | gpg --dearmor | sudo tee /usr/share/keyrings/symcon.gpg > /dev/null
sudo chmod 644 /usr/share/keyrings/symcon.gpg

The service is terminated and rebooted automatically afterwards.

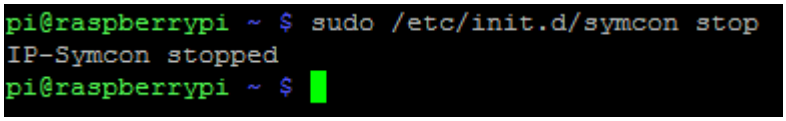
UNINSTALLATION

The IP-Symcon service needs to be terminated before uninstalling.

TERMINATE SERVICE/SERVER

The command to terminate IP-Symcon server needs to be entered via console. This is identical for Linux and Raspberry Pi.

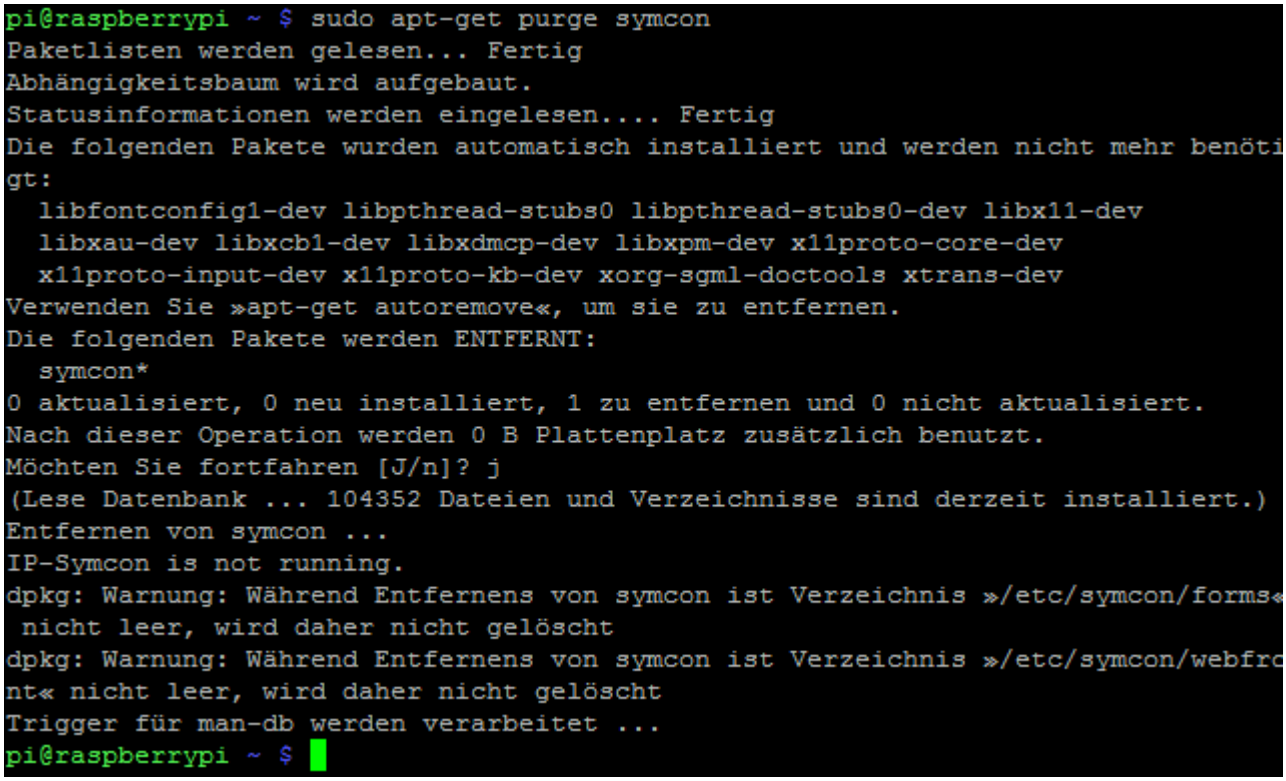
```
sudo /etc/init.d/symcon stop
```



UNINSTALL SERVICE/SERVER

The shown command needs to be entered via console. This is identical for Linux and Raspberry Pi.

```
sudo apt-get purge symcon
```



DELETE IP-SYMCON

Finalize the uninstallation by deleting the complete IP-Symcon-Data folder (/var/lib/symcon).

Docker



The installation is only possible with a valid subscription.



A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.



There are specific installation instructions for QNAP and Synology

CONTENT

Installation
Update
Hints
Limitations

INSTALLATION

REQUIREMENTS

Intel/AMD 64-Bit, ARM7, ARM64 CPU
Docker

HINTS

As containers cannot be "updated", they need to be deleted and recreated in the new version. This does not cause complications as the data is stored outside of the container.
Using KNX or HomeMatic requires the corresponding ports (KNX = 52000, HomeMatic = 5544) to be mapped. In addition, within the special switches NATSupport needs to be activated and the new property PublicIP needs to be set to the IP address of the Docker Host. If this is not done, no responses are received!

LIMITATIONS

Currently, the language is set to de_DE

Currently, the time zone is set to Europe/Berlin
If the container should not run interactive but detached, the flag -d needs to be used
Currently only port 3777 is mapped. If further ports are required, these need to be set as well!

INSTALL

```
docker pull symcon/symcon:stable
```

START

```
docker run --rm \
--name symcon \
--hostname symcon \
-p 3777:3777 \
-v /opt/symcon/data:/var/lib/symcon \
-v /opt/symcon/log:/var/log/symcon \
-v /opt/symcon:/root \
symcon/symcon:stable
```

The IP-Symcon WebFront should be reachable at http://ipadresse:3777/.
The IP-Symcon web based Management Console can be called via http://ipadresse:3777/console/. Alternativly, the Pro Console can be used.

UPDATE

Execute Install again
Execute Start again
This will delete and recreate the container
Data is stored outside the container and is persistant

COMFORTABLE UPDATES WITH PORTAINER

The tool "Portainer" helps with updating Docker containers and enables updating with the press of a button. The web based tool is available for free and can be installed as Docker container.

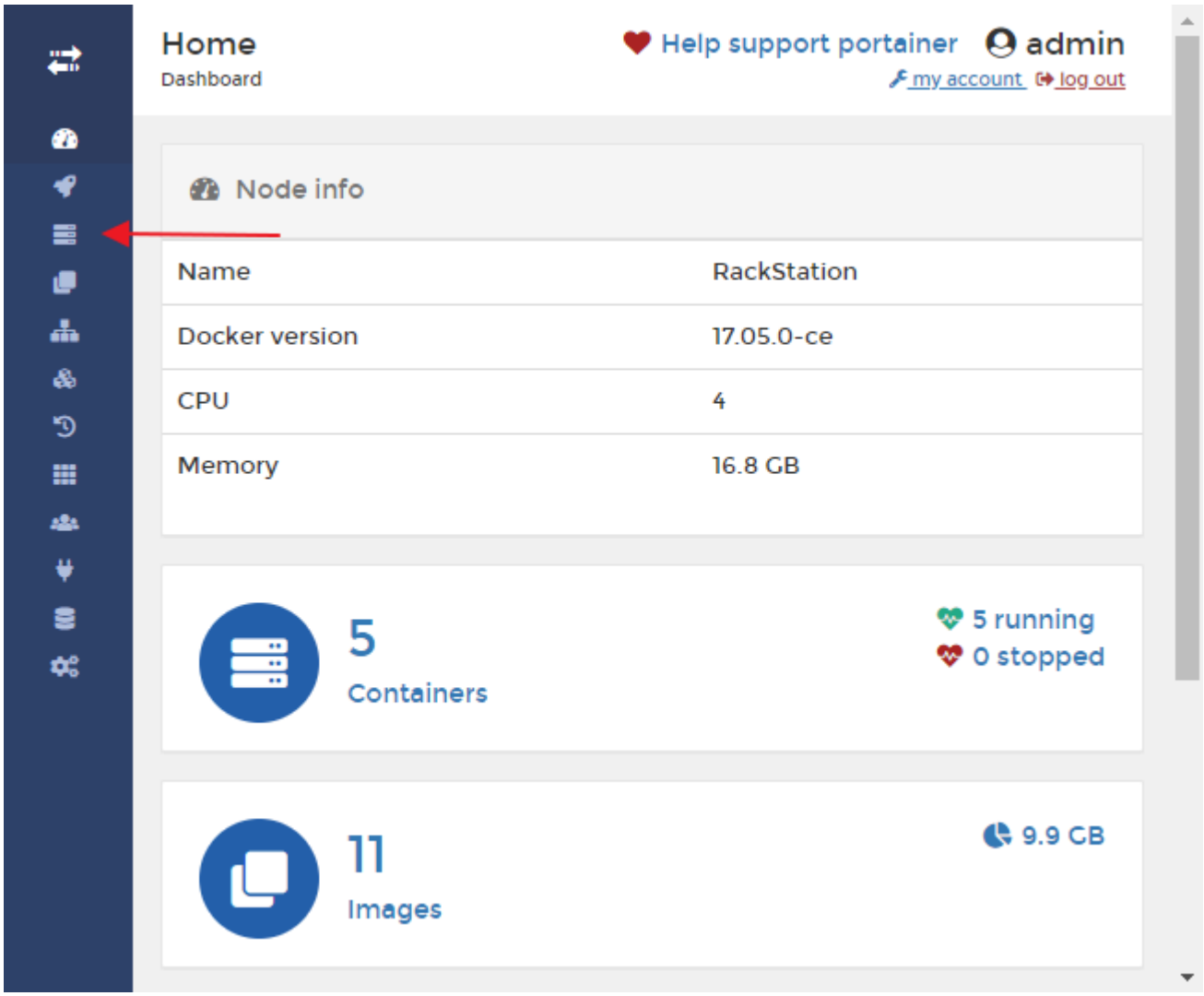
INSTALL

```
docker volume create portainer_data
docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock
-v portainer_data:/data portainer/portainer
```

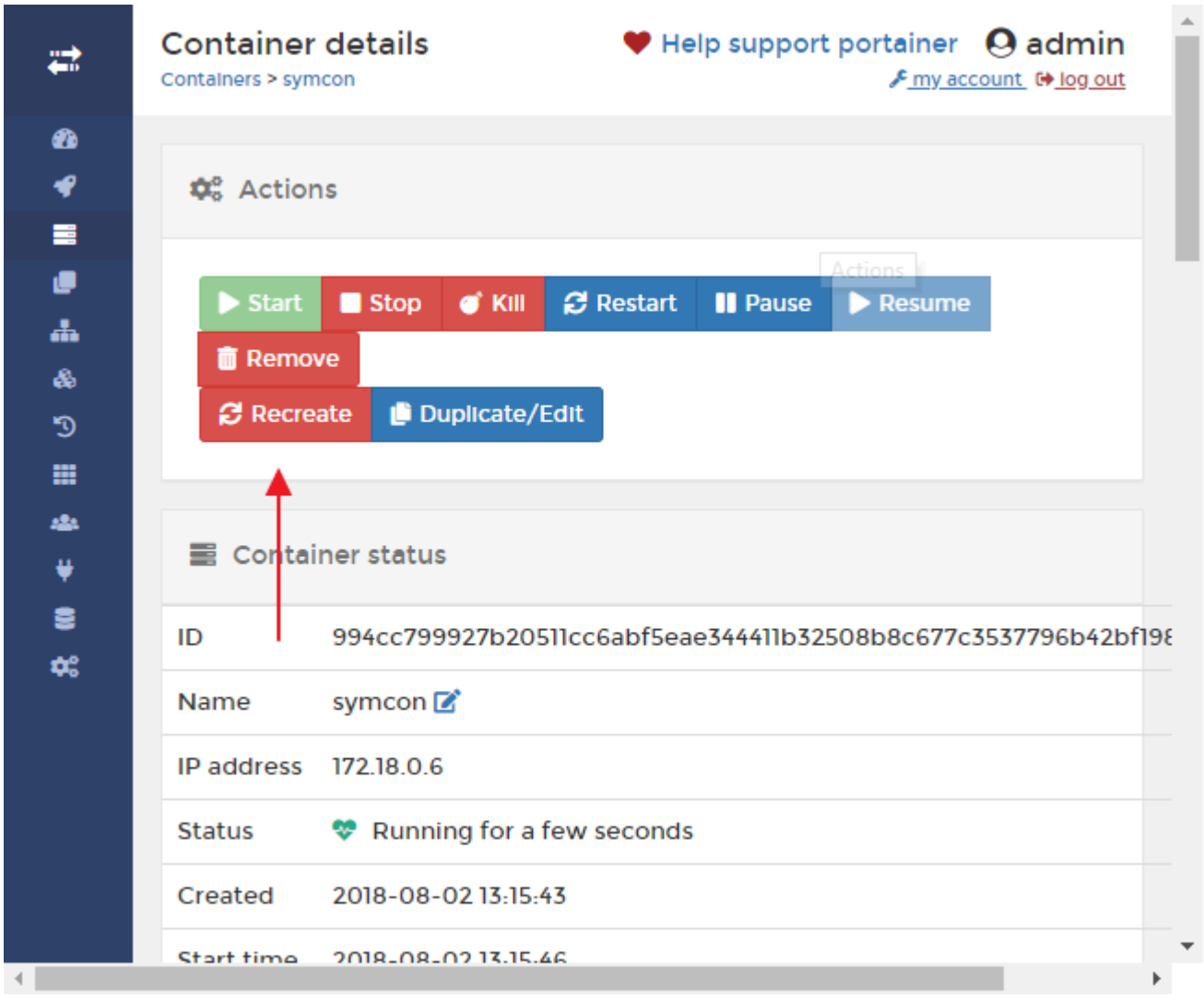


It may be required to run the command "docker run" as "sudo docker run"

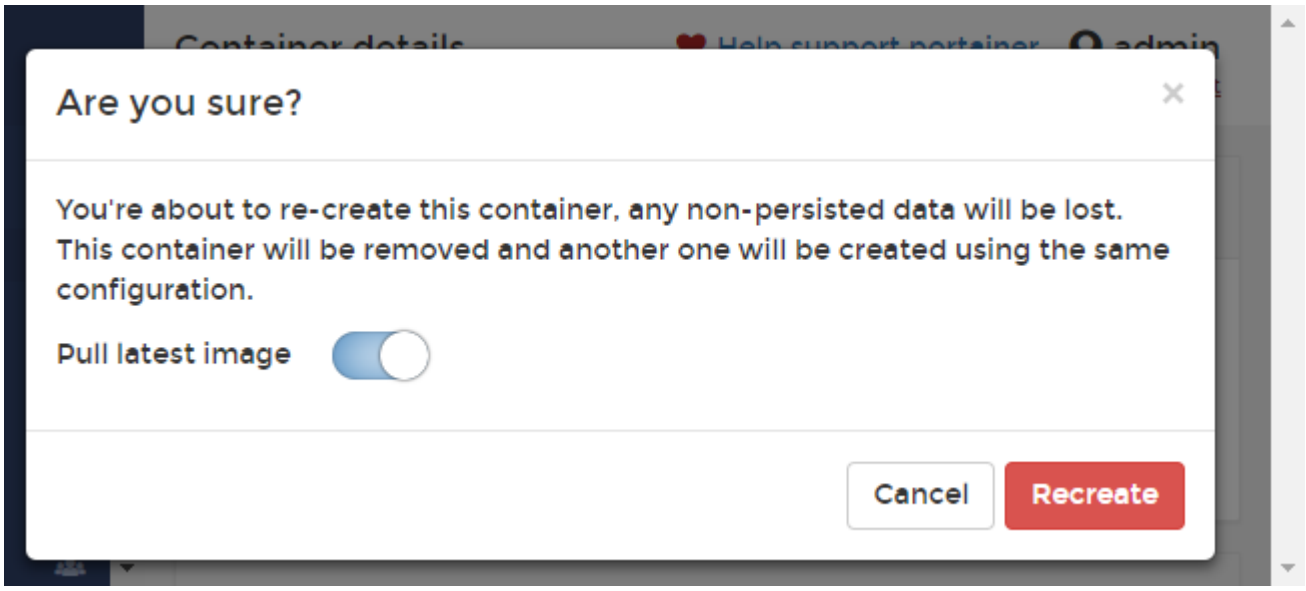
After the installation, the interface as available under http://127.0.0.1:9000/. Eventually, the IP address needs to be adjusted. During the first configuration, a password needs to be defined and a connection needs to be established to the local server in a next step. After succesfull installation, the interface should look as shown below.



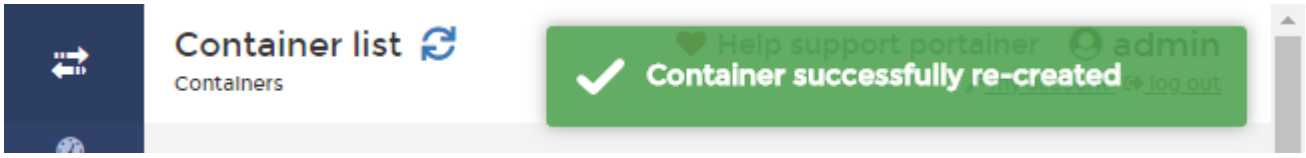
In the tab "Containers", the "symcon" container can be selected. The option "Recreate" enables a comfortable update. In this process, the container will be deleted, the new image downloaded, and installed with the same settings. Depending on the internet connection, this may take some minutes.



It is important, that the option "Pull latest image" is activated in the confirmation dialog.



After the update, the following message is displayed for a short time.



NATSUPPORT FOR KNX AND HOMEMATIC

Due to the container structure KNX and HomeMatic messages contain an IP address as response address that is used only internally and as such, responses could not be received as intended. To receive these messages correctly, NATSupport needs to be set up.

The Special Switches "NATSupport" needs to be activated and IP-Symcon restarted afterwards.

The port of the affected system needs to be mapped to the container. For KNX, this is 52000 by default and for HomeMatic this is 5544 by default. So, for example, the port 52000 would need to be mapped to 52000 for KNX.

A new field "PublicIP" appears in the affected I/O or Splitter instances. The external IP of Docker needs to be entered here. For KNX, this needs to be set at the splitter "EIB Gateway" and for HomeMatic at the I/O "HomeMatic Socket".

QNAP



The installation is only possible with a valid subscription.



A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.

CONTENT

Installation
Update
Uninstallation
Hints

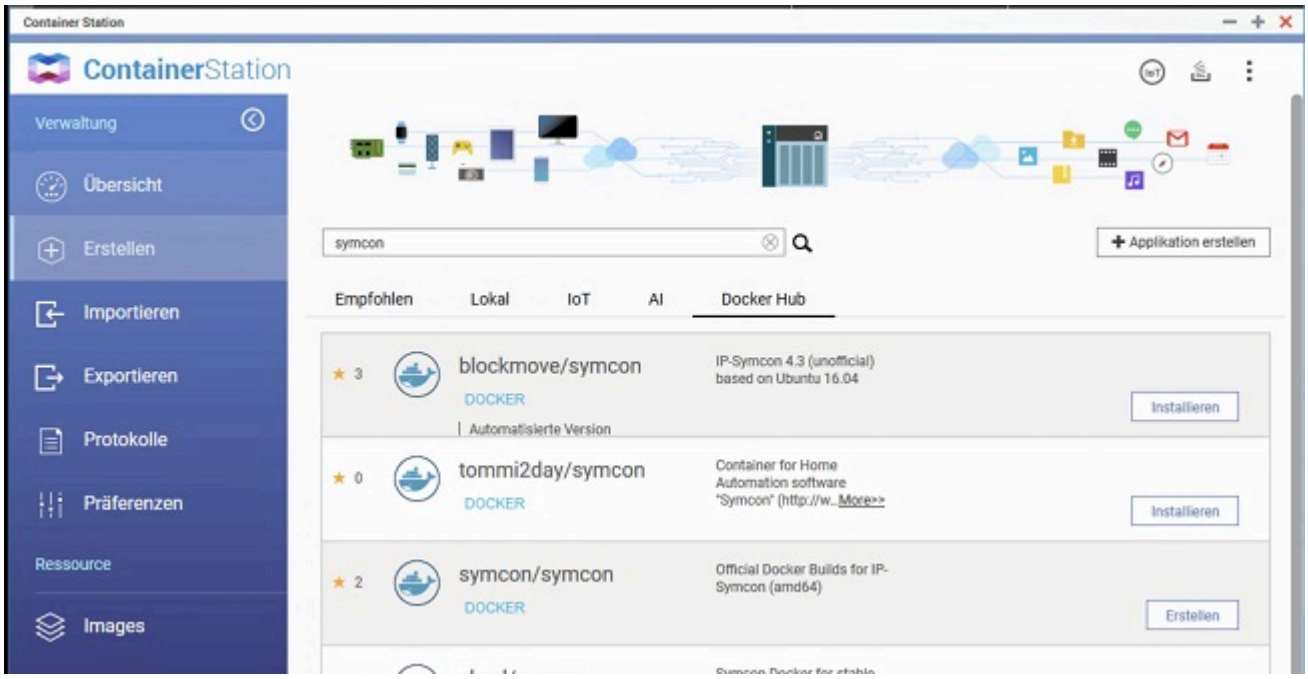
INSTALLATION

REQUIREMENTS

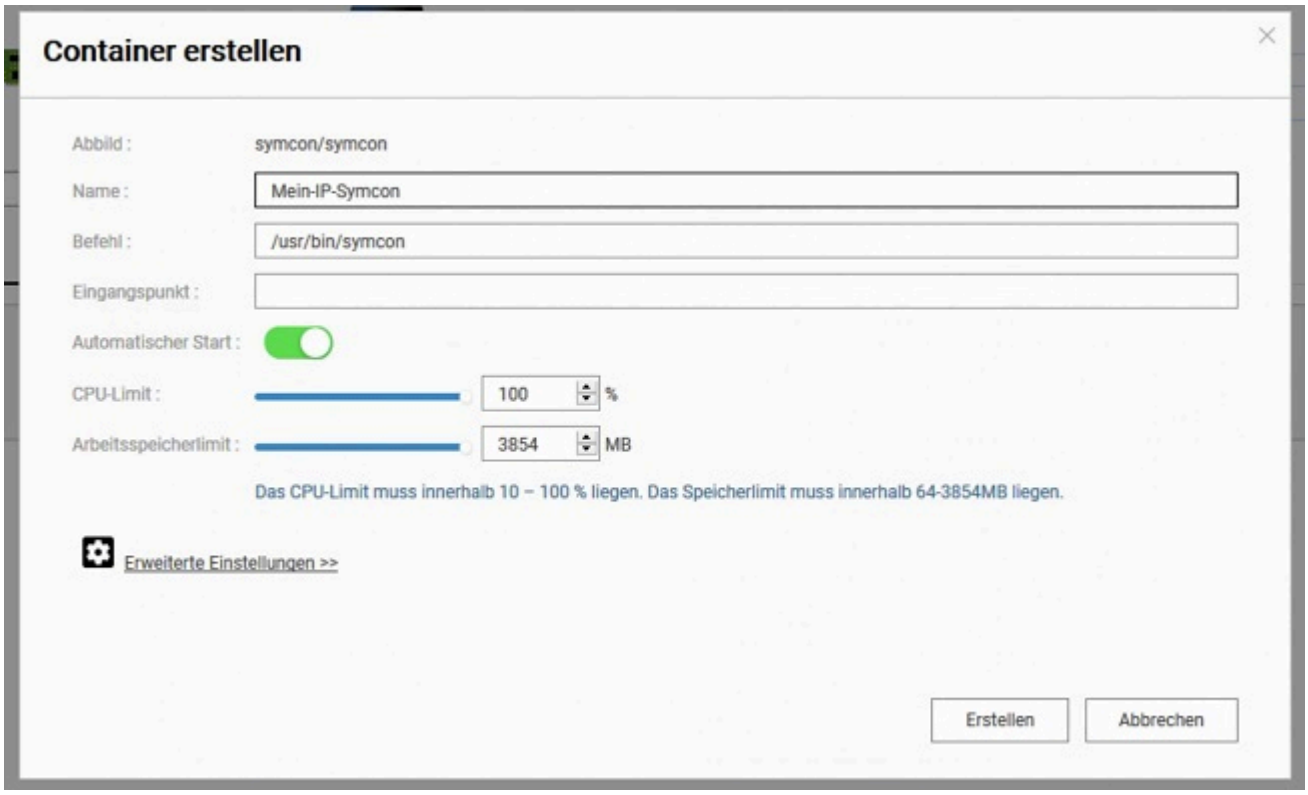
NAS with Container Station support

INSTALL

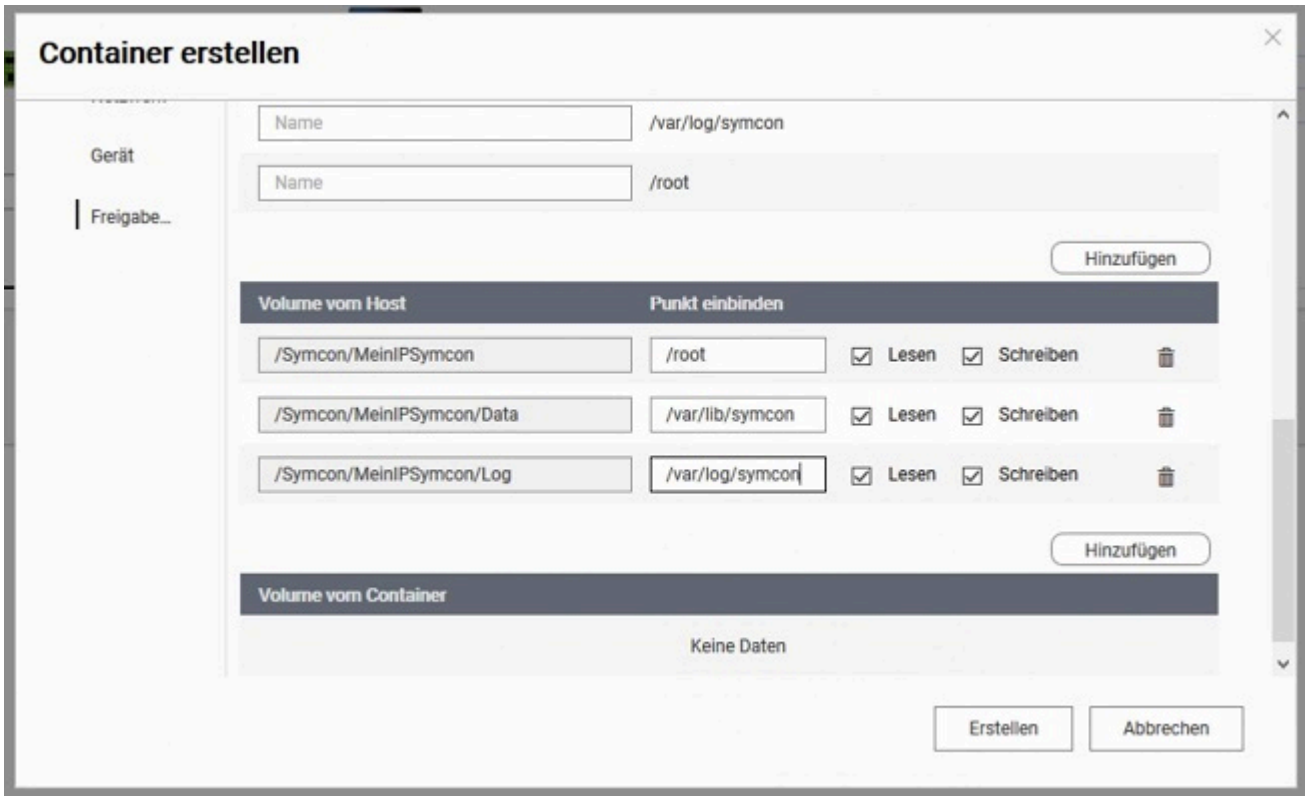
Open Container Station
Search for symcon/symcon in the Docker Hub and install/create the image (choose a tag, e.g., latest)



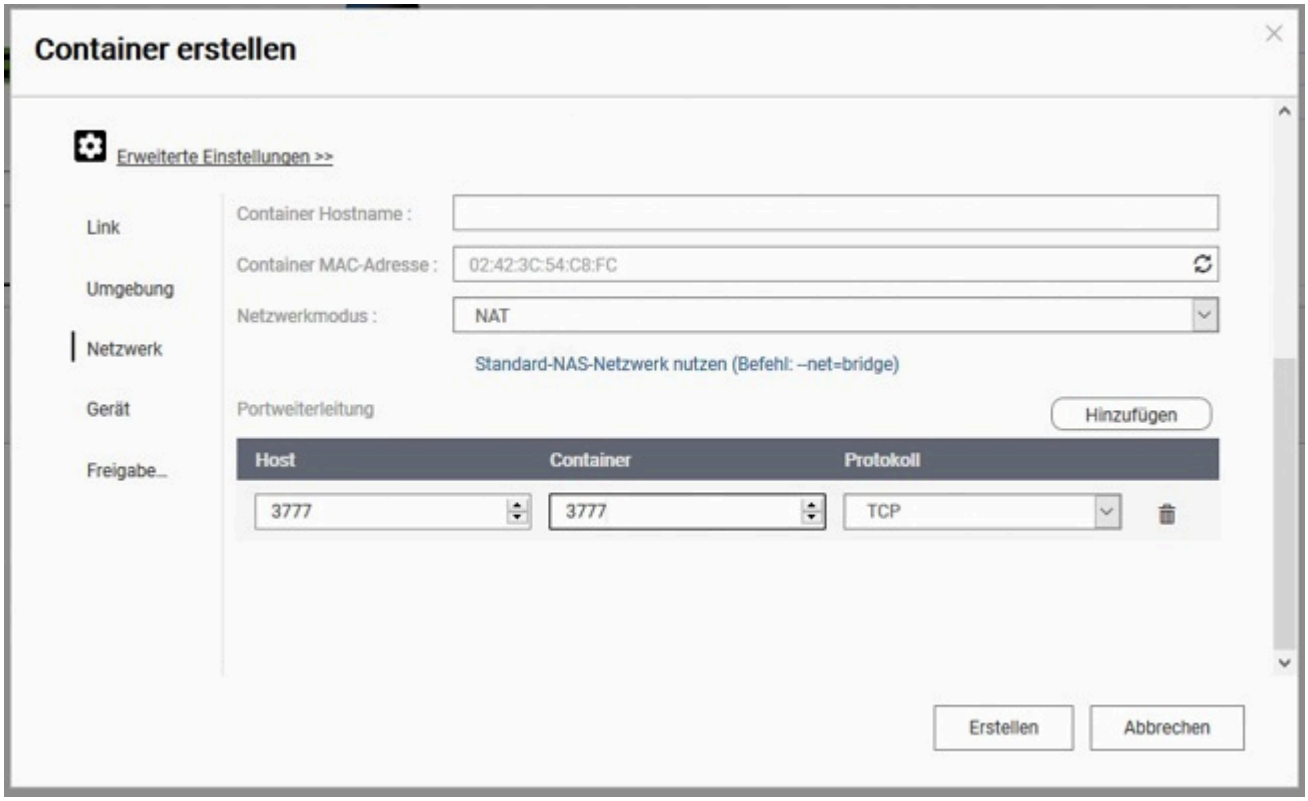
Choose a name and open Advanced Settings



Adjust Share Settings (/root, /var/lib/symcon, /var/log/symcon need to be stated!)



Adjust Network Settings (3777 to 3777 TCP)



Confirm with Create to create and start the container
The IP-Symcon WebFront should be reachable at http://ipadresse:3777/.
The IP-Symcon web based Management Console can be called via http://ipadresse:3777/console/. Alternativly, the Pro Console can be used.

UPDATE

Updating requires the removal and recreation of the container, as described under Install. The tool "Portainer" can significantly simplify updating. A tutorial can be found in the Docker Area .

UNINSTALLATION

For the uninstallation, the container needs to be stop and removed afterwards. The linked directories for the files need to be removed manually.

HINTS

NATSUPPORT FOR KNX AND HOMEMATIC

Due to the container structure KNX and HomeMatic messages contain an IP address as response address that is used only internally and as such, responses could not be received as intended. To receive these messages correctly, NATSupport needs to be set up.

The Special Switches "NATSupport" needs to be activated and IP-Symcon restarted afterwards.
The port of the affected system needs to be mapped to the container. For KNX, this is 52000 by default and for HomeMatic this is 5544 by default. So, for example, the port 52000 would need to be mapped to 52000 for KNX.
A new field "PublicIP" appears in the affected I/O or Splitter instances. The external IP of Docker needs to be entered here. For KNX, this needs to be set at the splitter "EIB Gateway" and for HomeMatic at the I/O "HomeMatic Socket".

Synology



The installation is only possible with a valid subscription.



A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.

CONTENT

Installation
Update
Uninstallation
Hints

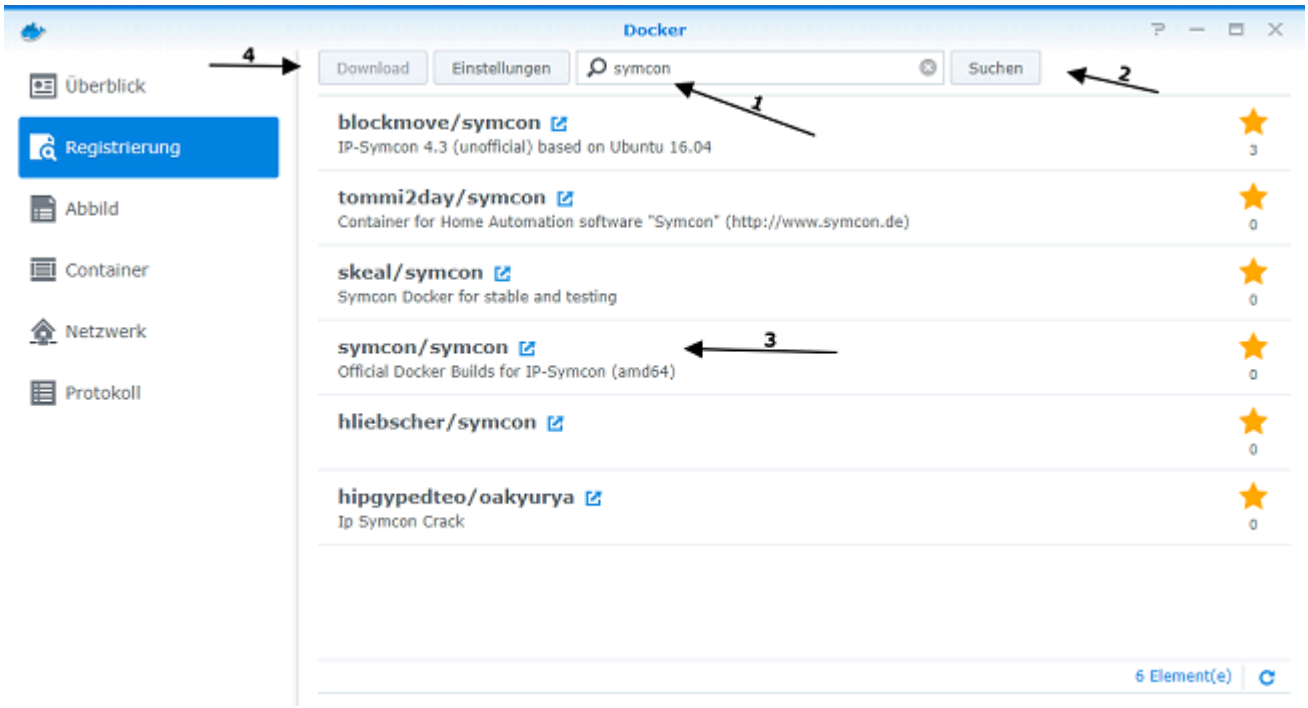
INSTALLATION

REQUIREMENTS

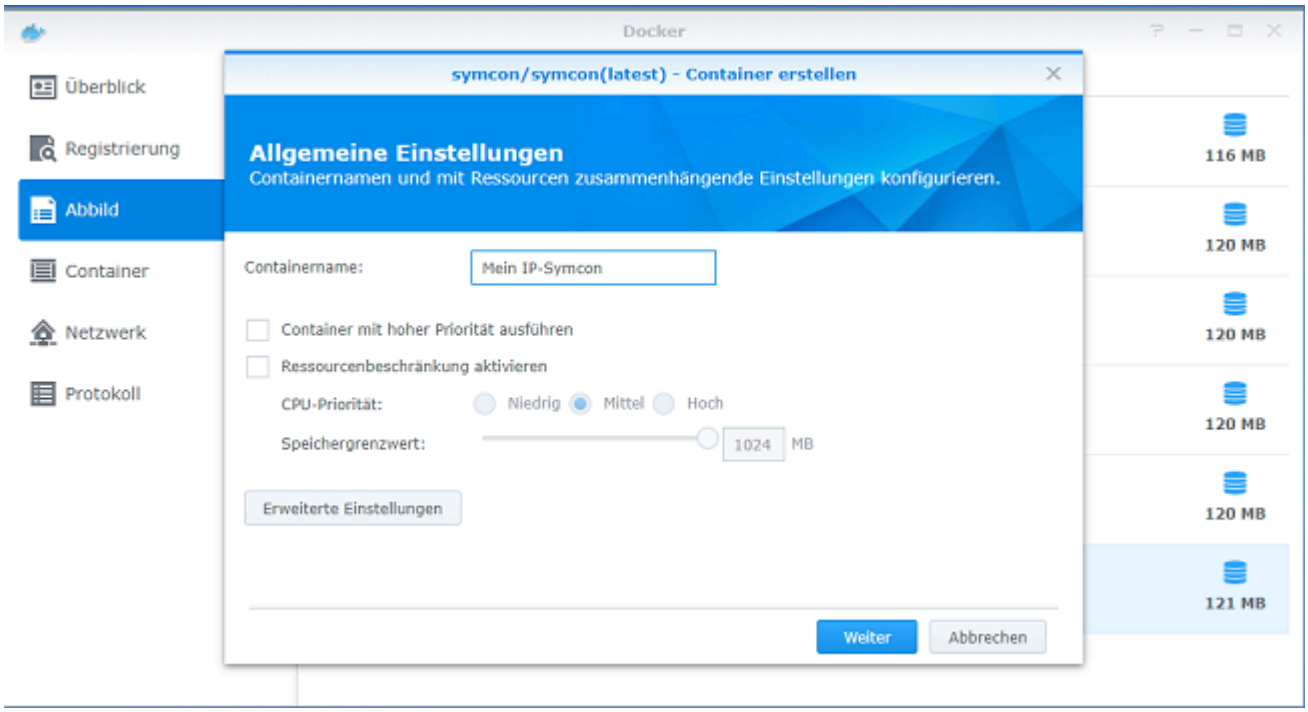
NAS with Container Manager support

INSTALL

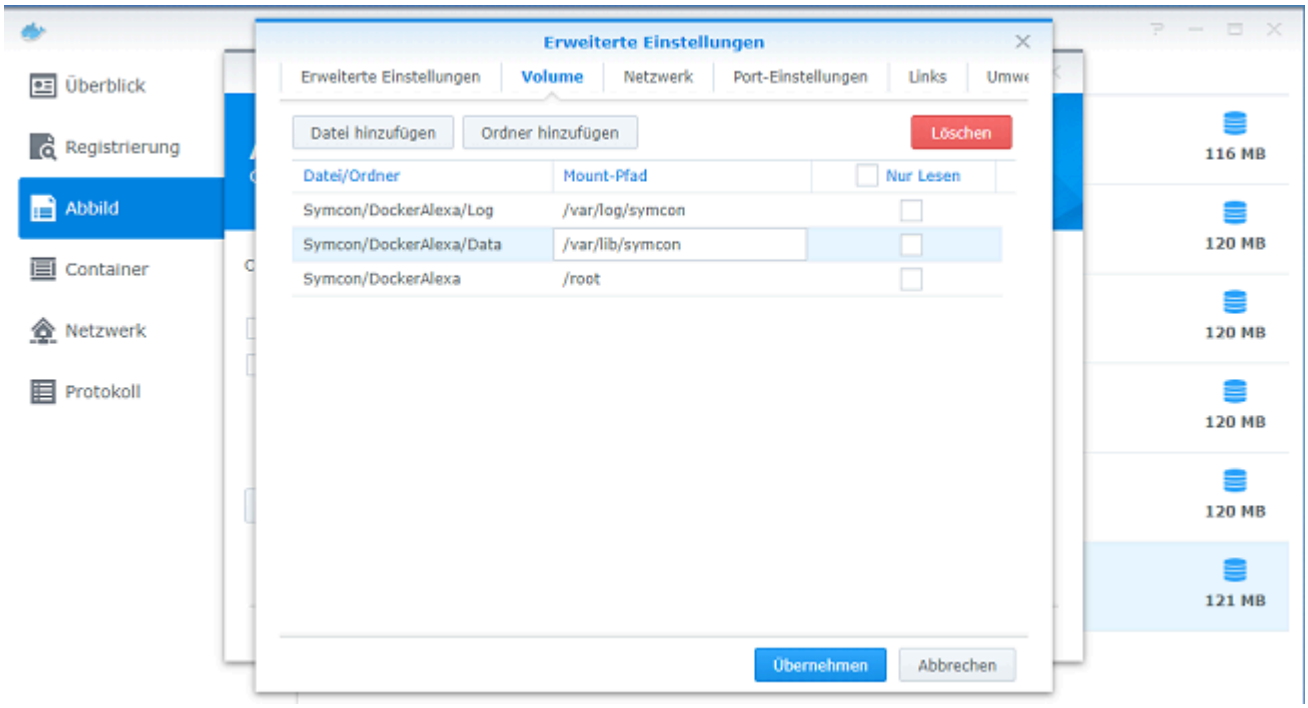
Search for Symcon in the registry and download the image (choose a tag, e.g., latest)



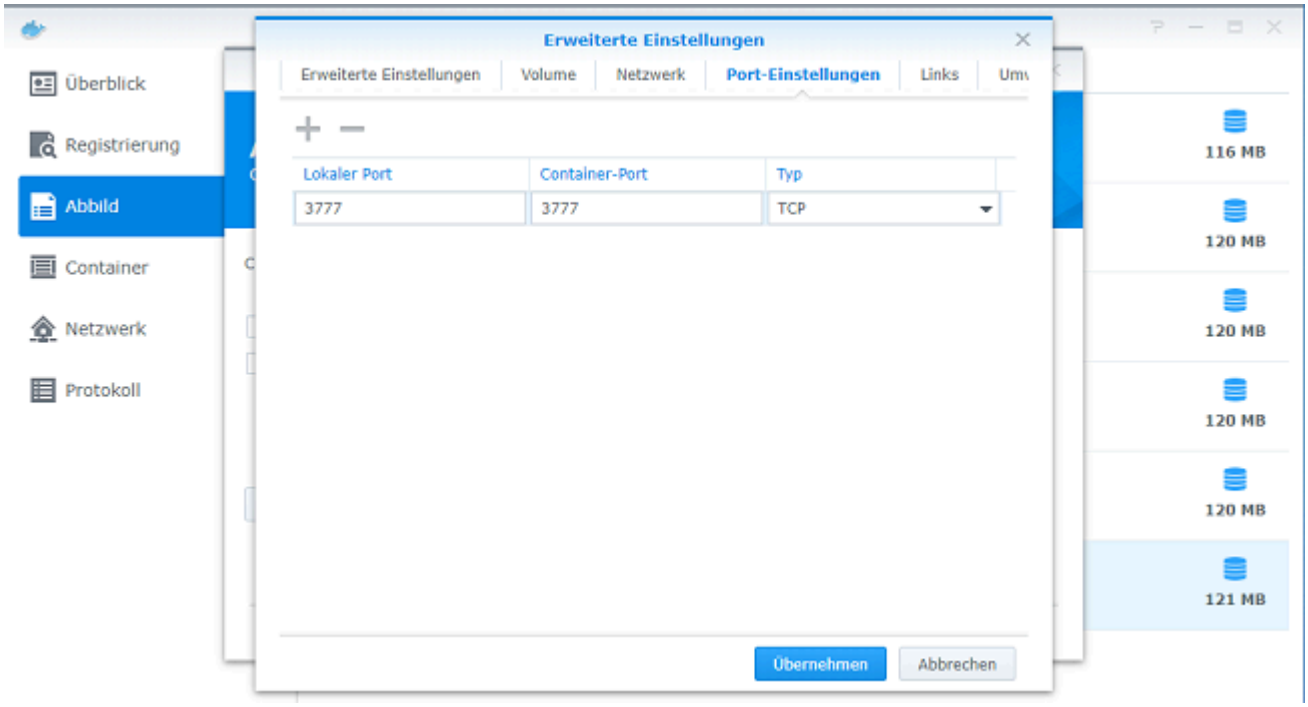
Choose the new image under Image and click Start
Choose a name



Adjust the Volume Settings (/root, /var/lib/symcon, /var/log/symcon need to be stated!)



Adjust Port Settings (3777 to 3777)



Confirm with OK, click Next and start the container with Apply
The IP-Symcon WebFront should be reachable at http://ipadresse:3777/.
The IP-Symcon web based Management Console can be called via http://ipadresse:3777/console/. Alternativly, the Pro Console can be used.

UPDATE

Updating requires the removal and recreation of the container, as described under Install.
The tool "Portainer" can significantly simplify updating. A tutorial can be found in the Docker Area .

UNINSTALLATION

For the uninstallation, the container needs to be stop and removed afterwards. The linked directories for the files need to be removed manually.

HINTS

NATSUPPORT FOR KNX AND HOMEMATIC

Due to the container structure KNX and HomeMatic messages contain an IP address as response address that is used only internally and as such, responses could not be received as intended. To receive these messages correctly, NATSupport needs to be set up.

The Special Switches "NATSupport" needs to be activated and IP-Symcon restarted afterwards.

The port of the affected system needs to be mapped to the container. For KNX, this is 52000 by default and for HomeMatic this is 5544 by default. So, for example, the port 52000 would need to be mapped to 52000 for KNX.

A new field "PublicIP" appears in the affected I/O or Splitter instances. The external IP of Docker needs to be entered here. For KNX, this needs to be set at the splitter "EIB Gateway" and for HomeMatic at the I/O "HomeMatic Socket".

SymBox



The installation is only possible with a valid subscription. However, an existing backup can be used to restore the software even without subscription. See Change of Platform



A regular backup is recommended to ensure that your system is as current as possible in case of a defect. Even more, an active subscription is required to download IP-Symcon anew in case of a complete loss of data.

INSTALLATION/UPDATE/RESET

For more informations see the Installation manual .

Catan

Important installation notes

The WBM is most easily accessed by connecting the Catan C1 to the PC via USB and then opening https://169.254.252.10/.
The Catan C1 currently does not support DHCP. Therefore, the IP address and gateway must be set manually. We recommend using the LAN1/LAN2 network ports (top sockets). The gateway is usually at .1. That means if your IP address is 192.168.178.250, then the gateway would be 192.168.178.1.
Before installing the Symcon app, it is mandatory to install the current firmware (version 2026.0.3 LTS). It is available for download on the Phoenix Contact website (Controller Catan C1) in the Firmware section. Installation is performed in the WBM under "System -> Update".
A prerequisite for installing the Symcon app is that the switch under "App Management -> Configuration -> Allow only signed apps" is disabled, as the Symcon app does not yet have a signature.
After installing the Symcon app, configuring the firewall under "Security -> Firewall" is recommended to ensure functionality after the PLCnext update to 2026.6, which enables the firewall by default. For this, create a new rule under "IP Input Rules" with "To Port" set to "3777" and the comment "Symcon". Depending on the system used, additional ports may need to be opened. Alternatively, the firewall can be disabled after the PLCnext update to 2026.6.
If a "Timeout" error appears while setting the license, it is very likely that the gateway is not configured correctly or internet access is blocked by the company firewall.
If the message "SSL peer certificate or SSH remote key was not OK: SSL certificate problem: certificate is not yet valid" appears while setting the license, the system time is incorrect. In this case, check in the WBM under "Configuration -> Date&Time" whether at least one NTP server is configured and set to *Active*. Date and time are displayed at the top of the same page and should be current. If no NTP server is configured or it is not active, create a new server (for example, "pool.ntp.org"). Due to a current WBM bug, activation is currently only possible in Firefox.
If problems occur during installation, support is always available to assist with the initial setup.

CONTENT

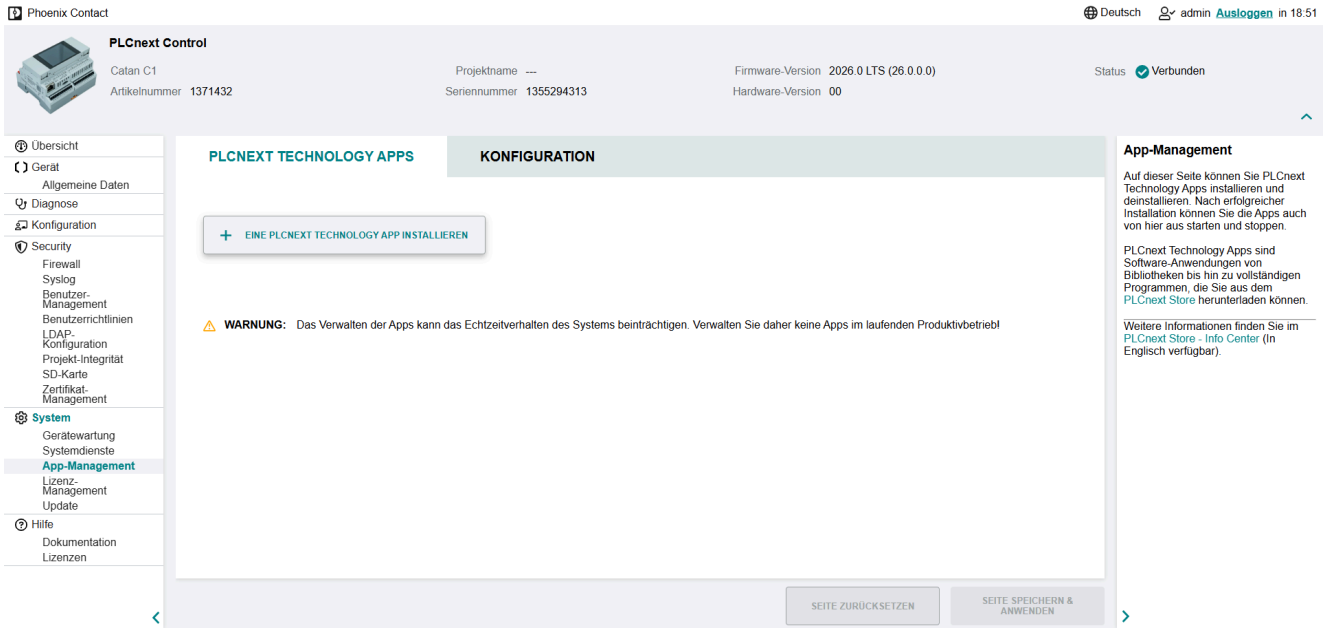
Installation
Update
Where is what?
Uninstall

INSTALLATION

Symcon can also be purchased and installed via the PLCnext Store.

Download: Appfile (64-Bit)
Symcon is installed as a PLCnext Technology App. The installation is carried out in WBM under Settings -> App Management. The option "Only allow signed apps" must first be

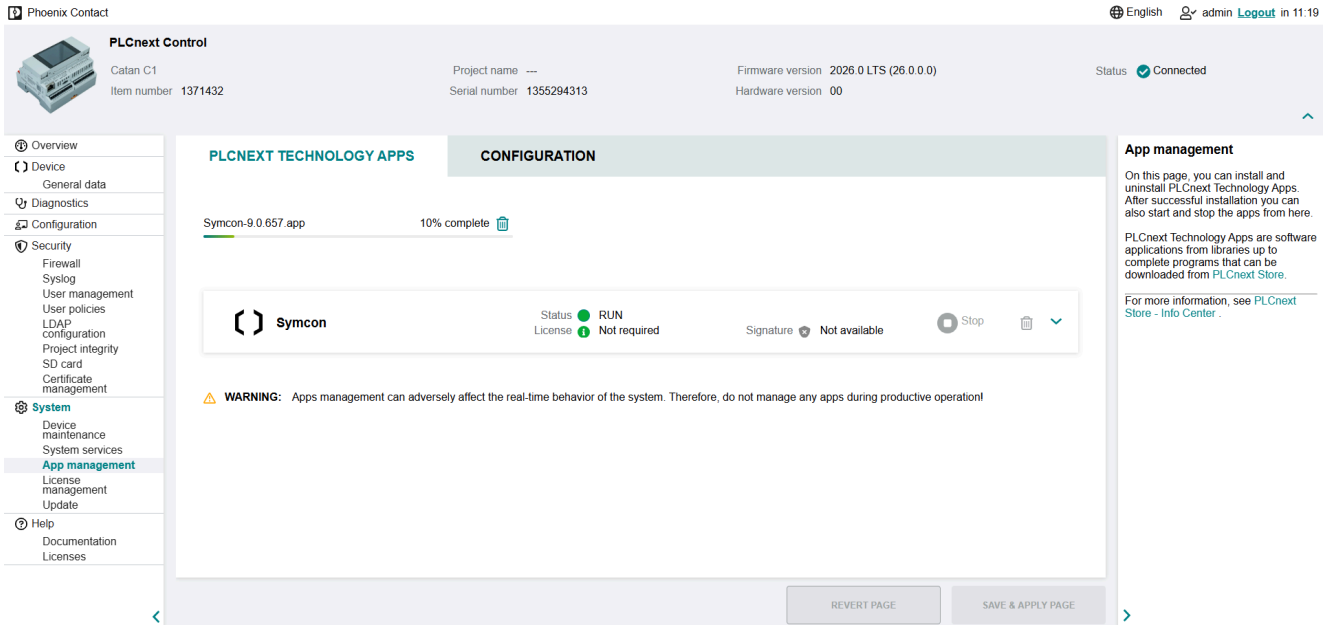
deactivated under Configuration. The Symcon app file can now be installed using the corresponding button.
Once the process is complete, the web-based Management console can be accessed via http://ipadresse:3777/console/. There you can change the License in the information menu at the top right.



A description of the special functions can be found here.

UPDATE

Current version: App file (64-bit)
To install an update, the .app file of the latest version must be reinstalled. Symcon is restarted during the installation. All data is retained.



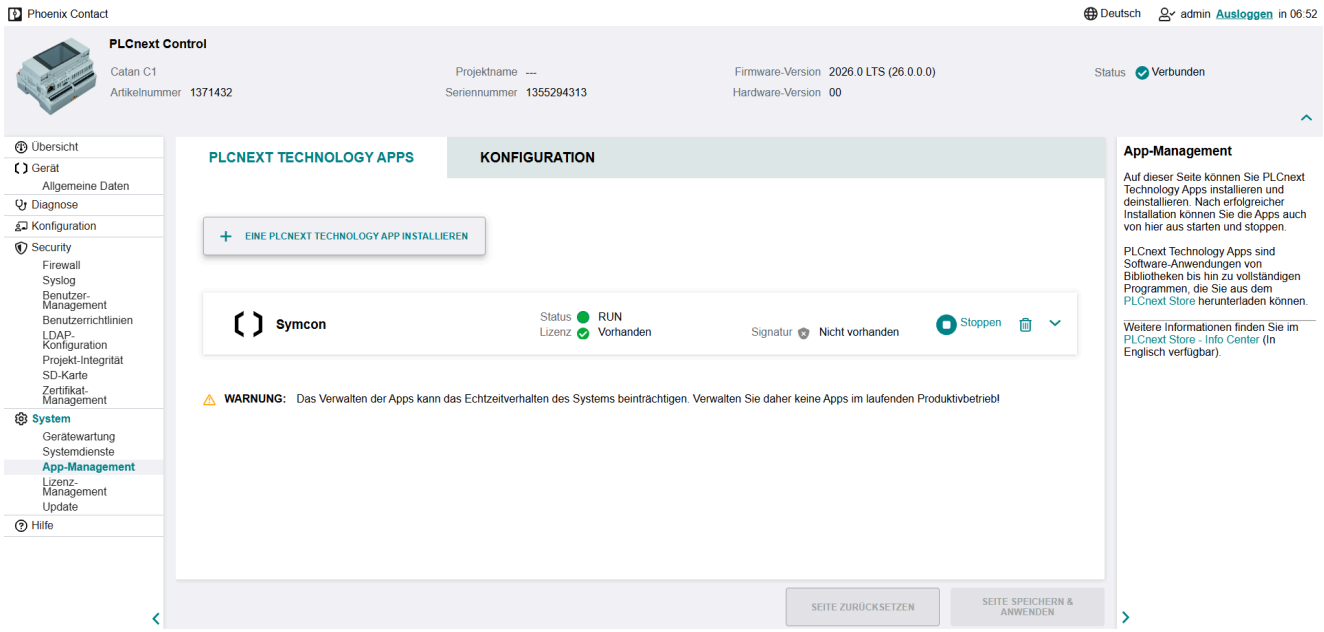
WHERE_IS_WHAT

WHERE IS WHAT?

/opt/plcnext/apps/home/data/60002171301055/data/ - Variable Data (Settings, scripts, media...)
/tmp/appsdata/60002171301055/logs/ - Log Files (Logfiles...)

UNINSTALL

Symcon can be uninstalled via the WBM under Settings -> App Management. To do this, the Symcon app must be selected. The uninstallation can be started via the trash can icon. All data will be lost when uninstalling.



Firewall

In order for IP-Symcon to be able to communicate with the various services from the Internet, the following settings must be made on a firewall.

RECOMMENDED SETTINGS

ADDRESS	PORT	PROTOCOL	FUNCTION
apt.symcon.de	443	TCP	Installation, Updates
api.symcon.de	443	TCP	Module Store, Personal area
symcon-store.s3.eu-west-1.amazonaws.com	443	TCP	Module Store (Data)
live.symcon.de	443	TCP	Installation, updates, connect, push notifications, display of the subscription term
zwdb.symcon.de	443	TCP	Z-Wave device database
dwd.symcon.de	443	TCP	WebFront DWD weather
ipmagic.de	50000	TCP	IP-Symcon Connect
github.com	443	TCP	Module Control (sofern genutzt)

SETTINGS ONLY RELEVANT TO SYMBOX

ADDRESS	PORT	PROTOCOL	FUNCTION
ipmagic.de	60000	TCP	SymOS Connect
ipmagic.de	53671	TCP	SymOS Connect (KNX)
ipmagic.de	53672	TCP	SymOS Connect (KNX)
ipmagic.de	54114	TCP	SymOS Connect (LCN)

CLOUDFRONT-EDGE-SERVER

The *.symcon.de addresses are mapped via AWS in the EU West 1 zone (Ireland). The endpoints terminate on AWS CloudFront and have a TTL of 60 seconds. In the case of dynamic resolution, it must therefore be ensured that the firewall determines the currently used IP addresses often enough.
Current lists of the IP addresses used can be found here: Locations and IP address ranges of CloudFront edge servers

Change of Platform

A change of platform is possible without any problems. No data or settings are lost in the process.

STEPS FOR A CHANGE OF PLATFORM

The following steps need to be executed to change from an initial platform to a target platform.

Install IP-Symcon on the target platform.
Terminate IPS service on the initial and the target platform.
Create backup on the initial platform.
Use Load backup to integrate the produced backup on the target platform.
Start the service on the target platform.



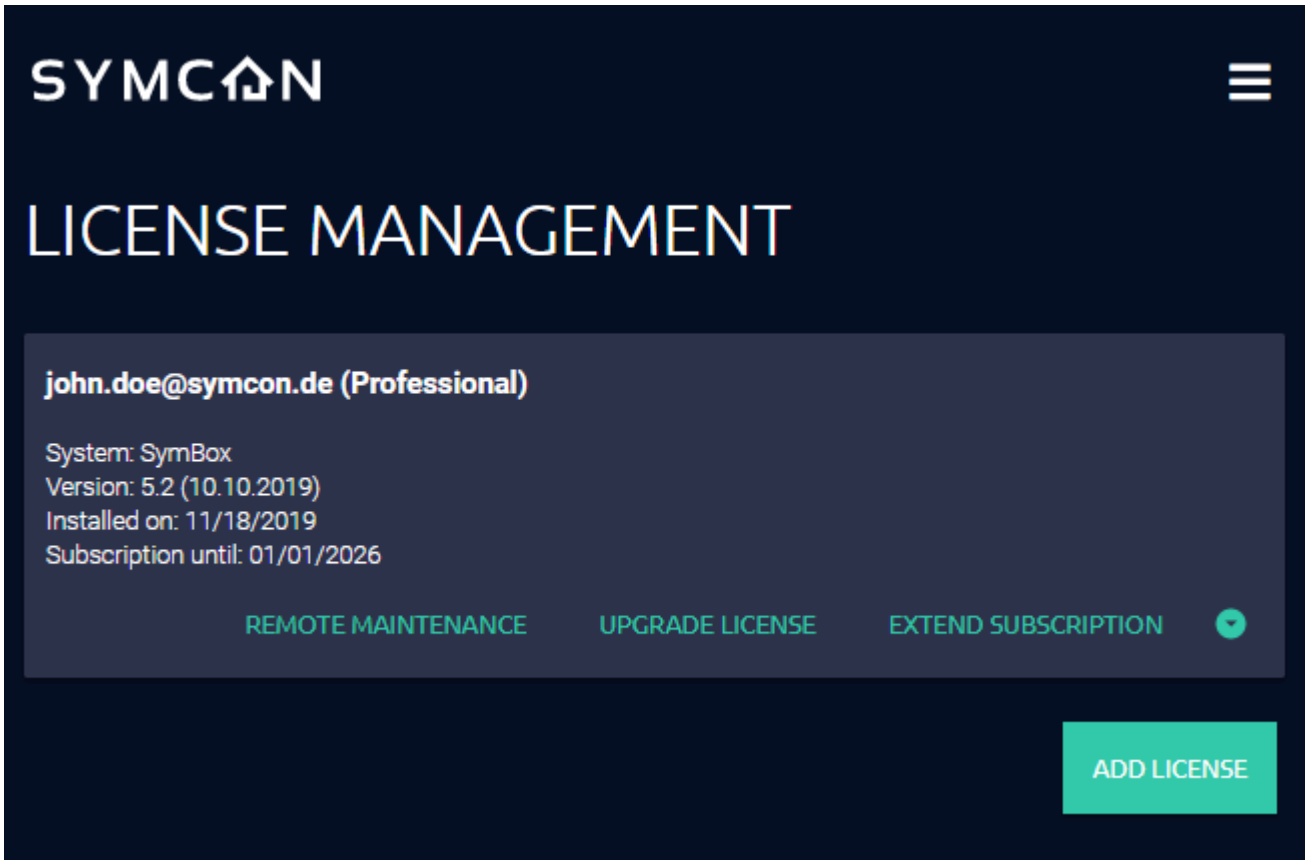
Watch out for the IP-Symcon versions of the initial and the target platform. If the versions are different, migrations need to be done eventually. For example, changing from version 3.4 to a higher version is impossible otherwise.

License Management

LICENSE MANAGEMENT

For easier handling and overview, it is possible to manage your license at the homepage in the License Management.
Among others, the remaining duration of a subscription can be checked or the license file can be sent again in the License Management.
The License Management can be found in the Personal Area . This can be reached via the person icon on the top right of the homepage. After logging in, the License Management can be opened.





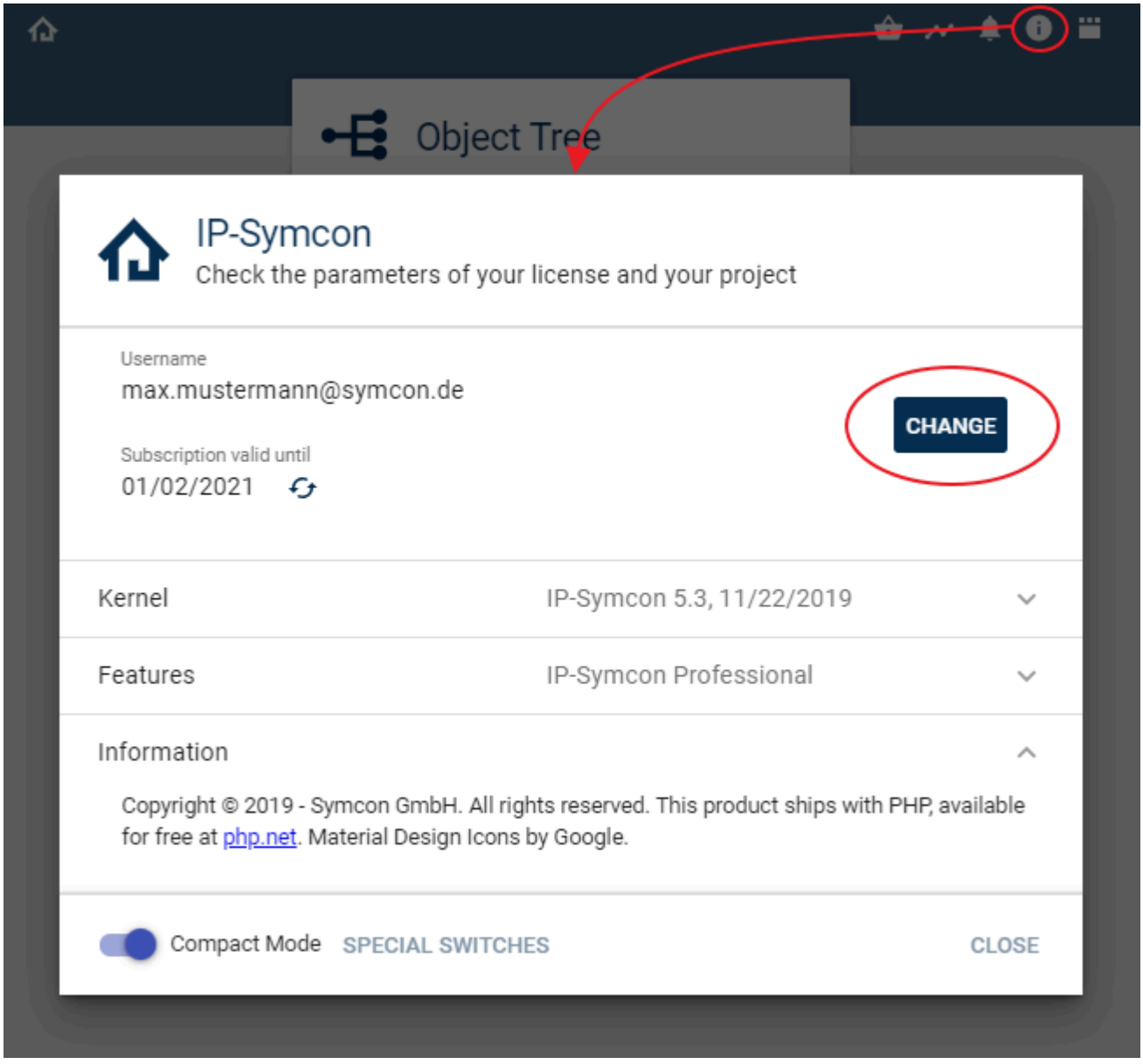
The accounts of forum and homepage are currently separate.
Combining these accounts is planned in the future.

CHANGE LICENSE

After any change to the license, the IP-Symcon Service needs to be restarted.

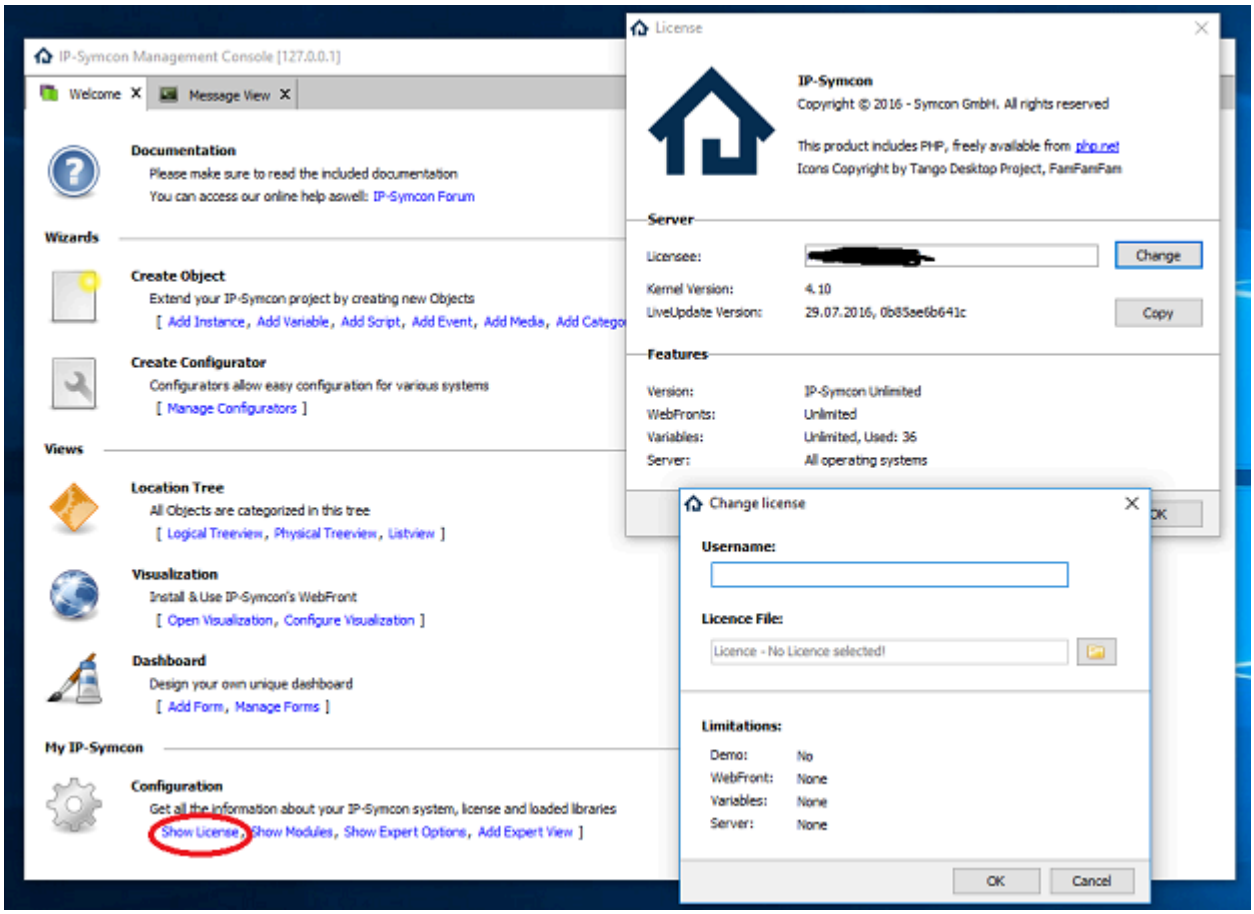
SINCE VERSION 5.0

The license can be changed in the Management Console in the information dialog.



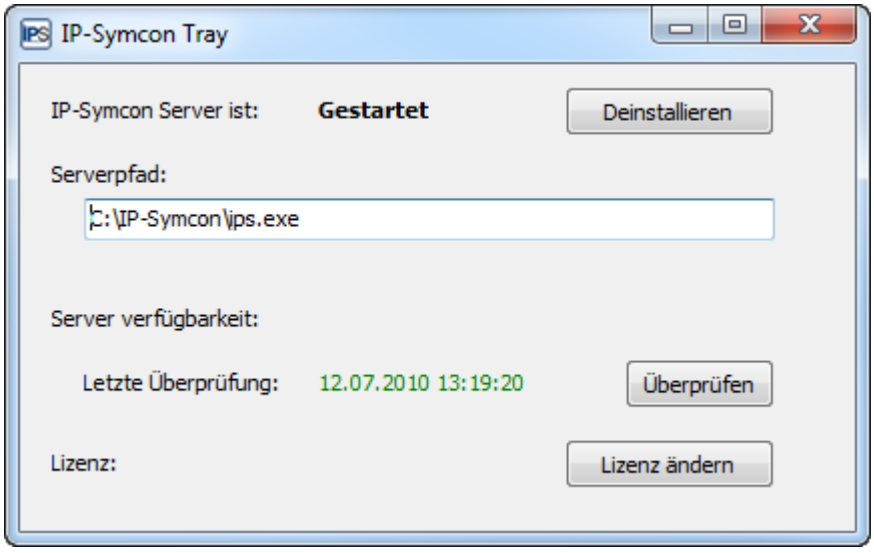
FROM VERSION 4.0 UP TO VERSION 4.4

The license can be changed in the management console in "My IP-Symcon -> Show License".



UP TO VERSION 3.4

A new license that was obtained due to a license upgrade can be entered by choosing the option "Information" in the context menu of the tray application. The context menu can be launched with a right click on the tray application.



The button "Change license" can open the dialog. Here, the user name and the new license file can be chosen. After confirmation, the new license is stored. Restart the IP-Symcon service to activate the new license.

Migrationen

Die Migrationen erläutern die Features und Fixes der jeweiligen Version.
Ebenso ist eine Beschreibung vorhanden, was bei Versionswechsel bzw. -update zu beachten ist.

MIGRATION	VERÖFFENTLICHUNG
V8.1->V9.0	Q1/2026
V8.0->V8.1	Q3/2025
V7.2->V8.0	Q1/2025
V7.1->V7.2	Q3/2024
V7.0->V7.1	Q1/2024
V6.4->V7.0	Q4/2023
V6.3->V6.4	Q2/2023
V6.2->V6.3	Q4/2022
V6.1->V6.2	Q2/2022
V6.0->V6.1	Q1/2022
V5.5->V6.0	Q3/2021
V5.4->V5.5	Q4/2020
V5.3->V5.4	Q2/2020
V5.2->V5.3	Q4/2019
V5.1->V5.2	Q3/2019
V5.0->V5.1	Q2/2019
V4.4->V5.0	Q3/2018
V4.3->V4.4	Q4/2017
V4.2->V4.3	Q3/2017
V4.1->V4.2	Q2/2017
V4.0->V4.1	Q4/2016
V3.4->V4.0	Q1/2016
V3.3->V3.4	Q2/2015
V3.2->V3.3	Q1/2015
V3.1->V3.2	Q3/2014
V3.0->V3.1	Q2/2014
V2.7->V3.0	Q3/2013
V2.6->V2.7	Q1/2013
V2.5->V2.6	Q3/2012
V2.4->V2.5	Q4/2011
V2.3->V2.4	Q2/2011

MIGRATION	VERÖFFENTLICHUNG
V2.2->V2.3	Q3/2010
V2.1->V2.2	Q1/2010
V2.0->V2.1	Q3/2009
V1.0->V2.0	Q1/2009
V0.0->V1.0	Q1/2005

V8.1->V9.0 (Q1/2026)

Wesentliche Änderungen
Migration
Breaking Changes
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

Aktualisierte Editionen
Optimierte Instanzerstellung
Matter (Beta)
Verbesserungen an der Visualisierung

AKTUALISIERTE EDITIONEN

Die verfügbaren Editionen wurden aktualisiert. Insbesondere neu ist die Community Edition, welche komplett kostenfrei installiert und dauerhaft genutzt werden kann. Aber auch bei den größeren Editionen wurde nachgelegt. Die Expert ist die Wahl für Hausbesitzer die mehr möchten und bietet 3000 Variablen sowie die Erweiterung für Türsprechanlagen. Für noch größere Systeme, insbesondere im professionellen Einsatz kommt die Edition Industrial mit dazu.
Diese Aktualisierung betrifft übrigens nur neu erworbene Editionen. Für bereits erworbene Editionen ändern sich die Bedingungen nicht.

OPTIMIERTE INSTANZERSTELLUNG

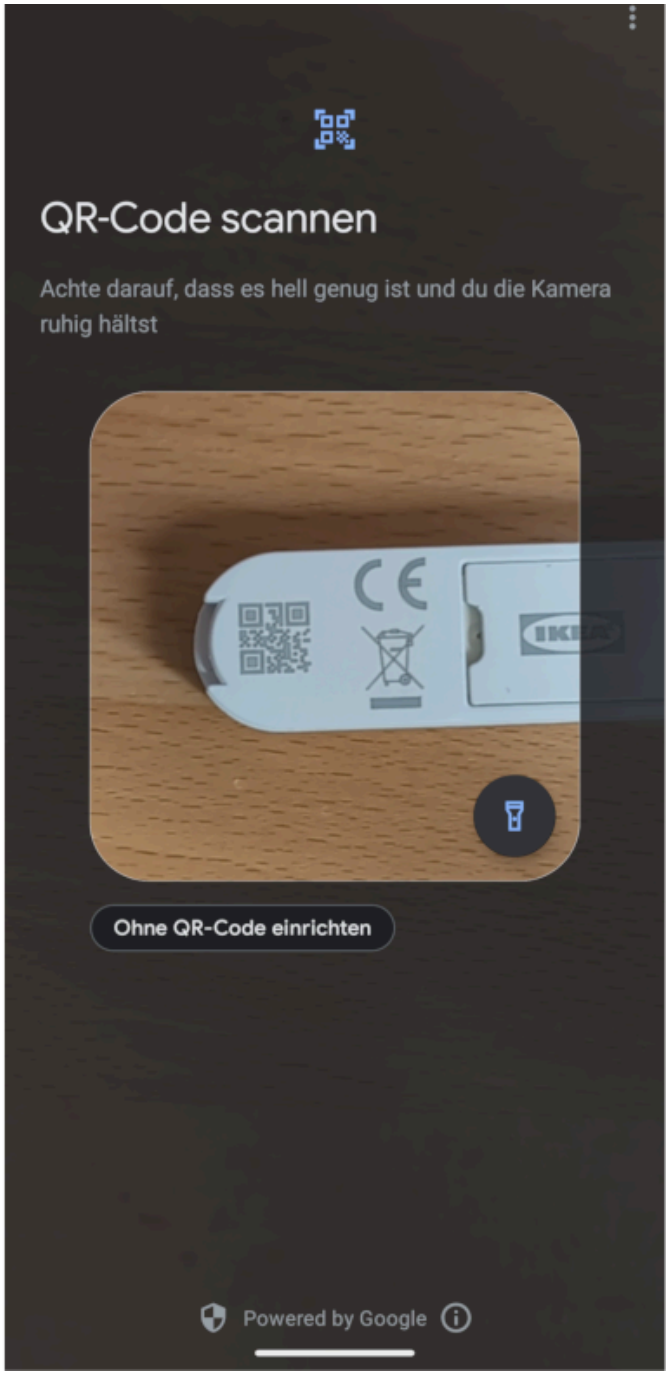
Der Erstellfluss beim Erstellen von Instanzen wurde komplett überarbeitet und bietet nun mehr Auswahlmöglichkeiten. Beim Anlegen einer Instanz kann nun gewählt werden ob diese beispielsweise mit dem bereits existierenden Gateway verbunden oder doch mit einer neuen Schnittstelle initialisiert werden soll.



MATTER (BETA)

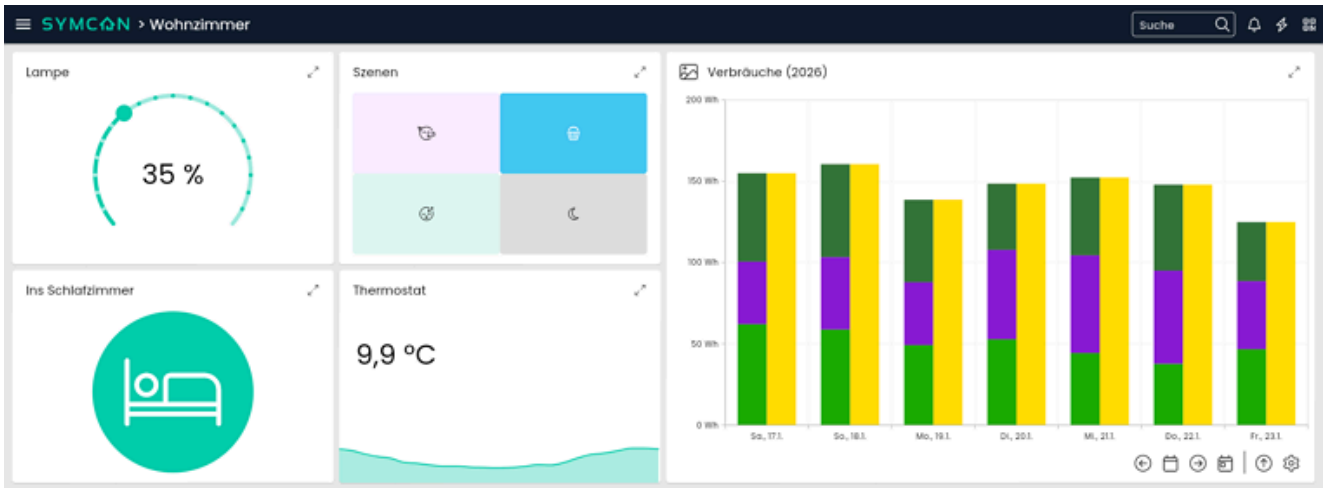
Matter kommt zu Symcon. Mit der 9.0 können alle Geräte, die zur Zeit erhältlich sind, eingebunden und verwendet werden. Der Anlernprozess funktioniert unkompliziert über die Symcon App oder über das Teilen als Code aus einem bereits bestehenden System heraus.





VERBESSERUNGEN AN DER VISUALISIERUNG

Natürlich wurde auch die Visualisierung weiter verbessert. So kann nun ein Sprung zu einem beliebigen Objekt per Automation oder Ereignis angestoßen werden. Außerdem haben diverse Darstellungen wie die Wertanzeige oder die Aufzählung neue Parameter und bieten somit noch mehr Freiheiten bei der Visualisierung.



MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

BREAKING CHANGES

Linux: Neuer Key für Debian-Pakete

LINUX: NEUER KEY FÜR DEBIAN-PAKETE

Der GPG-Schlüssel für das Symcon APT-Repository wurde erneuert, da der bisherige Schlüssel mit SHA-1 signiert war und von aktuellen Debian/Ubuntu-Versionen nicht mehr akzeptiert wird. Ohne diesen Schritt schlägt das Update mit einem Signaturfehler fehl: Die folgenden Signaturen konnten nicht überprüft werden, weil ihr öffentlicher Schlüssel nicht verfügbar ist: NO_PUBKEY 69FF6261F2D8B290
Bitte vor dem Update folgenden Befehl ausführen:

```
wget -qO- https://apt.symcon.de/symcon.key | gpg --dearmor | sudo tee
/usr/share/keyrings/symcon.gpg > /dev/null
sudo chmod 644 /usr/share/keyrings/symcon.gpg
```

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP 8.5
Neu: Datenfluss (Instanzverbindungen) werden ab sofort vollständig über die Verwaltungskonsole gemanaged
Neu: Spezialschalter ScriptWatch/VariableWatch sind standardmäßig inaktiv bei Neuinstallationen
Neu: Fehlermeldung verbessert, wenn zu viele Skripte gleichzeitig gestartet werden
Neu: Heating Control kann per Variablen An/Aus geschaltet werden
Neu: MQTT Client: Keep Alive Zeit kann eingestellt werden
Neu: Replication: Bessere und Intuitivere Konfiguration
Neu: SenderID in IPS_GetScriptThread zur besseren Nachverfolgbarekeit
Fix: IPS_GetProcessInfo funktioniert korrekt auch bei extrem vielen verfügbaren Prozessen
Fix: Ereignisse im Zeitraum der Umstellung von Sommer auf Winterzeit werden korrekt ausgeführt
Fix: Archiv konnte in einem Deadlock stecken bleiben, wenn sehr viele gleichzeitige Anfrage und Schreibvorgänge durchgeführt wurden
Fix: GridConfiguration wird vor dem Setzen validiert
Fix: Leere GridConfiguration wirft keinen Fehler mehr
Fix: Weitere Korrekturen für die bei manchen abstürzenden Streams
Fix: Beim Erstellen von Links wurde in der Persistent eine 0 statt 1 gespeichert (sofern per Skript erstellt und nie IPS_SetTargetID aufgerufen wurde)
Fix: Mirroring funktioniert wieder wie vorgesehen
Fix: UDP Socket konnte „stecken“ bleiben, wenn die BindIP ungültig wurde
Fix: SyncRemote: In seltenen Fällen konnte die Synchronisation abbrechen
Fix: Das Event Control konnte sich in einer Schleife verfangen, wenn fehlerhafte Instanzen manuell „repariert“ wurden

GERÄTE

Neu: Matter Unterstützung (Beta)
Neu: BACnet: Unterstützung von FDR (Foreign Device Registration)
Neu: VoIP: Spezialschalter ProxyPort zum Ändern des VoIP Inbound Ports
Neu: CMI: Freie Konfiguration, welche Daten abgefragt werden sollen
Neu: KNX: physikalische Adresse wird im Datenfluss mitgesendet
Neu: Xcomfort: Shutter unterstützt ~ShutterMoveStep Profil
Neu: Unterstützung für KNX DPT 5.100
Neu: Unterstützung für KNX DPT 9.009
Neu: ModBus Geräte: Abfrage über Datenblöcke möglich
Neu: ModBus Geräte: Schalter zum Deaktivieren von Fehlermeldungen
Neu: ServerSocket: Schalter zum Deaktivieren von Fehlermeldungen
Neu: HomeMatic HCU: Maintanance Variablen für alle Geräte und Gruppen
Neu: HomeMatic HCU: Mehr Variablen mit Darstellungen
Neu: HomeMatic HCU: Initial werden mehr Werte angezeigt
Fix: HomeMatic HCU: Wenn Kanäle mehrfach vorhanden sind werden alle erstellt
Fix: DoorIP: Nutze den Display Name bevorzugt statt dem Account
Fix: KNX: DPT 1.024 hatte Tag/Nacht im Profil vertauscht
Fix: KNX: Verbesserte Kompatibilität zu Enertex IP Interface

AKTIONEN

Neu: Aktion zum Öffnen eines Objekts in der Visualisierung

KACHEL VISUALISIERUNG

Verbesserte Vorschau Diagramme
- Neu: Position des Wertes kann angepasst werden
- Neu: Vorschau kann deaktiviert werden
- Neu: Wertebereich ist fix auf 24 Stunden

Verbesserte Diagramme
- Unterstützung für Darstellungen in Multi-Charts
- Stapeln
  - de-/aktivierbar
  - Gruppierung
- verbesserter Zeitversatz

Neu: Beim Verschieben oder Hovern des Rollladenreglers wird der aktuelle Wert angezeigt
Neu: Bilder werden gecached
Neu: Ein Design kann über die Visualisierung fest vorgegeben werden
Neu: Zentrierte Login Seite ohne „Ladekreis“
Neu: Darstellung Aufzählung verbessert:
- Icons können statt oder zusätzlich zum Text angezeigt werden
- Anordnung kann statt Spalte auch Zeile oder Gitter sein

Neu: Ähnliche Textgrößen der Wertanzeige werden einheitlich dargestellt
Neu: Hinweis in den Einstellungen, wenn sich App- und Serverversion unterscheiden
Neu: OpenObject
Neu: Anzeigemodus Kacheln(Liste) → Instanzkonfiguration
Neu: Info Dialog für Objekte in Listen
- Beim Klick auf eine Variable in einer Liste wird etwa die letzte Aktualisierung/Änderung angezeigt

Neu: Keine Limitierung der Anzahl von HTML-Inhalten im Browser (Performance Problem bei 20+ möglich)
Neu: Höhe vom Gitter wird nie gesenkt während die Höhe oder Position einer Kachel bearbeitet wird, erst danach
Neu: Zeige den Bearbeiten-Stift für alle Instanzen an (nicht nur für die mit untergeordneten Objekten)
Neu: Kachel Visualiserung Konfigurator: Begrüßung ist Standardmäßig aus
Neu: Seitenleiste kann permanent angezeigt werden
Fix: Suche konnte durcheinander kommen, wenn Server über mehrere IP-Adressen gleichzeitig geantwortet hat
Fix: Timeout für initiale Anfrage erhöht und Anfragen parallelisiert
Fix: Aufzählungen werden in kleineren Kacheln besser dargestellt
Fix: Design Einstellungen sind auf schmalen Bildschirmen besser nutzbar
Fix: Diagramme bei Übergang von Sommer-/Winterzeit korrigiert
Fix: Beim Wechseln der Visualisierung konnte der Server falsch erkannt werden, wenn zwei verschiedene Symcon Systeme in unterschiedlichen WLANs die gleiche IP-Adresse hatten
Fix: Notification Badge aktualisiert sich korrekt automatisch
Fix: In der MediaPlayer Darstellung wird der Fortschritt korrekt angezeigt
Fix: Beim Übernehmen von Text Werten, werden diese nicht doppelt übermittelt
Fix: Unterstützung für Reversed Shutter korrigiert

Fix: Aufzählungen in der Listenansicht brechen bei bestimmten Breiten nicht mehr um
Fix: In seltenen Fällen wurden die Hover Effekte nicht mehr korrekt angezeigt
Fix: Legende der Diagramme werden in der Farbe des Designs angezeigt

DARSTELLUNGEN

Wertanzeige
- Neu: Es kann entweder nur das Icon oder nur Text angezeigt werden
- Neu: Die Farbe des Icons/Wertes und des Hintergrunds kann einzeln verändert werden

Inhaltswechsler
- Fix: Voller platz wird genutzt
- Fix: Werte aktualisieren sich nach dem Wechseln
- Fix: Hintergrundfarbe wird aktualisiert

WEBFRONT

Neu: Unterstützung für neue Diagrammeeinstellung: Gestapelte Balken
Fix: Darstellung von Boolean Variablen verbessert

KONSOLE

Neu: Neuer Tab zum analysieren der RPC Befehle und verbrauchtem Traffic
Neu: Diagramm-Dialog an neues Format angepasst, inklusive Konvertierung vom alten Format zum neuen
Neu: Passe Anzahl Vorlagen pro Zeile im Vorlagenmanager an die Breite des Bildschirms an
Neu: Hinweise, dass die neue Kachelvisualisierung keine MJPEG Streams unterstützt
Neu: Vorlagenmanager kann Vorlagen der Legacy Darstellung anzeigen, falls welche existieren
Neu: Beim Wählen von Farben für Diagramme, wähle aus einem festen Set zufällig aus, statt immer neue Farben zu generieren
Neu: Instanzen können im Konfigurator direkt umbenannt werden
Neu: Validiere beim Erstellen von Instanzen ob diese aufgrund des Limits erstellbar sind
Fix: Neu verbinden via Prüfen-Dialog funktioniert wie vorgesehen
Fix: Mehrfaches Editieren von Medien ist jetzt korrekt möglich
Fix: Kennwörter mit gewissen Sonderzeichen wurden nicht korrekt übermittelt
Fix: Discovery-Instanzen aus dem Module Store bleiben wie vorgesehen bei Anpassung der Systeme bestehen
Fix: Vermeide kosmetische Fehlermeldung im Objektbaum beim betätigen von Enter ohne ausgewähltes Objekt

PRO-KONSOLE

Neu: Bibliotheken aktualisiert

SYMOS

Neu: Aktualisierung auf Kernel 6.12

PHP-SDK

Neu: Parameter popup.caption für PopupAlert
Neu: Parameter transparentCaption für SelectColor
Neu: UnregisterHook / UnregisterOAuth für IPSModuleStrict
Neu: List und Tree unterstützen nun den Parameter icon mit welchem pro Spalte ein Icon gesetzt werden kann
Neu: restriction „moduleType“ um eine Aktion auf spezifische Modultypen einzuschränken
Fix: RegisterHook überschreibt ggf. vorhandene „alte“ Hooks, sodass Modul Hooks immer korrekt registrieren können. (Sollte ein vorhandener Hook überschrieben werden, wird dies passend geloggt)

HTML-SDK

Neu: openObject

AKTUALISIERTE MODULE

AMAZON ALEXA

Fix: Rollladen-Darstellung wird wie vorgesehen unterstützt
Fix: Korrektes Handling bei der Gerätesuche von Variablen ohne gesetzte Darstellung

DYNAMISCHE E-MAIL

Neues Modul im Module Store, mit dem dynamisch generierte E-Mails versendet werden können. Der Inhalt und Betreff können basierend auf dem Inhalt beliebiger Variablen gefüllt werden.

EKEY BIONYX

Neu: Unterstützung für die Notification API (Input)
Neu: Protokoll Variable, welche granular konfiguriert werden kann

ENERGIE MANAGER

Neu: Vorlage für Goodwe, go-e, Keba P30 und Foxess ECS 2900
Fix: Wende Mindestlaufzeit und Nachlaufzeit für günstiges Laden an
Fix: Verhindere Nebeneffekte, die andere Geräte beim Nachtladen aktivierten

GOOGLE ASSISTANT

Neu: Unterstützung für Darstellungen
Fix: Korrekte Unterstützung von Rollläden

OCPP

Neu: Summiere alle Ladevorgänge je IdTag unter dem OCPP-Splitter
Neu: Besseres Konfigurationsformular
Neu: Schönere Darstellung der Variablen
Neu: Unterstützung von Autorisierung über RFID Karten

STROMPREIS (VORHERSAGE)

Neu: Beschaffe EPEX Preise über ENTSO-E anstatt die EPEX Homepage zu parsen
Neu: 15 Minuten Auflösung für EPEX und Tibber
Fix: Zeige immer bewegliches Wertefenster an, zeige dabei potentiell noch unbekannte Werte als nicht verfügbar an

SZENEN-STEUERUNG

Neu: Option zur Eingabe einer Schaltverzögerung

TAILSCALE VPN

Neu: Verbesserte Installationsroutine
Fix: Hostname wird korrekt gekürzt / Reset Button zum Deinstallieren

V8.0->V8.1 (Q3/2025)

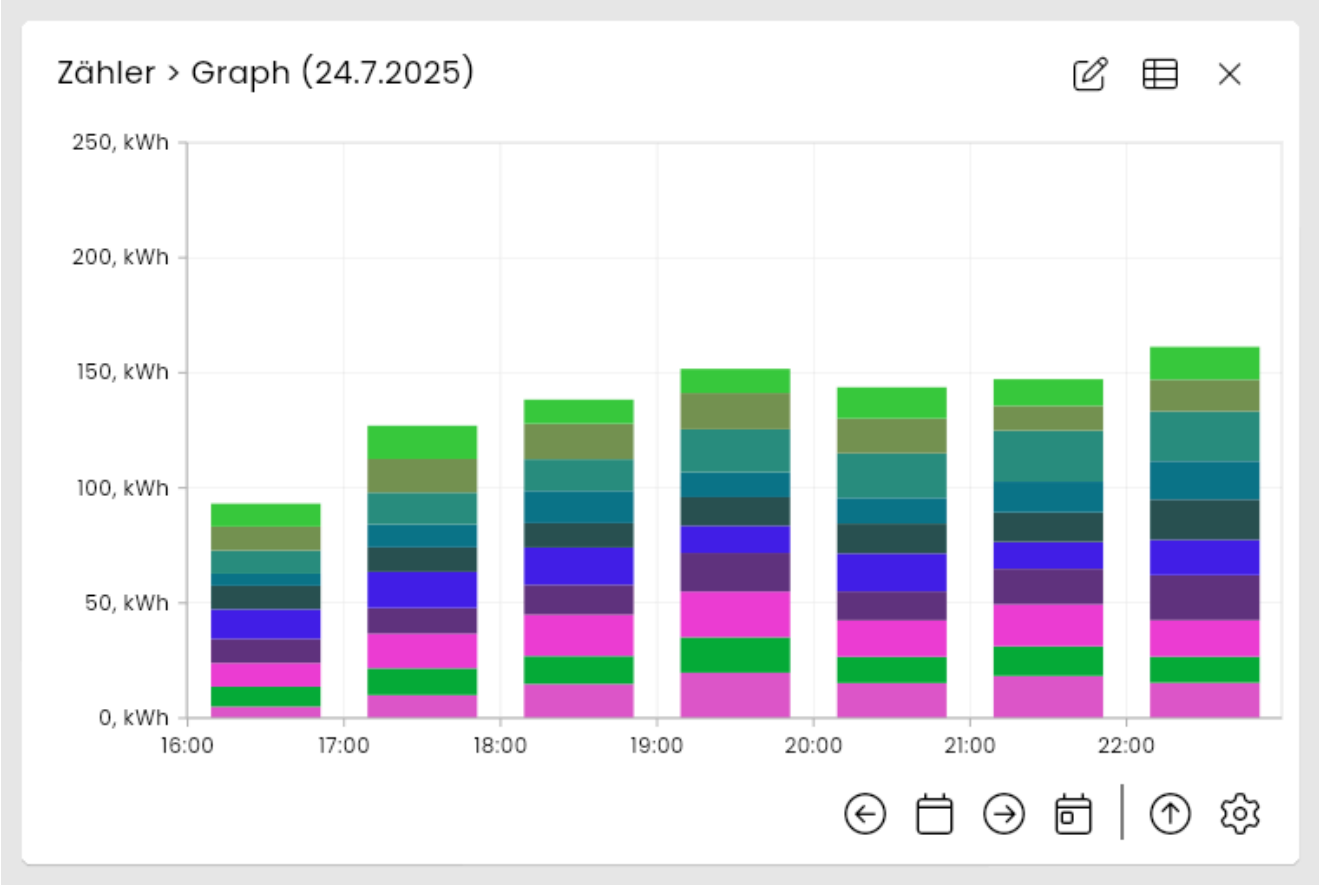
Wesentliche Änderungen
Migration
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

Moderne Diagramme
Energie Manager
HomeMatic HCU (Beta)
Enterprise Features für alle

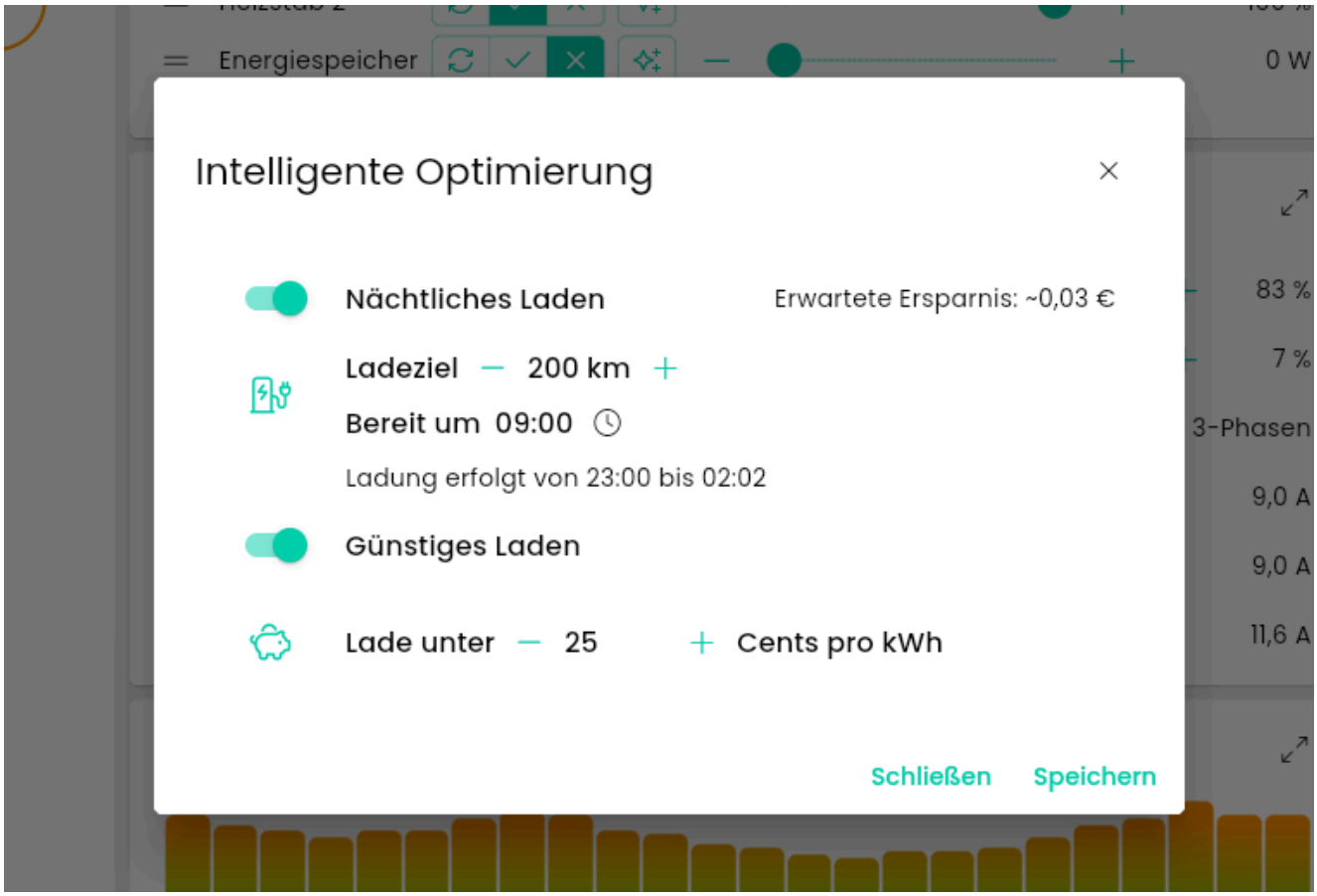
MODERNE DIAGRAMME

Diagramme informieren über den Verlauf von Werten und sind somit kaum aus Symcon wegzudenken. Diese Ansicht wurde komplett überarbeitet. Damit sehen die Diagramme nicht nur modern aus, sie bieten auch noch diverse neue Funktionen. Der Bereich der angezeigten Werte kann nun frei ausgewählt und stufenlos angepasst werden. Auch die Auflösung der dargestellten Werte kann nun frei ausgewählt werden. Soll der Verbrauch im Laufe des Tages lieber pro Stunde oder pro 15 Minuten angezeigt werden? All das und noch mehr kann unkompliziert eingestellt und betrachtet werden.



ENERGIE MANAGER

Auch der Energie Manager erstrahlt zur Version 8.1 in neuem Glanze. So werden mit dem Update viel mehr Geräte unterstützt, insbesondere auch Energiespeicher, welche nicht direkt schaltbar sind. Zur Vereinfachung der Konfiguration all dieser Geräte wurden Vorlagen eingeführt. So muss nur noch die passende Vorlage ausgewählt werden und die Konfiguration ist automatisch eingetragen. Und sollte die richtige Vorlage nicht dabei sein, kann die Konfiguration selbstverständlich manuell angepasst werden.
Darüber hinaus wurde die Unterstützung für die rechtlichen Vorgaben aus §9 EEG und §14a EnWG umgesetzt. Schließlich wurde für dynamische Strompreise günstiges Laden implementiert. Hiermit kann ein Energiespeicher oder ein E-Auto automatisch geladen werden, wenn der Strompreis gerade niedrig genug ist. So können die guten Konditionen optimal ausgenutzt werden.



HOMEMATIC HCU (BETA)

Mit der HCU kann nun auch das neueste HomeMatic Gateway in Symcon verwendet werden. Das Gateway bindet sich wie gewohnt nativ in Symcon ein und bietet die bekannten Komfortfeatures. Es wird per Discovery gefunden und die eingebundenen Geräte und Gruppen können direkt per Konfigurator erstellt und sofort genutzt werden.



ENTERPRISE FEATURES FÜR ALLE

Im kleinen Rahmen können ausgewählte Enterprise Features nun ganz ohne kostenpflichtige Erweiterung genutzt werden. Das ermöglicht es zum einen die Features vorab zu testen, natürlich auch im Rahmen einer Demo-Version. Oder vielleicht gibt es einfach einen kleinen Anwendungsfall wie ein einzelnes BACnet-Gerät. Oder der Wohnwagen mit eigenem Symcon soll ins Hauptsystem synchronisiert werden? Das ist jetzt für jeden nutzbar.

ENTERPRISE FEATURE	KOSTENFREIER NUTZUNGSUMFANG
Sync Remote	Ein Remote kann integriert werden
BACnet	Bis zu 3 BACnet Datenpunkte können integriert oder als BACnet Server bereitgestellt werden
OPC UA	Bis zu 3 OPC UA Knoten können integriert werden
Translation	Bis zu 10 Strings können je Sprache übersetzt werden
RBAC	Die Verbindung zum eigenen LDAP Server kann vorab getestet werden

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Standardmäßig werden Objekte mit leerem Namen statt „Unbenanntes Objekt“ erstellt
Bessere Erkennung von Abbrüchen bei Streams (Upload wurde unnötig zu lange beansprucht wenn die Verbindung über das Relay lief)
Neu: DoorIP: Neuer Befehl DOORIP_Reject zum Ablehnen von Anrufen
Neu: Shutter Control: Berechnung von Laufzeiten mit Millisekunden Präzision
Neu: Spezialschalter ProxySTUN zum Deaktivieren der STUN Anfragen
Neu: Spezialschalter ProxyTURN zum Deaktivieren des TURN Relays
Neu: Eltako FHK14 Instanz kann mit einem Thermokon SR0x kombiniert werden zur komfortableren Einzelraumregelung
Neu: Gitter-Konfiguration einer Visualisierung kann heruntergeladen/hochgeladen werden
Neu: EnOcean: Timeout zwischen zwei Befehlen kann eingestellt werden
Neu: Einlern-Wizard für das Shutter Control
Neu: Wireless M-Bus: Zentrales Schlüssel Management mit Import von diversen CSV Formaten (siehe Wireless M-Bus)
Neu: 15 Minuten Aggregation
Neu: Client Socket unterstützt SSL Verbindungen über Zertifikate
Neu: WebServer/ServerSocket akzeptiert jetzt Private Keys mit Kennwort
Neu: Neue XY-Darstellung wird von KNX DPT 242.600 unterstützt
Neu: Debug Funktion: IPS_DisconnectWebRTCConnections() zum Trennen von Kamera Verbindungen
Neu: Verbessertes Handling bei überschreiten des Connect Traffic Limits (Liste der Top-Verbraucher wird angezeigt, Automatisches Zurücksetzen nach 24 Stunden)
Neu: WebServer Instanz wird standardmäßig mit Port 443 und SSL erstellt
Neu: Links werden initial mit der ID 1 (ungültiges Ziel) erstellt
Fix: Tailscale Modul (Beta) funktioniert wieder korrekt und vergisst die Einstellungen nicht mehr
Fix: Archiv: Bei Daten >2GB in der Variable konnten die Statistiken als negative Werte angezeigt werden
Fix: Wireless M-Bus: WebHook Pfad für WebTech Geräte gekürzt, um Limitation einzuhalten
Fix: Shutter Control: Problem beim Nachführen der Position gelöst
Fix: ModBus: Schreiben von Hex-kodierten
Fix: Energieverteilung: Restriktion auf Int/Float Variablen
Fix: Berechnung vom Connect Verbrauch bei großen Dateien, die fragmentiert übertragen wurden, konnte ggf. zu hoch sein
Fix: Client Socket: Prüfung von Zertifikaten gegen offizielle Domains funktioniert korrekt
Fix: HomeMatic kann nun 0.0.0.0 (Alle) als Ereignisserver auswählen
Fix: Mirroring: Replikation als Standby legt die I/Os korrekt im Standby Status an
Fix: Mirroring: Replikation als Standby blockiert die zyklischen Ereignisse korrekt
Fix: Der Wochenplan konnte in sehr speziellen Fällen einmalig die falsche Aktion ausführen (Danke @bumaas)
Fix: Streams konnten beim Disconnect den Dienst nach und nach blockieren bis der interne WebServer nicht mehr reagiert hat

Fix: Streams zeigen eine Fehlermeldung an, wenn diese blockiert sind anstatt den ganzen Server lahm zu legen
Fix: GetValueFormatted funktioniert für Float-Variablen mit Intervallen
Fix: Man konnte die Root Kategorie löschen
Fix: Crashlogs für SymBox/Linux werden wieder korrekt versendet
Fix: Sporadische Abstürze mit DoorIP korrigiert
Fix: Inkonsistente Submodule Konfiguration wird wie bei der Git-Cli nicht als relevante Änderung gewertet im Module Control
Fix: Seltene Probleme mit fehlenden seriellen Ports unter Windows 11 korrigiert
Fix: Berechtigungen: Das Löschen von Benutzern ist jetzt auch korrekt möglich, wenn man über den Lizenzlimit ist

GERÄTE

Ab sofort sind für BACnet und OPC UA jeweils 3 Instanzen erstellbar (ohne kostenpflichtige Erweiterung) erstellbar. Bei Sync Remote ist 1 Instanz erstellbar. Beim Translation Control können bis zu 10 Strings pro Sprache zum Testen übersetzt werden.
Neu: KNX: Setzen der physikalischen Adresse der SymBox mit KNX kann direkt im Gateway durchgeführt werden. Optimal in Kombination der ETS Applikation mit Version 1.1 der SymBox
Neu: LCN UMR wird korrekt im Konfigurator erkannt
Neu: Wireless M-Bus kann nun mit dem Webtech SWAN Gateway verwendet werden
Neu: HCU-Integration
    Neu: Discovery Instanz
    Neu: Geräte werden mit Kategorien erstellt
    Neu: Grundlegende Unterstützung für Gruppen
    Neu: Unterstützung für mehr Geräte. Besonders externe
    Neu: Mehr Informationen im Konfigurator
    Neu: Die Gruppen SWITCHING und HEATING werden unterstützt
    Neu: Es werden nur Variablen für Kanäle erstellt, die in Benutzung sind
    Neu: Farbtemperatur für Geräte und Gruppen
    Fix: Korrekte min/max Werte für Temperatur und Farbtemperatur
Neu: MQTT Client kann den verbesserten Client Socket nutzen, um gesicherte Verbindungen über Zertifikate zu verwenden
Neu: ModBus: Eigene Faktoren können in der Konfiguration verwendet werden
Neu: M-Bus: Keep-Alive (alle 3 Minuten) für Gateways
Neu: wM-Bus: Besseres Caching von Compact Frame Formaten
Neu: CMI: Nur die aktivierten Abfragen werden auch wirklich angefragt
Neu: Eltako FUTH
Neu: DMX (Out) over RS485 (Experimentell)
Neu: Alle „alten“ HomeMatic Instanzen wurden intern zu HomeMatic CCU umbenannt
Neu: Eltako FIUS65 wird unterstützt
Neu: Alle Symcon LAN Gateway’s können aus den jeweiligen Discovery Instanzen neu gestartet und die BaudRate (sofern relevant) konfiguriert werden
Neu: HomeMatic HCU wird über die IP-Adresse konfiguriert (löst Probleme bei der DNS Auflösung)
Neu: HomeMatic HCU: Besserer Workflow bei der Einrichtung von Tokens
Neu: HomeMatic HCU: Geräte ohne „modelType“ werden unterstützt (wie z.B. die Vaillant per EEBUS)

Neu: HomeMatic HCU: Bessere Unterstützung für Fußbodenheizungscontroller und Wandthermostate
Neu: Eltako FTKE wird unterstützt
Fix: BACnet: Einige Geräte (DDC4000 von Kieback&Peter) erlauben nicht, dass der Subscriber Process Identifier gleich der Instance Number ist, wenn man den COV abonnieren möchte.
Fix: BACnet: PresentValue wurde fälschlicherweise als PresetValue geschrieben
Fix: Eltako RGBW: Status vom Weiß Kanal wird korrekt gesetzt
Fix: EnOcean: Fix für EnOcean Shutter Ex Funktionen
Fix: KNX: Bei der Konfiguration der physikalischen Adresse wird die letzte bekannte angezeigt statt der Standardadresse
Fix: KNX: Physikalische Adresse der SymBox darf nun auch 0.0.1 sein
Fix: KNX: Potenzieller Deadlock bei KNX IP-Secure Verbindungen gelöst
Fix: EnOcean: Wartezeit zwischen Befehlen bei FGW14 Kommunikation auf mind. 100ms erhöht
Fix: M-Bus: Suffix bei Wasserzählern korrigiert
Fix: HomeMatic HCU: Rückmeldungen bei unkonfigurierten Kanälen werfen keine Fehler mehr
Fix: EnOcean: Auswertung bei F6-05-00 und F6-05-02 korrigiert

ARCHIV

Neu: Spezialschalter ArchiveRecordLimit ist standardmäßig auf 50000 (ist auch der neue Minimalwert)
Fix: Werte Rohansicht von Zähleraggregation korrekt aus

LOGIKPLÄNE

Neu: Logikpläne können in einem nicht ausführbaren Zustand gespeichert werden
Neu: Das Kopieren/Einfügen zwischen mehreren Plänen auf einem System wird unterstützt
Fix: Copy & Paste für alle Browser
Fix: Bei Copy&Paste werden die Verbindung korrekt mitkopiert

AKTIONEN

Neu: Aktionen für Darstellungen angepasst
    Beinhaltet insbesondere eine komplette Überarbeitung der Gruppenschaltung
Fix: Gruppenschaltung: Fehler bei ungültigen Links gelöst
Fix: Gruppenschaltung: Standardmäßige Sortierung hinzugefügt

KACHEL VISUALISIERUNG

Diagramme
    Neu: Zoomen durch klicken. Vgl. alte Charts
    Neu: Freie zeitliche Auflösung
    Neu: Erweitere den Namen von Variablen in Diagrammen um übergeordnete Namen falls sie sonst nicht eindeutig sind
    Neu: Auflösung kann vorausgewählt werden
    Fix: Zoomen kann über Esc unterbrochen werden
    Fix: Label von Boolean Charts werden immer angezeigt

Fix: Verbesserter Fehler, wenn ein Variable nicht existiert
Fix: Graphen listen bei Anzeige des Intervalls im Titel das Zieldatum nicht mehr mit an, wenn dies nur als Ziel ist (z.B. bei Anzeige von 1.5. 00:00 bis 1.6. 00:00 wird nur 1.5 im Titel angezeigt)
Fix: Diverse Performance-Verbesserungen
Fix: Legende kann nicht mehr aus dem Chart heraus geschoben werden
Fix: Zeige Rohwerte korrekt an, wenn diese vor dem angezeigten Bereich liegen
Fix: Verhindere Fehler bei Graphen mit Minimum = Maximum und nicht-dynamischer Anzeige
Fix: Verbesserte automatisierte Bestimmung von Minimum/Maximum/Schrittweite
Fix: Verhindere Fehler beim Rauszoomen, falls der Startzeitpunkt sich nicht ändert (z.B. bei Stundenansicht von 0:00 bis 1:00 auf täglich)
Fix: Reicht der Platz nicht für alle Beschriftungen der X-Achse werden gleichmäßig Beschriftungen ausgeblendet
Fix: Bei Monats- oder Jahresgraphen bleiben die Beschriftungen nun konstant beim ersten Tag
Fix: Lade mehr Werte vor, damit die Linie zum nächsten Wert immer sichtbar ist
Fix: Ermögliche wie vorgesehen die Auswahl der Auflösung bei reinen Balkengraphen
Fix: Balkengraphen mit einem einzelnen Wert zeigen den Balken nun in korrekter Breite an
Fix: Ermögliche wie vorgesehen die Auswahl der Auflösung bei reinen Balkengraphen
Fix: Zeige Gradienten bei Variable mit Hintergrundfarbe und Graphen korrekt an
Fix: Formatiere unglücklich gerundete Zahlen mit Nachkommastellen korrekt
Fix: Zeige ersten Wert bei CSV-Ansicht wie vorgesehen mit an
Fix: Fehler beim Laden von Daten werden korrekt dargestellt
Neu: Die Farben der Designs können angepasst, hoch- und heruntergeladen werden
Neu: Energieoptimierer unterstützt Darstellung
Neu: Darstellungen von Energiespeichern ohne Aktion im Energieverbrauch Optimierer
Neu: Neue Icons von @da8ter
Neu: Visualisierung wird standardmäßig über /tile statt /preview geöffnet (/preview leitet weiter auf /tile)
Neu: Option nur das Icon in der Kategorieleiste anzuzeigen
Neu: Titel in der Legacy Ansicht und Objekt-Listen werden hervorgehoben und nicht eingerückt
Neu: xy Farbe kann als kombiniertes Element genutzt werden
Neu: Text innerhalb von Schiebereglern wird dynamisch skaliert
Neu: Text von Wertanzeige wird basieren auf festen Stufen skaliert, um Variationen der Größe zu reduzieren
Neu: Neues Wärmepumpen Icon wird in der Energieverteilung genutzt
Neu: Batterie-Icon der Energieverteilung unterstützt Darstellungen und mehr Profile
Neu: Info-Variable in der Energieverteilung wird zusätzlich im Chart dargestellt
Neu: Strategie der automatischen Titelgröße per Design anpassbar
Fix: Rollläden und normale Schieberegler funktionieren wie vorgesehen zusammen in einer Liste
Fix: Diagramm in Energieverteilung mit ungeloggter Infovariable funktioniert wie vorgesehen
Fix: Skaliere Kachelinhalt korrekt basierend auf skalierter Titelhöhe, auch HTML-Instanzen
Fix: Erhöhe Anteil der Wertanzeige bei Vorschaudiagramm von 25% auf 50%
Fix: Verhindere Animation der angepassten Titelgröße beim Kategoriewechsel
Fix: Unter gewissen Umständen konnte zwischen Hoch- und Querformat das Layout nicht korrekt geladen werden
Fix: Formatierter Wert von Variablen mit leerer/ungültiger Darstellung wie vorgesehen

Fix: Rotation von Lautstärkeregler von Mediaplayer angepasst
Fix: Führende Nullen werden bei der Datumsformatierung berücksichtigt
Fix: Werte von Textfeldern werden nicht mehr überschrieben, wenn sich ein übergeordnetes Element aktualisiert
Fix: Die breite des Shutter-Sliders in der Listenansicht wird korrekt berechnet
Fix: Heruntergeladene Designs werden in den Apps korrekt geöffnet
Fix: Hintergrundfarbe von Dialogen
Fix: Letzter Tag bei monatlichen zyklischen Ereignissen wird korrekt angezeigt
Fix: Längeres Timeout bei Verbindung
Fix: Schriftarten werden wieder korrekt geladen für HTMLBoxen und HTML Kacheln
Fix: Neu erstellte Variablen mit Farbdarstellung funktionieren wie vorgesehen
Fix: Favoriten sind Teil des Browser-Verlaufs
Fix: Verbesserte Fehlerbehandlung für Anfragen an den Server
Fix: Doppelte Darstellung von Kategorien beim ändern der Geräte Orientierung
Fix: Hintergrundbilder von Kategorien, werden nicht mehr angezeigt, nachdem sie verschoben wurden
Fix: Höhe des Titels wird bei kleineren Kacheln angepasst
Fix: Aufzählungen mit einem Element in Gitter-Ansicht funktionieren wieder
Fix: Leere Kacheln auf manchen Android Tablets
Inhaltswechsler
    Fix: Inhalt nutzt wieder die volle Höhe
    Fix: Werte und Hintergrundfarbe aktualisieren sich korrekt
    Fix: Das Leuchten aktiver Schatten wird nicht mehr abgeschnitten
Fix: DoorIP schaltet Werte korrekt
Fix: Handhabung von fehlerhaften Gitterkonfigurationen verbessert
Fix: Fehlerhafte Anzeige der Darstellung Mediaplayer, wenn sich die Darstellung einer Variable ändert
Fix: Der Gradient Farbtemperatur kann mit negativen Werten umgehen
Fix: Fehler beim schnellen Löschen eines Servers behoben
Fix: Bei Zeitzonen ohne Namen (z.B. Etc/Utc) wird kein grauer Kasten auf der Startseite mehr angezeigt

DARSTELLUNGEN

Neu: xy Kodierung für die Farbauswahl mit CIE Diagramm zur Auswahl
    Unterstützt eigene Farbräume
    Unterstützt eigene Farbkurven
Neu: Darstellung Dauer unterstützt Stunden:Minuten
Neu: Min/Max Feld für absolute Wertanzeigen

WEBFRONT

Fix: Letzter Tag bei monatlichen zyklischen Ereignissen wird korrekt angezeigt
Fix: Bessere Konvertierung von Darstellungen im WebFront
Fix: Bei leeren Namen in Links wird der Name korrekt vom Zielobjekt geerbt

KONSOLE

Neu: Unterstütze mehr Bildformate für Medienobjekte. Insbesondere webm funktioniert nun, es sollten aber alle üblichen Formate funktionieren
Neu: Import/Export von Ablaufplänen
Neu: Unterstützung für webp bei der Vorschau von Medien (Bild)
Neu: wM-Bus: Link Mode kann eingestellt/eingeschränkt werden
Neu: Das Kopieren/Einfügen zwischen mehreren Objektbäumen auf einem System wird unterstützt
Fix: Variablen mit der Darstellung Dauer werden korrekt formatiert und zählen bei Bedarf dauerhaft hoch bzw. runter
Fix: Kleine Performance-Verbesserung bei Aktualisierung von Variablen
Fix: Korrekte Formatierung von Variablen ohne Darstellung oder mit ungültiger Darstellung
Fix: Tree's merken sich beim Aktualisieren wieder die bereits ausgeklappten Knoten
Fix: M-Bus/wM-Bus: Profile (insbes. Icons) für Wasser Geräte korrigiert
Fix: Darstellungsmanager funktioniert wieder
Fix: Beim Versuch die Dauer-Darstellung zu aktualisieren verursachen Links ohne Ziel keine Fehler mehr
Fix: Zeige die Vorschau von PNG-Bildern wie vorgesehen an
Fix: Unterstütze Darstellungen im TestCenter
Fix: Seltene Fehlermeldung beim Wechseln von Darstellungen korrigiert

SYMOS

Neu: Reconnect Timer läuft nicht mehr nur alle 30 Minuten, sondern wie das Connect Control in IP-Symcon alle 2^Fehlversuche Minuten und 30% Varianz beim Reconnect. Damit sollten anfängliche Problem mit der Internetverbindung schneller gelöst sein.
Fix: Zeitzone kann wieder korrekt gesetzt werden

PHP-SDK

Neu: Feld "validPresentations" für SelectVariable zur Einschränkung auf bestimmte Darstellungen
Neu: Dokumentation für IPSModuleStrict
Neu: IPSModuleStrict nutzt jetzt die neue GetCompatibleParent Funktion.
Neu: IPSModuleStrict wird zu IP-Symcon 8.1 auch im Module Store offiziell unterstützt
Fix: Sofern keine weitere Property/Attribut hinzugefügt/gelöscht wurde, wurde die Migration in PHP-Modulen nicht korrekt ausgeführt
Fix: Type Hints für IPSModuleStrict verbessert
Fix: SelectValue unterstützt nun wie vorgesehen Variablen mit der Darstellung Wertanzeige

AKTUALISIERTE MODULE

AKTIVLISTE

Neu: Skript zum Deaktivieren aller aktiven Geräte
Neu: Buttons für einfacheres Umbenennen

ALARMIERUNG

Neu: Nacht-Alarm

AMAZON ALEXA

Neu: Beschreibendere Namen in den Tabellen
Neu: Unterstützung von neuen Darstellungen
Neu: Zeige mehr Zeilen pro Gerätetyp an
Fix: Korrektes Handling mit PHP 8.2

BACKUP

Neu: Bessere Unterstützung für große Backups

BENACHRICHTIGUNG

Neu: Unterstützung für neue Visualisierung

CSV ZIP EXPORT

Neu: Option zur Auswahl des Dezimaltrennzeichens

EKEY BIONYX

Neu: Verbesserungen zu den WebHooks
Neu: Unterstützung für die Notification API

ENERGIE MANAGER

Neu: Vorlagen für eine einfache Konfiguration
Neu: Unterstütze Einschränkung der Energieproduktion entsprechend §9 EEG
Neu: Deaktiviere Geräte, die auch nach mindestens fünf Minuten und zehn Updates nicht reagieren
Neu: Günstiges Laden/Günstige Laufzeit
New: Neue Variable über den Betriebsmodus von Energiespeichern (wird von der Visualisierung verwendet)
Neu: Unterstützung von Statusvariablen für Energiespeicher
Neu: Unterstützung von zwei Variablen im relativen Modus: Eine für Überschuss und eine für Defizit
Neu: Unterstütze eine Steuerung der Entladung von Energiespeichern
Neu: Unterstütze prozentuale Ladeziele für Elektrofahrzeuge
Neu: Möglichkeit den tatsächlichen Verbrauch für jeden Gerätetyp anzugeben
Neu: Unterstütze das indirekte Schalten von Energiespeichern ohne Aktion
Neu: Unterstütze Geräte mit nicht-Legacy Darstellungen
Neu: Unterstütze Energiespeicher, die eine einzelne Variable für Laden und Entladen verwenden
Neu: Verbessertes Formular für Energiespeicher

Neu: Leistung für E-Fahrzeuge kann pro Phase statt insgesamt geschaltet werden
Neu: Betrachte entladende Batterien als negativen Überschuss
Neu: Unterstütze Restriktionen nach §14a
Neu: Betrachte den Verbrauch von manuell gesteuerten Geräten im absoluten Modus
Neu: Energiepreise sind optional für Nachtladen
Neu: Optionale Variablen für An/Aus und Phasenwechsel von E-Fahrzeugen
Neu: Energieeinheit für E-Fahrzeuge auswählbar: Watt oder Ampere
Neu: Aktiviere Geräte mit niedrigerer Priorität wenn die Energie nicht für höhere Priorität ausreicht
Neu: Nächtliches Laden/Laufzeit
Neu: Gerätetyp Energiespeicher
Neu: Ünterstützung von Mindestlaufzeit und Nachlaufzeit
Fix: Verhindere ein "Zittern" in bestimmten Konstellationen
Fix: Aktualisiere Modus, falls sich der Gerätestatus ändert und der Modus nicht automatisch ist
Fix: Dimme gesperrte Geräte korrekt wenn keine Energie vorhanden ist
Fix: Nachtladen konnte in manchen Konstellationen die falsche Zeit wählen
Fix: Funktioniere weiterhin wenn das Format der Variable für Energiepreise ungültig ist
Fix: "Batterie" zu "Energiespeicher" umbenannt

LINGG&JANKE KNX QUICK

Neu: Verbesserte Darstellung als Liste

SZENEN-STEUERUNG

Neu: Erlaube das Editieren von Werten in der Konsole

TREPPENHAUSLICHTSTEUERUNG

Neu: Aktion beim Deaktivieren kann ausgewählt werden

WATCHDOG

Neu: Spalte zum Aktiv schalten der Aktoren eingefügt

V7.2->V8.0 (Q1/2025)

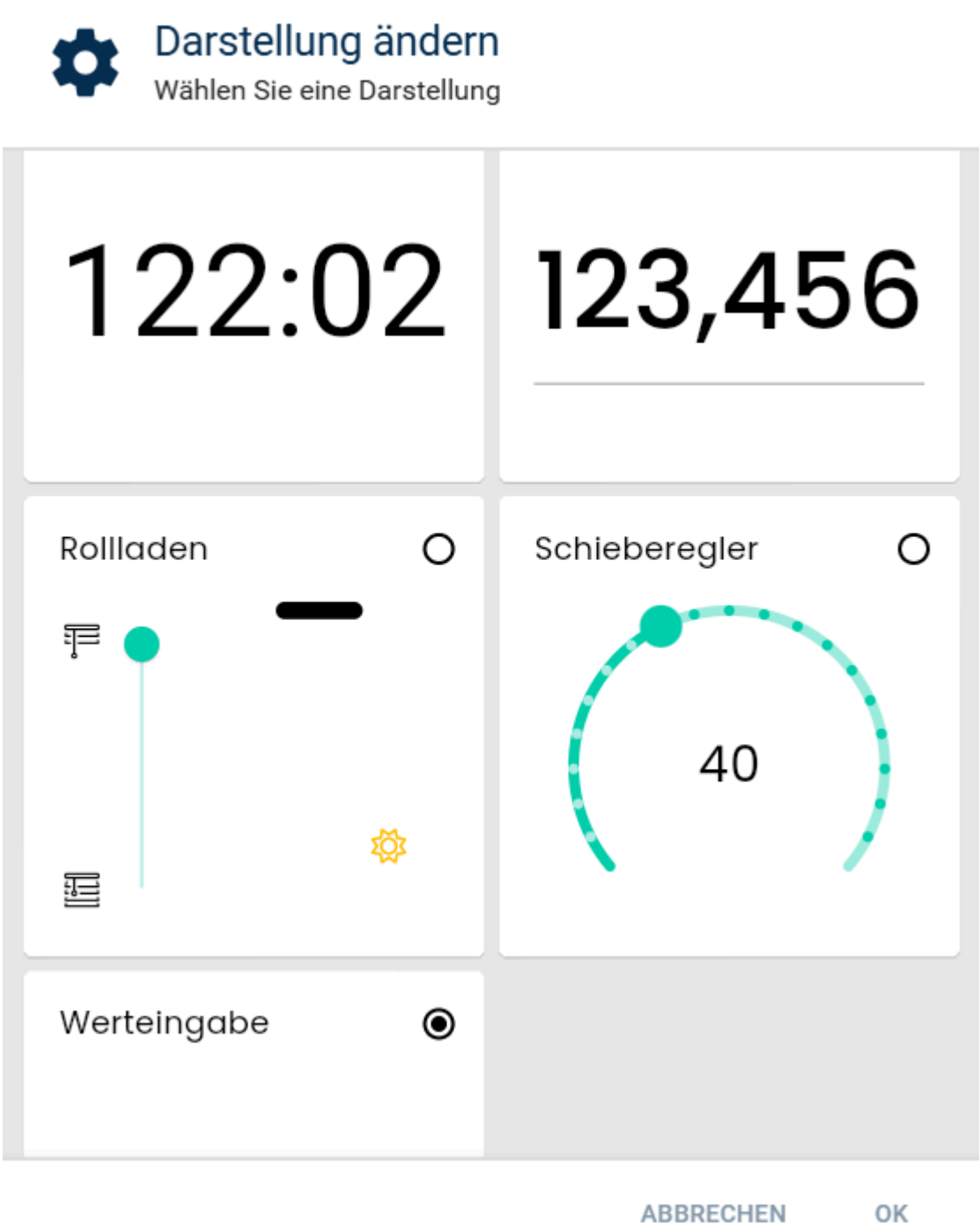
Wesentliche Änderungen
Migration
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

Darstellungen
Türsprechanlagen
Wireless M-Bus
Benutzerverwaltung

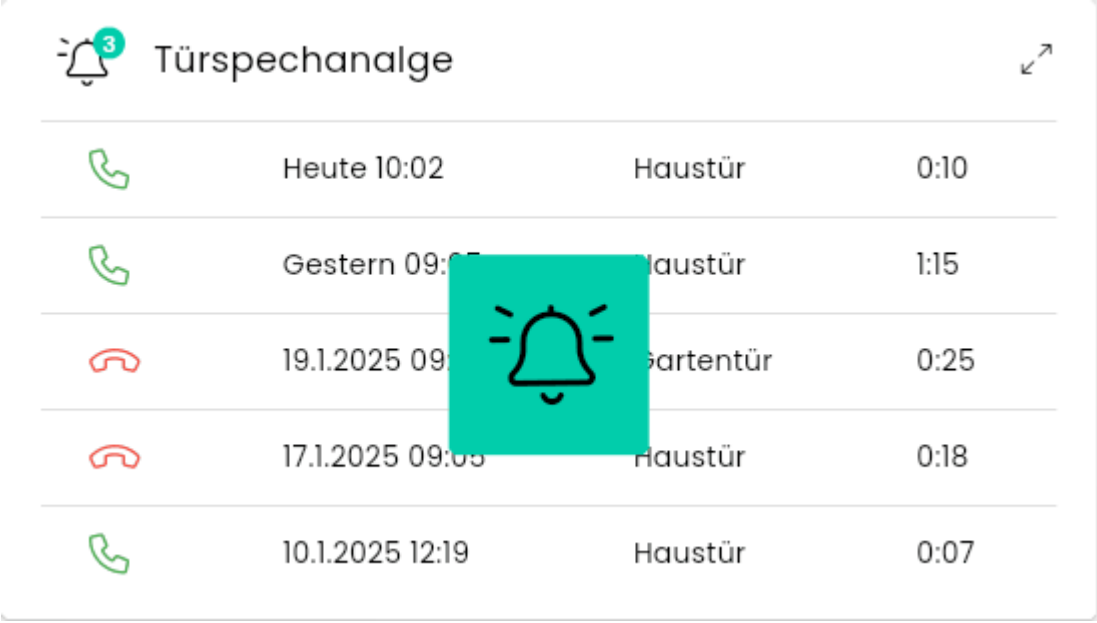
DARSTELLUNGEN

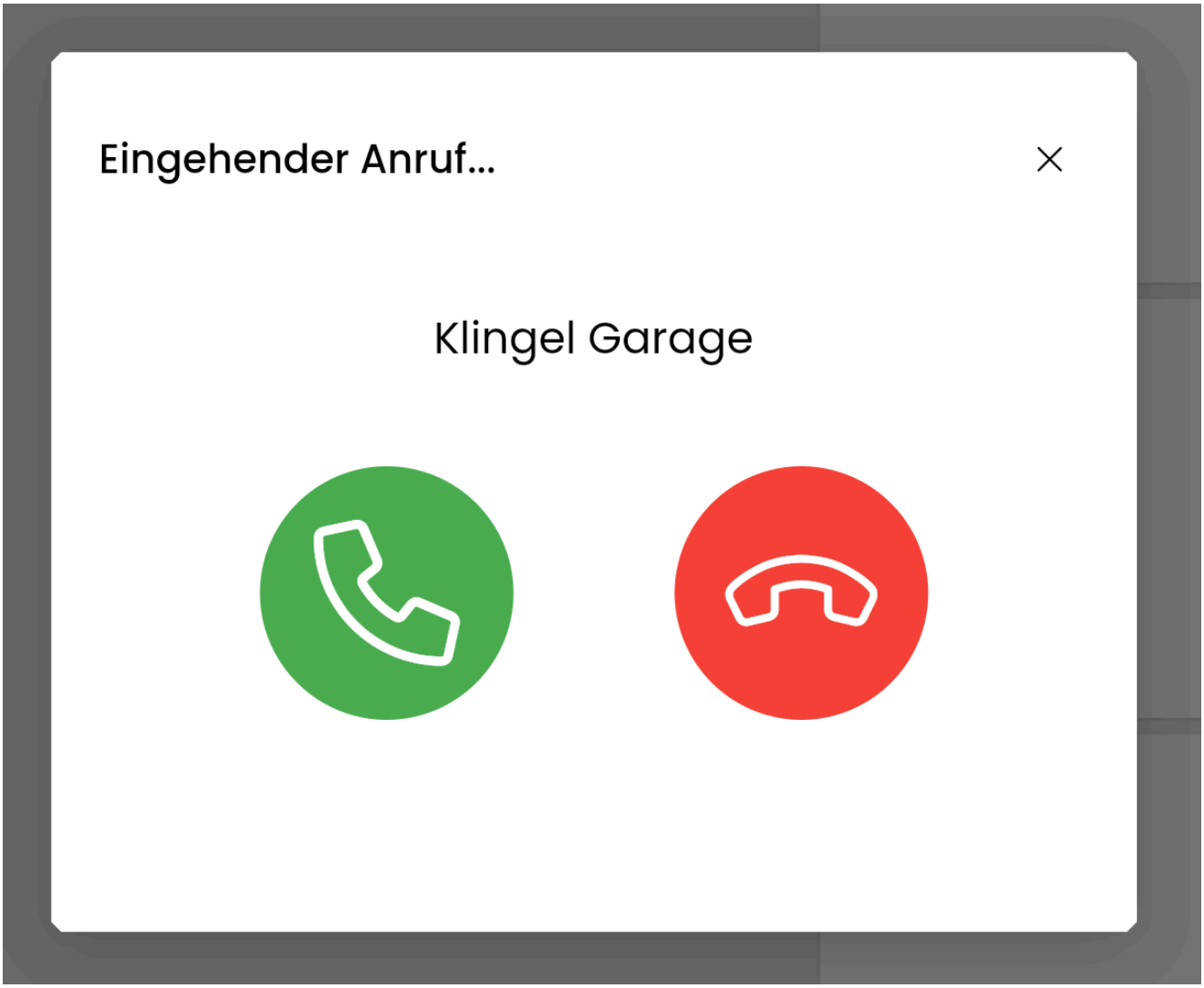
Darstellungen bieten eine neue Dimension bei der Konfiguration von Variablen. Nach der Wahl eines Darstellungstyps wie "Schieberegler", "Farbe" oder "Dauer", kann dieser individuell mit Hilfe von verschiedenen Parametern konfiguriert werden. Diese neue Struktur ermöglicht eine Vielzahl neuer Möglichkeiten für Variablen um diese in der Visualisierung zu verwenden.



TÜRSPRECHANLAGEN

Durch die Einbindung von Türsprechanlagen können diese nahtlos in der Visualisierung verwendet werden. Bei einem Klingeln kann direkt in einem Dialog das Bild der Kamera betrachtet und mit dem Besucher geredet werden um schließlich die Tür zu öffnen.





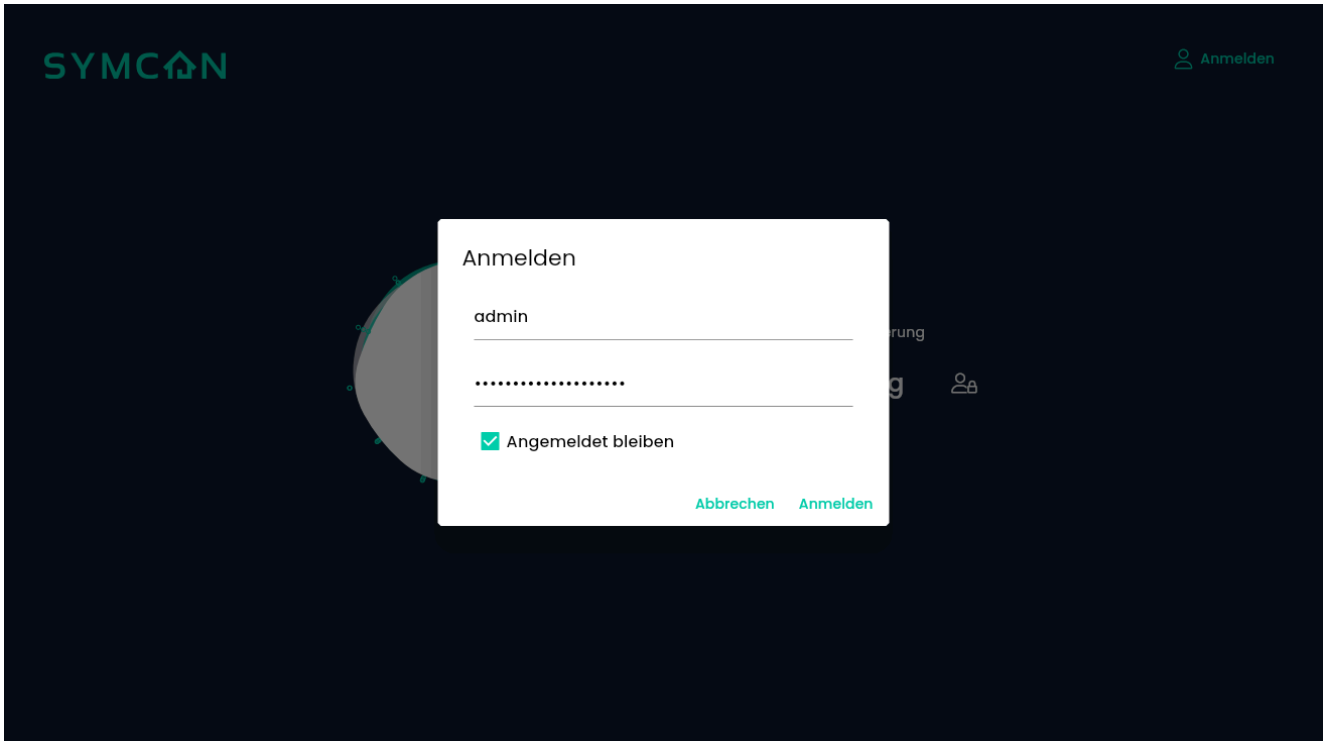
WIRELESS M-BUS

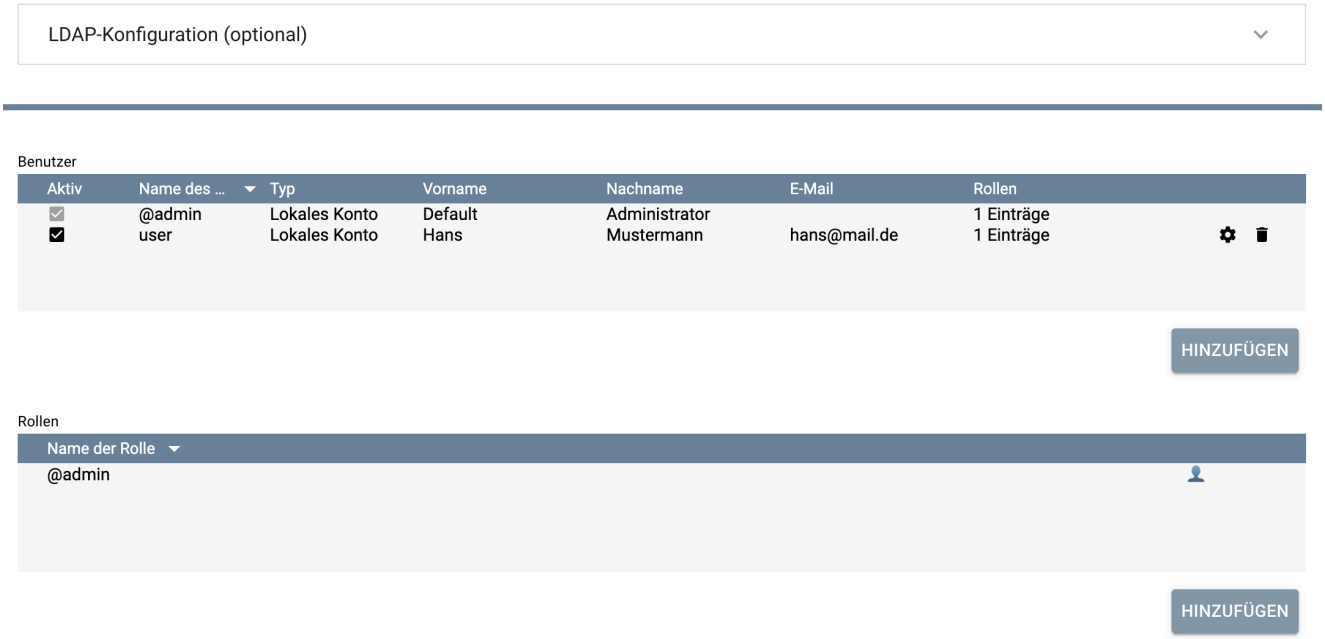
Zusätzlich zu den drahtgebundenen M-Bus-Geräten sind mit der Version 8.0 die kabellosen Geräte von Wireless M-Bus nutzbar. Diese können über die gängigen Gateways von WE und IMST angesteuert werden.

Name	ID	Hersteller	Version	Medium	Status	Verschlüsselu...	RSSI	Daten	InstanzID	
Wireless M-Bus Device (02355964, EFE, Heat)	02355964	EFE	0	Heat	0		N/A		#35671	
Wireless M-Bus Device (42069120, DME, Gas)	42069120	DME	0	Gas	179		N/A		#21970	
Wireless M-Bus Device (20119827, DWZ, Water)	20119827	DWZ	2	Water	0	OMS (Mode 5)	-61 dbm		Kein(e)	
Wireless M-Bus Device (20844457, KAM, Cold Water)	20844457	KAM	29	Cold Water	16	BSI (Mode 7)	-56 dbm		Kein(e)	
Wireless M-Bus Device (23160294, ITW, Heat)	23160294	ITW	0	Heat	19	OMS (Mode 5)	-52 dbm		Kein(e)	
Wireless M-Bus Device (43152149, EFE, Heat)	43152149	EFE	0	Heat	0	OMS (Mode 5)	-86 dbm		Kein(e)	
Wireless M-Bus Device (43152150, EFE, Heat)	43152150	EFE	0	Heat	0	OMS (Mode 5)	-86 dbm		Kein(e)	
Wireless M-Bus Device (43152155, EFE, Heat)	43152155	EFE	0	Heat	0	OMS (Mode 5)	-80 dbm		Kein(e)	
Wireless M-Bus Device (43152157, EFE, Heat)	43152157	EFE	0	Heat	0	OMS (Mode 5)	-87 dbm		Kein(e)	
Wireless M-Bus Device (43152159, EFE, Heat)	43152159	EFE	0	Heat	0	OMS (Mode 5)	-85 dbm		Kein(e)	
Wireless M-Bus Device (43152163, EFE, Heat)	43152163	EFE	0	Heat	0	OMS (Mode 5)	-81 dbm		Kein(e)	
Wireless M-Bus Device (43152169, EFE, Heat)	43152169	EFE	0	Heat	0	OMS (Mode 5)	-76 dbm		Kein(e)	
Wireless M-Bus Device (43152170, EFE, Heat)	43152170	EFE	0	Heat	0	OMS (Mode 5)	-90 dbm		Kein(e)	
Wireless M-Bus Device (43152172, EFE, Heat)	43152172	EFE	0	Heat	0	OMS (Mode 5)	-88 dbm		Kein(e)	
Wireless M-Bus Device (43152173, EFE, Heat)	43152173	EFE	0	Heat	0	OMS (Mode 5)	-82 dbm		Kein(e)	
Wireless M-Bus Device (43152175, EFE, Heat)	43152175	EFE	0	Heat	0	OMS (Mode 5)	-83 dbm		Kein(e)	
Wireless M-Bus Device (43152177, EFE, Heat)	43152177	EFE	0	Heat	0	OMS (Mode 5)	-77 dbm		Kein(e)	
Wireless M-Bus Device (43152178, EFE, Heat)	43152178	EFE	0	Heat	0	OMS (Mode 5)	-73 dbm		Kein(e)	
Wireless M-Bus Device (43152185, EFE, Heat)	43152185	EFE	0	Heat	0	OMS (Mode 5)	-85 dbm		Kein(e)	
Wireless M-Bus Device (43152186, EFE, Heat)	43152186	EFE	0	Heat	0	OMS (Mode 5)	-79 dbm		Kein(e)	
Wireless M-Bus Device (71927603, DME, Water)	71927603	DME	118	Water	112	BSI (Mode 7)	-46 dbm		Kein(e)	
Wireless M-Bus Device (76102435, DME, Heat)	76102435	DME	64	Heat	112	BSI (Mode 7)	-45 dbm		Kein(e)	

BENUTZERVERWALTUNG

Für Enterprise-Kunden bietet die neue Version eine Benutzerverwaltung (RBAC). Hier können verschiedene Benutzer angelegt und in beliebige Benutzergruppen kategorisiert werden. Visualisierungen können für Benutzergruppen oder einzelne Benutzer freigegeben werden. Benutzerkonten können entweder direkt in IP-Symcon angelegt werden oder per LDAP importiert werden.





MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP 8.3
Neu: VoIP: Lokale Nummer, die Angerufen wird in der Connection „Local“
Neu: Enterprise Funktion für Benutzerverwaltung und LDAP Synchronisation
Neu: Connect Control: Neue Buttons, um direkt die Visu, Konsole oder SymOS zu öffnen
Neu: Limit für Anzahl der Aktionen im Wochenplant entfernt
Neu: Limit für Anzahl von Assoziationen entfernt
Neu: Schalter in der Instanz der Kachelvisualisierung um die neuen Diagramme zu aktivieren
Neu: Für Logins über das RBAC/LDAP Feature, kann über den Spezialschalter „ServerTokenExpiration“ die Zeit bis zum Ablaufen des Token und einem erzwingen zum erneuten Login festgelegt werden.
Neu: GetValueFormatted für Darstellungen

Neu: ProxyTURN Spezialschalter zum Deaktivieren vom Streams Relay (bei Problemen mit zu hohem Upload)
Neu: ProxySTUN Spezialschalter, sofern die Streams nur lokal genutzt werden sollen und keine Kommunikation mit unserem STUN Server gewünscht ist.
Fix: DoorIP - Fehlende Push-Nachrichten
Fix: Notification Control: Sortierung nach Updated funktioniert nun wie erwartet
Fix: UDP Socket konnte stecken bleiben wenn der Bind beim Senden einen Fehler geworfen hat
Fix: Remote Sync: Problem mit Fehlermeldung „Endlosschleife“ korrigiert
Fix: Wird ein Profil auf einen leeren String gesetzt, wird die dazugehörige Darstellung gelöscht
Fix: DMX Output zu DMX Kanal umbenannt, da dieser Mittlerweile auch als DMX Input genutzt werden kann
Fix: Darstellungen mit Intervallen werden im WebFront korrekt angezeigt
Fix: Sync Remote: Probleme beim Löschen von Variablen gelöst
Fix: Fehler in Verbindung mit älteren Versionien der Energieverteilung behoben
Fix: WebSocket: WC_* Funktionen funktionieren korrekt, wenn Leerzeichen/Umlaute im Pfad enthalten sind
Fix: Neue Icons können für Push Nachrichten korrekt verwendet werden mit längeren Namen
Fix: Bei neuen SymBox’en wird korrekt die neue Visualisierung als Standard gewählt
Fix: Sicherheitslücke unter Windows im WebServer (Die Lücke betrifft nur die Windows Version und kann nicht über Symcon Connect ausgenutzt werden. Vielen Dank an @Dustin723 für’s finden und melden!)
Fix: Sync Remote: AC_* Funktionen konnten vom Master aus zum Remote nicht korrekt in PHP genutzt werden
Fix: Float-Werte werden in Regeln von Ereignissen korrekt verglichen
Fix: Rollladensteuerung: Bei manuellem Richtigungswechsel (und Empfang vom STOP) konnte wurde ein interner Stop Timer nicht beendet, wodurch der Rollladen bei der manuellen Fahrt ggf. falsch gestoppt wurde
Fix: Kalender setzt die Variablen wieder korrekt, wenn der Endzeitpunkt erst am nächsten Tag ist
Fix: $_IPS[‚TARGET‘] wurde bei TimerEvents nicht mehr korrekt gesetzt (vmtl. bereits seit der 7.0 defekt)

GERÄTE

Neu: Wireless M-Bus unterstützt IMST iM871A-USB und IMST iM891A-XL/iU891A-XL
Neu: EnOcean: Neuer RCM100 Modul für Rollladen
Neu: Wireless M-Bus Unterstützung
Neu: KNX: Konfigurator erlaubt das Hinzufügen von Keywords, die Entfernt werden sollen
Neu: KNX: Für die neuen KNX Geräte-Instanzen kann je GA ein optionaler Name vorgegeben werden
Neu: KNX: Konfigurator erlaubt den Export der genutzten GAs (Kann im „Secure Interface“ der SymBox in der ETS ab der 6.3 eingefügt werden)
Neu: EnOcean: Für den Modus FGW14 kann die BaseID angegeben werden
Neu: EnOcean: Beim Eltako Switch kann als Schaltmodus auch A5-38-08 verwendet werden
Neu: Charts werden jetzt immer im Vollbild angezeigt
Neu: EnOcean: Wartezeit bis Timeout kann eingestellt werden
Neu: EnOcean: Neue Instanz für den Eltako FWRB
Neu: M-Bus: Timeouts können im Gateway eingestellt werden (Sinnvoll z.B. bei 300 Baud)
Neu: wMBus: Wireless M-Bus Konfigurator zeigt Informationen besser an

Neu: EnOcean: Option, um beim Eltako Dimmer bei An auf die letzte Dimstufe anzuschalten
Neu: KNX: Physikalische Adresse der SymBox KNX Schnittstelle kann direkt aus der Splitter Instanz gesetzt werden. (Notwendig für die ETS Applikation 1.1 der SymBox)
Neu: Eltako FHK14 Instanz kann mit einem Thermokon SR0x kombiniert werden zur komfortableren Einzelraumregelung
Neu: EnOcean: Timeout zwischen zwei Befehlen kann eingestellt werden
Fix: Assoziationen mit %f /%d funktionieren wieder wie vorgesehen
Fix: Skripteditor unter Firefox 131.0.2 geht wieder
Fix: KNX: Einige TTL Dateien konnte nicht gelesen werden mit mit einem „stoi“ Fehler
Fix: KNX: Beim TTL Export wurden bereits erkannte Status GA nicht korrekt aus der Liste gefiltert
Fix: KNX: Beim TTL Export wurde der „unbekannt“ Tag nicht korrekt dargestellt
Fix: KNX: DPT 1.024 und 1.100 haben jetzt korrekte Profile (waren seit der 7.2 vertauscht)
Fix: OPCUA: Variablen werden wie vorgesehen nicht entfernt, falls der Datentyp vom Datenpunkt sich ändert
Fix: HomeMatic: Beim Trennen der Verbindung wurde ein falsches Paket gesendet
Fix: EnOcean: Korrigiere Profil für D5-00-01 (.Reversed fehlte)
Fix: OPC UA: Sofern der Server den Wert „Null“ gesendet hat, wurde die Variable gelöscht und dann, beim Empfang des korrekten Werts, mit anderer ID neu erstellt. IP-Symcon verhindert dies nun und loggt das Problem ins Meldungsfenster.
Fix: KNX: Der SubTag Lamelle wurde nicht korrekt ausgewertet bei neuen KNX Geräten
Fix: wMBus: Bei Mode 7 mit Informationen in der TPL wurde ggf. nicht korrekt entschlüsselt
Fix: MQTT Client: Retain Flag wurde nicht korrekt weitergeleitet
Fix: ModBus: Schreiben von Hex-kodierten Strings korrigiert
Fix: EnOcean: FHK14 Thermostat Schwellwert verbessert

KACHEL VISUALISIERUNG

Neu: Neue Graphen können verwendet werden
Neu: Darstellung von Nachtladen im Energieverbrauchoptimierer
Neu: Prognose für Energieoptimierer wird unterstützt
Neu: In den Servereinstellung werden verbundene Server/Visu hervorgehoben
Neu: Im Server Info Dialog wird der Verbindungstyp angezeigt
Neu: Annehmen von Anrufen direkt über die DoorIP-Instanz möglich
Neu: Nächtliches Laden für Batterien im Energieoptimierer
Neu: Beim Schalten von Variablen wird der neue Wert sofort angewandt und ggfs. nach kurzer Zeit zurückgesetzt, falls er vom Server nicht bestätigt wird. Dadurch fühlen sich diverse Bedienvorgänge flüssiger an.
Neu: Neue Typen Kälte und Klimaanlage für Energieverteilung
Neu: Hinweis und korrektes Handling, falls sich der Server in einer anderen Zeitzone befindet
Neu: Nächtliches Laden für Verbraucher im Energieverbrauch-Optimierer
Neu: eigenes Format für Uhrzeitdarstellung in der Appbar
Neu: Energieoptimierer
Neu: Nicht-stören-Modus für Door IP
Neu: LDAP: „angemeldet bleiben“ Option im Web
Neu: Ansicht von Aufzählung als Dropdown, Aufzählung nebeneinander und Wertedarstellung vereinheitlicht
Neu: Halloween Theme
Neu: Animation in der Energieverteilung sind wieder da.

Neu: Dem HTML-SDK und den HTML-Boxen stehen jetzt verschiedene Farben in Form von CSS-Variablen zur Verfügung (–accent-color, --content-color, --card-color). Die Variablen sind einfach über
var(--accent-color) abrufbar.
Neu: Warnung beim Löschen eines verbundenen Servers
Neu: Bei neuen Graphen werden wie ursprünglich die Namen der Boolean-Graphen eingeblendet
Neu: Die neuen Charts aktualisieren sich nach 2 Minuten, wenn der aktuelle Zeitpunkt angezeigt wird
Neu: Generator Typ für Energieverteilung
Neu: Einstellbares Format für den angezeigten Namen des Inhaltswechslers
Neu: Instanzkonfiguration
    Die Kategorieleiste kann in einer Zeile dargestellt werden.
    Option Kategorien nur in der Seitenleiste anzuzeigen
Neu: In den Apps werden Kacheln im Editiermodus über ein langes drücken verschoben
Neu: Nutze beim Kopieren der Versionsinformation ein formatiertes Datum anstatt eines Zeitstempels
Neu: Ladeanzeige beim Laden von neuen Werten für Graphen
Neu: Option für statische Begrüßung
Fix: Edit-Button wird auch in der Kachel ausgeblendet, wenn der entsprechende Schalter in der Visualisierungsinstanz deaktiviert wird
Fix: Löschen eins verbundenen Servers funktioniert wie vorgesehen
Fix: Der Schieberegler kann bei bestimmten Schrittweiten nicht mehr über seine Grenzwerte hinaus geschoben werden.
Fix: Sekundäre Variable in der Energieverteilung ist jetzt immer verfügbar
Fix: Bedienelemente für Aufzählungen überarbeitet
Fix: Breite von Slidern in Listenansicht vereinheitlicht
Fix: Komma als Dezimaltrennzeichen für Profile
Fix: LDAP: Fehler, wenn es nur Visualisierungen gibt, die nur für autorisierte Benutzer angezeigt werden.
Fix: LDAP: Drawer: Visualisierungen die nur für autorisierte Benutzer sind werden auch nur für diese angezeigt.
Fix: LDAP: Kein automatisches Öffnen, wenn eine versteckte Visualisierung mit Autorisierung vorhanden ist
Fix: Stringvariablen werden aktualisiert, wenn das Eingabefeld abgewählt wird
Fix: Größe von Textfelder in der Listendarstellung verringert, damit es besser in die Zeile passt
Fix: Fehler in Verbindung mit dem Passwortdialog behoben
Fix: Endlosschleife beim Verbinden mit einzelnem Server behoben
Fix: Liste gespeicherter Server wird richtig aktualisiert
Fix: Nicht-Geräte Instanzen können nur noch über Links angezeigt werden
Fix: Zuordnung von Fehlermeldungen verbessert
Fix: Korrigierte Ausrichtung von TextBoxen im Vollbild-Modus
Fix: Ist eine Visualisierung nur im lokalen Netz verfügbar, wird sie bei einem Zugriff über Connect als gesperrt dargestellt
Fix: Korrekte Prozentberechnung in der Energieverteilung, bei Prozentprofilen, die nicht von 0 bis 100 laufen
Fix: Events werden aus Dialogen heraus richtig aktualisiert
Fix: Erlaube Visu-Wechsel bei gelöschter Hauptkategorie
Fix: Bei Kategorie Kacheln wird nur ein Icon angezeigt
Fix: Icons werden beim Konfigurieren von Instanzen korrekt in der Titelzeile angezeigt
Fix: Größe und Position von Icons der Automationen angepasst

Fix: Alle Kombinierte Elemente werden als deaktiviert angezeigt, wenn sie deaktiviert sind
Fix: Objekte werden als deaktiviert angezeigt, wenn übergeordnete Objekte deaktiviert sind
Fix: Beim Konfigurieren von Instanzen wird die Sichtbarkeit des tatsächlichen Objektes genutzt, nicht die des Ursprungsobjektes
Fix: Icons dürfen jetzt 64 Zeichen lang sein für Push Nachrichten
Fix: Wird das Bild einer Kategorie aktualisiert, so wird die Kachel direkt aktualisiert
Fix: Variablen mit ~UnixTimestamp Profil können wieder verändert werden
Fix: Lesbarkeit des Zeitauswahldialogs im Darkmode verbessert
Fix: Graphen können nun aus der Energieverteilung geöffnet werden, falls der Button über HTML-Inhalt schwebt
Fix: Variablen mit HTML-Inhalt werden auch in der App neu geladen, falls der Variableninhalt aktualisiert, aber nicht geändert wird
Fix: Gehe in einen Fehlerzustand, wenn die Übergabe zu einer neuen Verbindung nicht funktioniert, anstatt bei „Verbinde…“ festzuhängen
Fix: Wird ein Reconnect durchgeführt warten Anfragen an den Server bis dieser abgeschlossen ist
Fix: Versuche bei einer Testnachricht ohne dazugehörige Visualisierung nicht eine nicht-existente Visualisierung zu öffnen
Fix: Skalierung in HTML-Elementen, die bei einigen mobilen Geräten falsch war, sollte nun korrekt funktionieren
Fix: Fange seltenen Absturz bei SVG-Graphen ab
Fix: Ehemalige Heuristik für HTML-Boxen wird weiterhin bei Zugriff auf ältere Symcon-Systeme angewandt
Fix: Datumsdarstellungen in Listenansicht nutzen wie vorgesehen nur den vorhandenen Platz
Fix: Lade HTML-Inhalte bei maximierten Kacheln nicht beim Ein- oder Ausblenden einer Tastatur neu
Fix: Fehler bei maximierter Thermostat Instanz behoben
Fix: Größe des Popups der Energieverteilung skaliert mit dem Inhalt
Fix: Die Zeit kann bei einmaligen Ereignissen wie vorgesehen ausgewählt werden
Fix: Datum/Uhrzeit Dialoge von Ereignissen nutzen den gleichen Stil, wie alle anderen
Fix: Fehler im HTML-SDK werden an den Benutzer weitergeleitet
Fix: Zurücksetzten des Kachel-Layouts bei fehlerhafter Gitterkonfiguration
Fix: Doorip - Dialog wird angezeigt, wenn bei geöffneter App auf eine Push-Benachrichtigung getippt wird
Fix: Energieoptimierer: Bedingungstext wird bei wenig Platz in eine zweite Zeile geschoben
Fix: Energieverteilung: geöffnete Dialoge werden nicht mehr am Bildschirmrand abgeschnitten
Fix: Buttons bei maximiertem Chart werden jetzt immer korrekt dargestellt
Fix: Chart Daten werden initial besser angezeigt
Fix: Sehr ähnliche Werte werden im Chart besser dargestellt
Fix: Slider wird nie ohne Label angezeigt
Fix: Hintergrund-Gradient wird bei maximierten Charts nicht mehr angezeigt
Fix: Im Wochenplan können Zeiträume < 30 Minuten ausgewählt werden
Fix: Rundungsfehler bei negativen Werden bei Schiebereglern
Fix: Uhrzeit wird bei dem Wert 0 korrekt dargestellt
Fix: Nicht existierendes Profil bei Variablen ohne Aktion führt nicht mehr zum Absturz
Fix: Versteckte Visualisierungen konnten IP-Autostart nicht verwenden
Fix: Geloggte String-Variablen nutzen immer die bestehende Tabellen-Darstellung
Fix: Die Einstellungen von verlinkten Inhaltswechsler werden wie vorgesehen übernommen
Fix: HTMLBox’en werden bei Variablenänderung korrekt aktualisiert
Fix: Overlay der Energieverteilung funktioniert korrekt über HTMLBoxen

Fix: Nicht existierende Objekte in den Favoriten führen nicht mehr zu Fehlern
Fix: Das Löschen von leeren Bereichen im Editiermodus funktioniert wie vorgesehen
Fix: Schieberegler werden hervorgehoben, wenn ihre Farbe dem Hintergrund entspricht
Fix: Streams verbinden sich nach einem Verbindungsabbruch neu
Fix: Die Rollladen Darstellung erkennt jetzt die neuen Variablendarstellungen
Fix: Scrollen in der Ausgabe von Skripten möglich
Fix: Verbesserte Logik für die Wiederverbindung von Streams
Fix: Punktbewegung der Energieverteilung aktualisiert sich wieder korrekt
Fix: URLs von hinzugefügten Servern werden aktualisiert
Fix: Bei mehreren Werten zum gleichen Zeitpunkt werden diese im neuen Graphen korrekt sortiert
Fix: Aktualisiere bei Verbindung via URL die Serverinformationen und verhindere so doppelte Einträge in der Serverliste oder unpassende Namen und Versionen
Fix: Hinweis, wenn die Stream Verbindung über das Relay stattfindet, wird wieder angezeigt
Fix: Kennwörter, welche UTF-8 Zeichen enthalten funktionieren wieder korrekt

DARSTELLUNGEN

Neu: Darstellungen sind nun nutzbar.
Neu: Legacy für die bisherigen Profile
Neu: Schieberegler
    Tausendertrennzeichen
    Intervalle
Neu: Shutter
    Zusätzliche Parameter für die Lamellenposition und Sonnenposition
Neu: Datum/Uhrzeit
    Datum und Uhrzeit können separat angezeigt werden
    Das Darstellungsformat kann angepasst werden
Neu: Dauer
    Dauer bis/seit einem Zeitpunkt
    Aktuell auf Stunden begrenzt
Neu: Darstellung Aufzählung
    Verschiedene Anordnungen: Spalte, Reihe, Liste
Neu: Darstellung Schalter
    Eigene Icons für Werte
    Anpassungen des Leuchtens
Neu: Werteingabe
    Präfix/Suffix
Neu: Wertanzeige
    Tausendertrennzeichen
    Intervalle
Neu: Webinhalt (ehemals ~HTMLBox)
    Direktes einbinden einer Webseite
Neu: Farbe
    Verschiedene Kodierungen: RGB, CMYK, HSV, HSL
Neu: Icons sind bei nicht Legacy-Darstellungen Teil der Darstellung
Neu: Vorlagen für Wertanzeige uns Schieberegler

WEBFRONT

Neu: Darstellungen werden im WebFront und den mobilen Apps angezeigt
Fix: Verhindere doppelte Präfix/Suffix in konvertierten Wertanzeigen

KONSOLE

Neu: Hinweis beim versuch eine Vorlage zu wählen, wenn keine existiert
Neu: Funktion zum Umwandeln von Profilen in neue Darstellungen
Neu: Tooltips für zu lange Icon-Namen
Neu: Vereinheitliche Dialoge für Referenzprüfung, verbessert insbesondere den Referenzdialog im Module Store
Neu: Bei neu erstellten Variablen wird standardmäßig eine passende Nicht-Legacy-Darstellung ausgewählt
Neu: Speichern von Vorlagen direkt aus dem Variablen-
Neu: Passe Anzahl der Elemente pro Zeile bei der Auswahl von Darstellung oder Vorlage an den verfügbaren Platz an
Fix: Ungültige Vorlagen können gelöscht werden
Fix: Beim Erstellen von Vorlagen kann, falls benötigt, die passende Gruppe ausgewählt werden
Fix: Benutzerdefinierte Darstellung von Statusvariablen kann gelöscht werden
Fix: Benutzerdefinierte Darstellungen wird wie erwartet bevorzugt, wenn vorhanden
Fix: Archivdialoge können nun auch bei sehr langen Variablennamen problemlos geschlossen werden
Fix: Größe der Kachel kann im Vollbild der Vorschau der Darstellungen wieder verändert werden
Fix: kosmetische Fehlermeldung beim öffnen des Variablen bearbeiten Dialogs verhindert
Fix: Aus dem Meldungsfenster heraus geöffneter Wert bearbeiten Dialog funktioniert wie vorgesehen
Fix: Mehrfachauswahl in Konfiguratoren funktioniert wieder wie vorgesehen
Fix: Übersetzung von Icon Namen verbessert
Fix: Fehler beim öffnen des Profil Dialogs
Fix: Objektbaum wird nach dem ändern einer Darstellung aktualisiert
Fix: Verbessertes Bearbeiten der Darstellung von mehreren Variablen
Fix: Deaktivierte Standardaktionen werden von Darstellungen berücksichtigt
Fix: Fehlerhafte Ketten Verbindungen
Fix: Fehlerhafte Erkennung ob eine Variable eine Aktion hat
Fix: Reaggregations-Dialog wird nur noch angezeigt, wenn benötigt
Fix: Gruppen von Vorlagen werden korrekt ausgewertet
Fix: Profile werden beim ändern von Darstellungen aktualisiert
Fix: Möglichen Fehler beim Speichern/Laden von Vorlagen behoben
Fix: Wende Konvertierung von Darstellungen bei der Wertdarstellung korrekt an, sodass keine Nachkommastellen bei Integer-Werten angezeigt werden
Fix: Vorlagenauswahl kann bei unglücklichem Timing nicht mehr die Auswahlkringel verhindern
Fix: Verhindere Fehler beim Versuch in einem Archiv mit sehr vielen Variablen aufzuräumen
Fix: Verdichtung kann wieder korrekt konfiguriert werden
Fix: Variablen mit der Darstellung Dauer werden korrekt formatiert und zählen bei Bedarf dauerhaft hoch bzw. runter

SYMOS

Fix: Aktivierung von Lizenzen (über Key) funktioniert über einen neuen Endpunkt ***

PHP-SDK

Neu: Neue Bedingung „hasNoIdent“ für Aktionen um diese nicht anzuzeigen, falls eine Instanz ein untergeordnetes Objekt mit einem bestimmten Ident hat
Neu: MaintainVariable* Funktion erlaubt jetzt auch die Angabe einer Presentation als Array
Neu: RegisterVariable* Funktionen erlauben die Angabe einer Presentation als Array
Fix: Listen sind jetzt in Trees nutzbar
Fix: IPSList als Type Hint für IPSModule und IPSModuleStrict erlauben

AKTUALISIERTE MODULE

TELEFONKETTE

Neu: Hinzufügen einer Ansage via Text To Speech(AWSPOLLY)

V7.1->V7.2 (Q3/2024)

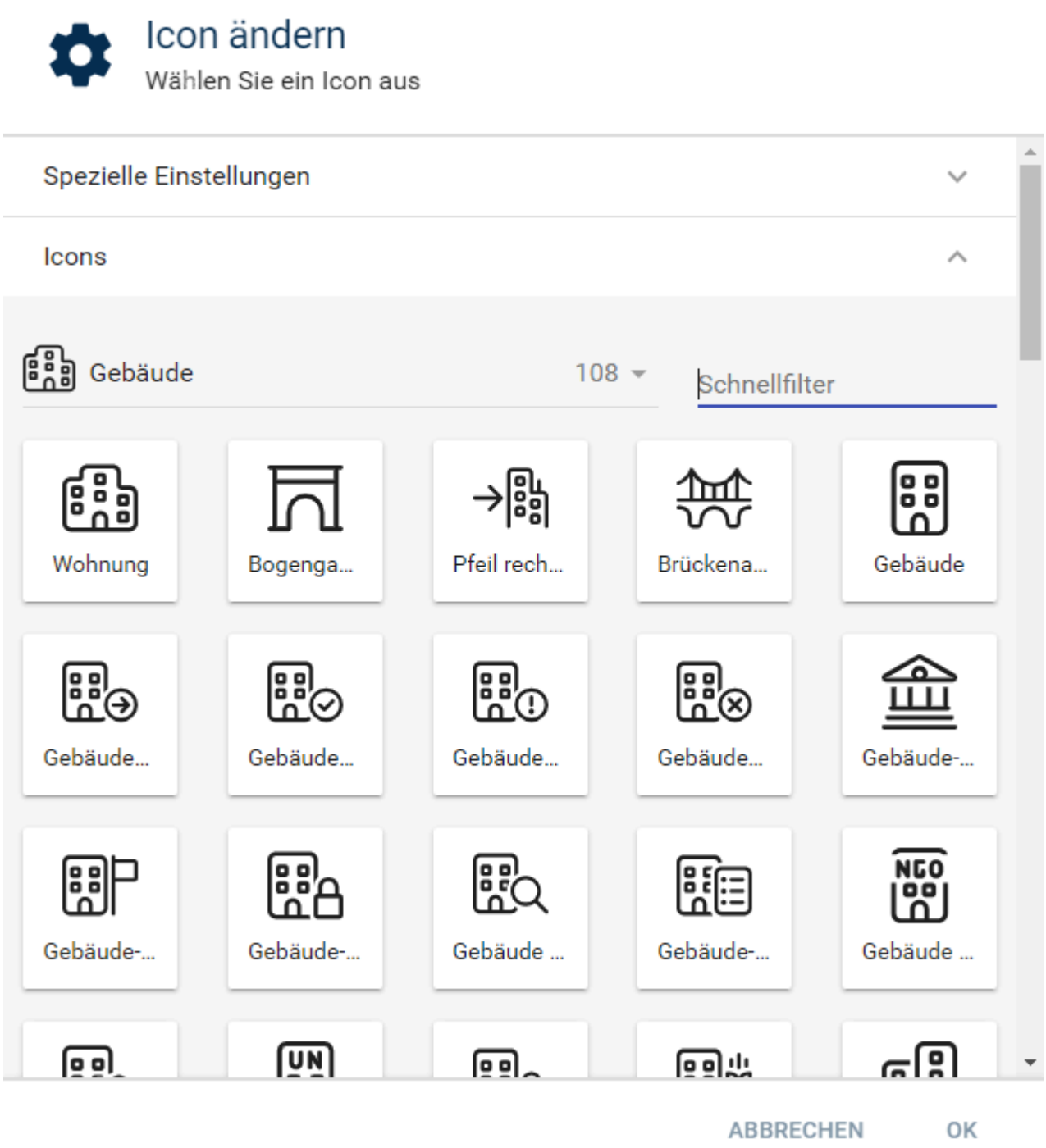
Wesentliche Änderungen
Migration
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

Neue Icons
KNX: Semantischer Export
Verbesserte Synchronisation entfernter Systeme
Stabile Streams in Visualisierung

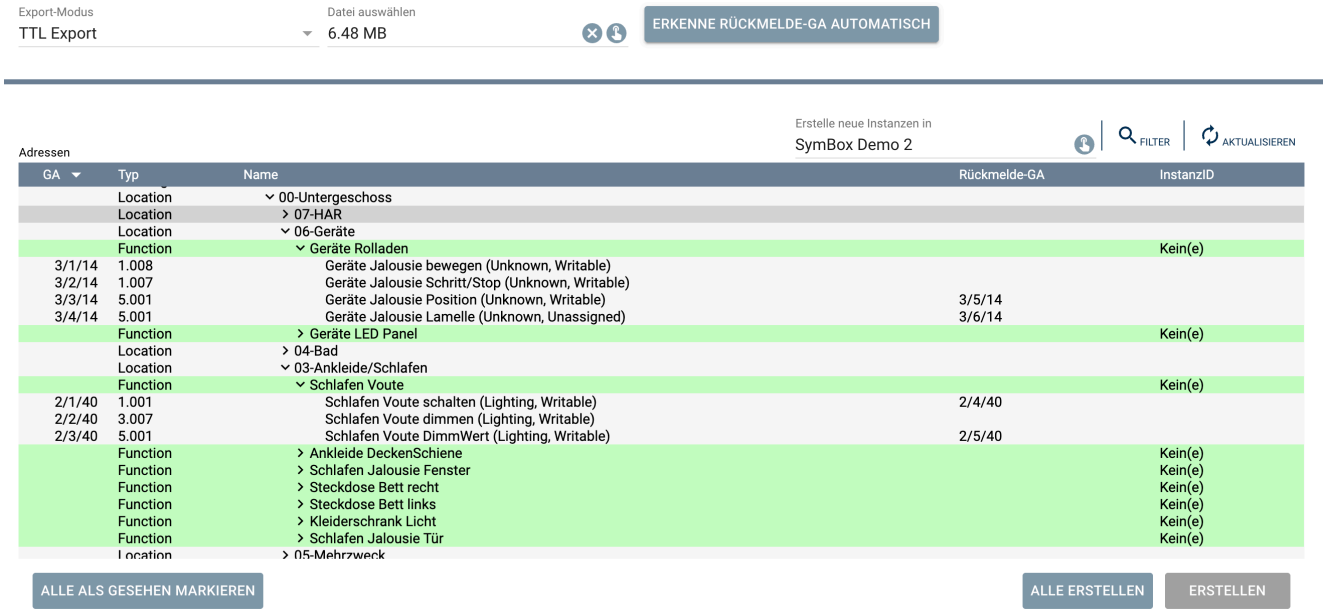
NEUE ICONS

Der Umfang der Icons wurde um ein vielfaches erhöht. Anstatt vorher 165 Icons kann nun aus den 3787 verschiedenen Icons von Font Awesome Pro für alle möglichen Anlässe ausgewählt werden. Damit bei dieser umfangreichen Iconmenge dennoch das optimale Icon gewählt werden kann, unterstützt die Verwaltungskonsole eine Aufteilung nach Kategorie sowie einen intelligenten Schnellfilter.



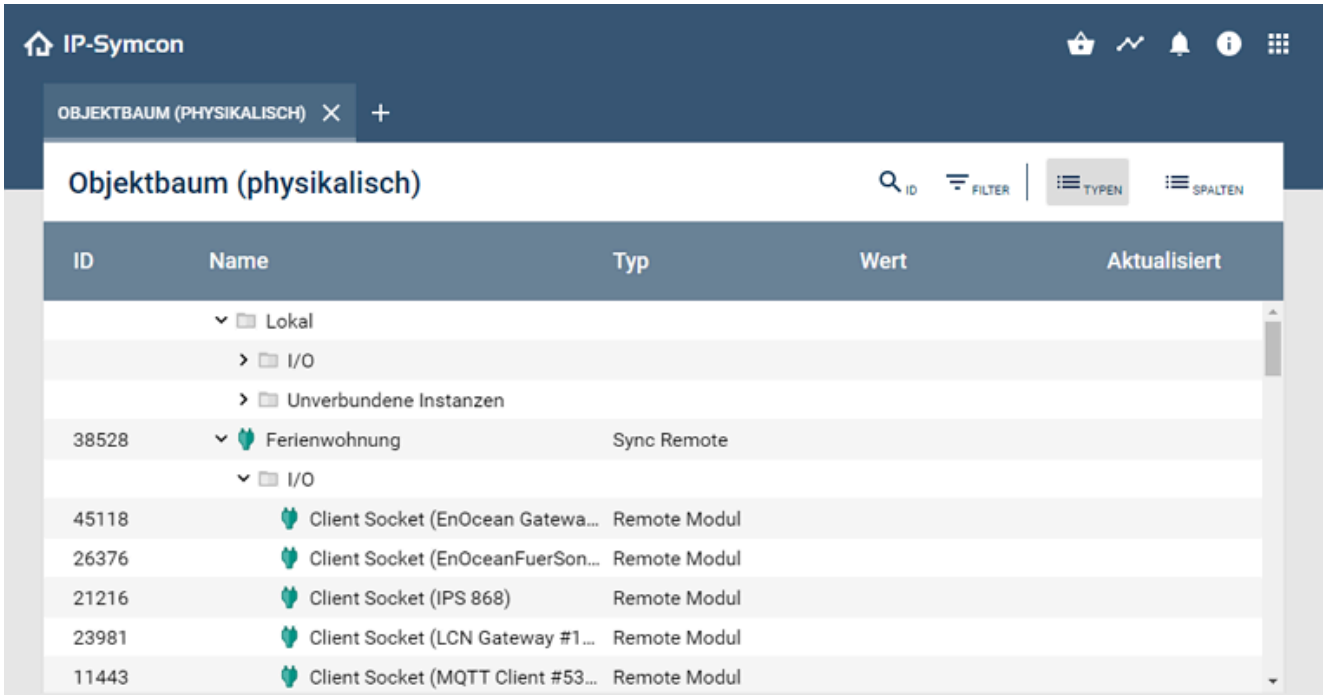
KNX: SEMANTISCHER EXPORT

Im KNX Konfigurator ist nun der semantischen Export (TTL) integriert. So kann die Zeit zur Einrichtung der Visualisierung auf ein Minimum reduziert werden. Seit der ETS 6.1.1 ist der semantische Export verfügbar, der viele weitere Informationen aus dem KNX Projekt exportieren kann. Sofern das KNX Projekt mit der Gebäudestruktur und den Gerätefunktionen projektiert wurde, kann diese Information genutzt werden, um die Visualisierung fast vollständig damit aufzubauen - ein manuell Nacharbeiten oder Zuweisen von Status-Adressen, Namen oder der Gebäudestruktur ist nicht mehr erforderlich.



VERBESSERTE SYNCHRONISATION ENTFERNTER SYSTEME

Die Einbindung des Sync Remote für unsere Enterprise-Kunden wurde weiter verbessert. Die Konsole zeigt nun auch Spezialkategorien wie Kern, Discovery, etc. des entfernten Systems an. Auch wird die Darstellung im physikalischen Objektbaum nun nach lokalen und entfernten Systemen aufgeteilt, sodass schnell der physikalische Aufbau eines bestimmten entfernten Systems ausgewertet werden kann.



STABILE STREAMS IN VISUALISIERUNG

Die Stabilität von Videostreams in der Visualisierung wurde verbessert. Vorher konnte es in manchen Konstellationen dazu kommen, dass Streams abbrechen oder IP-Symcon abstürzt. Die entsprechenden Fehler wurden behoben, sodass Streams nun in allen Fällen wie vorgesehen in der Visualisierung verwendet werden.

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: Logikpläne (ehemals Workflows) sind jetzt nativ integriert. (Das Module Store Modul kann gelöscht werden)
Neu: Button zum bequemen Erstellen von passenden Zertifikaten für alle SSL Instanzen wie Server Socket, Web Server oder OPC UA Client
Neu: Connect: Link zum Versenden per E-Mail wieder hinzugefügt
Neu: Sync Remote: Burst Abfragen nach einem RequestAction (um Traffic zu sparen)
Neu: Sync Remote: Unterstützung für Diff-Snapshots, um Traffic zu sparen
Neu: HTTP Client: VerifyHost/VerifyPeer Option
Neu: HTTP Client: Eigene Header können definiert werden
Neu: Translate: Anzeige der erkannten Systemsprache
Fix: Sync Remote: Fehler (Invalid Data Type) mit komplexeren Rückgabetypen gelöst
Fix: Fehlermeldung bei Aktion „Lese eine Objekteigenschaft aus“ gelöst

GERÄTE

Neu: KNX: TTL (Semantischer Export) kann im Konfigurator importiert werden
Neu: KNX: KNX Gerät als Instanz, welches mehrere DPTs unter einer Instanz abbilden kann
Neu: KNX: Gruppenadresse der Instanz wird in der Info Spalte angezeigt
Neu: DMX: DMX Input Instanz, um z.B. DMX Fader über einen ArtNET Input empfangen zu können
Neu: BACnet: Unterstützung für den Typ „Accumulator“
Neu: ModBus: Fix für Abfrage von Datenblöcken in Kombination mit „Little Endian (Bytes getauscht)“

KACHEL VISUALISIERUNG

Neu: Font Awesome Pro (Light + Brands) integriert (Insgesamt >3000 Icons)
Neu: Neue Fehlermeldung, wenn Endgerät keinen h264 Dekoder besitzt

Neu: Einstellungen für Energieverteilung wurden in die Kachelkonfiguration verschoben (Danke an @yansoph!)
Neu: Sprachen-Auswahl auf Login-Seite bei Web, sofern das Translation Modul aktiv ist
Neu: Kategorien die als Kachel dargestellt werden können ihr Icon als Inhalt anzeigen
Neu: Zeilenhöhe der Listendarstellung angepasst
Fix: Zustand des Wochenplans wird korrekt synchronisiert
Fix: Demo funktioniert wieder korrekt
Fix: Verbesserungen bei parallelen Streams in der Kachel Visualisierung (weniger Abstürze!)

WEBFRONT

Neu: Font Awesome Pro (Light + Brands) integriert (Insgesamt >3000 Icons)

KONSOLE

Neu: Sync Remote: Darstellung vom spezial Kategorien (z.B. I/O, Splitter…) direkt unterhalb der Sync Remote Instanz
Neu: Sync Remote: Sync Remote Instanzen werden im physikalischen Objektbaum beachtet
Neu: Sync Remote: Remote IDs werden im Skript Editor korrekt aufgelöst
Neu: Auswahl aller neuen Font Awesome Icons inkl. Filterfunktion
Fix: Z-Wave Assozationen konnten nicht gelöscht werden

SYMOS

Neu: Aktuell kann eine /mnt/system/.wireguard erstellt werden, die dann beim Start korrekt geladen wird und den wg0 aktiviert. Wir werden dies noch erweitern, sodass die Config Datei direkt über UI im SymOS geändert werden kann und auch zur Laufzeit übernommen wird.

PHP-SDK

Neu: Font-Awesome Pro Icons (Light/Brands) können direkt verwendet werden
Fix: SelectValue in Konsole korrigiert
Fix: CheckBox wird bei disabled = false korrekt dargestellt

AKTUALISIERTE MODULE

RECHENMODULE

Neu: Modul zum Skalieren zwischen zwei Wertebereichen

REPORT MODUL

Fix: Zeige Verbrauch korrekt an, Lücken beim Multi Report sind nun möglich

SNOM

Neu: Unterstützung für Expansion Modul

SPOTIFY

Fix: Korrektes Handling der Wiedergabe von gespeicherten Tracks

TAILSCALE VPN

Fix: Fehlende Übersetzung und Doku-Link ergänzt

V7.0->V7.1 (Q1/2024)

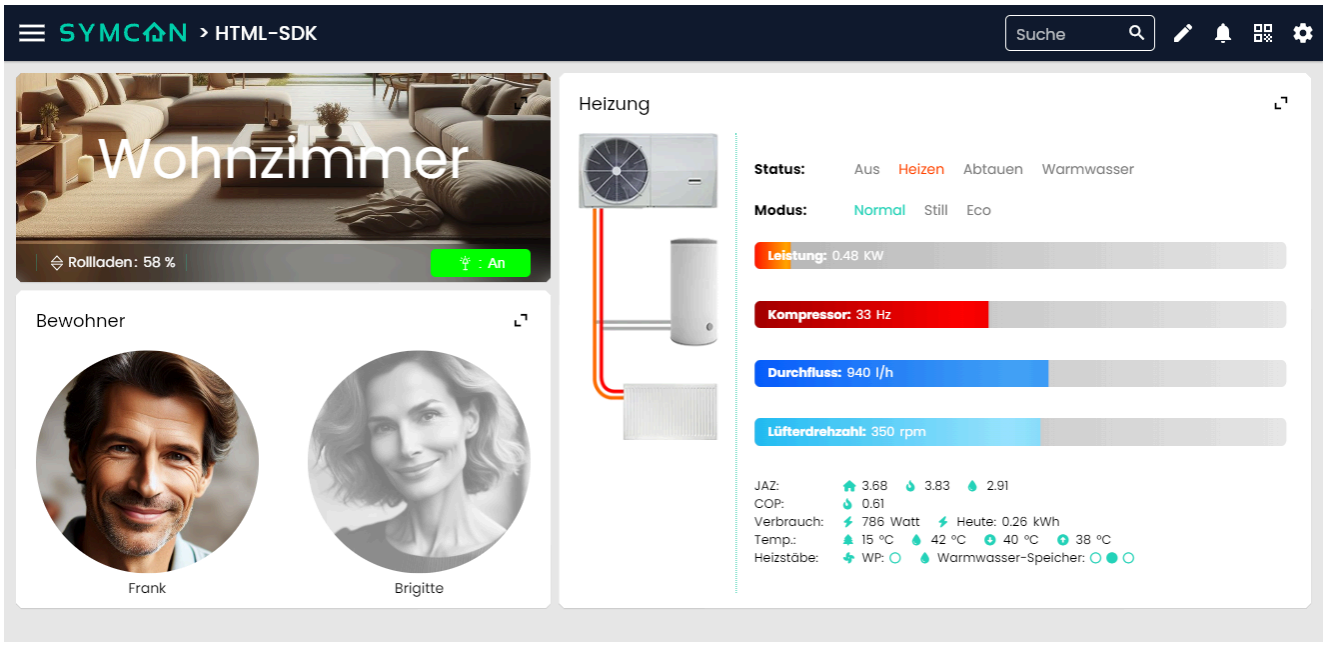
Wesentliche Änderungen
Migration
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

HTML-SDK
Logikpläne
Optimierte Darstellungen
VPN per SymBox

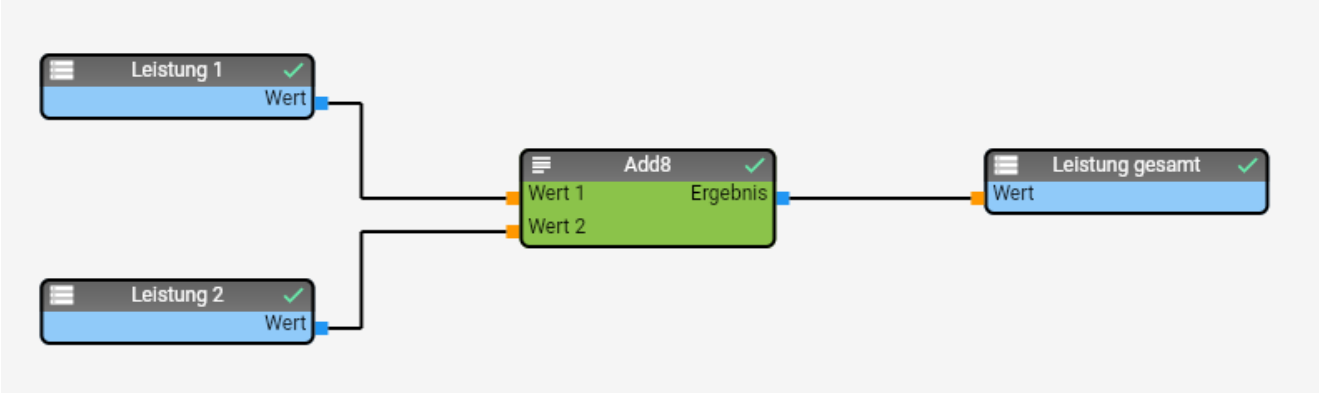
HTML-SDK

Das neue HTML-SDK ermöglicht es Entwicklern individuelle Darstellungen für ihre Module anzubieten. So werden ganz neue Darstellungen wie die Wärmepumpe oder Anwesenheitsanzeige ermöglicht, welche die Kachel Visualisierung noch einmal aufwerten.



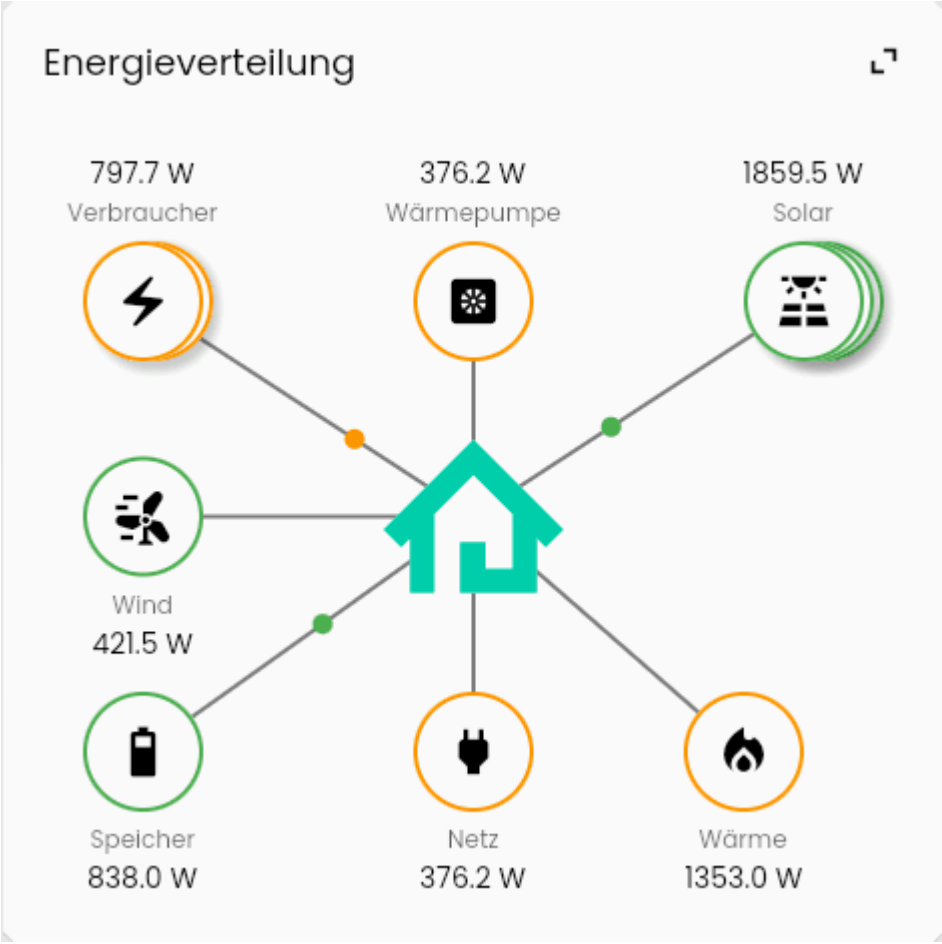
LOGIKPLÄNE

Jeder entwickelt seine Automationen anders. Der eine erfreut sich am unkomplizierten Konfigurieren und Analysieren von Ablaufplänen, der andere genießt die Flexibilität von PHP-Skripten. Insbesondere Elektriker denken wiederum in Schaltplänen. Und auch diese Art der Automation wird nun für alle als Logikplan nutzbar.
Die bisherigen IPSWorkflows von IPSView kommen als Logikpläne fest nach IP-Symcon und können uneingeschränkt verwendet werden und bieten so noch mehr Möglichkeiten die optimale Automation zu generieren. Die Dokumentation ist auch bereits migriert: Logikpläne



OPTIMIERTE DARSTELLUNGEN

Mit der Version 7.1 kommen auch einige neue Möglichkeiten für die Kachel Visualisierung. Nun kann die Darstellung von Kategorien individuell angepasst werden. Damit können beispielweise die Kategorien auf einer Startseite als Kachel mit Bild dargestellt werden, während sie auf Unterseiten als Kategorieleiste genutzt werden. In diesem Rahmen kommen auch die aus dem WebFront bekannten Inhaltswechsler zurück, welche eine neue mögliche Darstellung für Kategorien sind.
Auch die Energieverteilung bietet diverse neue Möglichkeiten. Insbesondere ist es nun möglich mehrere Einträge gruppiert werden, sodass fünf separate Verbraucher platzsparend gestapelt dargestellt werden können. Die individuellen Verbräuche sind weiterhin via Auswahl platzsparend auslesbar.



VPN (WIREGUARD) IN DER SYMBOX

Mit dem aktualisierten SymOS kann VPN über WireGuard/TailScale auf der SymBox eingerichtet werden. Damit kann auch aus der Ferne auf das Netzwerk hinter der SymBox zugegriffen werden. Dies ermöglicht eine unkomplizierte Konfiguration der entfernten Geräte, wodurch die Wartung deutlich vereinfacht wird.





Version: 1.58.2

Dienst: Läuft!

Tunnel: Verbunden!

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: Liste in der Connect Control Instanz, um für alle Visualisierungen den QR-Code anzeigen zu lassen oder diese direkt zu öffnen
Neu: Variablen unterhalb der Energieverteilung können deaktiviert werden
Neu: In der Kachel-Visualisierung Instanz gibt es einen Button, um diese die Kachel Visualisierung statt dem WebFront als Standard zu setzen. Das WebFront ist dann weiter über /legacy/ verfügbar.
Neu: Spezialschalter „ArchiveRecordLimitBool“ entfernt. Es gilt immer „ArchiveRecordLimit“.
Fix: Bei AC_FetchChartData* werden beim Erreichen der Limits die Fehler korrekt geworfen
Fix: Laden der php.ini robuster gemacht bei fehlerhaften Einträgen
Fix: $_IPS[‚CONFIGURATOR‘] wieder verfügbar, wenn auf dem WebFront eine Variablenaktion ausgeführt wurde
Fix: Doku Link für Kachel-Visualisierung korrigiert
Fix: Einige undokumentierte WFC_* Funktionen wie Visibility funktionierten nicht mehr korrekt seit einigen Versionen
Fix: Rate Limit für Google Assistant Status Reports erhöht
Fix: WebHook/OAuth Registrierung robuster gestaltet

GERÄTE

Neu: MQTT Konfiguratoren ermitteln den Datentyp so gut es geht automatisch für die Erstellung der Geräte-Instanzen
Neu: M-Bus: Neue MBUS_UpdateValuesEx Funktion zum Abfragen spezieller Daten
Neu: CMI: Für UVR610 wird der DL-Bus ebenfalls abgefragt
Neu: OPCUA: Konfigurator stark verbessert
Neu: OPCUA: RequestedMaxReferencesPerNode auf 100 gesetzt, da einige Geräte mit 0 (=Ohne Limit) nicht korrekt umgehen
Neu: LCN Shutter: Bei Relais Ansteuerung wird nur noch 1 Kommando statt 2 gesendet
Neu: Unterstützung für neues eHZ LAN Gateway (für Lastgangzähler nach IEC 62056), welches wir auf der Light+Building zeigen werden
Neu: ModBus: TimeoutTime Property auf Gateway Ebene, um die Wartezeit in Sekunden anzugeben (Aktuell nur über IPS_SetProperty verfügbar)
Fix: KNX: Sporadische Probleme bei KNX/IP (TCP/Secure) Verbindungen gelöst
Fix: EnOcean: Power/Energy Measurement für D2-01-0E korrigiert
Fix: ModBus: VALUES bei virtuellen Adressen enthalten jetzt die Ergebnisse der vorherigen virtuellen Lesevorgänge
Fix: KNX: Disconnect Request wurde nicht korrekt beantwortet, sondern eine Fehlermeldung im Meldungsfenster geloggt
Fix: Sync Remote: Fehlermeldungen beim Schalten über die Visualisierung gelöst
Fix: Sync Remote: Fehlermeldungen beim CSV Export gelöst
Fix: Sync Remote: Medien (Bilder) werden jetzt korrekt übertragen

KACHEL VISUALISIERUNG

Neu: Gruppieren in der Energieverteilung muss explizit aktiviert werden
Neu: Bessere Performance beim Rendern von Kachelansichten
Neu: Im Navigationsmenü kann zwischen den Visualisierungen des aktuellen Servers gewechselt werden

Neu: Darstellung des Wochenplans in einzelner Kachel optimiert
Neu: Visualisierungen eines Servers müssen nicht mehr explizit eingerichtet werden (App)
Neu: Name der Visualisierung wird im Tab-Titel angezeigt (Web)
Neu: Checkboxen in Chart Legende zusammengefasst
Neu: Minimale Laufzeit für Skript-Buttons
Neu: ‚Z2M.occupied_heating_setpoint' Profil funktioniert für Heizungskacheln Temperatur
Neu: IPSView nimmt die ganze Kachel ein (Benötigt noch Anpassungen von @Brownson)
Neu: ‚Lamella/Shutter.Reversed' wird unterstützt
Neu: Werte der Energieverteilung werden jetzt nach Typ gruppiert
Neu: Bunte Emojies
Neu: Energieverteilung bietet mehrere Variablen pro Gerät an und ermöglicht die Berechnung für Gruppen anzupassen
Neu: Neue Logik zur Verbindung zum Server
    Wechseln zwischen Verbindungen verbessert
    Neue Ansicht mit Log bei Neuverbindung
    Verständlichere Fehlermeldungen
Neu: Farbe des Speichers in der Energieverteilung angepasst (seit 7.1)
Neu: Höhe des Diagramm-Dialogs angepasst
Neu: CSV Export
Neu: Aktualisierungsrate der Begrüßung verbessert
Neu: Deep-Linking: QR-Codes können von anderen Apps gescannt werden
Neu: Fehlermeldung beim Schalten werden unten als rote Meldung dargestellt und nicht mehr gestapelt
Neu: Eine eventuelle Rückgabe von Automationen wird als Dialog angezeigt
Neu: Zeige mehr Informationen beim Versuch die Verbindung via App herzustellen an
Neu: Die Darstellung einzelner Kacheln kann aus dem Editier-Modus heraus verändert werden
Neu: Darstellungen für Kategorien:
    Kategorien können unabhängig voneinander in der Kategorieleiste oder als einzelne Kachel dargestellt werden
Neu: Inhaltswechsler
    Es kann ausgewählt werden, welche Objekte im Inhaltswechsler angezeigt werden sollen
    Elemente können nach einer festgelegten Zeit automatisch gewechselt werden
Neu: Für Listendarstellung von Instanzen: Nach Veränderung der aktiven Elemente werden neue Kinder automatisch angezeigt
Neu: Sollte versucht werden mehr HTML-Inhalte anzuzeigen als technisch Möglich, wird ein passender Fehler angezeigt.
Neu: Leistungsmodus der Energieverteilung wird gespeichert und wurde um statische Pfeile ergänzt
Neu: Aus dem Popup der Energieverteilung können Charts geöffnet werden
Neu: Wenn keine automatische Verbindung möglich ist, wird ein Button zum Neuverbinden angeboten
Neu: Energieverteilung
    Netz: Überschuss wird mit Verbrauch verrechnet
    Hat ein Gerät mehrere Variablen werden beide im Popup angezeigt
Fix: Chart-Buttons werden nicht mehr hinter Instanzen angezeigt
Fix: Anpassung der Skalierung des Logos in der Energieverteilung
Fix: Änderungen an Kategorien werden in der Kategorieleiste richtig angezeigt
Fix: Schieberegler stellen sich nun korrekt zurück, falls die Schaltoperation fehlschlägt
Fix: Namen einzelner Geräte werden in der Energieverteilung korrekt angezeigt
Fix: Schieberegler dürfen eine Schrittweite von 0 haben

Fix: Nicht-Geräte-Instanzen können verlinkt werden
Fix: Schnellwechsel nur für mehrere Visualisierungen verfügbar. Deaktivierte Visualisierungen sind deaktiviert
Fix: Autofill für Passwortdialog (Web-Only)
Fix: Fehler bei der Darstellung einer bei der Variablen gelöscht wurden
Fix: Einstellungen verlinkter Charts werden korrekt gespeichert
Fix: initial Anzeigen bei Charts wird berücksichtigt
Fix: Fehler wird angezeigt, wenn die aktuelle Kategorie gelöscht wurde
Fix: Minimaler Größenunterschiede der Kacheln behoben
Fix: Link zur Doku führt zur Doku und nicht in Forum
Fix: Fehlermeldung bei fehlenden Variablen im Chart
Fix: Uhrzeit in der Appbar aktualisiert sich sekündlich
Fix: Fehler bei automatischer Kachelauswahl
Fix: Präziser Farbwahl-Dialog funktioniert jetzt korrekt
Fix: Fehler in Verbindung mit dem ‚~Temperature.Difference' und ‚~Temperature.Fahrenheit' Profilen behoben
Fix: Slider ohne Schrittweite
Fix: Konfiguration Synchronisation auf mehreren Geräten
Fix: Charts mit widersprüchlichen Einstellungen in der Instanzkonfiguration
Fix: Fehler beim verschieben von Kacheln
Fix: Seitenübergangsanimationen reduziert (Web)
Fix: String-Eingabe zeigt die Korrekte Tastatur auf Samsung Geräten an (App)
Fix: Breitenabschätzung von Aufzählungen in Listen korrigiert
Fix: Auf älteren Geräten, die kein WebAssembly unterstützen, wird eine passende Fehlermeldung angezeigt, dass die
    Visualisierung auf diesen Geräten nicht unterstützt wird
Fix: Bei Diagrammen in den Apps mit Legenden in der Seitenleiste heben beim Hovern den korrekten Bereich hervor
Fix: Bei Visualisierungen mit langem Namen wird dieser im Navigationsmenü passend gekürzt
Fix: Höhe des Headers des Navigationsmenüs wird in den Apps korrekt dargestellt
Fix: Sehr lange Strings werden in der Darstellung gekürzt, können in TextBoxen aber komplett geladen werden
Fix: Stream Kachel zeigt Fehlermeldung, wenn Stream kein RTSP/RTSPS ist
Fix: Schieberegler in Dialogen aktualisiert springe nicht mehr zurück
Fix: Fehler behoben, der beim Löschen von Benachrichtigungen auftreten konnte
Fix: Schließe Verbindungen zum Server, wenn Visualisierung nicht geladen werden kann
Fix: Instanzen deren Darstellungstyp geändert wurde aktualisieren sich korrekt
Fix: Bei Verbindung zum lokalen Netzwerk und späterem Wechsel auf Connect wird das gespeicherte Kennwort verwendet
Fix: Korrektes Handling bei der initialen Erstellung eines Links in der Hauptkategorie
Fix: HTML-Boxen aktualisieren sich auch bei Variablenaktualisierung
Fix: Verhindere einige Verbindungsabbrüche
Fix: Inhaltswechsler
    Hintergrundfarbe wird von Variablen übernommen
    Titelgröße wird für jedes Objekt berechnet
Fix: Legenden werden nach Instanz-Einstellungen geladen
Fix: Fehler behoben, wenn Variablen in der Diagramm-Legende gefiltert und ausgewählt werden
Fix: Multigraphen werden korrekt in Inhaltswechslern angezeigt
Fix: Authentifizierung für HTML-Boxen in Apps hinzugefügt
Fix: Button zum neu verbinden funktioniert wie vorgesehen

Fix: Links auf die Hauptkategorie werden angezeigt
Fix: Neue Visualisierungen mit Passwörtern können wieder geöffnet werden
Fix: Neu hinzugefügte Server werden korrekt gespeichert
Fix: Füge Timeouts bei Verbindungsversuchen hinzu, damit diese nicht hängen bleiben
Fix: Verbesserte Performance bei Graphen mit sehr vielen Variablen
Fix: Stelle Zeiten im CSV-Dialog wie vorgesehen in der Zeitzone des Servers dar
Fix: Direkte Links auf passwortgeschützte Visualisierungen funktionieren wieder
Fix: String Variablen werden in den Zusatzinformationen der Energieverteilung korrekt angezeigt
Fix: Ungültige Variable in Begrüßungszeile wird korrekt abgefangen
Fix: Performance von Charts mit vielen Variablen verbessert
Fix: Darstellung der Zeit im CSV-Dialog korrigiert
Fix: Darstellung von Enumerationen in Listen verbessert

WEBFRONT

Fix: Assoziationen mit Platzhaltern und Aktion zeigen immer den korrekten Variablenwert an (konsistent zur neuen Visualisierung)
Fix: IPSView Views konnten nicht geöffnet werden, wenn /legacy/ verwendet wurde

KONSOLE

Neu: Zeige beim Speichern der Debug-Ausgaben in eine Datei den Ort dieser Datei an
Neu: Kopieren/Ausschneiden/Einfügen/Verlinken via Kontextmenü funktioniert nun analog zu Strg + C/X/V/L
Neu: Dokumentation „Button“ in der Konsole öffnet die Dokumentation automatisch in der korrekten Sprache
Neu: Zeige Logikpläne als Automation immer an und installiere bei Bedarf die benötigten Module aus dem Module Store
Neu: Konfiguration der Spalten bei PHP-Information wird gespeichert
Fix: Zeige korrekten Titel im Dialog beim erstellen von Kategorien via Alt + 0
Fix: Seltenen Fehler korrigiert, der beim Einklappen von unsichtbaren Knoten auftreten konnte
Fix: Beim Abdocken von IPSView kommt immer der Hinweis, wenn es ungespeicherte Änderungen gibt
Fix: Neu hinzugefügte Visualisierungen werden im Visualisierung-Widget korrekt hinzugefügt
Fix: String-Variablen können nun aus der Konsole reaggregiert werden (auch wenn dies lediglich die Rahmendaten wie Start- und Endzeit neu berechnet)
Fix: Im Module Store konnte bei Modulen aus der Vorschau fälschlicherweise ein Update angezeigt werden wenn auf dem eigenen System eine Beta- oder Testversion installiert ist, dies wurde behoben
Fix: Bei Sortierung nach Namen in der Timer Information wird nun primär nach Namen und nicht mehr nach Pfad sortiert
Fix: IPSView Updates können nur installiert werden, wenn eine gültige IPSView Subskription vorhanden ist
Fix: Verdichtung für String Variablen korrekterweise deaktiviert (Nur für Bool/Int/Float verfügbar)
Fix: Behebe Typo beim sicheren Löschen

Fix: Korrigiere Meldung, wenn ein Logikplan erstellt wird, das benötigte Modul aber noch nicht installiert ist
Fix: Wird die Prüfung beim sicheren Löschen abgebrochen wird wie vorgesehen der gesamte Löschvorgang abgebrochen
Fix: Korrigiere Validierung, ob das Modul zu einer Instanz existiert
Fix: Verhindere Fehler beim Bearbeiten von mehreren Ereignissen mit unterschiedlichen Aktionen
Fix: Springe wie vorgesehen zur letzten Position bei Objektauswahlen, wenn das aktuell ausgewählte Objekt nicht (mehr) existiert
Fix: Cache LocationControl ID im Location Widget (Verbessert die Performance massiv auf Systemen mit sehr viel Variablenänderungen)

PRO KONSOLE

Neu: Bibliotheken auf die neuste Version aktualisiert (Es wird mind. Windows 8 und Windows Server 2012 benötigt)
Neu: Beim Hinzufügen von Servern kann das System mit angegeben werden
Neu: SymOS Connect für KNX für SymBox'en direkt integriert
Neu: SymOS Connect für LCN für SymBox'en direkt integriert

SYMOS

Neu: WireGuard im Kernel aktiviert (wg utility ist auch verfügbar)
Neu: Kernel auf 6.1.61 aktualisiert
Neu: SymOS Connect für KNX ist offiziell verfügbar
Neu: SymOS Connect für LCN ist offiziell verfügbar
Neu: VPN über Tailscale Modul verfügbar
Fix: In manchen Situationen wurde der verfügbare Speicherplatz nicht korrekt ermittelt

PHP-SDK

Neu: HTML-SDK für die Kachel Visualisierung
Neu: Copy & Paste innerhalb von List und Tree via Strg + C/Strg + V möglich
Neu: Configurator unterstützt nun align und sortColumn analog zu List/Tree
Neu: link/download darf auch in Listen auf Column Ebene gesetzt werden
Neu: SelectCondition unterstützt die Eigenschaft „caption"
Neu: SelectAction verbessert mit Auswahl zwischen generellen Aktionen und Aktionen mit Ziel
Neu: Analog zu SelectCondition unterstützt nun auch SelectAction das Feld „caption"
Fix: description bei QrCode funktioniert jetzt wie vorgesehen
Fix: onClick wird nun wirklich nur bei angezeigtem leeren Text und nicht bei einem „falsey" deaktiviert
Fix: Bei on-Skripten und einer Liste ohne ausgewählte Zeile und add-Werte wird die aktuelle Zeile nun wie vorgesehen mit null initialisiert
Fix: Verhindere Fehler, wenn in einem Konfigurator die Sortierung deaktiviert wird
Fix: SelectAction verwendet wieder wie vorgesehen die gesamte verfügbare Breite

NEUE MODULE

EKEY BIONYX

Integration von ekey bionyx Geräten direkt in IP-Symcon
IP-Symcon empfängt die Aktivitäten des Fingerscanners über Webhooks

SNOM

Integration von Snom Geräten direkt in IP-Symcon

AKTUALISIERTE MODULE

AMAZON ALEXA

Fix: Korrektes Handling mit PHP 8.2

ANWESENHEITS-SIMULATION

Fix: Anpassungen für IP-Symcon 7.0

BENACHRICHTIGUNG

Fix: Einige kleinere Fehler behoben

ENERGIEVERBRAUCH OPTIMIERER

Neu: Unterstützung für E-Autos

GARDENA SMART SYSTEM

Fix: Intervall für die Gerätesuche erhöht

GOOGLE ASSISTANT

Fix: Korrektes Handling von PHP 8.2

SPOTIFY

Neu: Profile angepasst, damit sie mit dem Mediaplayer der neuen Visualisierung kompatibel sind

TEXT TO SPEECH (AWS POLLY)

Fix: Unterstützung für IP-Symcon 7.x

V6.4->V7.0 (Q4/2023)

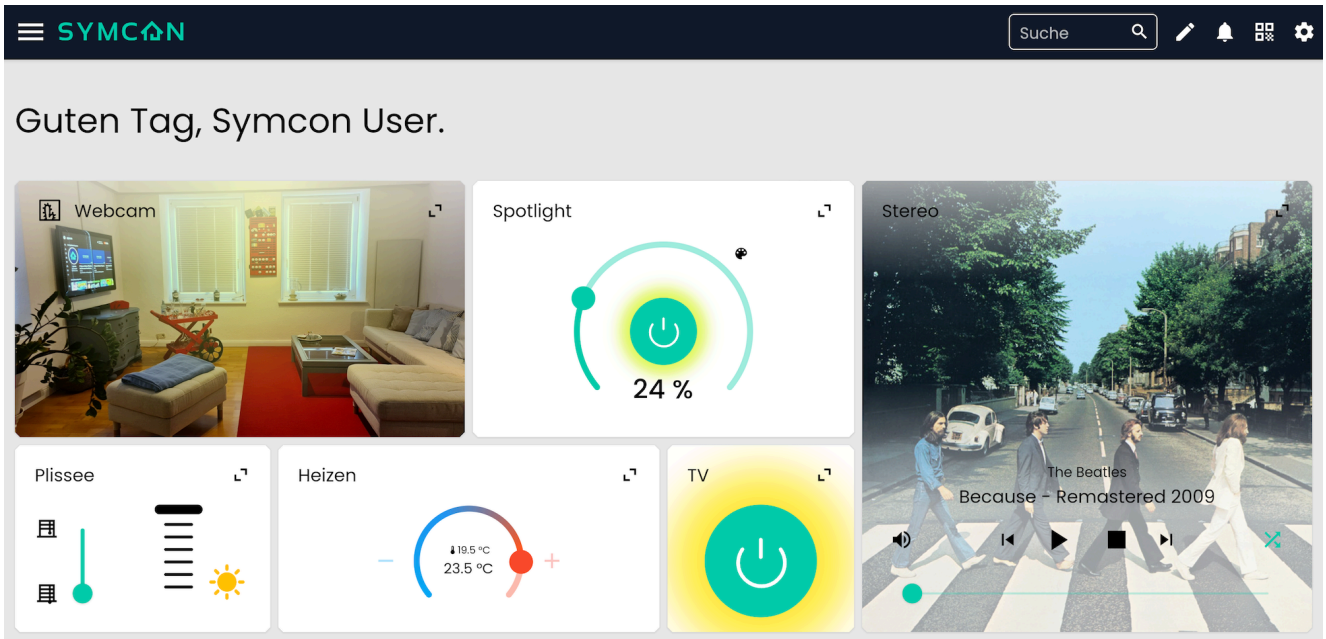
Wesentliche Änderungen
Migration
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

Neue Visualisierung
Integriertes Energie-Management
SymBox Pro mit KNX Data Secure
Stark verbesserte ModBus Anbindung

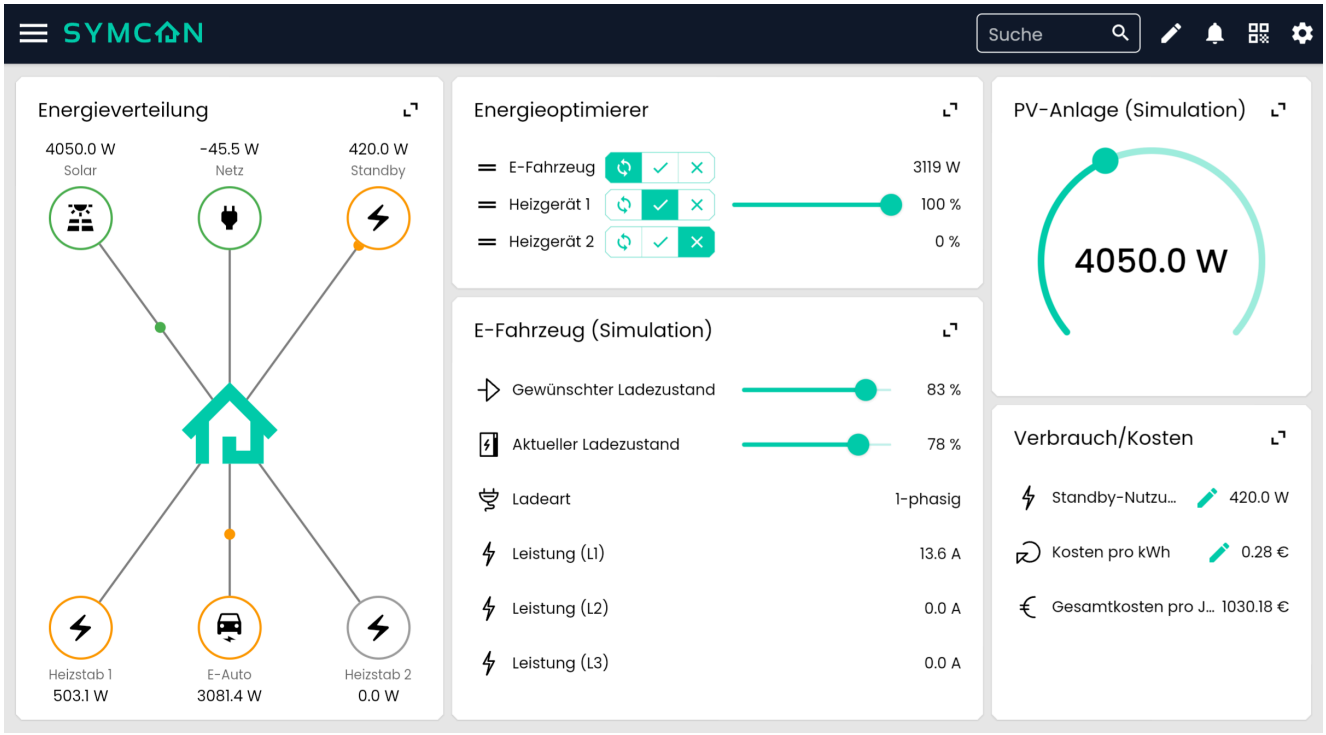
NEUE VISUALISIERUNG

Die neue, moderne Visualisierung im Kachel-Design ist ab sofort verfügbar. Mit hellem und dunklem Design, Drag&Drop Editor zum Positionieren der Kacheln und vielen spezialisierten Kacheln für die relevantesten Anwendungszwecke. Verfügbar für Web, iOS und Android bietet es eine plattformübergreifende, konsistente Darstellung. Eine Demo ist unter https://demo.symcon.de verfügbar!
Um die neue Visualisierung nutzen zu können, muss einfach eine neue Instanz vom Typ "Kachel Visualisierung" erstellt werden.
Download für iOS: https://apps.apple.com/de/app/symcon-visualization/id1566124129
Download für Android: https://play.google.com/store/apps/details?id=de.symcon.visualization



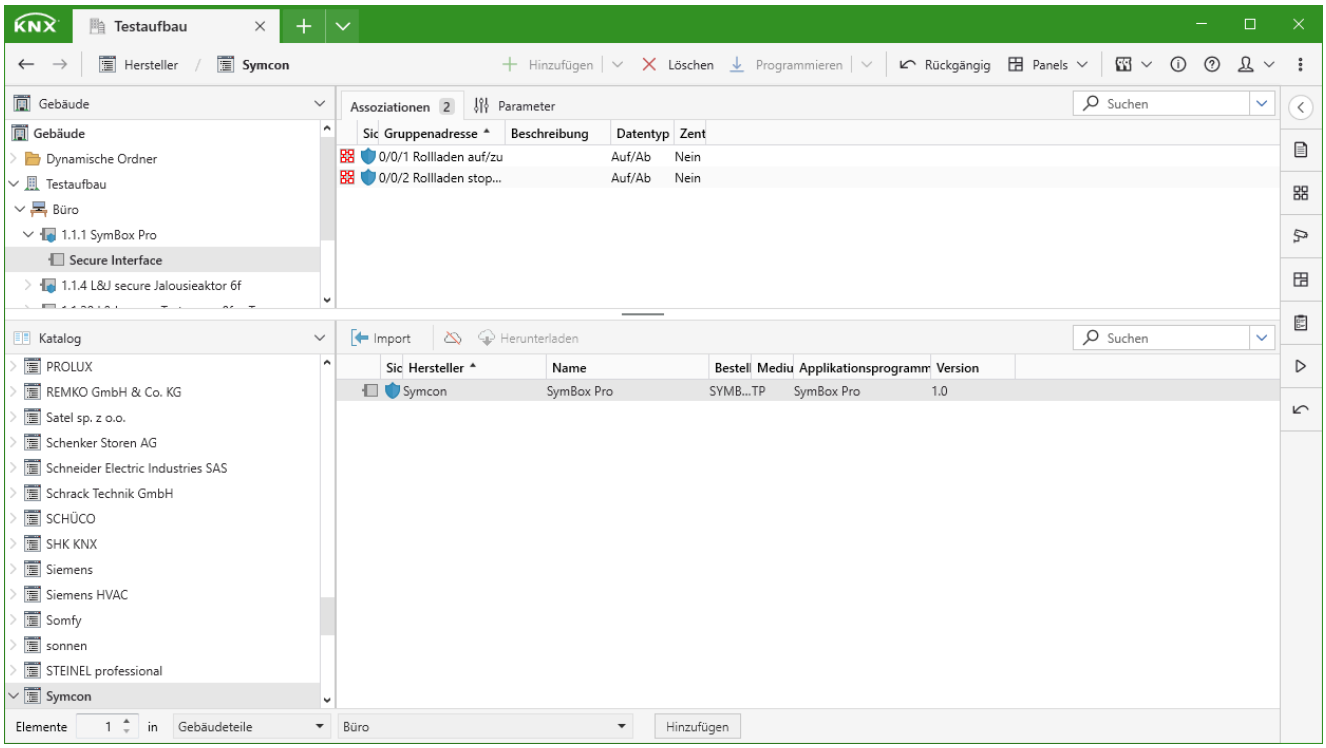
INTEGRIERTES ENERGIE-MANAGEMENT

PV-Überschuss Optimierung mit wenig Aufwand! Der neue Energieverbrauch-Optimierer und die Energieverteilung in der neuen Visualisierung helfen die vorhandene Energie optimal zu verbrauchen. Die neue Darstellung gibt dabei immer den Überblick über alle Werte. Vorlagen für ModBus machen die mehrfache Einrichtung zum Kinderspiel.



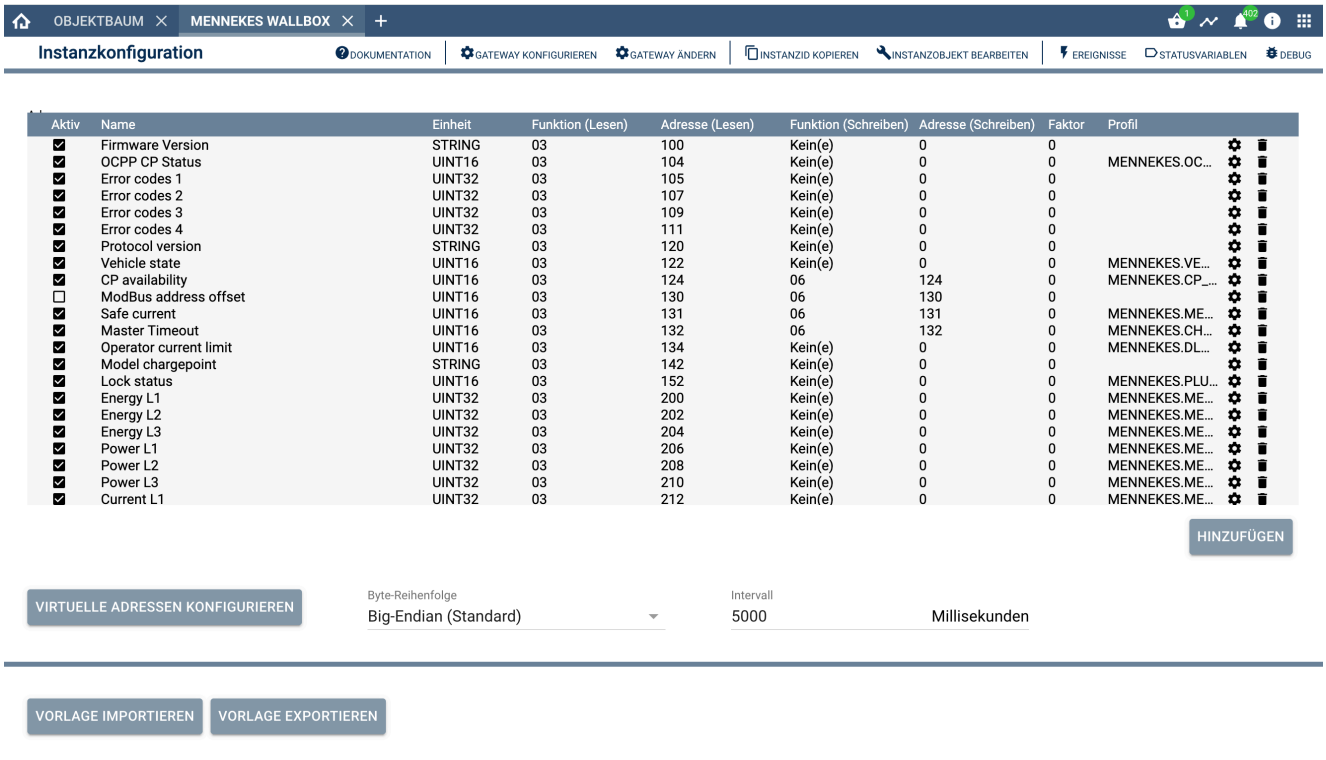
SYMBOX PRO MIT KNX DATA SECURE

Die SymBox Pro ist ab sofort offiziell von der KNX Association zertifiziert und ist im Produktkatalog der ETS auffindbar. Dadurch können Gruppenadressen direkt auf den Tunnel der SymBox Pro gezogen werden, womit sich Filtertabellen und die KNX Data Secure Verknüpfungen vollständig automatisch konfigurieren. Ein Video zur Einrichtung ist auf unserem YouTube Kanal zu finden: https://www.youtube.com/watch?v=aA8mcKEyHOY&t=1s



STARK VERBESSERTE MODBUS ANBINDUNG

Die ModBus Anbindung wurde überarbeitet, sodass direkt alle Adressen eines Geräts in einer Liste eingetragen werden können inklusive aller Namen, Parameter und Übersetzungen der Namen. Zusätzlich kann die Gerätekonfiguration jederzeit Exportiert und Importiert werden, wodurch die Einrichtung mehrerer Geräte ein Kinderspiel wird und die Vorlagen auch in unserer Community geteilt werden können. Wir haben im ersten Schritt Vorlagen für die Wallboxen/Ladesäulen von Mennekes, ABL, KEBA und ALFEN erstellt, welche in der Dokumentation heruntergeladen werden können. Die bereits geteilten Vorlagen aus der Community sind hier zu finden: https://community.symcon.de/c/86



MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

PHP wurde von 7.4 auf 8.2 aktualisiert. Insbesondere Nutzer der IPSLibrary/IPSLogger sollten VOR dem Update auf IP-Symcon 7.0 diese aktualisieren. Notfalls können die notwendigen Änderungen manuell nachgepflegt werden (Details). Die meisten PHP-Module wurden bereits aktualisiert, sodass diese bereits mit IP-Symcon 7.0 kompatibel sind.
Unter Windows wurden einige selten verwendete PHP-Extensions entfernt, sodass aktuell nur noch IMAP, OCI, PGSQL und SNMP optional installiert werden können. Die meisten anderen PHP-Extensions sind bereits standardmäßig auf allen Plattformen inkludiert (Details).
Die Pfade zum /user/ und /skins/ Ordner werden beim Upgrade verändert, sodass diese auf der Hauptebene und nicht mehr im /webfront/ liegen. Ggf. müssen PHP-Skripte angepasst werden, die auf diese Ordner hart kodiert zugreifen.
Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: Neue Visualisierung verfügbar (Siehe: https://community.symcon.de/t/133822)
Neu: Neue Streams-Integration in der neuen Visualisierung inkl. 4K Support (Siehe: https://community.symcon.de/t/134356)
Neu: PHP 8.2
Neu: Einige selten genutzte PHP-Erweiterungen unter Windows entfernt
Neu: Einige Profile für Rollladen/Lamellen verbessert, sodass direkt die neue Kachel in der Visu verwendet wird
Neu: Spezialschalter "DefaultVisualization", mit der man einstellen kann, welche Visualisierung standardmäßig geöffnet wird (0 = WebFront, 1 = Tile)
Neu: Spezialschalter "CompatibilityDeprecations", um auch die Deprecations wie utf8_* von PHP 8.2 zu aktivieren
Neu: Spezialschalter "DisableLogfile", um das Logfile fast komplett zu deaktivieren
Neu: Alle Bibliotheken aktualisiert. (z.B. auch OpenSSL auf 3.0)
Neu: Fehlende Icons für UnixTimestamp*, HexColor, TWColor ergänzt
Neu: Konstante SCRIPTTYPE_IPSWORKFLOW hinzugefügt
Neu: Konstante MEDIATYPE_IPSVIEW hinzugefügt (ersetzt MEDIATYPE_FORM)
Neu: Bei (z.B. UTF-8) Fehlern im Datenfluss wird die Meldung nur einmal für alle verbundenen Instanzen geloggt und zusätzlich mit mehr Informationen ausgegeben
Neu: Das Neuverbinden von fehlerhaften Instanzen verbessert und konfigurierbar gemacht
Neu: Weitere Modulnamen werden jetzt übersetzt (z.B. Energieverteilung)

Neu: ModBus ASCII wird unterstützt (für z.B. ABL eMH1)
Neu: KNX: DPT 14.080
Neu: KNX: DPT 21.1002, 21.1200, 21.1201
Neu: KNX: DPT 205.103
Neu: KNX: DPT 275.100
Neu: KNX: Passwort für Secure Schlüsselbund wird überprüft (und ggf. fehlerhaft gemeldet)
Neu: DMX: Option zum Dauerhaften senden von Paketen hinzugefügt
Neu: Bibliothek für VideoStreams+VoIP aktualisiert (Löst hoffentlich die Abstürze bei den Streams)
Neu: Farbtemperatur und Temperatur Regler werden auch in Listen eingefärbt
Neu: Bei Auswahl der Demo kann die Visualisierung gewählt werden
Neu: Unterstütze Übersetzung der Visualisierungsnamen (Enterprise)
Neu: Verbesserter Precheck für veraltete IPSLibrary/IPSLogger in der autoload.php
Neu: ModBus: Neuer STRING (HEX) Datentyp, um Binärdaten auszulesen
Neu: ModBus: Virtuelle Adressen werden nach den normalen Adressen positioniert
Neu: ModBus: „null“ Werte bei virtuellen Adressen werden als fehlender Wert interpretiert
Neu: ModBus: Beim Schreiben in virtuellen Adressen ist nun $VALUES verfügbar mit den aktuellen Werten
Fix: ModBus: Manuell erstellte Variablen unterhalb der Geräte-Instanz werfen keine Fehler mehr
Fix: ModBus: Leere „Leseaktion“ bei virtuellen Adressen funktioniert wie erwartet
Fix: Workaround für defekte php.ini nach Upgrade unter Windows
Fix: Sofern NATSupport in den Spezialschaltern aktiv ist, wird der Spezialschalter NATPublicIP den mobilen Apps ebenfalls mitgeteilt
Fix: Der Demoserver wird nicht als Server abgespeichert
Fix: Baumauswahl im Drawer wird erst bei Bedarf geladen und unterstützt so deutlich umfangreichere Bäume
Fix: Aktualisiere Änderungen an Kategorienamen innerhalb der Kategorieleiste
Fix: Zeige Legende in Graphen an der Seite korrekt an, falls Bedienungselemente ebenfalls an der Seite sind
Fix: Datumsauswahldialoge über HTML Boxen funktionieren wie vorgesehen
Fix: Problem mit HTMLBox über HTMLBox gelöst
Fix: ModBus: Überspringe Inf/NaN Werte, damit Vorlagen bei diesem nicht kritischen Fehler weiter abfragen. (Der Wert wird nicht in die Variable geschrieben und ein Hinweis ins Debug Log notiert)
Fix: BACnet: Dekodieren von UCS-2 Zeichenketten korrigiert (Dadurch konnte der Einlesevorgang der Datenpunkte abbrechen/stecken bleiben)
Fix: Geräte für Benachrichtigungen können in den Visualisierung Instanzen bearbeitet werden
Fix: KNX: SymBox mit KNX wird in der Discovery Instanz korrekt angezeigt
Fix: Precheck für autoload.php nach hinten verlegt, sodass alle Kernel Module verfügbar sind
Fix: UVR610 unterstützt ebenfalls M-Bus abfragen
Fix: Absturz korrigiert, wenn eine Ereignis Bedingung gelöscht wurde, die nicht existierte
Fix: "cert" Ordner wird nicht mehr erstellt, der seit langem nicht mehr genutzt wurde
Fix: Löschen von Skripten/Medien bewegt diese wieder in den ‚deleted‘ Order (seit 6.3)
Fix: __generated.inc.php wird jetzt vorab generiert und dann getauscht und löst dadurch Fehlermeldungen beim ModuleReload
Fix: Intensity.1 Profile benötigte keine Nachkommastellen
Fix: Security-Context für Visu’s wurde zu restriktiv berechnet, sodass Objekte unterhalb von Variablen, die mit Ereignissen verknüpft waren, manchmal nicht angezeigt wurden
Fix: Manuelles Auslesen von Lizenz Informationen im Util Control wieder möglich

Fix: ModBus Geräte konnte nicht mehr konfiguriert werden, da dynamische Formulare nicht geladen wurden (seit letztem Update)
Fix: CORS Fehler korrigiert beim Laden von Schriftarten in HTMLBoxen

GERÄTE

Neu: ModBus: Stark verbesserte Anbindung mit Im/-Export Funktion für Vorlagen
Neu: ModBus: Byte Order wurde im Splitter entfernt und ist jetzt nur noch in den Geräte/Coil/Adress Instanzen verhanden. Zusätzlich kann in der Geräte Instanz die Byte-Order pro Adresse überschrieben werden
Neu: ModBus: Byte Order Optionen „Little Endian“ und „Big Endian (Byte Swap)“ hinzugefügt
Neu: EnOcean: Zusätzliche Werte für STM550 (D2-14-41) verfügbar
Neu: JSON Dekoder/MQTT Device: Option zum Sperren der Variablen, sodass keine Variablen erstellt/gelöscht werden bei falschen Datenpaketen
Neu: KNX: Setzen der physikalischen Adresse/programmieren der Applikation verbessert (Nur SymBox)
Neu: LCN: Sperren für Regler können in der Visu ausgewertet/gesetzt werden
Neu: SMTP Unterstützt senden von HTML, wenn und im Inhalt gefunden wird
Neu: WebSocket Client und WebHooks unterstützen den binären Datentransfer
Fix: SSE Client (z.B. für Philips Hue V2) erkennt Verbindungsabbrüche besser
Fix: Hinweis im HTTP Client bzgl. Kompatibilität korrigiert
Fix: M-Bus: Bei Dekodierfehlern einzelner Felder, wird nicht die gesamte Verarbeitung abgebrochen
Fix: RegVar_SendText in Kombination mit ServerSocket korrigiert
Fix: Z-Wave: Wenn ein Gateway überlastet ist und Pakete „gedroppt“ hat, breche nach „Retry“ Fehlversuchen ab, anstatt es weiter zu versuchen
Fix: MQTT: Topics werden korrekt auf UTF-8 Konformität geprüft und mit besserer Fehlermeldung bemängelt
Fix: DigitalStrom: Einige Binärtypen ergänzt
Fix: ServerSocket: Timeout beim TLS Handshake implementiert, sodass dieser nicht mehr feststecken kann bei instabilen Verbindungen
Fix: VoIP: Bei vielen ausgehen Anrufen konnte die ConnectionIDs durcheinander geraten
Fix: EnOcean: Verbessertes Profil für Backup Battery (D2-14-41)
Fix: KNX: Absturz korrigiert, sofern fehlerhafte Pakete ohne Daten empfangen wurden
Fix: KNX: Erkannte Gruppenadressen für Statusrückmeldungen werden nicht doppelt hinzugefügt

WEBFRONT

Neu: DWD kann nun auch Vertikal dargestellt werden (Option verfügbar im Editor)
Neu: Wenn standardmäßig die neue Visualisierung aktiv ist, kann das WebFront über /legacy/ erreicht werden
Neu: /webfront/user Ordner wird nach user migriert
Neu: /webfront/skins Ordner wird nach skins migriert
Fix: Rote Fehlermeldung, welche beim Start kurz aufflackern konnte, wurde korrigiert

KONSOLE

Neu: Visualisierungs-Widget unterstützt nun auch die neue Visualisierung

Neu: IPSWorkflow wird nach der Konvertierung direkt geöffnet
Neu: Dialoge können nun in der webbasierten Konsole vergrößert werden (bisher nur in der Pro Konsole möglich)
Neu: Übersetzungen für Modulnamen/-aliasse werden geladen (noch nicht ganz vollständig für interne Module)
Neu: Moduldaten werden lokal gehändelt, wodurch beispielsweise Konfiguratoren deutlich schneller arbeiten
Fix: Erkennung von korrekten E-Mail-Adressen für Benutzernamen im Infodialog korrigiert
Fix: Verhindere eine falsche Ziel-ID beim mehrfachen Öffnen eines Ereignisses
Fix: Zeige bei Ereignissen beim Öffnen eine Meldung an, falls die aktuelle Aktion nicht zum Zielobjekt passt
Fix: Update Widget funktioniert nun auch korrekt für Docker armhf/arm64
Fix: Verhindere Fehler im Auswahlbaum, wenn ein Schnellfilter gesetzt ist und neue Objekte hinzugefügt werden
Fix: Vorschau von Bildern funktioniert nun auch bei Bildern mit Großbuchstaben in der Dateiendung
Fix: SelectCondition wird nun insbesondere in Listen korrekt dargestellt, wenn die betroffene Variable nicht mehr existiert
Fix: Wird man vom Testing-Kanal wieder ausgeladen, so verhält sich der Module Store so als wenn der Kanal geschlossen wäre
Fix: Duplizieren oder Verschieben mit aktivem Schnellfilter funktioniert nun wie vorgesehen
Fix: Keine Fehlermeldungen mehr beim Verschieben des Browserfensters
Fix: Sortierung auf Spalten wie Editieren/Löschen ist nicht mehr möglich
Fix: Besserer Fehler, falls Skripte durch Popup-Blocker o.ä. geblockt werden
Fix: Vorschau von GIFs funktioniert nun
Fix: Bei fehlerhaft geladenen PHP-Modulen wird jetzt eine bessere Fehlermeldung angezeigt
Fix: Bei fehlhaften dynamischen Formularen in Listen von PHP-Module wird eine bessere Fehlermeldung angezeigt
Fix: Beim Öffnen der Konsole werden gemerkte Tabs, die fehlerhaft sind, automatisch geschlossen anstatt irreführende Fehler anzuzeigen
Fix: Tooltip im Skript Editor funktioniert wieder korrekt
Fix: Copy&Paste im Skript Editor funktioniert korrekt, wenn in der Pro Konsole mehrere offen sind

SYMOS

Neu: IP-Adresse kann ohne vollständiges Recovery geändert werden (Dokumentation, Kapitel 5.7)
Fix: fnmatch Funktion fehlte auf der SymBox
Fix: Beim Neustart von IP-Symcon konnte fälschlicherweise eine Fehlermeldung auftreten, dass IP-Symcon bereits gestartet sei, obwohl dies nicht der Fall war
Fix: WebFronts zu Visualisierungen umgenannt
Fix: Resize der Partition/Dateisystem konnte bei SymBoxen, die in den letzten Wochen verkauft wurden, unvollständig sein. Das SymOS erkennt dieses Problem und fordert zur Korrektur auf. (Die SymBox'en, welche standardmäßig mit einer neueren SymOS Version ausgeliefert werden, kann man gut an dem neuen Sticker mit der Seriennummer/MAC-Adresse auf der Rückseite erkennen)
Fix: Sofern die Integrator Weiterleitung aktiv war, konnte man sich nicht korrekt über SymOS Connect einloggen
Fix: SymOS konnte Abstürzen, wenn Subskription über das Jahr 2038 hinaus ging

PHP-SDK

Neu: Preview für IPSModuleStrict verfügbar
Neu: Parameter "form" für List/Tree ermöglicht individuelle Bearbeiten-Dialoge (Dokumentation)
Neu: Neues Element QrCode um QR-Codes darzustellen (Dokumentation)
Neu: Ist bei einer List/Tree keine Zeile ausgewählt, werden die add-Werte übergeben
Neu: IPSModuleStrict bietet RegisterHook und RegisterOAuth (damit entfallen die alten Workarounds mit WebHookModule und WebOAuthModule und Einträge von Modulen werden damit automatisch aufgeräumt)
Neu: Migrate Funktion zum Migrieren der Persistenz (=Attributen/Properties) (Dokumentation)
Neu: Bei Configurator ist eine Kategorie wählbar unter welcher die Objekte erstellt werden sollen
Neu: Parameter „download“ für Elemente mit on-Callbacks um Downloads zu ermöglichen
Neu: Modulname und Aliasse werden anhand der locale.json übersetzt
Fix: visible und enabled funktioniert für SelectCondition nun wie vorgesehen
Fix: Bei dynamischen Formularen in Listen werden „add“ Werte beim Hinzufügen übernommen auch wenn kein dazugehöriges „edit“ Feld vorhanden ist.

ENTERPRISE

Fix: Sync: Version vom Remote darf älter sein. D.h. der Master in 7.0 kann Remotes in Version 6.4 synchronisieren
Fix: VHost Enterprise Feature funktionierte ggf. nicht korrekt für die iOS App

NEUE MODULE

OCPP

Integration von OCPP Geräten direkt in IP-Symcon
Anzeige des Verbrauchs einzelner Ladepunkte
Anzeige, ob gerade eine Transaktion läuft bei Ladepunkten

AKTUALISIERTE MODULE

ANWESENHEITS-SIMULATION

Neu: Verwenden von SelectVariable und Hinzufügen einer Status-Spalte

BETRIEBSSTUNDENZÄHLER

Neu: Preis für Kosten kann aus einer Variable gelesen werden

GARDENA SMART SYSTEM

Fix: Initiale Verbindung korrigiert
Fix: Überschreiben der Mähdauer korrigiert

HOME CONNECT

Fix: Endlosschleife bei Wiederverbindung

JSON EXPORT

Neu: CORS Header eingefügt um Integration in Web Anwendungen zu erlauben

SZENEN-STEUERUNG

Fix: Szene wird korrekt aufgerufen, auch wenn ein Gerät innerhalb der Szene fehlen sollte

TELEFONANSAGE

Fix: Fehler beim Starten korrigiert

VERBRAUCHSVERHALTEN

Fix: Bessere Vorhersage (Betrachtet nur positive Werte)

WEBGRAPH

Neu: Benutzeroberfläche optimiert
Fix: Referenzsuche, Kontinuierliche monatliche Darstellung im März korrigiert

V6.3->V6.4 (Q2/2023)

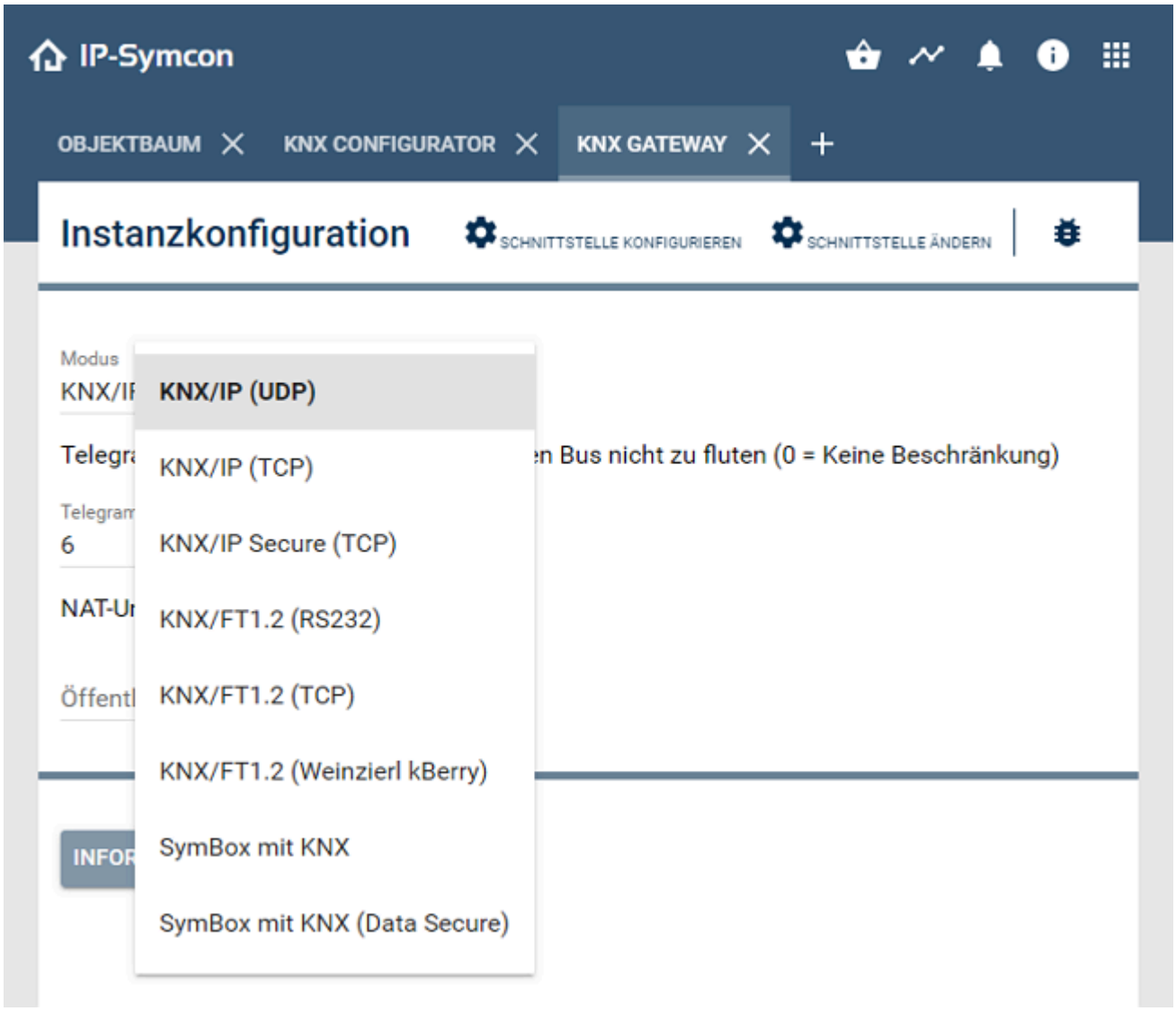
Wesentliche Änderungen
Migration
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

KNX Secure
Übersichtlichere Diagramme mit mehreren Variablen
Automatische Backups via FTP
OPC UA und mehr für Enterprise

KNX SECURE

Die Anbindung zu KNX wurde um KNX IP Secure und KNX Data Secure Unterstützung erweitert. So wird die Kommunikation von Visualisierung bis zu den Endgeräten auf IP und TP Ebene verschlüsselt und vor unberechtigtem Zugriff geschützt.



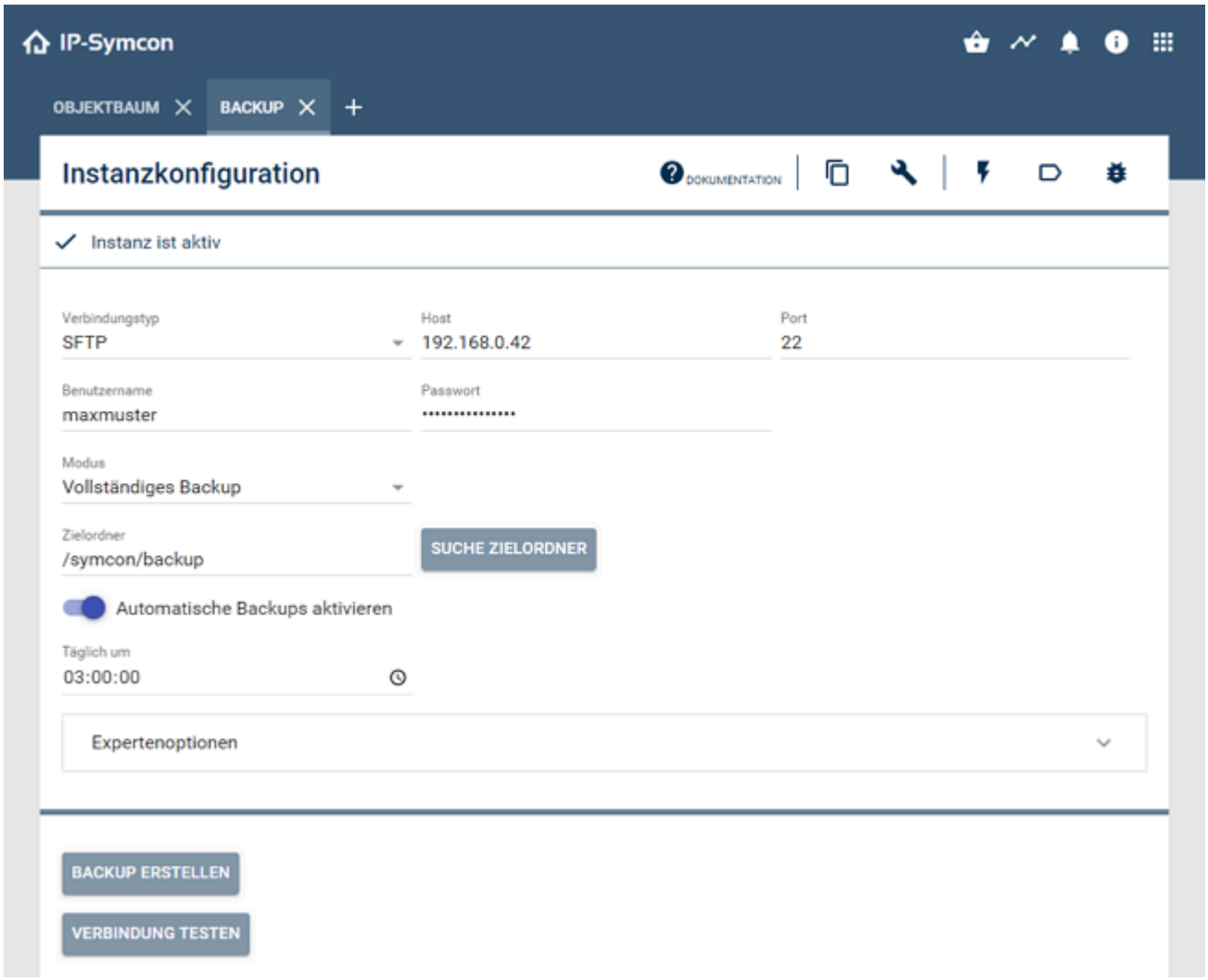
ÜBERSICHTLICHERE DIAGRAMME MIT MEHREREN VARIABLEN

Die Graphen wurden insbesondere für Graphen mit sehr vielen (z.B. hunderten) Variablen hin optimiert. Werden Variablen ausgeblendet, passen sich die dargestellten Achsen automatisch an die ausgewählten Variablen an und bieten so deutlich mehr Platz für die Verläufe. Darüber hinaus wird die Legende bei vielen Variablen nun als Extraspalte angzeigt und bietet dort diverse hilfreiche Features wie einen Filter oder Buttons zum Ein- oder Ausblenden aller Variablen, wodurch die optimale Auswahl an Variablen unkompliziert und effizient ausgewählt werden kann.



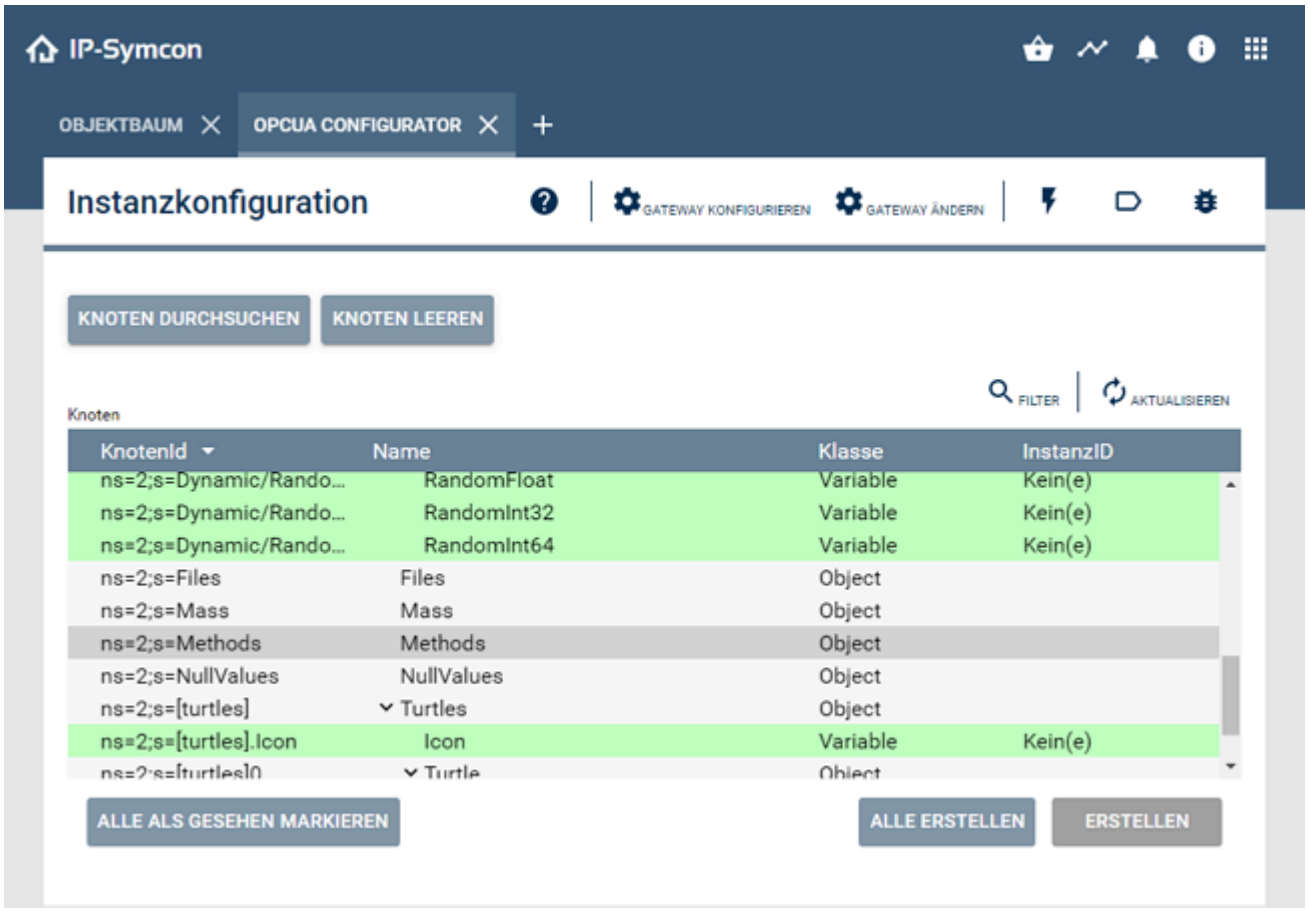
AUTOMATISCHE BACKUPS VIA FTP

Keine Lust mehr auf manuelle Backups? Dank dem neuen Backup-Modul für FTP/FTPS/SFTP muss gar nicht mehr daran gedacht werden. Das Modul erstellt nach Wunsch vollständige oder inkrementelle Backups und speichert diese typischerweise auf einem NAS-, Cloud- oder externen Speicher, damit es im Falle eines Falles wieder eingespielt werden kann.



OPC UA UND MEHR FÜR ENTERPRISE

Auch für unsere Enterprise-Kunden gibt es einige spannende neue Features. Nun können Sie OPC UA Geräte einbinden. Oder verwenden Sie die Sync-Module, welche das Zusammenführen von vielen IP-Symcon Systemen in ein Master System unterstützen.



Die Enterprise-Module sind kostenpflichtige Erweiterungen, welche zu jeder bestehenden IP-Symcon Lizenz erworben werden kann. Für Details steht unser Vertrieb zur Verfügung.

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: Referenzsuche berücksichtigt IPSView's

Neu: Spezialschalter zum Überschreiben der genutzten Sprache (Locale)
Neu: Funktion zum Migrieren von alten IPSWorkflows
Neu: VoIP_PlayWave validiert die WAV Datei, ob diese Unterstützt wird
Neu: Neues Profil ~Lamella
Fix: VoIP_PlayWave funktioniert wieder korrekt (inkl. PlayFinished Event)
Fix: StartTime wurde auf der SymBox falsch gesetzt, wenn der NTP Server zu langsam ist
Fix: Beim Beenden vom IP-Symcon Dienst warten wir maximal 5 Minuten auf das Beenden von PHP Threads
Fix: Löst das DNS-SD Fehler 87 Problem
Fix: SSDP: Besser Erkennung von Headern mit falscher Groß-/Kleinschreibung
Fix: Bessere Sanity Check für Datumsfuntionen
Fix: Installer überprüft, ob QWave verfügbar ist (Betrifft nur Windows Server 2008/2012)

AKTIONEN

Neu: Aktionen für Addition/Substraktion bei getriggerten Variablen

GERÄTE

Neu: KNX IP Secure / KNX Data Secure Unterstützung
Neu: MQTT: Möglichkeit ein abweichendes Topic für Aktionen zu definieren (z.B. mqtt/device/1234/set)
Neu: xComfort Jalousieaktor (Neue Generation mit mehr Feedback auf Kanal 0)
Neu: EnOcean D2-14-41
Neu: OPC UA Client (nur Enterprise)
Fix: KNX: In Profilen wurde „pulses“ nicht korrekt nach „Impulse“ übersetzt
Fix: ModBus: DeviceID ist im Formular auf 255 limitiert
Fix: SetPoint Berechnung beim Eltako FTR65DS, FTR65HS, FUTH55D, FUTH65D korrigiert

WEBFRONT

Neu: Diagramme laden nur noch die wirklich ausgewählten Charts (Damit können z.B. auch 500 Charts in einem Diagramm drin sein und es werden nur die ausgewählten geladen)
Neu: Diagramme merken sich im WebFront, welche Auswahl in der Legende getroffen wurde
Neu: Legenden von Diagrammen enthalten auch Boolean Graphen und können somit ausgeblendet werden
Neu: Listenbasierte Legende enthält Buttons für Alle ausblenden/Alle einblenden
Neu: Umschalten zwischen Schwebender/Listenbasierter Legende kann über die WebFront Visualisierung Instanz vorgenommen werden im Bereich Darstellung
Fix: WebFront startet wieder automatisch korrekt bei Neustarts
Fix: Bessere Erkennung von einem Verbindungsabbruch (insbesondere auf Safari Mobile)

KONSOLE

Neu: Bedingungen in Listen werden passend formatiert
Neu: Ändern sich die Werte in listenartigen Elementen einer Instanzkonfiguration bleibt die aktuelle Scrollposition erhalten
Neu: Speichere im Statuswidget welche Informationen dargestellt werden
Neu: Variablen mit nicht mehr existierenden Variablenaktionen werden als fehlerhaft markiert

Neu: Beim Versuch einen veralteten IPSWorkflow zu öffnen wird eine Migration ins aktuelle Format angeboten
Fix: Stelle Tooltips von langen Strings im Objektbaum korrekt dar
Fix: Verwende bei der Konfiguration von Diagrammen für Balkengraphen den korrekten Typ für Profile
Fix: Seltener Fehler, dass der Snapshot nicht geladen werden konnte (wenn mehrere Konsolen offen waren)
Fix: Mausklicks im Skripteditor landen wieder an der richtigen Position

SYMOS

Neu: NTP Einstellungen verbessert, sodass NTP Server auch per DHCP empfangen werden können
Neu: Connect: Neuer Spezialschalter CheckInterval zum Ändern des Prüfintervalls
Neu: Bei Logfiles kann mit „Neuste“ direkt an Ende gesprungen werden
Neu: Unter „Support → System anzeigen“ wird dargestellt, wenn eine RTC in der SymBox verbaut ist
Fix: Der Zurück Buttons bei Logfiles funktioniert wieder korrekt
Fix: KNX/ETS: Programmieren der physikalischen Adresse inkl. Applikation korrigiert
Fix: SymOS Oberfläche ich nun korrekt über IPv6 erreichbar
Fix: Interner Server Error beim Download von Logfiles korrigiert

PHP-SDK

Neu: Werden Tree und Configurator als Variable im Konfigurationsformular verwendet, beinhaltet nun jeder Knoten auch die Information „expanded“
Neu: Der Callback „onExpand“ für Tree und Configurator wird ausgelöst, wenn ein Knoten ausgeklappt wird
Fix: Ist ein Zahlenfeld im TestCenter leer wird nicht mehr auf einen ungültigen Wert geschaltet
Fix: Timer Intervall beim PHP SDK wird geprüft, sodass dieser nicht negativ wird
Fix: Bessere Fehlermeldung, wenn ein Button einen ungültigen Link zurückgibt
Fix: Bei der Erstellung von Instanzen werden OpenObjectButtons immer deaktiviert, damit die Konfiguration der Kette nicht unterbrochen wird
Fix: loadValuesFromConfiguration funktioniert wieder wie vorgesehen innerhalb von ExpansionPanels
Fix: SelectValue verursacht beim Öffnen der Konfiguration keine Fehler mehr und reagiert wie vorgesehen auf Änderungen

ENTERPRISE

Neu: Remote Sync (siehe: https://www.youtube.com/watch?v=njqcztynZjA)
Neu: Replication Sync (siehe: https://www.youtube.com/watch?v=njqcztynZjA)
Neu: Translation Modul kann Übersetzung direkt über DeepL vornehmen
Neu: Je WebFront können ein/mehrere Domains (VHosts) definiert werden, für die das WebFront angezeigt werden soll

NEUE MODULE

BACKUP (FTP/FTPS/SFTP)

Vollständige Backups jeden Tag zu einer gewünschten Uhrzeit
Optional auch Inkrementelle Backups (sodass das Backup immer nur aktualisiert wird)
Beliebige Ordner können gefiltert werden
Variable zur Uhrzeit des letzten erfolgreichen Backups
Variable zur letzten Backupgröße

AKTUALISIERTE MODULE

SPOTIFY

Neu: Funktion SPO_MakeAPIRequest um eigene Anfragen an die Spotify-API zu tätigen
Fix: Verhindere eine fälschliche Annahme, dass die Wiedergabe aktiv ist, sollten die Rückgabewerte der API problematisch sein

CSV ZIP EXPORT

Neu: Besserer Download und Download von Rohdaten

V6.2->V6.3 (Q4/2022)

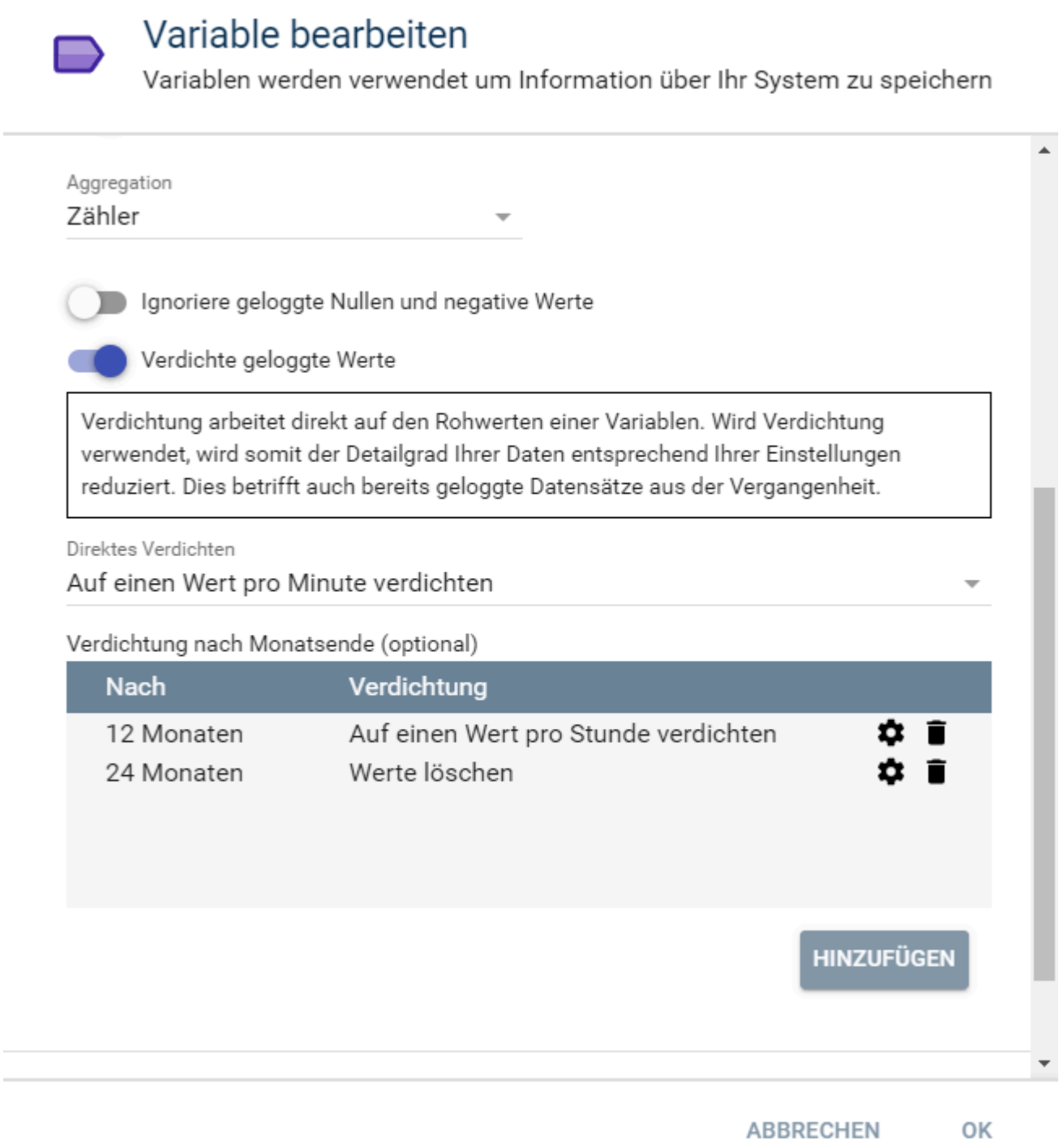
Wesentliche Änderungen
Migration
Liste der Änderungen
Neue und aktualisierte Module

WESENTLICHE ÄNDERUNGEN

Automatisches Verdichten von Archivdaten
Alle Graphentypen miteinander kombinierbar
Integration von IPSViewDesigner und IPSWorkflows
BACnet Server

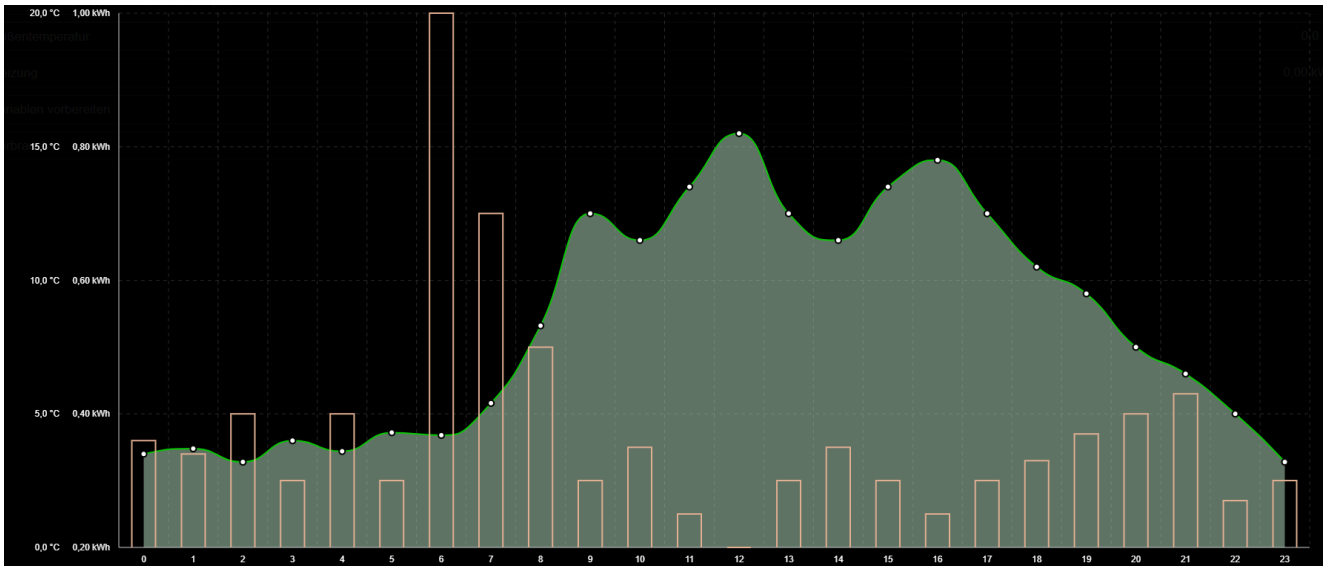
AUTOMATISCHES VERDICHTEN VON ARCHIVDATEN

Das Logging von Variablendaten ist wichtig um diverse Rückschlüsse zu ziehen und beispielsweise den Energieverbrauch zu verstehen und dadurch zu optimieren. Diverse Sensoren schicken hierbei hochauflösende Daten, wobei teilweise mehr als ein Datensatz pro Sekunde empfangen und im Archiv geschrieben wird. Dieser hohe Informationsgehalt schlägt sich natürlich im Speicherplatz nieder, obwohl die hohe Auflösung meisten Daten nach einer gewissen Zeit uninteressant werden. So reicht nach einem Jahr vielleicht ein Temperaturwert pro Stunde vollkommen aus und Werte, die älter als zwei Jahre sind, sind komplett uninteressant. Hierbei unterstützt die Verdichtgung, welche nach individuell einstellbaren Zeitabständen die Datensätze auf den gewünschten Detailgrad runterbricht oder auch komplett löscht. Dies kann pro Variable eingestellt werden, wodurch für jeden Fall die optimale Verdichtung gewählt werden kann.



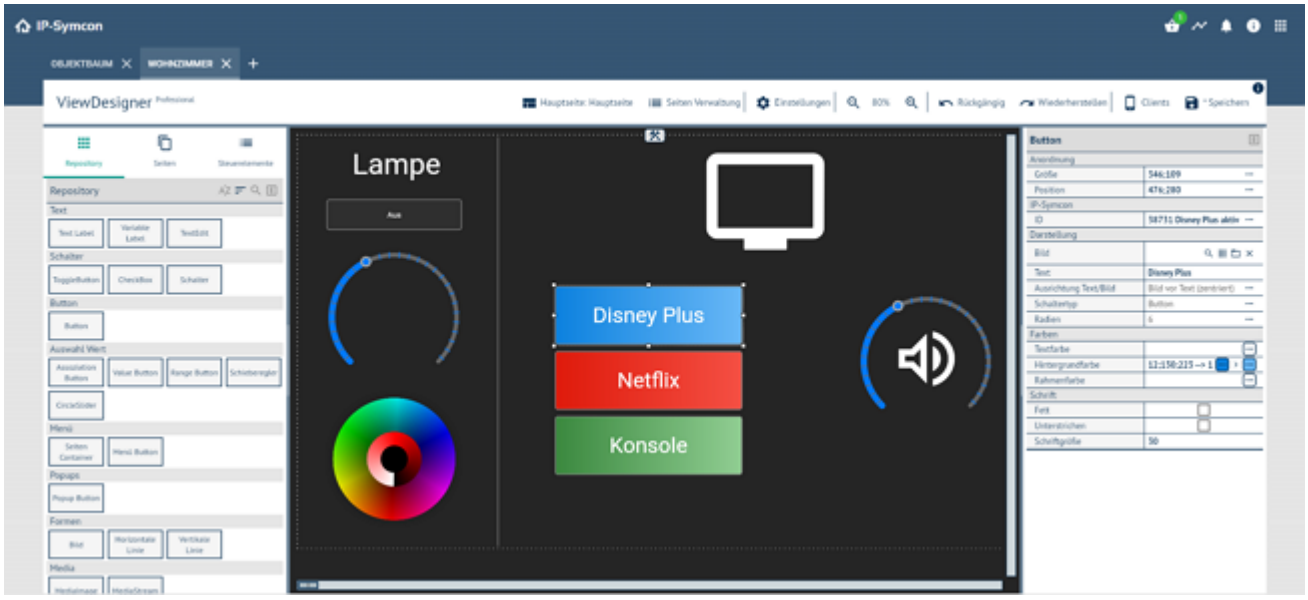
ALLE GRAPHENTYPEN MITEINANDER KOMBINIERBAR

Wie viel Energie verbraucht die Heizung in Abhängigkeit der Außentemperatur? Dank der neuen Kombination aller Graphentypen miteinander in einem einzelnen Diagramm, kann dies nun optimal zusammengeführt werden. Dabei wird der Zähler der Heizung wie gewohnt als Balkengraph und die Temperatur als Liniengraph dargestellt. In welchen Zimmern sind die Heizkörper aktiv? Erweitern Sie das Diagramm gerne auch noch um Booleangraphen um alle Informationen an einer Stelle zu präsentieren.



INTEGRATION VON IPSVIEWDESIGNER UND IPSWORKFLOWS

Kein Bedarf mehr für spezielle Anwendungen zu IPSStudio. Seit einiger Zeit können IPSViews bereits im WebFront dargestellt werden. Mit der neuen Version zieht nun auch die Konfiguration in die Konsole ein. So können IPSViews und Workflows plattformunabhängig direkt in der Konsole bearbeitet werden.



Zur Verwendung von IPSStudio in der Konsole ist eine Professional Lizenz für IPSStudio erforderlich.

BACNET SERVER

IP-Symcon kann nun nicht nur als BACnet Client, sondern auch als BACnet Server agieren. Dadurch kann IP-Symcon als Bridge agieren und Werte anderer Systeme an BACnet weiterleiten oder auch eigene Parameter zur Verfügung stellen. So können beliebige Werte in IP-Symcon anderen BACnet Clients zur Verfügung gestellt werden.

Das BACnet Modul ist eine kostenpflichtige Erweiterung, welche zu jeder bestehenden IP-Symcon Lizenz erworben werden kann. Für Details wenden Sie sich bitte an unseren Vertrieb.

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die aktuellste Version kann wie unter Installation beschrieben installiert werden.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: Verbesserte Sicherheit: Auch bei Anmeldungen über Localhost wird ein Kennwort erfordert, sofern der Fernzugriff aktiviert ist. Dies betrifft insbesondere Windows User, die nun ggf. ein Kennwort eingeben müssten.
Neu: Neue Version der IPSWorkflows wird unterstützt
Neu: Neue Version vom IPSViewDesigner wird unterstützt
Neu: Besserer Hinweis, wenn IP-Symcon initial von extern eingerichtet werden soll
Neu: Intern auf std::filesystem und std::variant (von C++20) umgestellt
Neu: Windows Installer setzt die IP-Symcon Version in den Datei-Eigenschaften
Neu: Windows Version kann auch unter Windows 10 (ARM64) installiert werden (ohne Gewähr und Support!)
Neu: Bibliothek zum Berechnen von Datum/Uhrzeit ausgetauscht
Neu: Bibliothek zum Lesen/Schreiben der php.ini ausgetauscht
Neu: Intern wurde alles auf Int64 umgestellt. Dadurch können Int64 Werte (insbesondere KNX DPT 29.x) vollständig dargestellt und archiviert werden. (Benötigt ein 64-Bit System. d.h. nicht für Raspberry Pi 32 Bit verfügbar. Dort ist zumindest PHP weiter auf 32 Bit limitiert)
Neu: Spezialschalter CompatibilitySloppyUTF8 der standardmäßig an ist, um die Konvertierung von nicht UTF-8 Konformen String wie bisher durchzuführen
Neu: CC_GetTrafficCounter um den exakten Verbrauch vom Connect Dienst der 24 Stunde Periode zu ermitteln
Neu: IPS_GetKernelDirSpace, um den freien Speicherplatz vom Datenlaufwerk anzuzeigen
Neu: Der Docker Tag testing ist ab sofort als Multi-Arch Image für amd64, arm64 und armhf verfügbar
Neu: GetValueFormatted/GetValueFormattedEx geben den unformatierten Wert auch ohne Profil zurück
Neu: Das Tray-Icon unter Windows öffnet bei Doppelklick die Pro Konsole. Der Verbindungsassistent ist separat im Kontextmenü verfügbar

Neu: SSE: Unterstützung für fragmentierte Events
Neu: IP-Symcon kann mit Einschränkungen auch ohne Root Rechte gestartet werden
Neu: IPS_ExecuteEx liefert Rückgabewert der Ausführung, wie auch IPS_Execute dies tut
Fix: Bessere Fehlermeldung im Logfile/Lizenz Ändern Dialog, wenn Subskription aufgrund einer veralteten Lizenz nicht ermittelt werden kann.
Fix: Bessere SSDP Suche unter Windows mit mehrere Netzwerkadaptern
Fix: Bessere Fehlermeldung, wenn Icons von defekten Skins nicht geladen werden konnten
Fix: Das Tray-Icon unter Windows sollte nicht mehr verschwinden
Fix: UTF-8 Problem im MQTT Konfigurator gelöst
Fix: UTF-8 Problem im Util Control gelöst
Fix: UTF-8 Problem mit Skriptausgaben von Ereignissen gelöst
Fix: Register Variable war ggf. falsche Fehlermeldung aufgrund fehlender Initialisierung einer internen Variable
Fix: Tray-Icon unter Windows prüft, ob der Dienst korrekt läuft, bevor er die Pro Konsole öffnet
Fix: Include-Pfad wurde unter Windows nicht korrekt gesetzt für Ereignisse mit ScriptText
Fix: Veraltete Spezialschalter werden nicht mehr angezeigt
Fix: Weiteres UTF-8 Problem im Util Control gelöst
Fix: IPS_Execute konnte (seit 4.0) keine Argumente mit Zeilenumbrüchen weiterreichen
Fix: Util Control zeigt die gefundenen Fehlerstellen für Umlaute/ShortTags wieder korrekt an
Fix: Referenzsuche/Skriptsuche kommt mit ungültigem UTF-8 besser klar
Fix: Weiteres UTF-8 Problem im MediaPlayer (Windows) korrigiert
Fix: Weitere UTF-8 Problem bei HID (Windows) korrigiert
Fix: Boolean (Variable) auf Boolean (Variable) Vergleich in bedingten Ereignissen korrigiert
Fix: Sicherheitslücke im WebServer korrigiert, welche nur mit direktem Netzwerkzugriff oder einer Port Weiterleitung ins Internet ausgenutzt werden kann

ARCHIV

Neu: Verdichten der Werte ist verfügbar!
Fix: Beim Entfernen aller Variablen aus dem Archiv wurde dies nicht korrekt abgespeichert
Fix: Bei einem Neustart konnten aktuelle Aggregationsdaten von Variablen mit Standardaggregation leicht abweichen. Dies wurde behoben.
Fix: Alle Archiv-bezogenen Dateioperationen werden mehrmals versucht, um Timing Probleme mit Backup-Software unter Windows zu umgehen.

AKTIONEN

Neu: Neue Aktion für den Ablaufplan zum Nachführen den Wochenplan-Ereignisses
Neu: Neue Aktion zum Senden von Push-Nachrichten mit Inhalt aus einer Variable
Fix: Aktionen mit potentiell nicht gesetzten Objekten, z.B. Push-Nachrichten, werden wieder korrekt formatiert
Fix: Archiv-Aktion konnte Werte für die Jahresaggregation nicht korrekt abrufen

GERÄTE

Neu: BACnet Server (Beliebige Variablen können im BACnet zur Verfügung gestellt werden)
Neu: KNX: DPT 1.024, 8.012, 12.1200, 12.1201, 13.1200, 13.1201, 13.016, 14.1200, 14.1201 hinzugefügt
Neu: Eltako FWWKW71L
Neu: Update Button in Konfiguration für OneWire, ModBus/Siemens Instanzen
Neu: I/O SSE: SSL-Zertifikat Prüfung kann deaktiviert werden

Neu: LCN Konfigurator zeigt Seriennummer, Firmware und Hardware an
Neu: LCN Quitterungen werden strikter erwartet/ausgewertet
Neu: Z-Wave: PHP-Befehle ZW_RequestPriorityGroup, ZW_UpdatePriorityGroup und ZW_UpdatePriorityReturnGroup
Neu: KNX DPT 9.030
Neu: MQTT: Neues Format Binär, welches binären Payload als HEX in String-Variablen speichert (Umgeht korrekt Probleme mit nicht validem UTF-8 in Strings)
Neu: BACnet: Relinquish Present Value Button direkt in der Instanz
Fix: BACnet: String im UCS-2 Format werden korrekt konvertiert
Fix: KNX: Einige Profile waren nicht korrekt UTF-8 kodiert
Fix: HomeMatic: Bessere Fehlermeldung beim Verbindungsaufbau
Fix: KNX Konfigurator funktioniert korrekt, auch wenn nicht unterstützte DPTs in der XML existieren
Fix: Kleine Anpassung den WebHooks, sodass diese mit OCPP kompatibel sind
Fix: SSE: Accept Header wird korrekt auf text/event-stream gesetzt, sodass z.B. die Hue Bridge kein application/json sendet
Fix: RegisterVariable: Interne Warteschlange wird auf 512kB limitiert
Fix: OZW: Bei mehreren Geräten konnten die DataPoint Instanzen dem falschen Splitter zugeordnet werden
Fix: OZW: Setze Wert auf 0, falls ---- empfangen wird (anstatt einer Fehlermeldung im Logfile)
Fix: ModBus Konfigurator konnte nicht geöffnet werden, wenn „fremde“ Instanzen am Splitter hingen
Fix: KNX DPT 12.1200, 12.1201, 13.1200, 13.1201 korrigiert
Fix: EnOcean: Eltako WWLW → Emuliere Status korrigiert
Fix: BACnet: Fehlermeldung beim Öffnen der Discovery Instanz gelöst (seit letztem Release)
Fix: BACnet: Nicht unterstützte NSDU Nachrichten werden nur im Debug Log gemeldet
Fix: MQTT: Ungültiges UTF-8 im Payload wird korrekt an PHP-Module weitergegeben
Fix: KNX Konfigurator mit mehreren IP-Gateway hat die Instanzen ggf. an das falsche Gateway verbunden

WEBFRONT

Neu: Bar/Line/Bool Graphen können alle in einem Diagramm kombiniert werden
Neu: Exakte Werte können via Hover auch für HD/RAW Werte abgelesen werden
Neu: IE11 wird nicht mehr korrekt unterstützt und wird als inkompatibler Browser identifiziert
Neu: Unterstützung einer listenbasierten Legende inkl. Filter sobald 10 oder mehr Graphen in einem Diagramm sind
Neu: Unterstützung für den neuen „Initial sichtbar“ Parameter bei Diagrammen
Fix: Legende vom Graphen mit gleicher MedienID aber anderem Offset werden korrekt getrennt
Fix: SVG: Werte-Overlays waren für einige Diagramme mit Achse auf der rechten Seite vertauscht
Fix: ~Door Profil hatte ein vertauschtes Icon
Fix: Sofern alle WebFronts unsichtbar waren, gab es eine kryptische Fehlermeldung

APP

Fix: ~Door Profil hatte ein vertauschtes Icon

KONSOLE

Neu: Module Store überarbeitet
    Module, die nur in anderer Sprache verfügbar sind, können dargestellt werden
    Zeige Text an, wenn keine Module gefunden wurden
Neu: IPS-Views und Workflows können direkt in der Konsole erstellt und bearbeitet werden
Neu: Optimiere Dialog zur Aktionsauswahl
Neu: Button um Mitteilungen im Status-Tab zurückzusetzen
Neu: In Objektauswahlfeldern kann bei sicherer Verbindung (HTTPS oder Pro Konsole) via Strg + C eine ObjektID kopiert und via Strg + V eingefügt werden
Neu: Beim Springen zu einem Objekt wird die Auswahl im Objektbaum nur aktualisiert, falls das Objekt noch nicht ausgewählt wurde. So bleibt eine mögliche Mehrfachauswahl erhalten
Neu: Scrolle beim „Gateway ändern“-Dialog zum aktuell ausgewählten Gateway
Neu: „Gateway ändern“-Dialog leicht visuell überarbeitet
Neu: Erhöhe Performance durch Verwendung von Lizenz- und Serverdaten aus dem Snapshot anstatt immer wieder anzufragen
Neu: Erhöhe Performance des Status Widget
Neu: Bessere Fehlermeldung, wenn nach der Lizenzänderung ein erneutes Einloggen notwendig ist
Neu: Dialog für Spezialschalter verbessert (Zeigt nicht übernommene Änderungen und Standardwerte an)
Neu: Meldung, falls der Speicherplatz knapp wird
Neu: Wird mit dem Schnellfilter im „Instanz erstellen“-Dialog nichts gefunden werden, kann direkt der Module Store geöffnet werden
Neu: Zeige Hinweis, falls einer der DebugDisable* Spezialschalter aktiv ist
Neu: Bei Diagrammen kann angegeben werden, ob ein Chart standardmäßig aktiv sein soll
Neu: „Passwort vergessen“-Button beim Login
Fix: Zeile Zeilenumbrüche in Beschreibungen im Module Store wie vorgesehen an
Fix: Visualisierungen können im Objektbaum nicht mehr verschoben werden
Fix: Verhindere, dass Zeilen bei inaktiven Tabs beim Wechsel des Kompaktmodus verschwinden
Fix: Verbesserte Fehlermeldung bei ungewöhnlichen Fehlern beim Verschieben von Objekten
Fix: Werden alle Dialoge geschlossen, beispielsweise von einem Konfigurator in einem Popup, so können diese wie vorgesehen danach wieder geöffnet werden
Fix: Bei Listen in Dialogen wird nun die Scrollposition korrekt initialisiert und es kommt nicht zu „leeren“ Listen
Fix: Da Firefox die Clipboard-API im Browser nicht vollständig unterstützt, wird dies nun überprüft und die Fehlermeldung entsprechend angepasst
Fix: Bei langen Strings wird der Variableninhalt im Dialog zum Variable bearbeiten nicht mehr doppelt angezeigt
Fix: Beim Bewegen eines Elements per Drag & Drop stellt der Zeiger nun korrekt dar, dass es sich um eine Bewegen-Aktion und nicht um eine Kopieren-Aktion handelt
Fix: Bei einer Anpassung der Graphenreihenfolge in einem Diagramm wird dies korrekt abgespeichert
Fix: Die Standardaktion wird nicht mehr modifiziert, wenn dies nicht explizit im Dialog getan wird
Fix: Zeige keine Warnung im Visualisierungs-Widget, falls keine Lizenz gesetzt ist
Fix: Wird ein Ereignis erstellt, wird wie erwartet im Objektbaum dahin gesprungen
Fix: Wird eine Liste oder ein Baum erneut geöffnet, wird die aktuelle Scrollposition aktualisiert. Dies verhindert den Fehler, dass die Profilauswahl unter bestimmten Umständen leer war
Fix: Wird bei zyklischen Ereignissen das Tagesmuster angepasst, so werden auch dazugehörige Änderungen korrekt übernommen, auch wenn die Standardeinstellungen beibehalten werden
Fix: Fordere nur eine Reaggregation aufgrund von veränderter Ausdünnung, wenn entsprechende Einstellung tatsächlich angepasst wurde
Fix: Verhindere kosmetische Fehlermeldung im Profildialog

Fix: Verhindere Fehler, beim duplizieren von Profilen und direktem Löschen von Assoziationen
Fix: Verhindere Fehler beim initialen Rendern eines Objektbaums unter Firefox
Fix: Liefert ein Konfigurator oder eine Discovery Instanz bei der Gerätesuche ungültige Werte, löst dies keinen Fehler aus
Fix: Unter Firefox wurden Listen in seltenen Fällen leer dargestellt und erst beim Scrollen aktualisiert
Fix: Mehr Informationen, wenn die „Timestamp is missing“ Fehlermeldung kommt
Fix: Zeige Listen in Dialogen wieder korrekt an, z.B. Ereignisliste von Skripten oder Auswahl der Systeme bei der Gerätesuche
Fix: Dialog zum prüfen im Konfigurator behoben

PRO-KONSOLE

Neu: Kleine Anpassungen für den neuen IPSViewDesigner
Fix: Werden die Skripteinstellungen in einem abgedockten Skripteditor geändert, wird dies für alle Fenster übernommen
Fix: Kopieren von ObjektIDs aus Auswahlfeldern in der Pro Konsole führte zu einem Fehler

SYMOS

Neu: Kernel auf 5.10.110 aktualisiert
Neu: SymOS: Einrichten der Lizenz speichert das Lizenzdatum und zeit es bei „Lizenz anzeigen“ an
Neu: Aktuelle IP-Adresse kann über Support → System anzeigen angefragt werden

PHP-SDK

Neu: SelectValue Element (Dokumentation)
Neu: Feld „discoveryInterval“ für Configurator ermöglicht eine Einschränkung der automatischen Gerätesuche
Neu: Standardwert für SelectObject und dessen Derivate ist nun 1, also „Nichts ausgewählt“
Fix: Existiert das Ziel in einem SelectAction nicht mehr, so wird dies passend dargestellt
Fix: Zeige SelectColor in List oder Tree über dem Grau der ausgewählten Zeile an
Fix: Wird ein Icon im SelectIcon ausgewählt, wird wie vorgesehen die „Änderungen übernehmen“-Zeile eingeblendet
Fix: Bessere Fehlermeldung, wenn eine verschachtelte Liste kein gültiges JSON ist
Fix: Verschachtelte Listen haben nun in PHP-Funktionen wie vorgesehen Zugriff auf $id
Fix: Soll ein PopupAlert geöffnet werden während der Tab mit der Instanz nicht aktiv ist, wird das Popup erst geöffnet sobald der Tab wieder aktiv wird
Fix: Spalten mit SelectDate/SelectTime/SelectDateTime in List und Tree werden korrekt sortiert

NEUE MODULE

SNMP

Offizielles SNMP Modul der Symcon GmbH. Unterstützt SNMP V1/V2/V3 und kann alle OIDs eines Geräts per Walk anzeigen. Per Checkbox können die gewünschten Werte als Variablen in IP-Symcon erstellen werden, welche dann zyklische aktualisiert werden.

AKTUALISIERTE MODULE

AMAZON ALEXA

Fix: Verbesserte Fehlermeldungen

ANWESENHEITSSIMULATION

Fix: In seltenen Fällen wurde die Simulation fälschlicherweise gestoppt

BENACHRICHTIGUNG

Neu: Unterstützung für neues Telegram Bot Modul

GOOGLE ASSISTANT

Neu: Link um den Skill bei Google zu bewerten
Fix: Geräte, die 30 Minuten keine Updates schicken, werden nicht mehr als offline angenommen

RECHENMODULE

Fix: Fehlende Konfigurationselement wieder hinzugefügt

REPORT MODUL

Neu: Temperature Variable ist optional

SNMP

Neues Modul im Module Store

SPOTIFY

Fix: Gehe korrekt mit abgelaufenen oder falschen Token um
Fix: Verhindere Fehler bei deaktivierter Wiedergabe

TELEFONANSAGE

Neu: Benutze das neue SelectModule Element

VIRTUELLE MESSSTELLE

Neu: Verbesserte Berechnung

V6.1->V6.2 (Q2/2022)

WESENTLICHE ÄNDERUNGEN

Gleichzeitiges Editieren
Verbesserter Module Store
Verbesserte Performance
Neue Module von der Symcon GmbH

GLEICHZEITIGES EDITIEREN

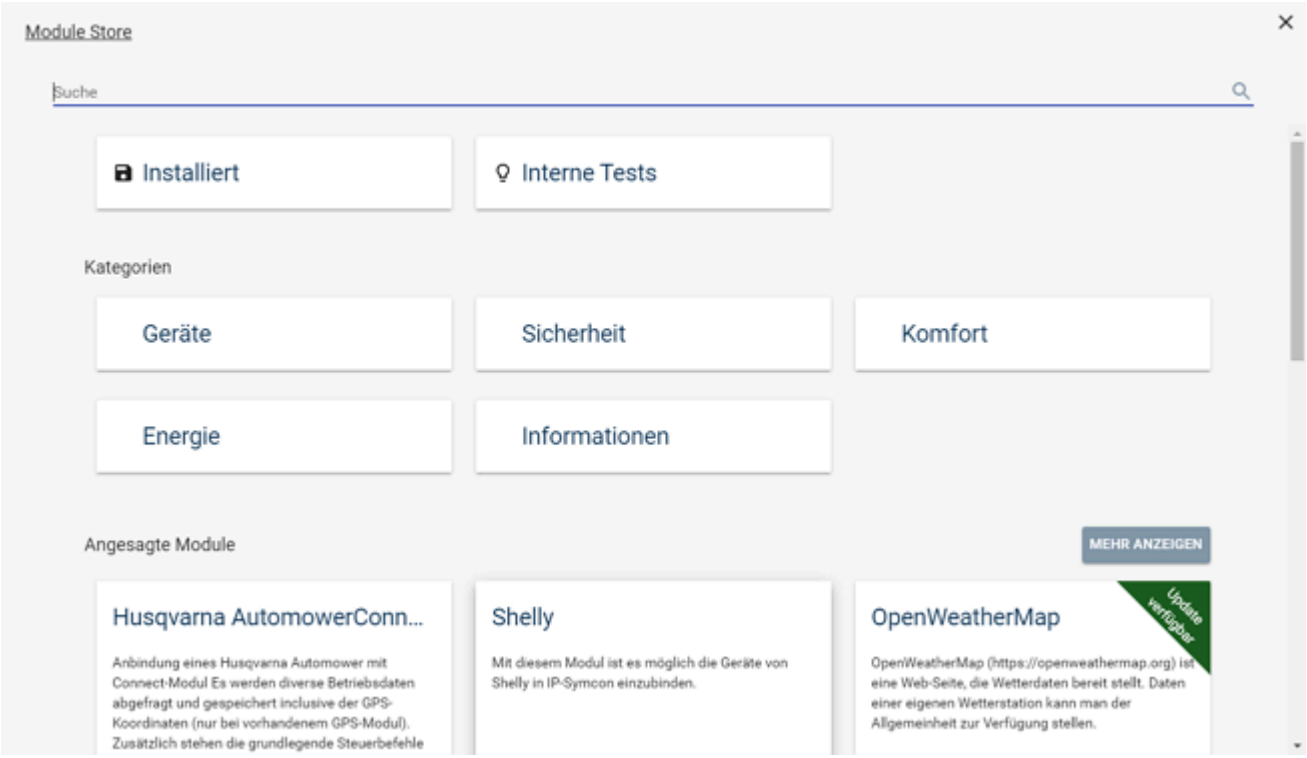
Mit der neuen Version können in der Konsole beliebig viele Objekte gleichzeitig bearbeitet werden. So kann beispielsweise schnell und unkompliziert das Logging für viele Variablen aktiviert oder die Sichtbarkeit im WebFront für viele Objekte gleichzeitig konfiguriert werden. So gelangt man bei einer großen Menge von Operationen mit viel weniger Klicks ans Ziel.



VERBESSERTER MODULE STORE

Der Module Store wurde an diversen Stellen verbessert und erweitert. So weist der Module Store nun auf den ersten Blick auf das Icon auf verfügbare Updates hin und ermöglicht auch eine direkte Installation aller Updates. Dazu kommt eine Anzeige verschiedener interessanter Modullisten: Welche Module sind angesagt? Welche Module sind ganz neu oder haben gerade erst ein Update erhalten? Schließlich ist nun zusätzlich zum Stable und Beta Kanal auch der Testing Kanal verfügbar. Dieser ermöglicht es Entwicklern gezielt Tester

für interne Modultests einzuladen. Dann können nur eingeladene Benutzer über den Module Store die Testing-Version des Moduls installieren.



VERBESSERTE PERFORMANCE

Zur 6.2 haben wir an diversen Stellen noch weitere Verbesserungen der Performance durchgeführt. Werden Objektbäume mit mehreren Tausend Objekten geöffnet, so geschieht dies nun nahezu sofort. Auch das Ausklappen von Kategorien mit extrem vielen Kindern geht nun noch schneller als jemals zuvor. Auch der Start von IP-Symcon wurde beschleunigt. Werden die Variablen ins Archiv geladen, so werden alle Prozessorkerne parallel ausgenutzt, wodurch IP-Symcon nach einem Start schneller zur Verfügung steht. Schließlich wurde der MQTT Server optimiert und kann nun performant mit viel mehr Geräten gleichzeitig umgehen.

NEUE MODULE VON DER SYMCON GMBH

Auch bei der Entwicklung von Modulen hat sich einiges getan. So sind einige neue Module im Module Store zu finden bzw. Erweiterungen zu existierenden Modulen:

- Betriebsstundenzähler (Dokumentation)
- Virtuelle Messstelle (Dokumentation)
- CSV Zip Export (Dokumentation)
- Variablen Vergleich (Dokumentation)
- Verbrauchsverhalten (Dokumentation)
- Virtueller Zähler (Dokumentation)
- Symcon Report (Erweitert um den Energie Report, Dokumentation)
- JSON Exporter (Dokumentation)

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Es wird unter Linux mindestens Ubuntu 20.04 erfordert
Es wird beim Raspberry Pi mindestens Raspbian OS (Bullseye) erfordert

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: MessageQueue wird parallel abgearbeitet und verbessert die Latenz, wenn z.B. PHP Module diese vorher zu lange blockiert hatten
Neu: Datenfluss für PHP-Module optimiert (sollte CPU-Last leicht senken)
Neu: Datenfluss-Filter (RegEx) von PCRE zu RE2 gewechselt (wesentlich bessere Performance)
Neu: Archiv aktualisiert Aggregationen beim Start parallel
Neu: Beim Ausführen von Skripten werden PHP-Module erkannt und als solche mit SenderID angegeben
Neu: IP-Symcon prüft, ob die Subskription für die IP-Symcon Version noch gültig ist
Neu: ServerSocket unterstützt IPv6 sofern verfügbar
Fix: Wir prüfen jetzt vor dem Installieren, ob die Linux Version aktuell genug ist und brechen ansonsten mit einer passenden Fehlermeldung ab
Fix: Fehlermeldung beim Verlinken von access_id.log korrigiert
Fix: Prüfung, ob Raspbian Bullseye / Ubuntu 20.20 oder neuer installiert ist funktioniert wieder korrekt
Fix: Prüfung der Demo-Lizenz/Subskription war nicht vollständig korrekt
Fix: Einige veraltete Aktionen wurden unter Windows nicht aufgeräumt
Fix: Abstürze mit HomeKit auf älteren Macs
Fix: URL von ip-symcon.de auf symcon.de 1 in den UPNP Anfragen angepasst
Fix: Speicherleck unter Windows bei SSL-Verbindungen gelöst
Fix: Seltenes Speicherleck bei RTSP Streams und Verbindungsabbrüchen gelöst
Fix: Hoher Speicherverbrauch (durch zu viel Caching) bei vielen PHP-Threads und hohen Speicherverbrauch in Skripte (z.B. durch große IPSView Dashboards) gelöst
Fix: Durch eine Umstellungen von GitHub müssen wir alle git:// Skins/Module zu https:// konvertieren (Betrifft nur Skins und alte Module)
Fix: Bei zu schnellem Shutdown von IP-Symcon konnte dies am Ende abstürzen
Fix: Sys_GetCPUInfo lieferte unter Windows 0 beim ersten Aufruf

AKTIONEN

Neu: Mathematische Funktionen können nun sowohl mit einem konstanten als auch dem Wert einer anderen Variable verwendet werden

GERÄTE

Neu: W&T unterstützt CO2 Sensoren
Neu: BACnet: Datenpunkt to Geräten werden einer gerätespezifischen Kategorie erstellt
Neu: MQTT: Button in den Konfiguratoren, der erlaubt die Liste zu leeren
Fix: KNX: Verwerfe Pakete von alten Verbindungen
Fix: MQTT: Client gibt Hinweis, wenn ein Neu-Verbinden nach einer Änderung erforderlich ist
Fix: ModBus: String-Werte filtern korrekt Null-Characters am Ende
Fix: BACnet: Absturz bei einigen Geräten, wenn beim Anfragen der Objektliste nicht wie erwartet die Länge gemeldet wurde
Fix: BACnet: Lade Objektliste in einem Stück, sofern das Gerät dies unterstützt
Fix: KNX: Button zum Aktivieren des Programmiermodus auf der SymBox funktioniert wie erwartet
Fix: Gerätesuche für unsere LAN-Gateway ist schneller
Fix: IRTrans: Konfigurationsseite konnte bei Fehlerhafter Verbindung nicht geöffnet werden
Fix: KNX: Problem beim Trennen der Verbindung beim Beenden von IP-Symcon gelöst
Fix: KNX: DPT 19: Wochentag wurde falsch gesendet, wenn RequestAction genutzt wurde
Fix: MQTT: Fehlerhafte Verbindungen werden besser/schneller getrennt, damit sich (z.B. die Shelly's) korrekt neu verbinden können

KONSOLE

Neu: Store wurde überarbeitet
    Module werden nun alphabetisch sortiert
    Eine grüne Zahl am Module Store Icon zeigt an, wenn Updates vorhanden sind
    Button um alle installierten Module gleichzeitig zu aktualisieren
    Spezialkategorien für Top Downloads, Angesagt Module, Neueste Updates und Neueste Module
    Unterstützung der Spezialkategorie für Testing (allerdings kann aktuell noch nicht im persönlichen Bereich eingeladen werden)
Neu: Mehrfachauswahl wird nun bei allen Operationen im Objektbaum unterstützt, insbesondere auch bei „Objekt bearbeiten"
Neu: Zeige Dateipfad in PHP Informationen an
Neu: Verbesserte Performance für alle Arten von Bäumen und Listen in der Konsole
Neu: Mehrere bewegbare Dialoge werden leicht versetzt geöffnet
Neu: Doku-Widget bietet nun einen Button, um die Dokumentation direkt zu öffnen
Neu: Filter im Archiv
Neu: Aggregationstyp wird im Icon im Archiv angezeigt
Neu: Zeige Profiltyp im Profil-Manager an
Fix: Beim Löschen von Variablen aus Diagrammen wird dies korrekt gespeichert
Fix: In offenen Skripten suchen schlägt nicht mehr fehl, wenn manche der Skripte während der aktuellen Session noch nicht ausgewählt wurden
Fix: Verhindere fälschliche Anzeige, dass der Browser veraltet ist, falls die Verbindung langsamer ist
Fix: Lokalisierte Darstellung von Kommazahlen im Archiv und „Variable bearbeiten"-Dialog
Fix: Das Achievement-Widget (Aprilscherz 2020) verursacht keine Fehler mehr bei neu erstellten Objekten, die direkt wieder gelöscht werden
Fix: Wird der Name eines Ereignisses geändert, wird dies direkt im Objektbaum dargestellt
Fix: Verhindere fälschliche geforderte Reaggregation bei neu erstellten Variablen

Fix: Variablen werden in Dialogen wie dem Auswahlbaum einer auslösenden Variable eines Ereignisses korrekt zentriert dargestellt
Fix: Kann eine Subskription aufgrund einer fehlenden Internetverbindung nicht geprüft werden, wird dies dem Benutzer mitgeteilt
Fix: Listen wurden manchmal leer dargestellt und erst beim Scrollen korrekt dargestellt

PRO-KONSOLE

Keine Änderungen

SYMOS

Neu: Kernel auf 5.10.92 aktualisiert
Neu: IPv6 ist verfügbar (noch nicht aus SymOS heraus konfigurierbar)
Neu: Anzeige der SymBox Version auf der Support Seite mit Speicher/RAM Größe
Neu: Zeige Compute Module Version auf der Support/System-Seite
Neu: Backups werden mit dem Format Jahr/Monat/Tag anstatt Tag/Monat/Jahr erstellt für eine bessere Sortierbarkeit
Neu: „Backup wiederherstellen“ Button direkt auf der Backup-Seite
Fix: Einspielen von Backups ist auch ohne Internet-Verbindung wieder möglich
Fix: Beim Zurücksetzen auf Werkseinstellung wird auch der Update-Kanal korrekt zurückgesetzt
Fix: Logfile-Ansicht funktioniert wieder korrekt, sobald man auf „Mehr“ klickt
Fix: VoIP auf der SymBox wieder verfügbar
Fix: Backups über 2GB können korrekt wiederhergestellt werden
Fix: Backups im Safari werden als zipbackup heruntergeladen, um nicht automatisch entpackt zu werden

PHP-SDK

Neu: „Nichts ausgewählt“ entspricht bei SelectObject und dessen Derivaten nun dem Wert 1, wodurch die Hauptkategorie mit der ID 0 bewusst ausgewählt werden kann
Neu: Flag „deprecated“ für veraltete Aktionen um Sie korrekt in ihre Nachfolger zu konvertieren
Fix: PHP-Module konnten Abarbeitung von KERNELSTARTED im MessageSink behindern

V6.0->V6.1 (Q1/2022)

WESENTLICHE ÄNDERUNGEN

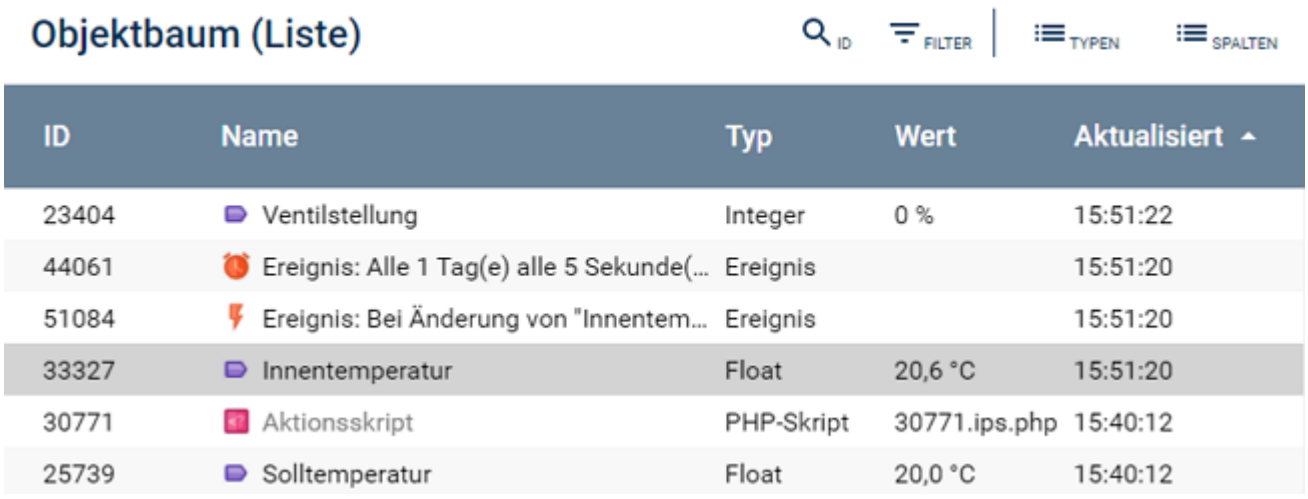
Mehr Performance
Objektbaum in Listenansicht
Gerätesuche in Sprachassistenten
SymOS mit 64 Bit

MEHR PERFORMANCE

Die Performance von allen Baum- und Listendarstellungen in der Konsole wurde noch einmal deutlich verbessert. Werden tausende Objekte gleichzeitig dargestellt oder Kategorien mit mehr als 1000 Unterobjekten geöffnet, läuft dennoch alles flüssig weiter. Und auch wenn sich an allen Stellen im Objektbaum gleichzeitig etwas ändert, stört dies die Nutzung nicht durch Verzögerungen oder Ruckler, es kann einfach flüssig weiter gearbeitet werden.

OBJEKTBAUM IN LISTENANSICHT

In der Listenansicht des Objektbaums können alle Objekte in IP-Symcon gleichzeitig dargestellt werden. Damit kehrt die klassische Ansicht aus der Legacy-Konsole in die aktuelle Konsole zurück und ermöglicht eine Sortierung der Objekte nach beliebigen Spalten. Damit kann schnell gesehen werden, welche Objekte im Baum häufig aktualisiert werden oder welche Instanzen fehlerhaft oder inaktiv sind.



GERÄTESUCHE IN SPRACHASSISTENTEN

Alle Sprachassistenten bieten nun eine Gerätesuche. So müssen nicht mehr alle Geräte einzeln eingepflegt werden. Stattdessen wird die Gerätesuche einmal gestartet und entdeckt passende Geräte. In der Auswahl kann dann schnell ausgewählt werden, welche dieser Geräte tatsächlich im Sprachassistenten integriert werden sollen und schon ist der Assistent blitzschnell bereit zur Verwendung.



SYMOS MIT 64 BIT

IP-Symcon und das SymOS nutzen auf der SymBox jetzt das volle Potential der 64-Bit-Architektur. Dadurch läuft IP-Symcon noch performanter und ermöglicht noch größere Projekte auf der SymBox.

WEBINAR ZUM THEMA IP-SYMCON 6.1

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: access_*.log Dateien vom WebServer werden automatisch rotiert
Neu: Receive-/Forward-DataFilter in PHP-Modulen können größer als 16kb sein und sollten wesentlich performanter ausgewertet werden
Neu: Profile Watt und Electricity.Wh
Neu: IPS_GetSystemLanguage, um die erkannte Systemsprache auszugeben
Neu: Spezialschalter zum Debuggen (Deaktiviert jeweils GetSnapshot, GetSnapshotChanges, WebSocket Rückkanal, Streams)
Neu: Spezialschalter, um RTSP Streams über TCP statt UDP zu empfangen
Neu: Sofern "MessageQueueWatch" aktiv ist, werden Statistiken pro Instanz geführt (IPS_GetInstanceMessageStatistics())
Neu: Statistiken für Anzahl der Nachrichten im Datenfluss
Neu: Einige Bibliotheken aktualisiert
Fix: max_execution_time ebenfalls für ini_set deaktiviert
Fix: Google Assistant Request Limit wird bei vielen Geräten korrekt erweitert, wenn die GA Instanz erst nach der Connect Control Instanz geladen wurde
Fix: Notification Control unterstützt Views von IPSView wieder korrekt
Fix: IP-Symcon wird korrekt gefunden, wenn ein & im Namen der Hauptkategorie stand
Fix: Ablaufpläne mit Wartezeit blockieren nicht mehr beim Herunterfahren
Fix: Zeitzone Etc/UTC (Default auf Ubuntu) startet korrekt
Fix: Beim Start werden unvollständige Objekte aus dem Snapshot entfernt
Fix: Limitierung bei RTSP/MJPEG Streams greift jetzt korrekt und verhindert massiven Speicherverbrauch, sofern Clients keine Daten mehr abgenommen haben
Fix: Beim Ändern von „Erweiterten Bedingungen“ mit Variablenvergleichen wurden die neue Werte als 0 statt dem eigentlich Wert in der Settings gespeichert, sodass nach einem Neustart die „Erweiterte Bedingung“ falsch konfiguriert war
Fix: Dekodieren von Absendern bei POP3 Nachrichten korrigiert (Bei IMAP bereits seit 2016 korrekt)
Fix: DNS-SD: Unter Windows konnte in seltenen Fällen sehr viele gleiche Fehlermeldungen auftreten
Fix: Löschen von Skripten/Medien legt diese wieder im „deleted“ Ordner ab
Fix: WebFront: Sofern Diagramme im WebFront sind und diese Variablen enthalten, welche Unterobjekte besaßen, wurden diese in seltenen Fällen nicht im WebFront angezeigt
Fix: FlowWatchTxLimit/FlowWatchRxLimit/MessageQueueWatchLimit kann mit 0 deaktiviert werden

ARCHIV

Fix: Falls es Daten aus der Zukunft gibt, üblicherweise durch eine Korrektur der Uhrzeit, so werden diese bei der minütlichen Aggregation einfach nicht angezeigt anstatt Fehler zu verursachen

AKTIONEN

Fix: Ereignisse mit fehlerhafter Aktion werden korrekt im Log protokolliert anstatt die Abarbeitung mit einer Exception zu beenden

GERÄTE

Neu: HomeMatic: Konfigurator zeigt Geräte als Baum an, sodass die Kanäle unterhalb angezeigt werden
Neu: HomeMatic: Datenfluss unterstützt Komplexe Werte (als JSON-kodiert)
Neu: HomeMatic: Datenfluss unterstützt Anfrage für HMScript
Neu: HomeMatic: Konfigurator/Socket kann Gruppen (Port 9292) auslesen
Neu: HomeMatic: Konfigurator zeigt gute Gerätenamen an
Neu: Technische Alternative CMI: Vollständige Unterstützung von Protokoll-Version 6.0
Neu: Eltako: Schalter für Serie 62 werden besser Unterstützt (Eigene Instanz)
Neu: BACnet: Unterstützung für MultiState
Neu: BACnet: COV kann ein-/ausgeschaltet werden
Neu: BACnet: Priorität beim Senden kann eingestellt werden
Neu: BACnet: BACNET_RelinquishPresentValue zum „Ausnullen“ es vorher gesendeten Wertes
Neu: BACnet: Strings mit ISO 8859-1 Kodierung werden unterstützt
Neu: KNX: Status-GA Erkennung kann nun auf verschiedenen Ebnen suchen
Neu: KNX: Für Weinzierl kBerry und SymBox mit KNX kann der Programmiermodus für die physikalische Adresse aus der Gateway Instanz aktiviert/deaktiviert werden
Neu: LCN: Warten auf Quittierungen ist möglich und standardmäßig aktiv
Neu: Z-Wave: Schönere Positionierung von Buttons in Popups der Konfiguration
Neu: ModBus: Option zum Byteweise verdrehen von Strings
Neu: ModBus: Optionen werden je nach gewähltem Datentyp angezeigt/versteckt
Neu: W&T unterstützt CO2 Sensoren
Fix: KNX: Fehlermeldung vom expliziten Lesen von Werten auf dem Bus, sofern die GA unter „Mehr?“ definiert war
Fix: BACnet: Priorität wird standardmäßig als 8 (Handbetrieb) gesendet
Fix: BACnet: Daten vom Typ Enum werden korrekt gesendet
Fix: BACnet: Fehler im Geräte-Konfigurator sofern keine Objekte verfügbar sind
Fix: BACnet: Forwarded-NPDUs werden im BVLC Header unterstützt
Fix: BACnet: Ignoriere WhoHas Service Requests, auf welche wir nicht antworten müssen
Fix: Elatako FAHB130 sollte Daten korrekt empfangen
Fix: Z-Wave: NodeID wird bei Nonce Requests angezeigt
Fix: MQTT: Korrigiert „Packet was not fully consumed“ Fehlermeldungen, welche bei abgebrochenen Verbindungen auftreten konnten
Fix: digitalStrom: Workaround für Bug in dSS Firmware 1.19.2
Fix: Z-Wave Konfigurator fragt Node Informationen seltener ab
Fix: ModBus: Auslesen von Strings wurde Byteweise verdreht
Fix: BACnet: Unterstützung von Geräte mit UCS-2 Kodierung in Strings
Fix: JSON Decoder kommt mit Sonderzeichen im Identifikator zurecht
Fix: KNX: DPT 3.x kann korrekt geschaltet werden

I/OS

Neu: WebSocket Client kann benutzerdefinierte Header mitsenden
Fix: HTTP/WebSocket Bibliothek aktualisiert

KONSOLE

Neu: Performance von Listen/Bäumen mit vielen Elementen drastisch erhöht
Neu: Objektbaum kann als Listenansicht dargestellt werden
Neu: Die Gerätesuche informiert auch über deaktivierte/fehlerhafte Konfiguratoren/Discoveries

Neu: In Bedingungen können Variablen mit Variablen verglichen werden
Neu: Objekte, die neu erstellt wurden und deren Elternobjekt direkt geändert wird, flackern nicht mehr in der Hauptkategorie
Neu: Diagramme vom Typ Bool, Standard, Zähler können alle miteinander kombiniert werden
Neu: Prüfe im Profilmanager, ob die Assoziationswerte von Float-Profilen zu den eingestellten Nachkommastellen passen
Neu: Logik beim Duplizieren von Aktionen im Ablaufplan optimiert
Neu: Wird im Eingabefeld der Iconauswahl etwas eingegeben, so öffnet sich eine Autovervollständigung und bietet passende Icons an
Neu: Das Icon von Variablen mit Aktion wird dunkler dargestellt als von Variablen ohne Aktion
Fix: Assoziationen von Stringprofilen werden in Erstellungsreihenfolge angezeigt
Fix: Schöne Fehlermeldung für den „out-of-memory“ Bug von Firefox (Version 92 bis 93)
Fix: Werden Zeiten im Events- oder Timertab sortiert, kommt „Nie“ unabhängig von der Sortierrichtung immer nach unten
Fix: Wird ein Objekt unter sich selbst dupliziert, wird eine Endlosschleife verhindert
Fix: Spezialkategorien werden wie vorgesehen gefiltert
Fix: Für alle Tastenkombinationen wird das Cmd von Macs als Strg gewertet
Fix: Aktualisiere verwendete Bibliotheken
Fix: Kontextmenü kann unter Safari wie vorgesehen mit Strg + Klick geöffnet werden und schließt sich nicht direkt wieder
Fix: Fehler behoben, wenn die Position mehrerer Objekte gleichzeitig aktualisiert wird (typischerweise ein Objekt und ein Link darauf)
Fix: Behebe Fehler, wenn bei einer erweiterten Bedingung initial ein String gewählt wird
Fix: Eingaben für Link-Parameter nutzen die volle verfügbare Breite
Fix: Bessere Fehlermeldung, wenn die Verbindung abbricht
Fix: Verhindere Fehler beim Verschieben von Aktionen im Ablaufplan auf die oberste Ebene
Fix: Typo in „Autor“ im Instanz-Dialog
Fix: Wird bei der Betrachtung von Archivwerten das Datum manuell eingegeben, wird das Datum erst ausgewertet sobald der Wert gültig ist
Fix: Beim Hinzufügen von Bedingungen mit prozentualen Variablen wird der neue Wert echt gerundet und so meistens der eingegebene Wert auch angezeigt wird
Fix: Bei sehr vielen Instanzen kommt es während der Gerätesuche nicht mehr zum Überlauf

PRO-KONSOLE

Neu: Die Breite von Dialogen kann angepasst werden und wird auch gespeichert
Neu: Upgrade auf neuste Bibliotheken
Neu: MacOS wird für als Universal App für Intel/Apple Silicon erstellt
Neu: „Versionsinformationen kopieren“ Button
Fix: Beim Öffnen der Konsole, öffnet sich diese ohne ein flackerndes Zwischenfenster

SYMOS

Neu: Wechsel auf 64-Bit für SymOS und IP-Symcon

PHP-SDK

Neu: Parameter „color“, „bold“, „underline“ und „italic“ für Label
Neu: Erweiterte Datenfluss-Funktionen SendDataToParentEx und SendDataToChildrenEx mit der alternative Pakete versendet werden können (wovon die Ziel-Instanz nur das erst-beste Verarbeitet)

Neu: Parameter poup.closeCaption für PopupButton
Neu: Parameter popup.buttons für PopupButton und PopupAlert
Neu: Parameter des Popups von PopupButton und PopupAlert können abgesehen von items dynamisch geändert werden
Neu: Wird innerhalb von einer Sekunde nach einer Schaltung im TestCenter der neue Wert nicht von der Variable bestätigt, setzt sich der dargestellte Wert zurück
Neu: Element SelectIcon zur Icon-Auswahl
Neu: Ist bei einem Image onClick gesetzt, so wird das Mauszeiger über dem Image zum Zeiger
Fix: Lazy-Container (ExpansionPanel) stellen Ihren Inhalt auch vor dem ersten Ausklappen bereit
Fix: Hat eine Liste nicht die volle Breite, wird der „Hinzufügen“-Button dennoch rechts unter der Liste angezeigt
Fix: „Falsey“ Werte für SelectAction werden als „Keine Aktion“ interpretiert
Fix: SetBuffer bei PHP-Module prüft und limitiert den Speicherverbraucht auf max. 1 MB pro Instanz. Das Limit pro Buffer liegt bei 512kB.

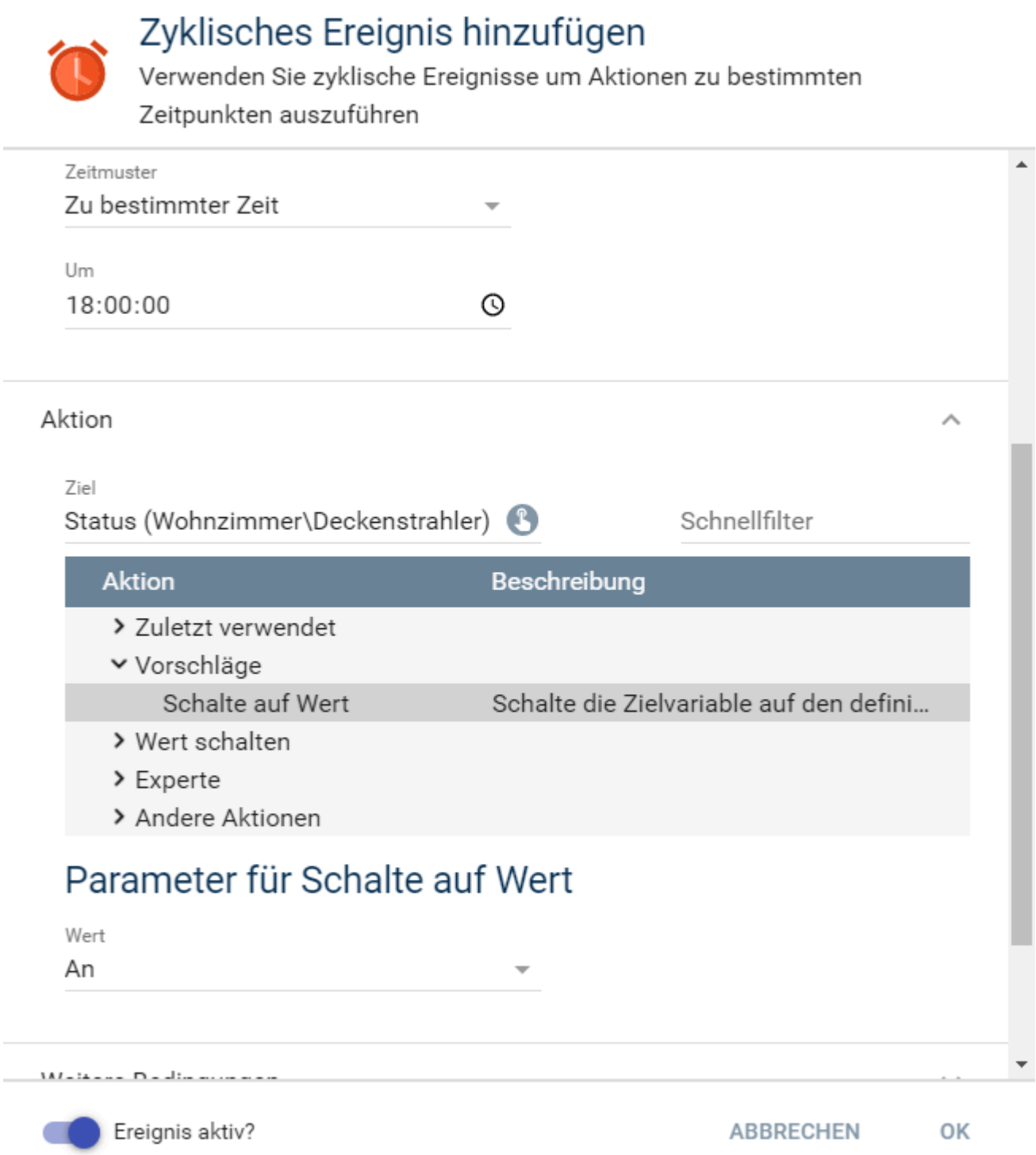
V5.5->V6.0 (Q3/2021)

WESENTLICHE ÄNDERUNGEN

Aktionen
Ablaufpläne
Gardena Smart System und Home Connect
Erweiterte Unterstützung von Betriebssystemen

AKTIONEN

Als neuer Baustein zur Automation stehen jetzt Aktionen zur Verfügung. Diese ermöglichen die Bedienung und Konfiguration aller Objekttypen. Aktionen werden primär von Ereignissen und den neuen Ablaufplänen verwendet. So kann unkompliziert auf die individuellen Funktionen einer Instanz zugegriffen werden, wie das Verschicken von Push Nachrichten über das WebFront oder E-Mails via SMTP. Es können Ereignisse aktiviert oder alle Lampen innerhalb einer Kategorie eingeschaltet werden. Die gewohnte Individualisierbarkeit von IP-Symcon ermöglicht es auch Entwicklern eigene Aktionen zu gestalten, welche zu ihren Modulen passen.



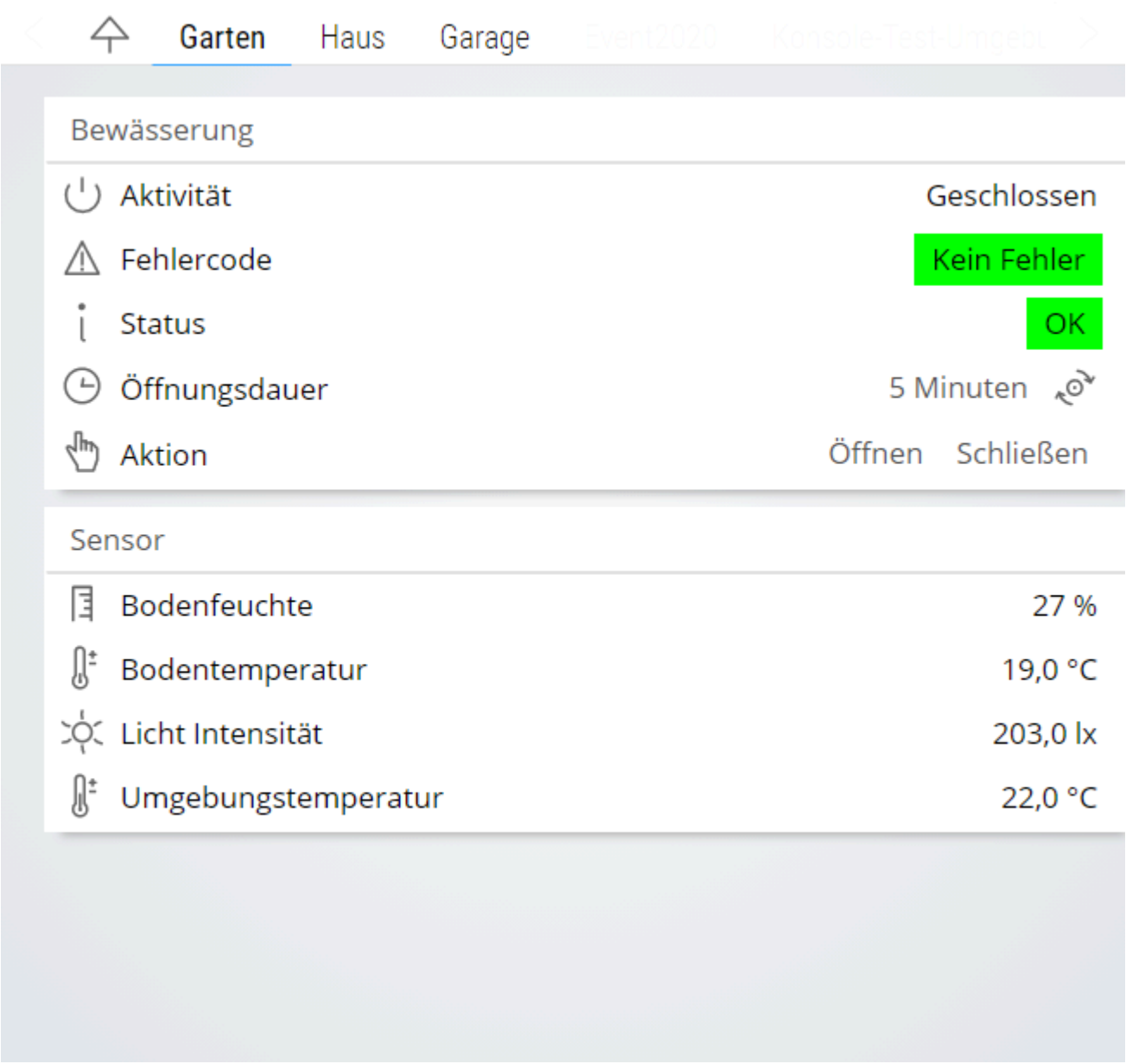
ABLAUFPLÄNE

Die Version 6.0 führt Ablaufpläne ein, eine neue Art der Automation. Hier können mehrere Aktionen hintereinander ausgeführt werden. Durch "Wenn ..., dann"-Aktionen können Aktionen sogar ineinander verschachtelt werden. Auf diese Weise kann beispielsweise ein Licht unkompliziert angeschaltet und nach 30 Sekunden wieder ausgeschaltet werden. Hilfreiche Analys-Tools finden schnell die Ursache, falls mal etwas nicht so passiert, wie erwartet.



GARDENA SMART SYSTEM UND HOME CONNECT

Mit der neuen Version kann nun auch der Garten und die Küche automatisiert werden. IP-Symcon ist nun zu GARDENA smart system bzw. Home Connect kompatibel. Hierfür wurden Module entwickelt, welche über den Module Store installiert werden können.



ERWEITERTE UNTERSTÜTZUNG VON BETRIEBSSYSTEMEN

IP-Symcon kann bereits auf diversen Betriebssystemen verwendet werden. Mit der Version 6.0 steht IP-Symcon nun zusätzlich als natives Paket für Raspberry Pi OS (64-Bit Beta) und als Docker-Version für Arm64 und Apple Silicon (M1) Geräte zur Verfügung.

WEBINAR ZUM THEMA IP-SYMCON 6.0 INKL. AKTIONEN UND ABLAUFPLAN

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.



Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die Legacy Konsole ist nicht zu IP-Symcon 6.0 mehr kompatibel und wurde entfernt. Wir empfehlen den Wechsel auf unsere Pro Konsole.
Das WebFront Retro/Retro Mobile ist zu IP-Symcon 6.0 mehr kompatibel und wurde entfernt. Zur Verwendung über die mobilen Apps werden diese in der Version 5.5 oder neuer benötigt. Zur Verwendung mit IPSStudio wird dies in der Version 5.0 oder neuer benötigt.
Zur Verwendung mit dem AIO CREATOR NEO von mediola wird dies in der Version vom 2021-07-06 oder neuer benötigt.
Zur Verwendung mit IPSRCommand (Download) wird die Version 1.05 oder neuer benötigt.
Werden Ereignisse per Skript erstellt, so muss diesen per IPS_SetEventAction eine Aktion zugewiesen werden, damit sie fehlerfrei auslösen können
WebFronts ohne Kennwort sind standardmäßig nicht mehr von extern nutzbar. Ein WebFront ohne Kennwort muss ab IP-Symcon 6.0 explizit im „Sicherheit“ Bereich der WebFront Instanz dafür freigeschaltet werden.
Sys_GetURLContent hat bisher das SSL Zertifikat nicht korrekt überprüft. Dies wurde korrigiert.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP auf 7.4.20 aktualisiert
Neu: Der Ablaufplan wurde als neuer Typ der Automation eingeführt
Neu: ARM64 (aarch64) Version für Raspberry Pi und Docker (inkl. Apple M1)
Neu: RTSPS und HTTPS wird bei Streams unterstützt (gültige Zertifikate werden benötigt!)
Neu: max_execution_time in PHP auf 0 gesetzt, weil es Ursache für viele Abstürze war.
Neu: set_time_limit darf seit der 6.0 nicht mehr aufgerufen werden. Einige PHP-Module und Skript müssen angepasst werden, sodass diese die Funktion nur auf Versionen kleiner 6.0 aufgerufen wird.
Neu: Windows Daten Pfad kann nach der Installation verändert werden
Neu: Notification Control verbessert, sodass es die neuen Möglichkeiten der Liste nutzt
Neu: Hinweise im Instanz-Debug von I/O, wenn Demo-Version abgelaufen ist
Neu: WebFront wird ebenfalls deaktiviert, sobald die Demo-Version abläuft
Neu: Einige der Kern Instanzen (z.B. Archiv) sollten nur einmal erstellt werden dürfen. Wir prüfen dies nun beim Erstellen.
Neu: Intern werden alle Netzwerkzugriffe über cURL abgebildet (SSL-Zertifikate werden jetzt überall konsistent und korrekt geprüft und es gibt viel bessere Fehlermeldungen bei z.B. Timeouts oder Zertifikatsfehlern)
Neu: Image Grabber unterstützt jetzt auch Digest Authentication
Neu: VerifyPeer/VerifyHost kann im HTTP Client, POP3, IMAP, SMTP Modul und HomeMatic Socket aktiviert/deaktiviert werden
Neu: Docker Builds werden jetzt mit einem (Docker) Suffix markiert, da diese in Zukunft für mehrere Plattformen verfügbar sein können
Neu: User-Agent wird in allen internen Netzwerkzugriffen und allen PHP Netzwerkzugriffen auf IP-Symcon/Version gesetzt
Neu: Legacy Konsole/Dashboard wird (da inkompatibel) nicht mehr ausgeliefert (Reduziert das Setup um ~5MB)

Neu: Verbindungen mit Legacy Konsole/Dashboard werfen eine aussagekräftige Fehlermeldung
Neu: Neue Profil Millivolt
Neu: Bessere Absturz-Berichte werden gesendet
Neu: Google Assistant Request Limit erhöht sich automatisch, wenn viele Variablen eingerichtet sind
Fix: Absturz bei Fehlerhaften Daten in IPS_SetConfiguration korrigiert
Fix: DNS-SD: IPv4 Adressen landen nicht mehr im IPv6 Array
Fix: IP-Symcon bricht Startvorgang ab, sofern Port 3777 nicht verfügbar ist
Fix: In seltenen Fällen konnte die Konsole den Snapshot nicht laden wenn der interne Timestamp auf 0 stand
Fix: Kleine Optimierung beim Ausführen von Text-Skripten (+ im Log wird die Größe nach der erfolgreichen Ausführung korrekt angezeigt)
Fix: Sys_GetMemeryInfo funktioniert unter Linux mit >4 GB RAM
Fix: WebHooks liefern statt HTTP/1.0 200 X ein HTTP/1.1 200 OK, sofern keine andere Status-Zeile per PHP definiert wurde (HTTP/1.0 200 X ist laut HTTP Specs vollkommen valide, aber z.B. Loxone scheint damit aktuell Probleme zu haben)
Fix: Notification Control verursacht keine Fehler mehr, wenn kein Gerät ausgewählt ist
Fix: PHP-Fehler werden als Rohtext dargestellt und nicht mehr als HTML
Fix: Wochenplanaktionen mit schwarzer Farbe können gelöscht werden
Fix: Graphen: Nachkommastellen vom Profil werden beachtet sofern möglich
Fix: Graphen: Am 01. eines kurzen Monats werden Graphen wieder korrekt gerendert
Fix: Icon-Namen aller Objekte dürfen keine Leerzeichen enthalten. Dies wird korrekt validiert.
Fix: Das Zurücksetzen von lokalen Änderungen im Module Control konnte in sehr seltenen Fällen fehlschlagen
Fix: Das Aufräumen der Logdateien konnte bei sehr weniger Speicherplatz fehlschlagen, sodass das Logging deaktiviert wurde (SymBox)
Fix: Der Update Kanal unter Ubuntu/Rasperry Pi wird korrekt erkannt, wenn https verwendet wird
Fix: TextParser: Genutzte Variablen werden bei Referenzen einbezogen
Fix: Bei niedrigem Speicherplatz wurden ggf. zu viele Logfiles gelöscht, wodurch das Aufräumen
Fix: XDebug Extension wird korrekt als Zend-Extension geladen (Windows)
Fix: VoIP funktioniert auf der SymBox wieder korrekt (Nur SymBox)
Fix: Neue Installationen mit dem Windows-Installer verhinderten das korrekte Einrichten von Lizenzen und fielen in den Demo-Modus zurück (Nur Windows)
Fix: $_IPS['TRIGGER'] Systemvariable fehlte seit dem 6.0 Release
Fix: Suche/Ersetzt in Dateien funktioniert weiterhin korrekt, wenn eine Skriptdatei fehlt
Fix: Abarbeitung von EventLimit war um eins versetzt (seit 4.0)
Fix: Absturz korrigiert, sofern beim Beenden das Schreiben der Settings fehlgeschlagen ist
Fix: Absturz im WS Client korrigiert, welcher bei ständigem Verbinden/Trennen auftreten konnte
Fix: Absturz/Speicherleck bei RTSP Streams wenn diese oft und schnell im WebFront hin und her geschalten wurden
Fix: Speicherleck in Sys_GetNetworkInfo unter Windows korrigiert
Fix: Link zum alten Community-Dashboard angepasst

ARCHIV

Neu: Spezialschalter „ArchiveWatch“, der einige Debug-Ausgaben vom Archiv aktiviert
Fix: Bei AC_GetAggregatedValues konnte manchmal die Duration falsch zurückgegeben werden
Fix: Werte können mit . und , als Dezimaltrennzeichen hinzugefügt werden

Fix: Beim Reaggregieren konnten ungültige UTF8 Zeichen an die Konsole gesendet werden, die dadurch neu geladen hat

AKTIONEN

Neu: Aktionen wurden eingeführt und ersetzen die Muster aus Ereignissen
Neu: Aktion um ein Ereignis zu aktivieren/deaktivieren
Neu: Aktionen um die Rechenarten (Addieren, Subtrahieren, Multiplizieren, Dividieren) mit dem Wert einer anderen Variablen zu verwenden
Neu: Aktion für Gruppenschaltung aller Variablem mit einem definierten Profil innerhalb einer Kategorie
Neu: Aktion um eine Push-Nachricht via WebFront zu verschicken
Neu: Aktionen um Variable auf den Wert einer anderen Variablen zu setzen oder schalten
Neu: Aktion um eine E-Mail via SMTP-Instanz zu verschicken
Neu: Neue Aktion zum Warten mit einer Wartezeit, welche per Variable festgelegt wird
Neu: Neue Aktionen zum Schalten mit einem Invertierten/Multiplizierten/Dividierten Wert
Neu: Neue Aktion zum Konkatenieren von String-Werten
Neu: Aktion um aggregierte Werte via Archiv abzufragen, insbesondere auch fließend die letzten 24 Stunden und dergleichen
Neu: Die Aktion um Instanzfunktionen auszuführen ermöglicht es nun, den Rückgabewert in eine Variable schreiben zu lassen
Neu: Wenn-Dann Aktion, welche ein Ereignis als Bedingung hat
Neu: Aktion für „Wenn …, dann“ mit mehrfachen Bedingungen
Neu: Neue Aktion „Bedingung“, um Ablaufplan bei nicht-erfüllung abzubrechen

GERÄTE

Neu: MQTT Client: Die ClientID kann angepasst werden
Neu: MQTT: Retained Messages können eingesehen und gelöscht werden
Neu: MQTT: Retained Messages werden nach 7 Tagen (Einstellbar) aufgeräumt ohne Update
Neu: MQTT: Retained Messages werden verzögert in die Persistenz übertragen, um die Performance zu verbessern
Neu: MQTT Geräte Instanzen unterstützen JSON, welches automatisch in Variablen/Instanzen zerlegt wird
Neu: ModBus: Im ModBus Gateway können ganze Blocke definiert werden, welche am Stück ausgelesen werden sollen. Dadurch kann die Performance wesentlich verbessert werden. Die Timer in den jeweiligen Instanzen sollten dann deaktiviert werden!
Neu: ModBus: Unterstützung for den Datentyp: String
Neu: ModBus: Konfigurator, welcher eine Übersicht aller eingerichteten ModBus Instanzen anzeigt (Vorbereitung für einen Export/Import zur einfachen Konfiguration von ModBus Geräten)
Neu: KNX Discovery erkennt IP-Gateways anhand der gemerkten MAC Adresse, sofern die IP-Adresse sich geändert hat
Neu: KNX: „Status emulieren“ Eigenschaft
Neu: KNX: Module von EIB konsistent nach KNX (Legacy) umbenannt
Neu: KNX: Schlagwörter zum Erkennen von Status-GAs im Konfigurator sind jetzt konfigurierbar
Neu: KNX: XML-Export ist im Konfigurator standardmäßig ausgewählt
Neu: KNX: Sofern der OPC-Export ausgewählt wird, gibt es eine Warnmeldung, dass der XML-Export bevorzugt genutzt werden sollte
Neu: EnOcean: Unterstützung für Rauchmelder D2-14-30

Neu: EnOcean: Unterstützung für hochauflösende Ansteuerung von Eltako FRGBW71L
Neu: EnOcean: Unterstützung der Konfiguration von D2-01-* Geräten für Energie-/Leistungsmessung
Neu: JSON Decoder Modul zum leichten Einlesen von JSON Dateien über den HTTP Client
Neu: Eltako FABH130
Neu: LCN Gateway steht standardmäßig auf 200 Dimschritte
Neu: Z-Wave: Expertenoption zum „Sperren der Variablen“ um das Erstellen von falschen Variablen zu verhindern, sofern Sensoren fehlerhafte Daten senden.
Neu: Image Grabber/HTTP Client: Auswahl zwischen Basic/Digest Authentifizierung
Neu: Eltako: FD62NP-230V, FD62NPN-230V, FL62-230V, FL62NP-230V, FR62-230V, FR62NP-230V werden unterstützt
Neu: KNX: Wenn Fehler beim Verbindungsaufbau auftreten zeigen wir eine bessere Fehlermeldung
Neu: Eltako: FJ62/12-36V DC, FJ62NP-230V werden unterstützt
Fix: Eltako FAFT60 Batterie Anzeige wurde nicht korrekt in Volt umgerechnet
Fix: KNX: „Status emulieren“ ist jetzt auch unter Windows verfügbar
Fix: EnOcean: Konfigurator zeigt das generische „Geräte hinzufügen“ nicht mehr als neues Gerät an
Fix: ModBus: Konfigurator zeigt das generische „Geräte hinzufügen“ nicht mehr als neues Gerät an
Fix: Shutter Control: Bessere Beschreibung der Zeitangaben
Fix: MQTT: Server Konfiguration konnte manchmal wegen UTF-8 Fehler nicht geöffnet werden
Fix: HomeMatic Socket blockiert beim Verbinden nicht mehr die MessageSink
Fix: ModBus Gateway intern umgebaut, sodass der Datenfluss für PHP-Module korrekt funktioniert
Fix: OneWire Test-Center ist konsistent verfügbar
Fix: Text Parser / JSON Decoder erstellen standardmäßig einen HTTP Client bei Neu-Erstellung
Fix: MQTT Client Configurator erstellt Client Devices (und nicht mehr Server Devices)
Fix: ALL4000 konnte nicht eingerichtet werden
Fix: KNX: Icon für DPT 14.068 korrigiert
Fix: HomeMatic: LOW_BAT wird korrekt das Battery Profil zugeordnet
Fix: EnOcean: General Profile Telegramme werfen keine Fehlermeldung mehr
Fix: JSON Decoder: Seltene Fehlermeldung beim Schreiben Werten korrigiert
Fix: EnOcean: D2-01-* Typen laut aktueller EEP Beschreibung angepasst
Fix: KNX: Erkenne Status-GA durchsucht nun alle Keywords einzeln

I/OS

Neu: Neuer Datenfluss für UDP Socket, um Unicast/Broadcast gleichzeitig über einen I/O abbilden zu können
Neu: Server Socket: Zertifizierungsstelle kann optional angegeben werden
Neu: SSE Client kann eigene, in der Konfiguration definierte, Header mitsenden
Neu: SSE Client versendet ein eigenes Datenpaket, welches das SSE Protokoll direkt geparst ausgibt
Neu: Serial Port erkennt Symlinks welche auf ttyUSB* gehen (Insbesondere nützlich für udev Regeln)

KONSOLE

Neu: Initiales Kopieren/Verlinken/Auslöserauswahl springt zur Position des Zielobjektes
Neu: Spalte „Geändert“ im Objektbaum
Neu: Bearbeiten-Dialog im Archiv zeigt nun die gewählte Variable an

Neu: Verwendete Bibliotheken aktualisiert
Neu: Schönere Fehlermeldungen, wenn der Login aus unbekannten Gründen fehlschlagt
Neu: Beim Umbenennen wird der aktuelle Name ausgewählt, sodass er beim Eingeben eines neuen Namens direkt überschrieben wird
Neu: Der Variable-bearbeiten-Dialog unterstützt nun auch ein Schalten von Variablen
Neu: Werden Variablen geloggt, kann dies nun am dazugehörigen Icon im Objektbaum erkannt werden
Neu: Anzeige von einer roten Warnmeldung wenn es starke Verzögerungen auf dem Rückkanal gibt
Neu: Schnellfilter im Objektbaum filtert auch die Beschreibungsspalte, falls diese sichtbar ist
Neu: Die Konsole überprüft nun Browserfeatures und nicht die Browserversion um ggfs. eine Warnung anzuzeigen. Dies verhindert insbesondere die Warnung bei aktuellen Edge-Browsern.
Neu: Im Skripteditor kann nun via Kontextmenü auf die Zwischenablage zugreifen (Kopieren/Ausschneiden/Einfügen)
Neu: Der Module Store ist nicht mehr Beta
Neu: Hat eine via Konfigurator gelöschte Instanz physikalisch untergeordnete Instanzen, so wird angeboten auch diese zu löschen
Neu: Mehrere Zeilen können im Ablaufplan ausgewählt und beispielsweise gleichzeitig dupliziert werden
Neu: Enter und Entf können als Tastaturkürzel im Ablaufplan verwendet werden
Neu: Im Info-Dialog wird beim Kopieren die Architektur einbezogen
Fix: Bei Bedingungen geht beim Wechsel von Variablen mit Assoziationen der Wert nicht verloren
Fix: Die Konsole generiert beim Speichern eine neue Skriptdatei, sofern die alte gelöscht wurde
Fix: Bearbeite Wochenpläne wie vorgesehen, wenn nur eine Wochenplanaktion gesetzt ist und diese angeklickt wird
Fix: Nicht-Gerät-Instanzen können bei der Erstellung nicht (unsichtbar) in eine Kategorie geschoben werden
Fix: Gebe eine sprechende Fehlermeldung zurück, wenn im Archiv versucht wird ein Wert mit einem neunstelligen Zeitstempel hinzuzufügen
Fix: Kaputte Module im Module Store konnten unter gewissen Umständen nicht geöffnet werden. Dies ist wieder möglich
Fix: Verhindere kosmetische Fehlermeldung, welche gelegentlich auftrat, wenn eine Instanz beim Schließen das Tabs gespeichert wurde
Fix: Bei erweiterten Bedingungen mit Wochentagsregeln, erkennt die Konsole nun korrekt, wenn die Regel per Skript gelöscht wird
Fix: Verhindere Fehler, wenn der Skripteditor eine Autovervollständigung versucht, während er innerhalb eines String ist
Fix: Endlosschleife/Defekter Objektbaum bei IPS_GetLocation korrigiert, wenn ein Ereignis mit Auslöser auf eine Variable unterhalb vom selben Ereignis erstellt wurde
Fix: Defekte Module im Module Store können editiert werden
Fix: Anzeige einer Warnmeldung, wenn die Demo abgelaufen ist
Fix: Beim Aktivieren einer Lizenz über das Demo Widget wird korrekt der „Neustart“-Balken angezeigt
Fix: Module Store: Beim Erstellen von Instanzen wurden benötigte Spliiter + I/O Instanzen erstellen, eine Konfiguration war bisher jedoch nur im Nachhinein möglich. Mit diesem Update können diese Instanz direkt Konfiguriert werden, genau wie im „Instanz erstellen“ Dialog.
Fix: Behebe Fehler, welcher beim Neu laden zu „Uncaught Cannot find object with Id“ führen konnte
Fix: Update-Widget funktioniert korrekt für arm64 Versionen
Fix: Würde der Wert einer Profilassoziation überschrieben werden, erscheint eine Frage, ob dies wirklich passieren soll
Fix: Seltener Fehler behoben, welcher Emojis als Symbol in einer Liste abgeschnitten hat

Fix: In seltenen Konstellationen konnte ein Ereignis ein Icon für Erfüllung der erweiterten Bedingungen anzeigen, obwohl keine Bedingungen gesetzt wurden. Dies wurde behoben
Fix: Icons in Listen und Bäumen sind nicht mehr dragbar
Fix: Anzeige von Boolean Assoziation „true“ im Profilmanager gefixt
Fix: Erstellen von Medien (Dokumenten) über das „+“ korrigiert

PRO-KONSOLE

Neu: Links werden im Standardbrowser und nicht mehr innerhalb der Konsole geöffnet
Fix: Verhindere kosmetische Fehlermeldung, wenn beim Schließen eines Tabs mehrere Tablisten existieren
Fix: Medien können wie vorgesehen heruntergeladen werden

WEBFRONT

Neu: WebFront Retro/Retro Mobile entfernt
Neu: WebFronts ohne Kennwort sind standardmäßig nicht mehr von extern nutzbar
Neu: Wird ein Server manuell hinzugefügt, so wird geprüft, ob es sich dabei um eine IP-Symcon handelt
Fix: Nach einer versuchten (und fehlgeschlagenen) Verbindung zu einem IP-Symcon vor 5.0, ist der Verbindungsassistent weiter verwendbar
Fix: Charts in Panels werden beim Tab-Wechsel mit korrekter Zeit angezeigt
Fix: WebFront Visualization Instant hat den SecurityContext zu stark reduziert, wenn in den Standardeinstellungen die Startkategorie der mobilen App verändert wurde

SYMOS

Neu: KNX/LCN kann über SymOS Connect und ETS/LCN Pro aus der Ferne programmiert werden. Diese Funktion befindet sich in einer internen Beta-Version und eine Freischaltung dafür kann über support@symcon.de angefragt werden.
Fix: Backups mit der Größe > 2,1GB sind jetzt möglich
Fix: Beim Booten der SymBox wurde der „IP-Symcon startet“ Dialog manchmal übersprungen

PHP-SDK

Neu: SDK für Aktionen
Neu: Neues Element SelectAction
Neu: Neues Element ScriptEditor
Neu: Neues Element ColumnLayout
Neu: Inhalt von RowLayout und ColumnLayout kann dynamisch angepasst werden
Neu: Listen können durch den Benutzer bei gesetztem Parameter „changeOrder“ umsortiert werden
Neu: Parameter für SelectVariable um erlaubte Variablen einzuschränken
Neu: on-Skripte können nun auch als Array eingetragen werden
Neu: Image unterstützt nun width und onClick
Neu: Parameter ‚loadValuesFromConfiguration‘ für List und Tree
Neu: Configurator versucht die Spaltenbreite bei Knappheit besser zu verteilen und zeigt ggf. einen Scrollbalken
Neu: List und Tree unterstützen nun „enabled“
Neu: Parameter „moreDigits“ für NumberSpinner

Neu: Werden bei einem NumberSpinner ohne „moreDigits“ mehr Nachkommastellen als erlaubt eingegeben, wird dies als Fehler angezeigt
Fix: Dynamisch gesetzte Spaltennamen von Listen werden übersetzt
Fix: Werden dynamisch neue Werte für ein Element Tree gesetzt, wird die aktuelle Sortierungsspalte beibehalten
Fix: Ist beim Configurator visible auf false gesetzt, so wird das Element nun auch korrekt versteckt
Fix: Wird der Suffix eines NumberSpinner dynamisch angepasst, so wird der neue Wert wie vorgesehen übersetzt

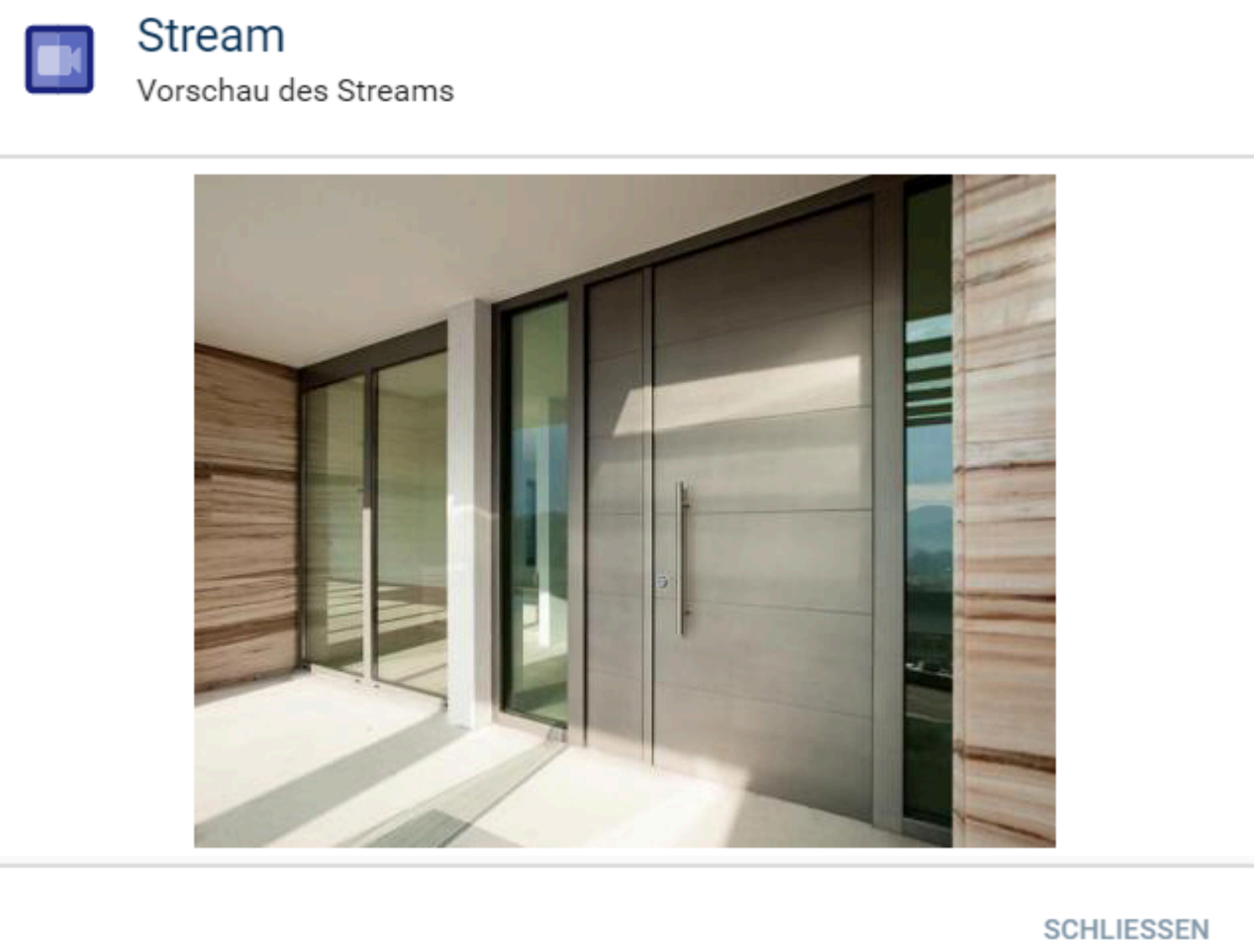
V5.4->V5.5 (Q4/2020)

WESENTLICHE ÄNDERUNGEN

Verbesserte Streams
Mobile Apps überarbeitet
Optimiertes Archiv
Neue Geräteanbindungen

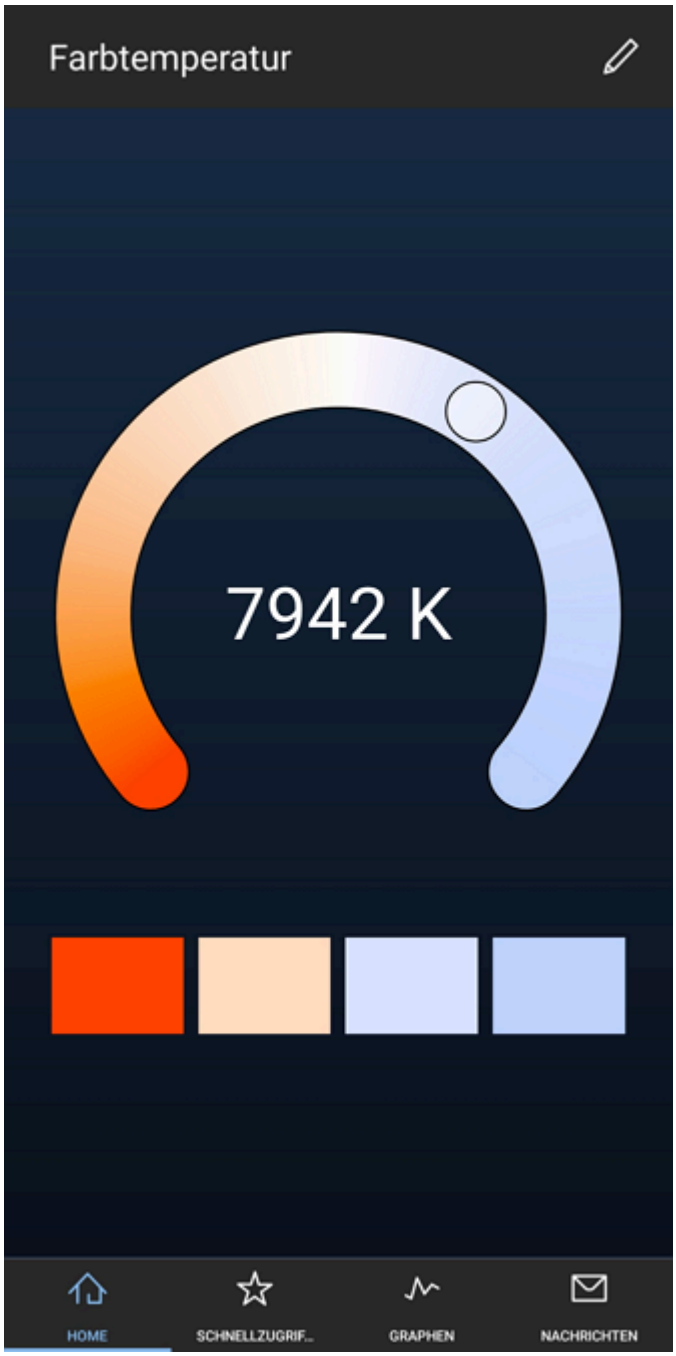
VERBESSERTE STREAMS

Die Behandlung von Streams wurde optimiert. Egal ob ein Stream als RTSP oder MJPEG vorliegt, dieser ist nun nahezu live in der Visualisierung verfügbar, ganz gleich ob lokal oder per Fernzugriff. Auch können Streams nun unter Android, iOS, WebFront und der Verwaltungskonsole betrachtet werden.



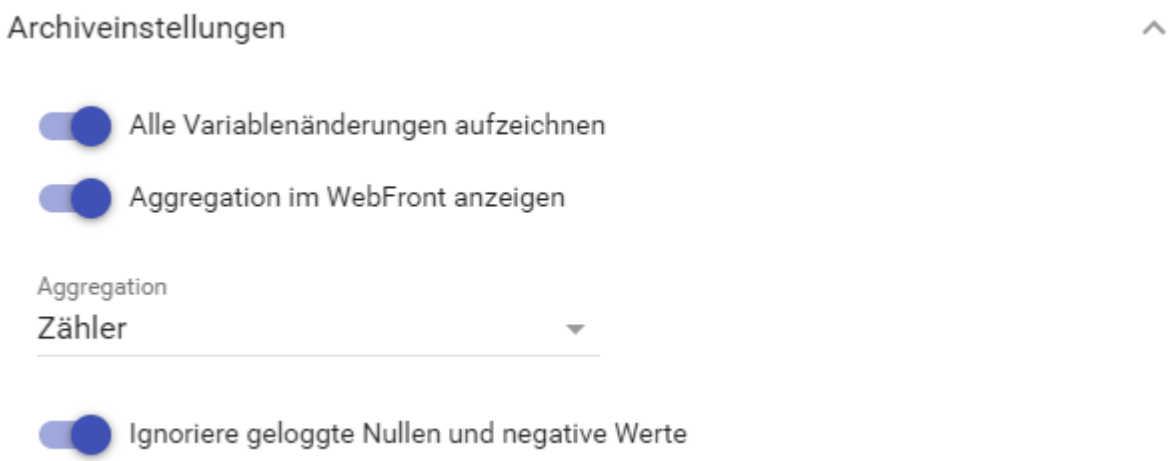
MOBILE APPS ÜBERARBEITET

Die Darstellung des Objektbaums in den Apps und im WebFront wurde gleichgezogen. Dadurch ist nun in den Apps eine tiefere Verschachtelung darstellbar. Um direkt in die richtige Visualisierung zu kommen, kann dies nun in der App voreingestellt werden. Noch mehr Aufwand spart man über die Schnellzugriffsleiste. Hier kann komfortabel auf Graphen, Favoriten und Nachrichten zugegriffen werden. Für die Einstellung der Farbtemperatur von Lampen gibt es nun eine optimierte Darstellung. Die oben beschriebenen Optimierungen der Streams gelten selbstverständlich auch für die Apps.



OPTIMIERTES ARCHIV

Das Archiv wurde grundlegend überarbeitet. Dabei wurde es unter der Haube komplett optimiert, so dass nun beispielsweise Aggregationen deutlich schneller laufen. Gleichzeitig wurde der Speicherverbrauch drastisch gesenkt, da die Aggregationsdaten nun nicht mehr im Speicher gehalten werden. Zusätzlich ist es als neues Feature nun möglich bei einem Zähler Nullen und negative Werte zu ignorieren, welche in manchen Fällen bei Fehlfunktionen gesendet werden.



NEUE GERÄTEANBINDUNGEN

Mit der neuen Version wurden diverse neue Geräte und Schnittstellen implementiert. So kann IP-Symcon nun auch nativ als MQTT Client verwendet werden und für EnOcean wurden zwölf zusätzliche EEPs eingebunden. Darüber hinaus kann CMI nun auch mit UVR65, UVR67, UVR 610 und CAN-EZ3 verwendet werden. Dazu kommen noch einige Optimierungen für KNX, S7 und eKey.

MIGRATION

Bitte erstellen Sie sicherheitshalber ein Backup, sollten wider erwarten Probleme bei der Aktualisierung auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP 7.4.11 (Bitte beachten: https://www.php.net/manual/de/migration74.deprecated.php)
Neu: RTSP Streams werden nach MJPEG umgewandelt
- Bild wird sehr zeitnah und fast live angezeigt
- Nun auch Unterstützung für Safari und iOS verfügbar
- Benötigt die Apps in Version 5.5
- Auflösung, Qualität und Frame-Rate wird beim Connect-Dienst reduziert, um Traffic zu sparen

Neu: MJPEG Streams werden über IP-Symcon geroutet
- Direkt über den Connect Dienst verfügbar

Basis Authentifizierung wird über IP-Symcon erledigt und ist somit uneingeschränkt nutzbar
Benötigt die Apps in Version 5.5
Neu: Tray Icon für Windows komplett erneuert mit High DPI Support
Neu: Schalter um "Verschachtelung" zu aktivieren ist standardmäßig an und nur noch auf Legacy-Systemem verfügbar
Neu: Schalter, um Enums anstatt CheckBox'en in den Apps zu nutzen (benötigt Apps in der Version 5.5)
Neu: Connect ist für Neuinstallationen automatisch aktiviert
Neu: Wochenplan kann maximal 32 Aktionen (vorher 8) haben
Neu: Autostart im WebFront unterstützt die CIDR Notation
Neu: IP-Autostart mit CIDR 0.0.0.0/0 wird auch über Connect-Dienst ausgewertet
Neu: Mac Version wird korrekt von Apple Notarisiert, sodass Gatekeeper glücklich ist
Neu: WebFront ist jetzt vom Typ Visualisierung und nicht mehr Konfigurator
Neu: WebServer ist sofort beim Start verfügbar (und nicht erst beim Kernel Betriebsbereit)
Neu: Durch besseres Caching sollte die Konsole ca. 1 - 3 Sekunden schneller Starten
Neu: GetValueFormattedEx, welches einen weiteren Parameter akzeptiert. (Wert, der zum Formatieren genutzt werden soll)
Neu: Web Server SSL Konfiguration geschieht jetzt über das Konfigurationsformular (Zertifikate im cert Ordner werden migriert und gelöscht)
Neu: Web Server nutzt jetzt SSL 1.2 (SSL Labs attestiert IP-Symcon damit wieder die Note A)
Neu: Settings haben eine Schema-Version um zukünftig inkompatible Downgrades zu verhindern
Neu: Instanzen werden beim Starten in der Reihenfolge Core, Visualization, Discovery, Configurator, Device, Splitter, I/O erstellt, wodurch Probleme wie z.B. Verbindungsprobleme bei MQTT gelöst werden.
Neu: WebFront Visualisierung: Hinweis, dass die Standard Graphen Optionen nur für das WebFront gelten
Neu: OpCache ist jetzt standardmäßig aktiv - kann jedoch in den Spezialschaltern wieder deaktiviert werden
Neu: max_memory_limit darf nun auf maximal 128MB erhöht werden
Neu: CC_GetRequestLimitCount und NC_GetRequestLimitCount mit dem die Anzahl der Anfragen, welche gegen das tägliche Limit zählen, abgefragt werden kann
Neu: NotificationWatch Spezialschalter welcher alle Anfragen an den Push-Nachrichten Server ins Log ausgibt
Fix: Proxy Stream URLs filtern Benutzername/Passwort aus der Adresse im Logfile
Fix: Lock Profil nutzt das adaptive Icon
Fix: Duplizieren von Variable ohne Logging korrigiert
Fix: Demo-Überprüfung unternimmt nun 5 Versuche á 10 Sekunden (manchmal war nach einem Neustart kein Internet verfügbar und die Demo startete ohne korrekte Endzeit und lief nur 6 Stunden)
Fix: Duplizieren von Instanzen konnte Fehlschlagen, wenn die Instanzen unterschiedlich verbunden waren und dabei eine Instanz keine Verbindung hatte
Fix: Attribute werden in der Persistenz direkt beim Register initialisiert
Fix: In seltenen Fällen konnten Fehler in der Konsole erscheinen, wenn Objekte während des Öffnens gelöscht wurden
Fix: Sys_GetNetworkInfo liefert nur aktive Netzwerk-Adapter
Fix: Bei nicht mehr existierenden Variablen wird das Logging auf Inaktiv gestellt
Fix: PRCE JIT unter MacOS korrigiert
Fix: Nachrichten werden in korrekter Reihenfolge vor KR_READY verarbeitet, sodass z.B. Siemens Gateway die IS_ACTIVE Nachricht korrekt erhalten und verarbeiten
Fix: User-Agent wurde bei Anfragen nicht korrekt gesetzt
Fix: Löst hoffentlich Abstürze in Verbindung mit dem ServerSocket
Fix: Anzahl der CPU’s konnte auf z.B. Intel NUCs nicht korrekt ermittelt werden

Fix: MJPEG Streams, welche keine HTTP Ranges unterstützen, konnten nicht genutzt werden
Fix: Seltene Abstürze beim Beenden, wenn während des Herunterfahrens „die Glocke“ gearbeitet hat
Fix: Proxy: Im Log werden username, password, user, pw, usr, pwd Werte maskiert
Fix: Netzwerk Bibliothek auf Stand der 5.4 zurückgesetzt um ggf. Probleme mit der neuen Version auszuschließen
Fix: MessageQueueWatch Meldungen zu KL_NOTIFY (lila) befördert
Fix: Notification Control: Beim Versenden von Test-Nachrichten war „Sent!“ nicht übersetzt
Fix: Datenfluss ignoriert Pakete > 16kb sofern der ReceiveDataFilter/TransmitDataFilter gesetzt ist um Abstürze in der RegEx Engine zu umgehen. Dies wird im Debug Log der betroffenen Instanz geloggt.
Fix: Fehlerhafte JSON-RPC Requests konnten IP-Symcon zum Absturz bringen
Fix: WebServer konnte bei starker Nutzung vom IPSView Web-Client abstürzen
Fix: Abstürze unter Raspberry Pi & SymBox aufgrund von fehlerhafter PCRE/JIT Bibliothek gelöst
Fix: Endlosschleife/Absturz bei IPS_GetLocation korrigiert, wenn ein Ereignis mit Auslöser auf eine Variable unterhalb vom selben Ereignis erstellt wurde

ARCHIV

Neu: Optimierung des Archivs
Neu: Archiv nutzt jetzt intern Attribute anstatt Properties. (Die AC_* Funktionen benötigen kein IPS_ApplyChanges mehr!)
Neu: Bei Zähleraggregationen können Nullen ignoriert werden
Neu: Aggregationsdaten einer Zählervariable beginnen nun direkt beim ersten Wert, nicht erst beim ersten Delta
Neu: Wird das Logging einer Variablen erstmalig aktiviert, wird direkt der aktuelle Wert geloggt
Neu: Erlaube Migration der Daten von Integer Variablen zu Float Variablen
Neu: Erlaube parallele Aggregation mehrere Variablen
Neu: Ignoriere Nullen wurde erweitert, sodass es auch negative Werte filtert
Fix: Die Referenzen des Archivs werden bei Bedarf aufgeräumt. So sollte es nicht mehr möglich sein, dass Variablen als Referenz das Archiv angeben, dort allerdings nicht zu finden sind.
Fix: Alternative Implementation für die kontinuierliche Aktualisierung der Datengröße, behebt möglicherweise die negativen Größenangaben
Fix: Bei einer Reaggregation mit vielen CPUs konnte diese evtl. in einem Deadlock enden
Fix: Problem behoben, wenn es im DB-Ordner Daten gab, aber keinerlei Referenz auf die Variable mehr existierte
Fix: Problem beim Starten mit der letzten Version korrigiert
Fix: Statistiken von String-Variablen wurden nicht korrekt berechnet
Fix: Archiv: In seltenen Fällen konnte ein Endlosschleife entstehen und alles an RAM verbrauchen, sodass IP-Symcon abstürzt
Fix: Archiv: Seit der Optimierung aus der vorletzten Beta konnten Race-Conditions auftreten und den Dienst zum Absturz bringen
Fix: Archiv: Viele Graphen bei Systemen ohne SSD konnten das Gesamtsystem verlangsamen
Fix: Archiv: Beim Duplizieren von Variablen konnte der initiale Wert doppelt geloggt werden
Fix: AC_AddLoggedValues versucht mehrfach auf die Datei zuzugreifen, sofern diese aktuell „blockiert“ wird

GERÄTE

Neu: CMI: Unterstützung für UVR65, CAN-EZ3, UVR610, UVR67
Neu: EnOcean: EnOcean Gateway zeigt die EURID an
Neu: EnOcean: D2-15-00, D2-07-00, D2-03-00, D2-03-0A, D2-03-10, D2-A0-01
Neu: EnOcean: A5-20-06, A5-10-1E, A5-10-22, A5-10-23
Neu: EnOcean: F6-01-01 und F6-10-01
Neu: MQTT Client (inkl. Device, Splitter und Konfigurator)
Neu: MQTT: Nutzt die Verbesserung vom Server Socket besser, indem er weniger blockiert
Neu: MQTT: Nachrichten mit Retain Flag werden auch bei IP-Symcon Neustart korrekt retained
Neu: KNX Shutter unterstützt das prozentgenaue Positionieren von Lamellen
Neu: eKey: Unterstützung für Rare Net Protokoll
Neu: ALL3653 zu den unterstützen Geräte hinzugefügt
Neu: S7 unterstützt nun den Typ "String"
Neu: HomeMatic IP ist standardmäßig beim Erstellen aktiv
Neu: Bessere Fehlermeldung, wenn HomeMatic Kanal (z.B. RF) nicht verfügbar ist
Fix: Min/Max bei DPT9 realistischer gestaltet
Fix: LCN Module Konfigurator werden korrekt übersetzt
Fix: VoIP: Manchmal konnte VoIP stecken bleiben, wenn ein Anruf abgelehnt wurde
Fix: LCN sollte sich beim Start direkt verbinden, ohne das bei KR_READY eine Erneutes verbinden angestoßen wird
Fix: Die Initialisierung von KNX, FHZ, IRTrans, MQTT Client, Siemens S5/S7/Logo, WMRS200, Z-Wave wird asynchron gemacht und verzögert nicht mehr den Start
Fix: EnOcean: D2-01-xx Einzelkanal Rollladenaktoren, wie z.B. der OPUS Shutter, können jetzt korrekt angesteuert werden
Fix: EnOcean: F6-05-00 und F6-05-02 fehlten die Konfigurationsformulare
Fix: MQTT: Beim Empfang von unvollständigen Datenpaketen, konnten bereits vollständige Datenpakete verworfen werden
Fix: MQTT: Keep-Alives werden beachtet und Clients nach Timeout Disconnected (+LWT versendet)
Fix: Web Server: Beim Importieren von Zertifikaten konnten unter Windows diese nicht korrekt aufgeräumt werden
Fix: MQTT: Wenn die Session Cleanup Zeit versehentlich zu klein Eingestellt wird, werden noch aktive Sessions nicht fälschlicherweise mit aufgeräumt
Fix: MQTT: Löst hoffentlich ein Problem, wobei Geräte fälschlicherweise das LWT empfangen und dann nicht mehr geschaltet werden können
Fix: WebServer hat in einigen Fällen nach dem Import die SSL-Zertifikate nicht in die Persistenz geschrieben
Fix: Z-Wave Geräte nutzen jetzt korrekt das Battery.100 Profil
Fix: MediaPlayer warf eine Fehlermeldung, wenn ein WAC_Stop direkt nach IP-Symcon start ausgeführt wurde
Fix: SSDP Suche fand einige Geräte nicht, welche sich zu 100% an die Spezifikation gehalten haben
Fix: KNX Konfigurator: Wenn ein XML-Export viele GAs hatte, welche keine DPT Zuordnung haben, konnte der Konfigurator ewig laden und IP-Symcon zum Absturz bringen
Fix: Ausgehende Anrufe per VoIP konnte nicht korrekt abgebrochen werden
Fix: DNS-SD: Abstürze unter Linux, Ubuntu, RPi wenn gleichzeitig eine Anfrage lief und die Instanz übernommen wurde
Fix: MQTT: Abstürze bei defekten Paketen korrigiert. (Jetzt kommt eine Fehlermeldung „Consumed Packet is larger than buffer“)
Fix: MQTT: Abstürze/Unendlich viele Log-Nachrichten mit „Skipping Cleanup“ korrigiert
Fix: Siemens S7/Logo: Verbinden konnten fehlschlagen, wenn Anfragen bereits vor der erfolgreichen Verbindung per Skript angefordert wurden
Fix: confirm-Dialoge bei Z-Wave waren nicht übersetzt
Fix: confirm-Dialog beim Zurücksetzen vom Z-Wave Gateway hinzugefügt
Fix: Neue TCM515 Revision sendet neue Bestätigungsmeldungen, welche nicht korrekt ausgewertet wurden

Fix: LCN Shutter hat den Status nicht korrekt nachgeführt, wenn die Invertierung aktiv war
Fix: Alte EIB RGB/RGBW Instanzen können wieder korrekt Rückmeldungen empfangen
Fix: EEP A5-14-08 wird jetzt korrekt geladen und ist verfügbar

I/OS

Neu: I/O für Server Side Event (SSE Socket)
Neu: WebSocket Client (inkl. SSL/TLS) [Datenfluss wie bei ClientSocket]
Neu: WWW Reader zu HTTP Client umbenannt
Neu: Server Socket unterstützt SSL/TLS
Neu: ServerSocket bietet die Option zum ignorieren von Verbindungsabbrüchen
Neu: ServerSocket kann 5 Verbindungen gleichzeitig verarbeiten
Neu: Client Socket hat SSL/TLS Unterstützung
Fix: Probleme beim Senden von sehr vielen Paketen über die I/Os (insbesondere Server Socket bei MQTT) gelöst
Fix: ServerSocket hat Feld für Zertifikate angezeigt, obwohl SSL deaktiviert war
Fix: Beim gleichzeitigem Öffnen/Schließen von I/Os konnte IP-Symcon abstürzen

KONSOLE

Neu: Medienkategorie entfernt - Medien ohne übergeordnetes Objekt werden nun direkt auf der Hauptebene des Objektbaumes angezeigt
Neu: Speichere die letzte Spaltenauswahl in Objektbäumen für zukünftige Objektbäume
Neu: Duplikation eines Objektes in einer Spezialkategorie geschieht direkt vor Ort ohne Positionsauswahl
Neu: Spezialkategorie "Visualisierungen" beinhaltet WebFronts
Neu: Konsole wartet bis der Kernel Betriebsbereit ist und zeigt eine Meldung, dass IP-Symcon noch startet
Neu: Konsole zeigt an, wenn der Server heruntergefahren wird und startet die Konsole neu, sobald IP-Symcon wieder verfügbar ist
Neu: Per Doppelklick auf einen Stream wird eine passende Vorschau gegeben
Neu: Permanente Benachrichtigung, sofern die Lizenz/Spezialschalter geändert wurde; der Neustart jedoch nicht durchgeführt wurde
Neu: Unbehandelte Fehler werden sofort angezeigt, damit diese gemeldet werden können
Neu: Im Archive Control werden beim Auswählen einer nicht mehr existierenden Variablen nicht verwendbare Aktionen ausgegraut
Neu: Im Archive Control kann nun auch nach der Statusspalte sortiert werden
Neu: Werden Variablen mit geloggten Werten gelöscht, so wird gefragt ob die geloggten Werte ebenfalls gelöscht werden sollen
Neu: Inhalt des Variableneditors (Doppelklick auf Variablenwert) kann markiert und somit auch kopiert werden
Neu: Grundlegende Eigenschaften von IPS-View-Formularen können nun auch in der Konsole angepasst werden
Fix: Verwendete Bibliotheken aktualisiert
Fix: Nach dem Schließen von Dialogen wird der Fokus wie vorgesehen wieder auf den Objektbaum geschoben, damit Tastenkürzel direkt weiter verwendet werden können
Fix: Spezialkategorien unterstützen kein "ObjektID kopieren" mehr
Fix: Instanzen, die über einen Konfigurator erstellt wurden, können nun wie vorgesehen andere valide Verbindungen finden obwohl sie sich initial anders verbunden haben
Fix: Mögliches Problem gefixt, wodurch der gespeicherte Zustand der Konsole beim Start verloren gehen konnte

Fix: Shortcuts im Skripteditor funktionieren nun auch unter Mac und verwenden auch Cmd statt Strg
Fix: Verifkation in einer ValidationTextBox wird nicht mehr ein Zeichen zu spät angewandt
Fix: PopupAlerts konnten zu groß werden
Fix: Strings bei "Bei bestimmter Wert" werden nicht mehr mit 'null' initialisiert
Fix: I/Os können in der physikalischen Ansicht korrekt gelöscht werden
Fix: Wird für eine neue Variable der Aggregationstyp Zähler beim Aktivieren des Loggings ausgewählt, wird dies wie vorgesehen übernommen.
Fix: Das Schnellausführungswidget wird nicht mehr leer angezeigt, wenn während dessen Ladevorgang auf ein Tab gewechselt wird
Fix: Bei fehlerhaften Debug-Meldungen werden die problematischen Rohdaten angezeigt
Fix: Im ArchiveControl wird nicht mehr versucht nicht-Variablen-Objekte mit gelöschten Variablen gleicher ID gleichzusetzen
Fix: Bei einem Fehler beim Speichern im Profilmanager, wird dieser auch ausgegeben
Fix: Verhindere Fehler beim Öffnen eines ausgelösten Ereignisses dessen Auslöser nicht mehr existiert
Fix: Sonderzeichen in Variablenaktionen mit Strings werden korrekt beim Öffnen des Ereignisses wieder dargestellt
Fix: Instanzaktionen mit Stringparametern können nun auch wie vorgesehen mit Sonderzeichen verwendet werden
Fix: Autovervollständigung im Skripteditor um mehr Konstanten und Systemvariablen erweitert
Fix: Werte Auslösungwert in ausgelösten Ereignissen bei "leerem" Wert korrekt aus
Fix: Drag & Drop im Objektbaum leicht optimiert
Fix: Sonderzeichen im DebugLog werden nun wie vorgesehen ausgewertet
Fix: Der Lizenz-Benutzername wird nun einheitlich als solcher bezeichnet
Fix: Das Schnellausführungswidget zeigt die Objektinformation hinter der ID wie vorgesehen nur mit dem Namen ohne Pfad an
Fix: Beim Öffnen eines Moduls nach einem Update konnte ein Fehler kommen. Dieser wurde behoben
Fix: Erkenne bei Tastenkombinationen im Objektbaum wie vorgesehen, wenn kein Objekt ausgewählt ist und tue entsprechend nichts
Fix: NumberSpinner setzt jetzt die minimum/digits Parameter korrekt um
Fix: Eingabe von Datum/Uhrzeit im Safari korrigiert bei der manuellen Archiv-Daten Eingabe
Fix: Sporadische Fehlermeldungen beim Editieren vom Wochenplan korrigiert
Fix: Wird im Such- oder Filterfeld des Objektbaums etwas per Strg + V eingefügt, wird nicht zusätzlich ein möglicherweise kopiertes Objekt im Objektbaum eingefügt
Fix: Float-Variablen mit Assoziationsprofilen werden nun wie vorgesehen bei erweiterten Bedingungen eines Ereignisses gehändelt
Fix: Die Werte von Variablen mit Prozentprofilen werden wie vorgesehen bei erweiterten Bedingungen dargestellt
Fix: Drücken von STRG+V im Suchfeld fügt keine neuen Instanzen im Objektbaum hinzu
Fix: In seltenen Konstellationen, könnten Textfelder mit $-Zeichen in der Instanzkonfiguration zu Fehlern führen. Dies wurde behoben
Fix: Profile konnten nach einem Neustart in der Pro Konsole fehlen
Fix: Im Snapshot konnte das compatibility Feld doppelt vorhanden sein
Fix: Bei erweiterten Bedingungen mit Wochentagsregeln, erkennt die Konsole nun korrekt, wenn die Regel per Skript gelöscht wird
Fix: Wird beim Hinzufügen einer ObjektID im Skripteditor die Auswahl per Enter bestätigt resultiert dies nicht mehr in einem Zeilenumbruch
Fix: Die ausgewählte Instanz bei einem SelectModule-Feld wird nun wie erwartet angezeigt
Fix: Verhindere kosmetischen Fehler mit _isPrimaryDialog
Fix: Defekte Module im Module Store können editiert werden

PRO-KONSOLE

Neu: "Letzte Verbindung" wird kontinuierlich aktualisiert solange ein Fenster offen bleibt
Neu: Filter-Möglichkeit für Server
Neu: Bessere Fehlermeldung, wenn selbst-signierte Zertifikate verwendet werden
Fix: Beim Umdocken einer Instanzkonfiguration konnten Werte in verschachtelten Elementen (z.B. RowLayout) verloren gehen. Dies wurde behoben
Fix: Bei "Erneut suchen" von Servern wird die Verbindung zum Netzwerk komplett neu aufgebaut. So können Server gefunden werden, obwohl man sich erst nach dem Öffnen des Verbindungsassistenten mit dem Netzwerk verbunden hat
Fix: Schlägt das Laden von Lizenzen aus der Lizenzverwaltung fehl, z.B. weil kein Internet vorhanden ist, so wird der Benutzer abgemeldet und bleibt nicht bei "Wird eingeloggt..." stehen
Fix: Unter Linux wird der in der sources.list gewählte Update Kanal korrekt erkannt
Fix: Unter Linux wird geprüft, ob der update-manager verfügbar ist
Fix: Sind Tabs mit gleichem Namen in verschiedenen Tablisten offen, so werden sie namentlich differenziert als wären sie in der selben Liste
Fix: Verwendete Bibliotheken aktualisiert
Fix: Login zur Lizenzverwaltung funktioniert nach Überarbeitung des persönlichen Bereichs wieder
Fix: Neue Benutzer oder Benutzer mit geändertem Benutzernamen können sich wieder via E-Mail anmelden

WEBFRONT

Neu: Unterstützung für IPSView WebViews (Als Element und direkt als Medien-Objekt)
Neu: Unterstützung für neue RTSP Streams (werden direkt im Bild und nicht als iFrame dargestellt)
Neu: Unterstützung für MJPEG Streams, welche direkt über IP-Symcon / Symcon Connect abgerufen werden
Neu: Streams unterstützen im Inhaltswechsler die Skalieren und Aspect Ratio Einstellung
Neu: Streams werden, wenn über Connect dargestellt, nach 1 Minute pausiert
Neu: Aktionen im Wochenplan sind scrollbar und können maximal 32 Aktionen darstellen
Neu: Dialog für Verbindungsabbrüche überarbeitet
Fix: Legende mit „großer“ Legende haben das Aktivierungshäkchen falsch skaliert
Fix: Beim Erstellen von einer Menüleiste direkt gefolgt von einer Kategorie konnte die Startkategorie nicht gewählt werden
Fix: Ein WFC_SwitchPage wird als Interaktion vom Idle Widget ausgewertet
Fix: Die Meldung, dass die Zeitzone nicht stimmt, konnte nicht „weggeklickt“ werden
Fix: Einstellungen aus 3.x und älter mit versteckten Root-Objekte wurde im Editor nicht korrekt übersprungen
Fix: WebFront läuft bei Verbindungsverlust sofort neu, sofern möglich
Fix: Wir probieren das Neu-Verbinden jetzt am Anfang schneller und zum Ende hin mindestens alle 10 Sekunden
Fix: Popup Module laden Streams nicht mehr im Hintergrund, sondern nur, wenn diese wirklich sichtbar offen sind
Fix: Charts in Panels werden beim Tab-Wechsel mit korrekter Zeit angezeigt

SYMOS

Neu: Alle SymBox'en die nach August 2019 gekauft wurden, erhalten ~4 GB zusätzlichen Speicher!

Neu: Benachrichtigung für alle, welche noch ein CM1 in der SymBox nutzen
Neu: Wechsel von uclibc auf glibc (Wir nutzen glibc auf den anderen Plattformen auch schon)
Neu: SymOS aktualisiert, sodass es ebenfalls die neusten Bibliothek-Versionen enthält
Fix: Backups können jetzt mehrere GB groß sein, sofern es genug Speicherplatz auf der SymBox gibt
Fix: Größe reduziert, sodass die Updates auf der SymBox 1 mit alter/kleiner Systempartition besser funktionieren

LEGACY-KONSOLE

Neu: Neuer Typ Visualisierung wird unterstützt
Neu: Alle Instanzen werden über die webbasierte Konsole konfiguriert
Neu: Hinweis, dass die Legacy Konsole ab 2021 entfernt wird

PHP-SDK

Neu: Parameter "name" im Configurator kann Namen der erstellten Instanzen überschreiben, insbesondere auch für Splitter und I/Os in einer Kette
Neu: RegisterOnceTimer, um asynchrone Sofort-Aktionen auszuführen
Neu: Image Element kann jetzt auch Streams per mediaID anzeigen
Neu: Neues Element: SelectProfile
Neu: Formularelement "SelectInstance" kann mit "validModules" auf bestimmte Module restriktiert werden
Neu: Neues Formularelement "SelectModule" um per Dropdown Instanzen eines bestimmten Modultyps auszuwählen
Fix: Das TestCenter passt sich nun dynamisch an Änderungen der Aktionsvariablen an
Fix: Die Parameter "info" und "position" werden nun auch wie vorgesehen bei erstellten Verbindungen angewandt
Fix: Wenn eine Zeile in einer Liste nicht editierbar ist, wird auch die CheckBox ausgegraut

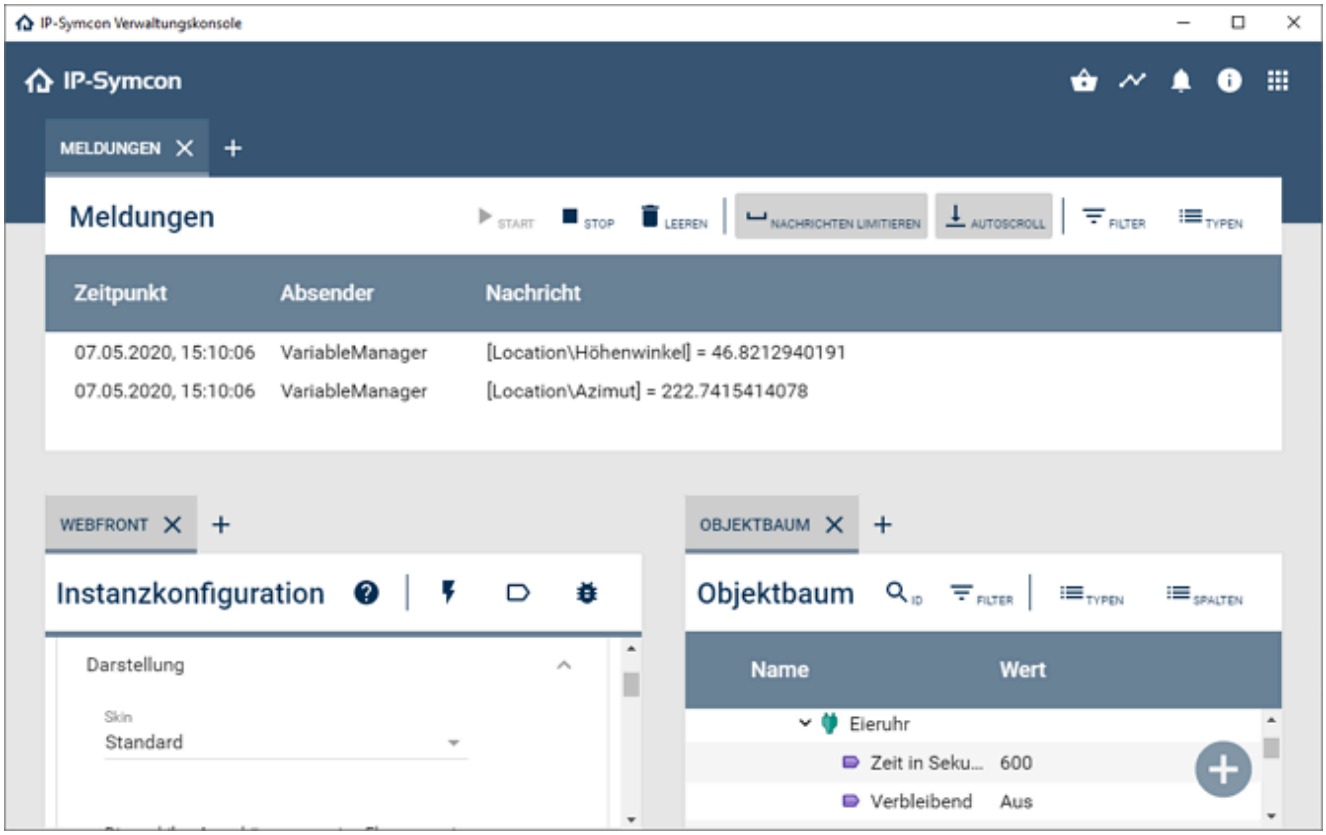
V5.3->V5.4 (Q2/2020)

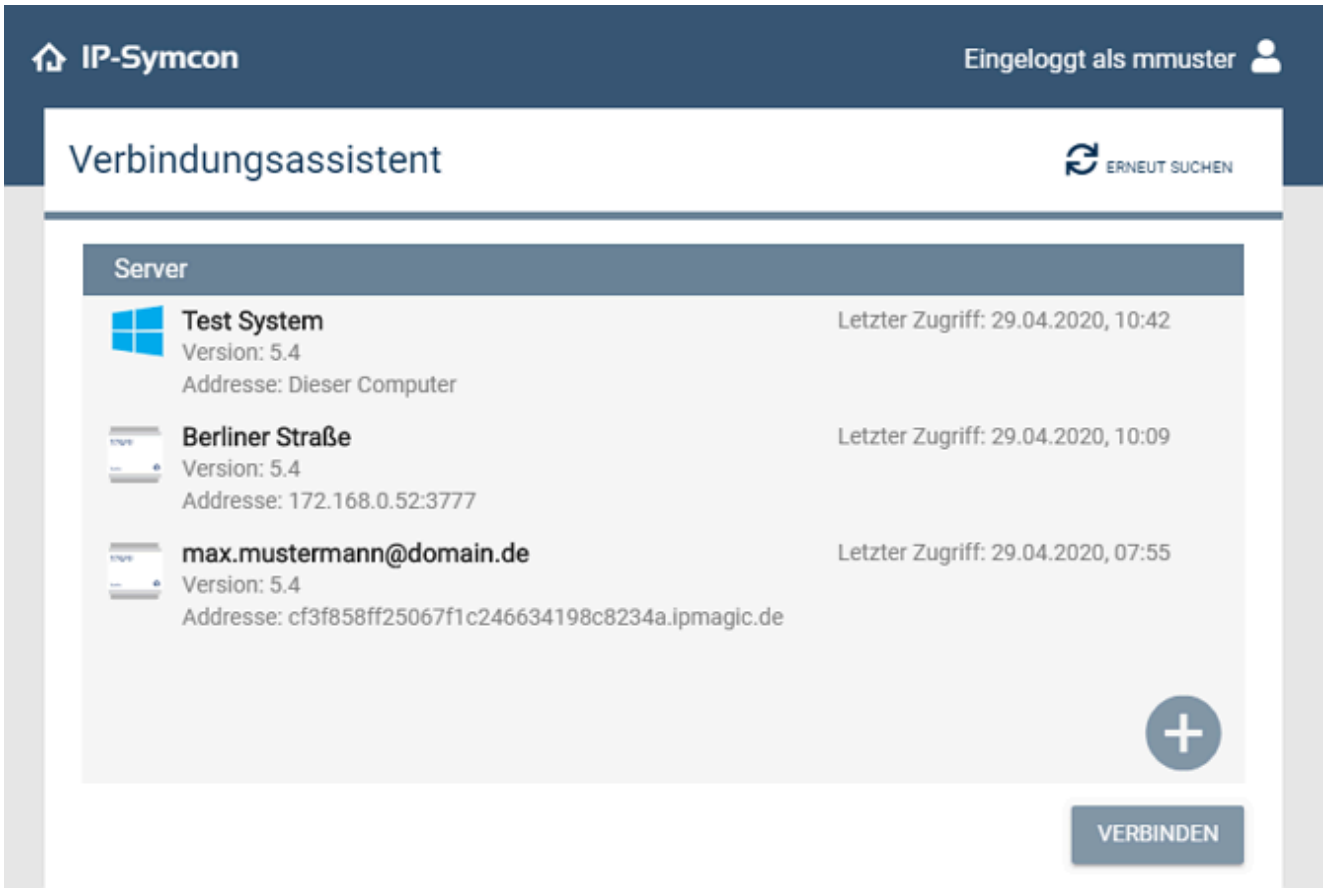
WESENTLICHE ÄNDERUNGEN

Pro Konsole
SymOS Connect
Hovereffekte bei Graphen
Verbesserte Geräteunterstützung

PRO KONSOLE

Eine neue Konsole für die Konfiguration von IP-Symcon wurde eingeführt. Als native Anwendung für Windows, Ubuntu und Mac bietet die Pro Konsole eine deutlich erhöhte Performance. Die neue Konsole bietet nicht nur alle Funktionen der webbasierten Konsole, sondern auch noch einige mehr. So können beispielsweise Tabs ab- und umgedockt werden um optimal mit mehreren Tabs zu arbeiten. Im Verbindungsassistenten können mehrere IP-Symcon Server verwaltet werden. Server im lokalen Netzwerk und entfernte Server aus der Lizenzverwaltung werden automatisch gefunden. Weitere Server können manuell hinzugefügt werden. Zusätzliche Informationen gibt es in der Dokumentation zur Pro Konsole.



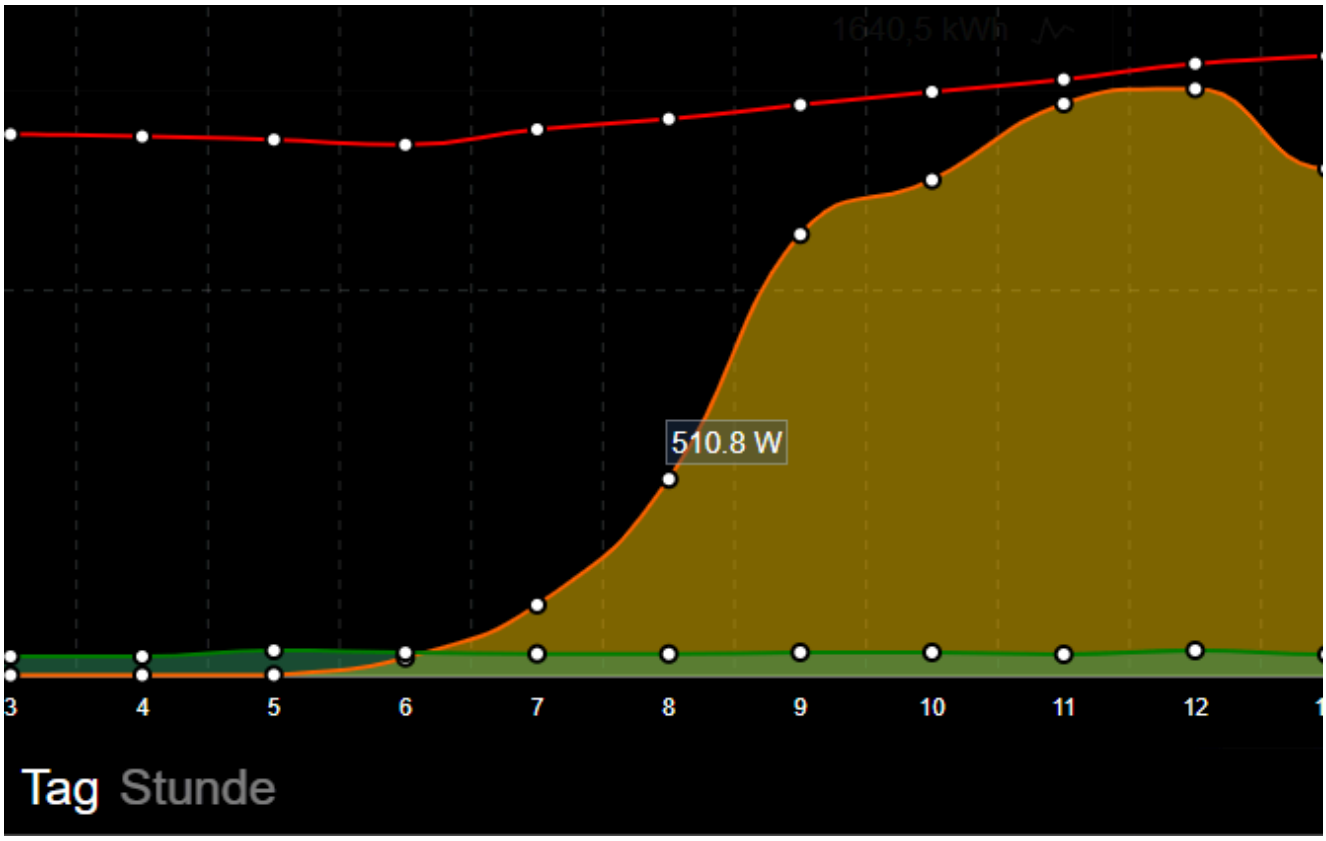


SYMOS CONNECT

Mit SymOS Connect wird ein komfortabler Fernzugriff auf das SymOS der SymBox eingeführt. So kann von überall auf das SymOS zugegriffen, Updates durchgeführt oder IP-Symcon neu gestartet werden. Das SymOS Connect ist unabhängig von der Subskription, d.h. ein Zugriff ist immer möglich.

HOVEREFFEKTE BEI GRAPHEN

Die Graphen im WebFront wurden weiter verbessert. Durch ein Hovern mit der Maus über einem Datenpunkt oder einem Balken wird der dazugehörige exakte Wert angezeigt. So bieten die Graphen einen deutlich erhöhten Komfort und können viel präziser ausgewertet werden.



VERBESSERTE GERÄTEUNTERSTÜTZUNG

Die native Unterstützung von Geräten wurden an diversen Stellen verbessert. So wurde der KNX Import optimiert, so dass nun beispielsweise auch Teile von Exports eingebunden werden können. Für Z-Wave Geräte werden nun einige weitere Informationen der Geräte ausgelesen und können für vielfältige Anwendungsfälle verwendet werden. Bei xComfort werden nun auch die neuen Fensterkontakte CDWA/* unterstützt.

MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 5.3 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 5.4 auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Es wird Raspbian Buster für Raspberry Pi erfordert! (Generelle Versionenübersicht)
Es wird Ubuntu 18.04 für Linux 64 Bit erfordert! (Generelle Versionenübersicht)
Für Microsoft Windows steht ausschließlich eine 64-Bit Version zur Verfügung.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP auf 7.3.16 aktualisiert
Neu: Bibliotheken aktualisiert
Neu: PHP Threads standardmäßig auf 25 erhöht
Neu: VoIP: Anrufe können gezielt per Skript angenommen/abgelehnt werden
Neu: DNS-SD: Ex* Varianten der Abfrage Funktionen hinzugefügt um Timeouts selber definieren zu können
Neu: Profile ~Playback und ~PlaybackPreviousNext zur Steuerung von Medienwiedergabe
Neu: Snapshot für mobile Apps filtert weniger Objekte (ggf. werden dadurch mehr Objekte in der App angezeigt)
Neu: Spezialschalter: NATPublicIP (Globale Public IP Option für den NAT-Support)
Neu: Spezialschalter: LogfileFilter (RegEx zum Filtern von Nachrichten)
Neu: Reagiere unter Windows auch auf das PRESHUTDOWN Event, wodurch der Dienst mehr Zeit zum Herunterfahren haben sollte
Neu: Maximale Ausgabegröße kann über ini_set('ips.output_buffer', Größe in Bytes); vergrößert werden.
Neu: Benachrichtigungen können global im Notification Control deaktiviert werden
Neu: Limitation für täglich versende Push Nachrichten (250). Per Spezialschalter (NotificationLimit) auf 1000 erhöhbar
Neu: Limitation für tägliche Reconnect's zum Connect Dienst (25). Per Spezialschalter (ConnectLimit) auf 100 erhöhbar
Neu: Google State Report Anfragen auf 1000 pro Tag limitiert
Fix: Abstürze beim Auflegen von VoIP Verbindung unter Windows gelöst
Fix: Z-Wave: Fehlende Übersetzungen nachgepflegt
Fix: Z-Wave: Assoziationen können wieder in Gruppen hinzugefügt werden
Fix: HomeMatic: AIR_PRESSURE darf auch vom Typ Float sein
Fix: KNX: XML-Import verbessert, wenn mehrere DPTs/DPSTs genannt werden
Fix: Seltener Deadlock in Verbindung mit Ereignissen korrigiert
Fix: ekey: InputID wird nun korrekt auf "nicht definiert" gesetzt
Fix: Modbus: Problem bei Modbus TCP/IP in Verbindung mit viele Geräte-IDs am selben Strang gelöst
Fix: Übersetzungen für z.B. Archiv vervollständigt
Fix: Legacy Heating Control hat das Resend Interval bei Experten-Skripten nicht berücksichtigt
Fix: IP-Symcon ist abgestürzt wenn eine php.ini von MacOS Systemen unter Windows/Linux verwendet wurde
Fix: DNS-SD: Modul limitiert die Anzahl der Einträge auf 60.
Fix: Connect: Unter Linux wurden fälschlicherweise die ssh config Dateien gelesen
Fix: Connect: "Per E-Mail senden" funktioniert wieder korrekt

GERÄTE

Neu: Xcomfort: Unterstützung für neue xComfort Fensterkontakte (CDWA/*)
Neu: xComfort Konfigurator: Unterstützung für Temperatur-Sensoren am RC Touch
Neu: Z-Wave: RSSI/SecurityKey Information zu ZW_GetInformation hinzugefügt
Neu: Z-Wave: ZW_GetInformation liefert Zeitstempel, wann Pakete gesendet/empfangen wurden
Neu: KNX: XML-Import kann nun auch mit Exports von Teilen (z.B. nur ein Zimmer) umgehen
Neu: KNX: XML-Import ohne zugewiesen DPT bietet beim Erstellen einen Dialog zum auswählen der gewünschten DPT
Fix: Z-Wave: Erkennung von "fehlerhaften" Knoten verbessert
Fix: Z-Wave: Abfrage der Manufacturer Specific Klasse reduziert (fast alle Geräte antworten nicht korrekt und wir brauchen die Informationen eigentlich nicht)

Fix: Z-Wave: Fibaro RGBW (Neue Generation) funktioniert nun korrekt
Fix: Z-Wave: Fehlermeldungen im Debug bzgl. falscher Länge (sofern Security genutzt wurde) korrigiert
Fix: Z-Wave: Ignoriere ein weiteres Fibaro-Gerät bei Multilevel-Sensorabfrage
Fix: KNX: DPT 11 hat beim Senden vom Datum den Tag um 1 versetzt versendet
Fix: M-Bus: Kleine Wartezeit eingefügt, sofern mehrere Pakete hintereinander abgefragt werden
Fix: ModBus: Bessere Kompatibilität beim Auslesen von Coils bei Geräte die sich nicht sauber an die Spezifikation halten
Fix: KNX Discovery konnte Konfigurator nicht erstellen, wenn der IP Interfacename genau 30 Zeichen lang war
Fix: HomeMatic: Wenn der Socket inaktiv ist, fragt der Konfigurator die Raum-Informationen auch nicht ab
Fix: UVR1611 setzt den Port korrekt auf 40000

PHP-SDK

Neu: Der Inhalt eines Konfigurators kann wie eine Liste für Aktionen verwendet werden, wird allerdings nicht als Eigenschaft gespeichert
Neu: Flag noDiscovery im Configurator ermöglicht es Geräte aus der Gerätesuche herauszuhalten, z.B. für ein generisches "Gerät hinzufügen"
Neu: Flags editable und deletable für List und Tree geben an ob ein einzelnes Element editiert/gelöscht werden darf
Neu: Datenfluss gibt Rückgabewerte von Children an den Parent weiter (z.B. Device an Splitter)
Neu: Parameter multiline für ValidationTextBox
Fix: minimum und maximum vom Numberspinner können wie vorgesehen per Dynamik angepasst werden
Fix: validate vom ValidationTextBox und PasswordTextBox können wie vorgesehen per Dynamik angepasst werden

WEBFRONT

Neu: Hover-Werte bei Balkengraphen und Linengraphen (Nur normal, nicht bei HD/Raw)
Neu: Hinweis, wenn die Externe Seite HTTP Seiten laden will, obwohl das WebFront über HTTPS geladen wurde (Moderne Browser blockieren sogenannten Mixed Content)

KONSOLE

Neu: Performance bei Drag & Drop und Anpassung vieler Baumelemente gleichzeitig, z.B. beim Filtern, optimiert
Neu: Suchen/Ersetzen-Dialog sortiert die angezeigten Skripte
Neu: Zeige Pfad zum Objekt als Tooltip des Tabs an
Neu: Die Variablenaktionen "Addieren" und "Subtrahieren" verwenden für den Summanden/Subtrahenden nun die Nachkommastellen des Profils
Neu: Wird der Filter im Objektbaum angepasst wird zum aktuell ausgewählten Objekt gescrollt
Neu: Tabs können via Mittelklick geschlossen werden
Neu: "Wer benutzt dieses Profil?"-Dialog betrachtet nun auch Charts
Neu: Button im Debug Log hinzugefügt, um zur dazugehörigen Instanzkonfiguration zu springen

Neu: Skripteditor intern komplett umgebaut, dadurch sollte der Editor deutlich schneller laden
Neu: Nachrichtenlänge in Debuglogs auf 255 gekürzt und "Erweitern"-Funktion im Kontextmenü hinzugefügt
Neu: Frage nach dem Löschen von Instanzen nur einmal nach, ob unbenutzte Elterninstanzen gelöscht werden sollen, nicht nach jeder einzelnen Instanz
Neu: Kompaktmodus überarbeitet: Keine "Ränder" mehr und ein paar Größen angepasst
Neu: Dialog um in allen Skripten zu ersetzen fragt vorm Ersetzen, ob ungespeicherte Skripte gespeichert werden sollen
Neu: Dialog zum Auswählen eines Profils ist nun im klassischen Look von OK/Abbrechen-Dialogen
Neu: Der Inhalt von Mitteilungs-Popups kann markiert und somit auch kopiert werden
Neu: Initialer Wert von neuen Assoziationen wird auf Basis der aktuell höchsten Assoziation und Schrittweite vorgeschlagen
Neu: Bei Klick auf "Neu Laden" nach einem Verbindungsabbruch wird geprüft, ob die Konsole wieder verfügbar ist bevor neu geladen wird
Neu: Versuche bei mehreren geöffneten Konsolen den Snapshot von einer anderen offenen Konsole abzufragen. So sollte insbesondere das Öffnen von abgedockten Fenstern der Pro Konsole beschleunigt werden
Neu: Spalten im Objektbaum werden automatisch ausgeblendet, wenn der Platz nicht ausreicht
Neu: Referenzsuche im Module Store (unter Weitere Optionen für installierte Module)
Neu: Beim Löschen von Modulen im Module Store wird geprüft, ob dazugehörige Instanzen noch existieren
Neu: Alle X <Zeiteinheit> im zyklischen Ereignis wird mit 1 initalisiert
Neu: Filter im Statuswidget werden gespeichert
Neu: Ist beim Drücken von Strg + O im Skripteditor eine InstanzID ausgewählt, springe direkt zu dieser
Neu: "Nachfolgende Ereignisse ausführen" wird nun als Dropdown dargestellt
Neu: Einige Eingaben in Ereignisdialogen sind nun auf voller Breite
Neu: Die Eingabe von auslösenden Werten in auslösenden Ereignissen ist nun ein Nummernfeld bei numerischen Variablen
Neu: Im Objektbaum können Objekte nun via Rechtsklick -> Objekt verschieben verschoben werden
Neu: Formatiere Variablenwerte bei Erweiterten Bedingungen entsprechend des Variablenprofils
Neu: Markiere Filter im Objektbaum, wenn der Filter aktiv ist
Neu: Statusdialog ist nun modal und kann auch als Tab geöffnet werden
Neu: PHP Code der gewählten Aktion kann im "Befehle testen"-Dialog per Strg + C in die Zwischenablage kopiert werden
Neu: Variablenaktion "Auf auslösenden Wert schalten" kann nun bei prozentualen Variablen den Wert prozentual oder absolut übertragen. (50% von Intensity.255 werden 50% bei Intensity.100)
Neu: "Zu Objekt springen" kann im Skripteditor bei Auswahl einer ObjektID per Rechtsklick oder das Menü Textbearbeitung ausgeführt werden
Neu: Im Instanz erstellen/bearbeiten-Dialog passt sich die Ortsanzeige für Modultypen an, welche in eine eigene Sonderkategorie kommen
Neu: Im Archive Control wurde die Namensspalte auf eine Namens- und eine Ort-Spalte aufgeteilt
Neu: Wurde ein Log gestoppt, wird dies nicht mehr gespeichert, bei einem Neustart der Konsole läuft das Log also weiter
Fix: Werden Spalten hinzugefügt oder entfernt, bleiben Sonderaktionen für spaltenspezifische Doppelklicks bei den korrekten Spalten
Fix: Sortierung der Transferprüfung im Module Control ist nun unabhängig von Groß- und Kleinschreibung

Fix: Info zur Variablenanzahl im Lizenz-Widget für die Unlimited Edition passend formatiert
Fix: Schöne Fehlermeldung beim Versuch die Subskriptionsüberprüfung ohne Lizenz zu aktivieren
Fix: Variableneditor ist nun auf 100 Einträge beschränkt
Fix: Größe des Wertefensters des Variableneditors korrigiert
Fix: Icons für den Icon-Auswahl-Dialog laden deutlich schneller
Fix: Anpassung der Fenstergröße in Safari werden in der Formatierung korrekt umgesetzt
Fix: Biete im Skripteditor keine Autovervollständigung innerhalb von Strings oder Kommentaren
Fix: Formulierung bei Nachfolgender Ausführung von ausgelösten Ereignissen optimiert
Fix: Dialog zur Discovery-Auswahl scrollt nun wie vorgesehen nur den Inhalt des Baumes
Fix: Aktionsbeschreibung in Ereignisdialogen passt sich auch bei Variablen ohne Aktion korrekt an
Fix: Die Gerätesuche wertet Konfigurator-Einträge ohne InstanzID wie vorgesehen als nicht erstelltes Gerät aus
Fix: Bewegbare Dialoge werden korrekt aufgeräumt nachdem sie geschlossen wurden
Fix: Dropdowns werden wie vorgesehen geschlossen, wenn auf einen bewegbaren Dialog geklickt wird
Fix: Prüfung ob Von und Bis bei zyklischen Ereignissen korrekt gesetzt sind
Fix: Einige verwendeten Bibliotheken aktualisiert
Fix: Ausklappende Kontextmenüs auf kleinen Bildschirmen werden korrekt formatiert
Fix: Nummern in Listen, z.B. für NumberSpinner, werden korrekt numerisch sortiert
Fix: Ladenachrichten beim Start der Konsole werden nun wie vorgesehen auf englisch übersetzt
Fix: Warte mit dem neu laden der Instanzkonfiguration bis die Antwort auf ApplyChanges zurückkommt
Fix: Debuglog versucht nicht mehr nach Verbindungsabbruch das Debug aktiv zu halten
Fix: Werden bei einem Löschvorgang ein Objekt und seine Kinder markiert, kommt es nicht mehr zu einer Fehlermeldung
Fix: Ändert sich ein übergeordnetes Objekt einer auslösenden Variable eines Ereignisses, wird die Bezeichnung des Ereignisses wie vorgesehen aktualisiert
Fix: Verhindere Verbindungsabbruch bei Konfiguratoren mit sehr vielen Geräten
Fix: Generierte Aktionsskripte werden übersetzt
Fix: Formulierung im Lizenz-Widget angepasst, so dass deutlich wird, dass nicht die Lizenz, sondern die Subskription abläuft
Fix: "Befehl einfügen" formatiert den eingefügten Code auch mehrzeilig wie vorgesehen
Fix: Prüfe ob der zurückgegebene Link eines Buttons Zeilenumbrüche enthält und gebe ggf. passende Fehlermeldungen
Fix: Korrekte HEX-Darstellung im Debug-Log - Es wird nicht versucht gelegentlich Zeichen zusammenzuziehen, da diese als UTF-8 interpretiert werden
Fix: Anzeige vom Teilpfad zur Tabunterscheidung wird auch bei mehreren gleichnamigen Tabs gleichzeitig korrekt ausgewertet
Fix: Ident wird wie vorgesehen aktualisiert
Fix: Eigenschaften, die via GetConfigurationForParent gesetzt werden, werden auch übernommen, falls die dazugehörigen Elemente noch inaktiv sind, z.B. in einem eingeklapptem ExpansionPanel
Fix: Wurde in der HTML-Darstellung der Skriptausgabe auf eine externe Seite gelinkt, kann das Skript wie vorgesehen erneut ausgeführt werden
Fix: Ausgelöste Ereignisse mit einer Boolean-Variable ohne Profil als Auslöser funktionieren wieder wie vorgesehen
Fix: Gruppen in Wochenplänen werden nun in korrekter Reihenfolge gespeichert
Fix: Initialer auslösender fester Wert bei ausgelösten Ereignissen funktioniert bei Boolean-Variablen ohne Profil wie vorgesehen
Fix: Bei Wechsel der auslösenden Variablen sollte der bisher ausgewählte Wert wie vorgesehen stets passend zur neuen Variable konvertiert werden

Fix: Ein seltener Fehler beim Ändern eines ausgelösten Ereignisses auf einen anderen Variablentyp gefixt
Fix: Wird im Skripteditor als Teil der Ereigniskonfiguration Enter gedrückt, wird dies nicht mehr als Bestätigung des Dialogs gewertet
Fix: Tabs hatten manchmal einen falschen Namen (Name not yet loaded)

SYMOS

Neu: SymOS Connect. Direkter Zugriff auf SymOS über unsere Cloud*
Neu: Ohne SymOS Kennwort sind keine Anfragen über SymOS Connect erlaubt
Neu: SymOS Web-Server komprimiert Daten per GZIP und unterstützt Caching
Neu: IP-Symcon wird bei einem Crash automatisch neu gestartet
Neu: Weiterleiten ins WebFront (SymOS wird dabei übersprungen)
Neu: Integrator Nachricht kann über Einstellungen gesetzt werden
Neu: Checkbox um #integrator Nachricht zu deaktivieren
Neu: Unterbau von Polymer 1 auf Polymer 3 migriert (Im Prinzip fast neu geschrieben)
Neu: Kernel auf 4.14.114 aktualisiert
Neu: #delete heißt nun #files
Neu: Splashscreens entfernt
Fix: Dateien werden nach Datum sortiert
Fix: Sofern der Speicherplatz gering ist, werden alte IP-Symcon/SymOS Versionen gelöscht
Fix: Sonderzeichen beim SymOS Kennwort konnten (selten) Probleme beim Einloggen machen
Fix: Löst hoffentlich das Problem, dass DHCP nach Auslaufen der Lease keine IP-Adresse mehr bezogen hat
Fix: E-Mail Adresse mit längeren Domain Namen (z.B. swiss) werden akzeptiert

PRO KONSOLE

Neu: Die Pro Konsole wurde veröffentlicht

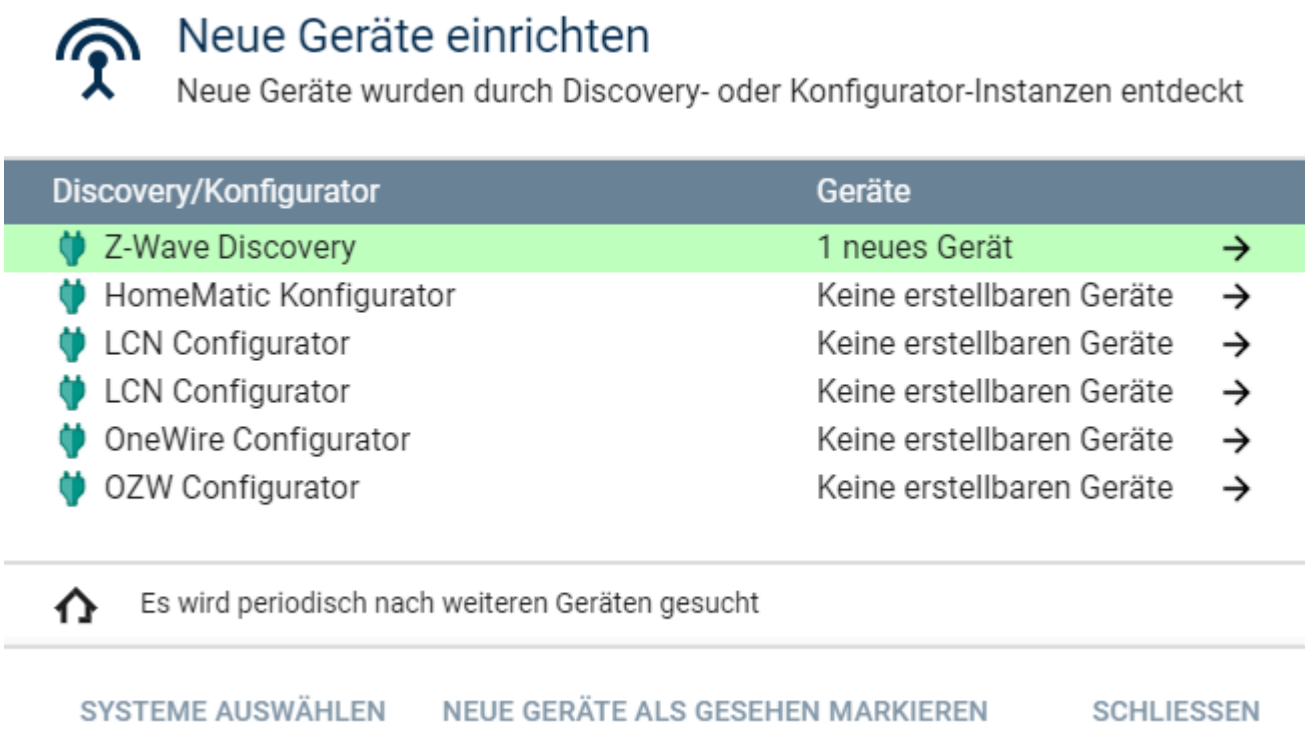
V5.2->V5.3 (Q4/2019)

WESENTLICHE ÄNDERUNGEN

Gerätesuche
Verbesserter KNX Import
Konfigurator für M-Bus
Komfortverbesserungen in der Verwaltungskonsole

GERÄTESUCHE

Mit der neuen Version sucht IP-Symcon automatisch in regelmäßigen Abständen nach neuen Geräten und zählt diese in der Gerätesuche auf. Nach der Systemauswahl erfolgt die Detektion von Gateway und Geräten durch die Gerätesuche und diverse neue Discovery- und Konfigurator-Instanzen bei den meisten Systemen automatisch und es müssen nur noch die zu erstellenden Geräte ausgewählt werden. Weitere Informationen gibt es in der Dokumenentation zur Gerätesuche und in unserem Grundlagen-Webinar .



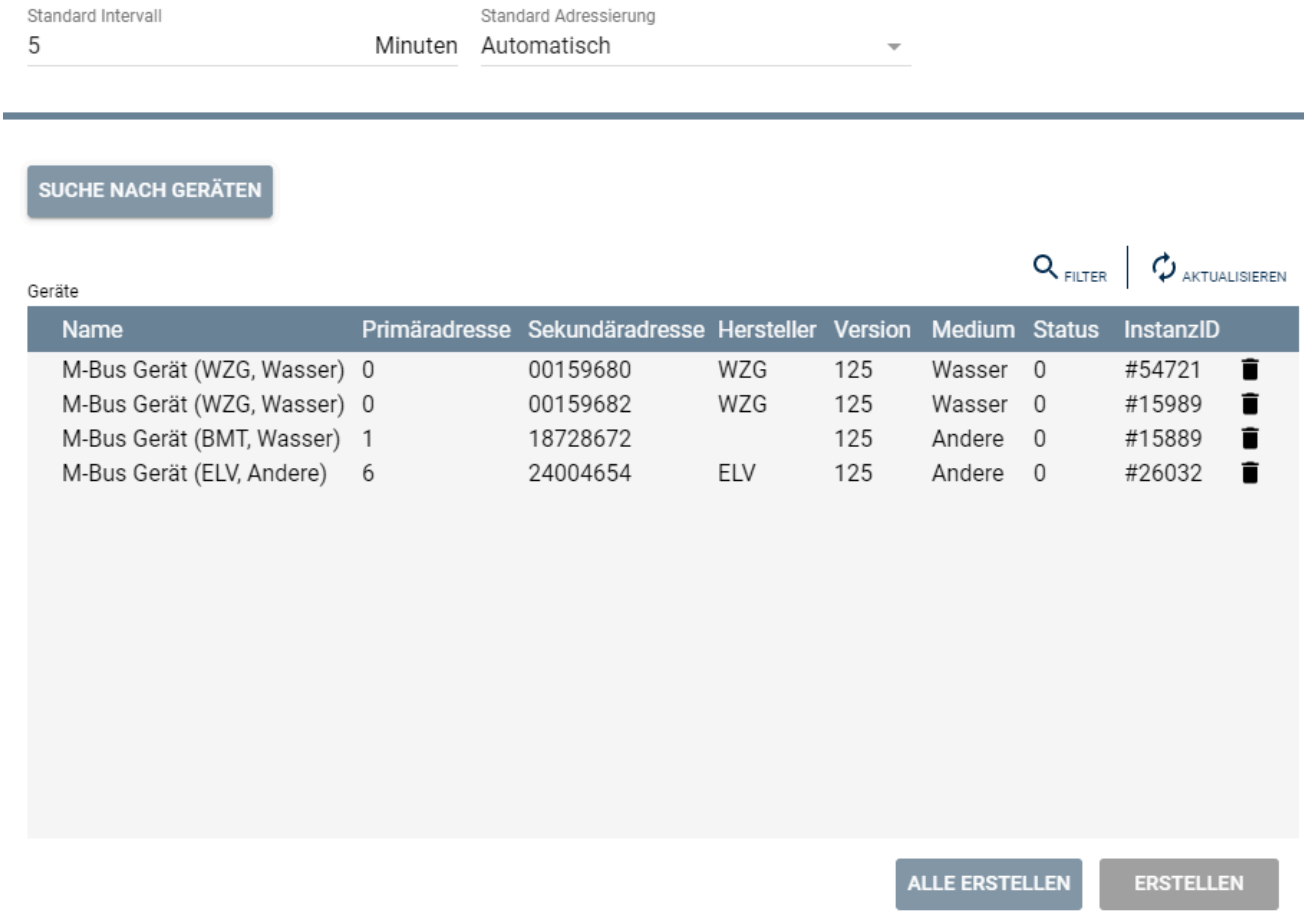
VERBESSERTER KNX IMPORT

Die Import-Funktion im KNX-Konfigurator wurde erweitert. Zusätzlich zum OPC Import ist es nun auch möglich die modernen DPTs aus einer generierten XML-Datei zu importieren und danach komfortabel im Konfigurator anzulegen. Es stehen dazu folgende Video Tutorials zur Verfügung: XML-Export mithilfe der ETS , XML-Export in IP-Symcon verwenden



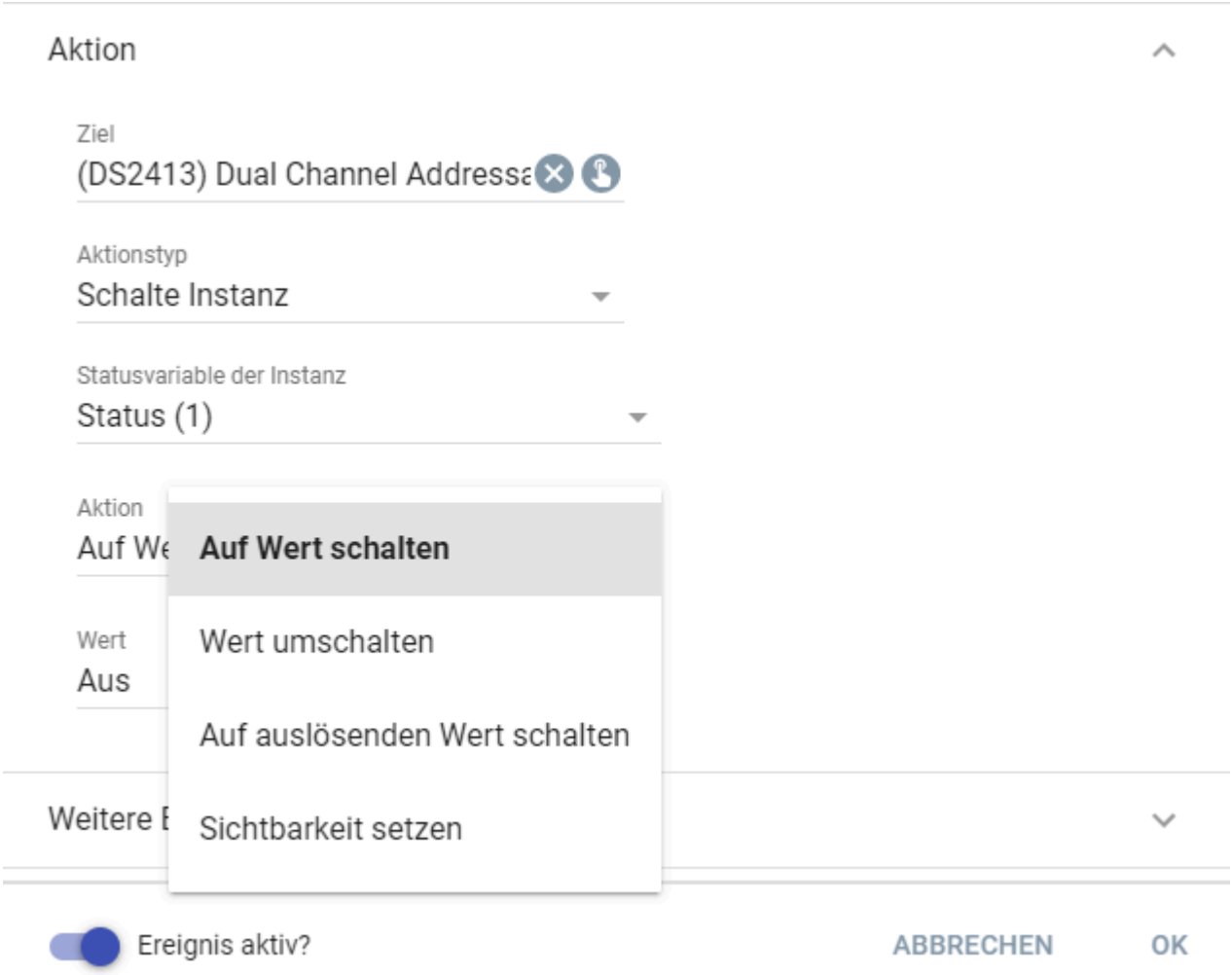
KONFIGURATOR FÜR M-BUS

Auch für M-Bus gibt es nun einen Konfigurator. Dieser sucht den gesamten Bereich der Sekundäradressen von M-Bus ab und findet so alle angeschlossenen M-Bus-Geräte, damit diese unkompliziert in IP-Symcon erstellt werden können.



KOMFORTVERBESSERUNGEN IN DER VERWALTUNGSKONSOLE

Diverse kleine Erweiterungen haben den Komfort der Verwaltungskonsole verbessert. So ist es nun möglich die Reihenfolge der Tabs via Drag & Drop anzupassen. Der Browsertitel passt sich an den Namen der Hauptkategorie an. Die Aktionen für Variablen in Ereignissen wurden erweitert. Insbesondere ist es möglich bei ausgelösten Ereignissen den auslösenden Wert direkt weiterzuleiten. So können unkompliziert diverse automatisierte Verhalten implementiert werden.



MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 5.2 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 5.3 auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Beim Upgrade in Windows wird die Konfiguration von IP-Symcon analog zu anderen Betriebssystemen in den ProgramData-Ordner verschoben. Dies geschieht automatisch.
Es wird Raspbian Buster für Raspberry Pi erfordert! (Generelle Versionenübersicht)
Es wird Ubuntu 18.04 für Linux 64 Bit erfordert! (Generelle Versionenübersicht)
Für Microsoft Windows steht ausschließlich eine 64-Bit Version zur Verfügung.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: Windows Version speichert alle Daten ab sofort unter C:\ProgramData\Symcon (Mehr Informationen)
Neu: Wenn IP-Symcon abstürzt wird unter Ubuntu/RPi/MacOS versucht automatisch ein Stacktrace im /var/lib/symcon/minidump Ordner zu speichern.
Neu: MiniDumps unter Windows sind jetzt testweise signifikant kleiner
Neu: HasAction Funktion zum Prüfen, ob Variable eine gültige Aktion besitzt
Neu: SSDP Control (Beispiel: YC_SearchDevices($id, "upnp:rootdevice"))
Neu: DNS-SD kann Geräte suchen
Neu: DNS-SD nutzt unter Linux die native Avahi API
Neu: UDP Socket unterstützt nun korrekt Broadcasting über alle vorhandenen Interfaces
Neu: Datenfluss angepasst, sodass mehrere alternative Pakete versendet werden können und nur das "beste" an die Instanz ausgeliefert wird. (Bisher nicht für PHP-Module verfügbar)
Neu: Update-Kanal wird unter Linux auch unter /etc/apt/sources.list.d/* geprüft
Neu: WFC_PushMessageEx um eine WebSocket Nachricht an einen bestimmten verbundenen Client via (RemoteIP/RemotePort) zu senden.
Neu: Die Docker, Ubuntu und Raspberry Pi Versionen nutzen für DNS-SD nun Avahi Core anstatt Avahi vom Betriebssystem. (Dadurch entfällt die letzte große Paket-Abhängigkeit)
Neu: Ubuntu/Raspberry Pi zeigt nach der Installation den Pfad zur Konsole an
Neu: MQTT: Retain
Neu: MQTT: Session Management (Siehe Timeout und Queue Limit Einstellungen im Server)
Neu: MQTT: Last Will/Testament
Neu: MQTT: Publish eines Clients wir nur in IP-Symcon ausgewertet und nicht an die anderen Clients weitergeleitet
Neu: Im Tray unter Windows gibt es einen Button zum Öffnen des ProgramData Pfades
Fix: Connect Dienst hat ein Timeout von 120 Sekunden anstatt von 10 Sekunden (Das sollte die Abbrüche der Verwaltungskonsole bei langen Operationen lösen)
Fix: MessageQueue wird beim Beenden vollständig abgearbeitet (Löst Problem mit fehlenden Werten in der settings.json die direkt vor dem Beenden geschrieben wurden)
Fix: "Enable Broadcast" aus UDP Socket entfernt, das dies nur für den Multicast Socket anwendbar war
Fix: Bei Modul-Updates wurde der Instanz-Status nicht korrekt zurückgesetzt, sodass diese weiterhin als Aktiv gemeldet wurden, obwohl diese gar nicht verfügbar waren
Fix: Potenzieller Fix für seltenes Crash-Problem, wenn man oft I/Os schließt/öffnet unter Linux
Fix: MQTT: Unvollständige/Fehlerhafte Pakete konnten IP-Symcon zum Absturz bringen
Fix: VoIP: VoIP_SendDTMF erlaubt nun auch * und #
Fix: IntervalBox'en und 'label' auf NumberSpinner und 'caption' aktualisiert
Fix: MQTT leitet Nachrichten nicht an den Ursprung weiter um Endlosschleifen zu verhindern
Fix: Fehlerhafte Module konnten nicht über den Store gelöst werden
Fix: MQTT: Geräte ohne ClientID bekommen eine passende ClientID zugewiesen
Fix: MQTT: Last Will mit QoS > 0 wird aktuell mit QoS = 0 versendet

GERÄTE

Neu: Discovery Instanzen für alle Symcon LAN Gateways (IPS-868, M-Bus, 1-Wire, Z-Wave, EnOcean)
Neu: HomeMatic Discovery
Neu: IPS-868 Konfigurator
Neu: EnOcean D2-11-*
Neu: EnOcean EEP D2-05-00, EEP D2-05-01, D2-05-02
Neu: M-Bus Konfigurator, welcher alle angeschlossenen M-Bus Geräte suchen kann
Neu: M-Bus Geräte können über Sekundär-Adressierung angesprochen werden
Neu: M-Bus: Anzahl der gelesenen Pakete kann limitiert werden
Neu: M-Bus Konfigurator: Standard Intervall / Standard Adressierung kann vorgegeben werden

Neu: KNX Konfigurator kann XML Exports der ETS Gruppenadressen einlesen (Damit werden auch die neuen DPT Instanzen erstellt)
Neu: KNX Konfigurator in der Legacy Konsole entfernt, da dieser nicht mehr mit IP-Symcon 5.3 kompatibel ist
Neu: KNX Konfigurator versucht GA's mit dem Status-Suffix automatisch als Rückmelde GA zuzuordnen, sofern gewünscht
Neu: KNX/DPT Instanzen haben im Namen zusätzlich den generischen Datentyp
Neu: LCN: Setzen vonn Schwellwerten (1x5 über Register 0, 4x4 über Register 1-4)
Neu: Z-Wave: Es können benutzerdefinierte Werte bei Parameters aus der ZWDB angegeben werden
Neu: VoIP: VoIP_SendDTMF zum Senden von DTMF Tönen
Fix: VoIP: Besserer Fix für das 401 Problem sobald die Session abläuft
Fix: M-Bus: Position wird über mehrere Pakete hinweg korrekt gesetzt
Fix: Z-Wave: Öffnen der Z-Wave Instanzen bei vielen angelernten Knoten sollte wesentlich schneller gehen
Fix: Z-Wave: Problem beim Öffnen von Instanzen mit Zipato RFID/Lock gelöst
Fix: Z-Wave: Workaround für fehlerhafte Fibaro Geräte, welche eine schlampige Implementation der Multi Channel Association Klasse haben. Wir entfernen diese Klasse entsprechend, sodass IP-Symcon die alten simplen Association verwendet
Fix: Z-Wave: Option für "Simple Association erzwingen" wird nur noch angezeigt, wenn diese auch sinnvoll ist (Benötigt vorhanden sein der Multi Channel Klasse und die Multi Channel Association Klasse ab Version 3)
Fix: LCN Display: Anzeigedauer wird nicht in Sekunden angeben, sondern ist von der Modul-Konfiguration abhängig
Fix: LCN: Ampere-Werte werden nun korrekt in Milliampere dargestellt
Fix: xComfort 1-Kanal Heizaktor kann nun auch korrekt ausgeschaltet werden
Fix: xComfort: Router wird vom Konfigurator korrekt erkannt

KONSOLE

Neu: Die Glocke oben rechts in der Verwaltungskonsole zeigt nun erstellbare Geräte von Discovery und Konfigurator Instanzen an
Neu: Im Module Control kann die Migration auf den Store für beliebige Module deaktiviert werden
Neu: Ausgelöste Ereignisse können den Ereignis-Wert direkt an das Ziel-Objekt weiterleiten.
Neu: Zeige im Skripteditor einen Hinweis, wenn Teile der Skriptausgabe durch den HTML-Filter versteckt werden
Neu: Wird der Aggregationstyp einer Variablen geändert, dann wird gefragt, ob diese reaggregiert werden soll
Neu: Das Schnellausführungs-Widget speichert das Skript der letzten Ausführung
Neu: Browsertitel passt sich an den Namen der Top-Level-Kategorie an
Neu: Schlägt die Erstellung einer Instanz via Konfigurator fehl, dann räumt der Konfigurator auf
Neu: Tabs können nun per Drag & Drop neu angeordnet werden
Neu: Icons im Objektbaum haben wieder einen Bubble-Effekt beim Klicken
Neu: Lade Konfiguration nicht neu, wenn das Übernehmen der Änderungen fehlschlägt
Neu: Verwaltungskonsole kann via Browser als "App" installiert werden
Neu: Beim Hovern über einem Variablenwert und innerhalb des Variable-bearbeiten-Dialogs werden Rohwert und formatierter Wert angezeigt
Neu: Widget um Absturzberichte zu verschicken
Neu: Update-Buttons für die verbleibende Subskriptionsdauer im Info-Dialog und im Lizenz-Widget
Neu: Tabs mit gleichem Namen erhalten Teile des Pfades im Tabnamen, damit die Tabs differenziert werden können

Neu: Tutorial an neue Gerätesuche Mechanismen angepasst
Neu: Geräte können in Konfiguratoren als Gesehen/Ungesehen markiert werden
Neu: Als gesehen markierte Geräte werden in Konfiguratoren initial nicht angezeigt
Neu: "Springe zum Auslöser" für auslösende Ereignisse per Rechtsklick im Objektbaum
Fix: In der Legacy Konsole können Wochentage bei monatlichen Ereignissen korrekt gesetzt werden
Fix: Spalten gehen nicht mehr kaputt, wenn man die Positionsspalte einblendet, im Gegenzug eine andere Spalte ausblendet und dann neu Lädt (Dies ist der Webinar Bug!)
Fix: Neu angelegte Ereignisse konnten unter bestimmten Bedingungen nicht mehr geöffnet werden
Fix: Progressbar mit "indeterminate" Zustand wird korrekt angezeigt
Fix: Übersetze suffix des NumberSpinner wie vorgesehen
Fix: Lade Konfiguration beim Verwerfen der Änderungen neu und lädt somit auch die Einträge von Feldern ohne Eigenschaft neu
Fix: Widget für statistische Daten kann sich nicht mehr mehrfach öffnen, wenn noch keine Entscheidung getroffen wurde
Fix: Korrigiere Dialoge für Microsoft Edge
Fix: Zoomen sollte auch in iOS 13 wie vorgesehen deaktiviert sein
Fix: Der mobile Safari von iOS 13 wird nun wie vorgesehen korrekt erkannt
Fix: In der Timerübersicht werden die Statusicons nun wie vorgesehen eingefärbt
Fix: In der Timerübersicht werden die Timer wie vorgesehen nach Kategorien gruppiert
Fix: Blockiere die Auswahl eines Links als Elternobjekt
Fix: Skripte, die kein Variablentemplate sind, werden nun wie vorgesehen in der Ereignisübersicht einer Instanz angezeigt
Fix: Wird der Zeitzonen-Hinweis auf Wunsch des Benutzers nicht angzeigt, wird dennoch wie vorgesehen die Zeitzone des Servers eingestellt
Fix: closeCaption bei Popups wird nun wie vorgesehen übersetzt, dies betrifft insbesondere Z-Wave
Fix: Der Ausklapp-Button einer eingeklappten Kategorie im Objektbaum verschwindet nun wie vorgesehen wenn das letzte Kind gelöscht wird
Fix: Bestimmte Benutzername/Passwort Kombinationen führen im Chrome nicht mehr zu einem Login-Fehler
Fix: Eigenschaften in Expansion Panels werden wie vorgesehen geladen
Fix: Beim Versuch ein IPSStudio-Formular zu editieren kommt ein Hinweis, dass dies nur mit IPSStudio möglich ist
Fix: Download-Button im Dokument-Dialog wird nicht mehr unterstrichen
Fix: Die Beschreibung eines Objektes wird nun einheitlich als Beschreibung bezeichnet, nicht mehr als Info
Fix: Der Inhalt von Toolbars kann nicht mehr überlaufen. Dies betraf insbesondere die Suchfelder im Kompaktmodus
Fix: Die caption des Formularelements Label wird nun korrekt aktualisiert, auch wenn im Text Links enthalten waren
Fix: Jedes SelectDateTime-Feld hat seinen eigenen Wert und stellt nicht mehr die Werte von vorherigen SelectDateTime-Feldern dar
Fix: Zeige Mehrfachoptionen in einem Konfigurator auch bei mehrfachem Öffnen korrekt an
Fix: location wird nun auch bei Mehrfachoptionen im Konfigurator wie vorgesehen ausgewertet
Fix: value kann nun im Formularfeld Select wie vorgesehen verwendet werden
Fix: SelectColor lädt nun wieder wie vorgesehen Wertevorgaben
Fix: Nach dem Suchen&Ersetzen lädt die Konsole die betroffenen Skripte neu
Fix: Beim Schließen eines Tabs wird nun nicht mehr gelegentlich auf den ersten Tab gewechselt
Fix: SelectLocation überarbeitet, nun verschwindet der Marker nicht mehr nach einem Neuladen und die Bedienung ist flüssiger
Fix: In Firefox wird die Größe des Editieren-Dialogs von Listen nun korrekt bestimmt

Fix: Listen mit einer Spalte können wie vorgesehen im elements-Bereich verwendet werden
Fix: Achte bei der Sortierung in Listen wie vorgesehen nicht auf Groß- und Kleinschreibung

WEBFRONT

Fix: RTSP Streams wurden bei bestimmten WebFront Kennwörtern nicht geladen
Fix: Dokumente wurden nicht automatisch aktualisiert, sofern diese z.B. über IPS_SetMediaContent verändert wurden

PHP-SDK

Neu: minimum, maximum und onChange sind beim HorizontalSlider nun optional
Neu: Neuer Parameter stepsize für den HorizontalSlider
Neu: Im Konfigurator können individuelle Namen für Statusvariablen definiert werden
Neu: Neues Formularfeld OpenObjectButton
Neu: Validierung für NumberSpinner, ValidationTextBox und PasswordTextBox
Neu: ExpansionPanel und PopupButton unterstützen nun onClick
Neu: List und Tree unterstützen nun onEdit, onDelete und onAdd
Neu: Skripte in der Instanzkonfiguration können nun die gesamte Elemente vom Typ List oder Tree komplett iterieren und nicht nur auf den aktuell gewählten Wert zugreifen
Neu: Parameter "link" für Label kann einstellen ob URLs in der Caption als Links dargestellt werden oder nicht
Neu: Unterstütze die Option width für fast alle Formularfelder
Fix: Slider, welcher initial per value gesetzt wurde, sendet diesen Wert nicht mehr direkt per onChange wieder raus

ÄNDERUNGEN 5.3.X

ALLGEMEIN

Neu: KNX DPT 9.029 hinzugefügt
Neu: SSDP Announce beinhaltet die genutzte Platform
Neu: Performance vom Datenfluss verbessert
Neu: HomeMatic: Keep Alive Timeouts können konfiguriert werden (0 = Aus)
Neu: M-Bus Konfigurator: Standardeinstellung für die Paketlimiterung verfügbar
Neu: Neue Spezialschalter zum Steuern der WebSocket/RTSP Limitationen (ServerSoftQueueSizeLimit, ServerSoftQueueBytesLimit, ServerSoftQueueTransferTimeout, ServerHardQueueSizeLimit, ServerHardQueueBytesLimit, ServerHardQueueTransferTimeout) *
Neu: Unterstützung für neue xComfort Fensterkontakte (CDWA/*)
Fix: PHP zurück auf 7.3.5, um ggf. Abstürze mit PHP 7.3.8 auszuschließen. (Nur Linux, SymBox, Ubuntu, Docker)
Fix: M-Bus: Sofern zwei Geräte mit der selben Sekundäradresse am Bus waren, konnte IP-Symcon abstürzen
Fix: SSDP: Auf einigen Systemen konnte die Instanz nicht korrekt erstellt werden
Fix: LCN: LCN Konfigurator (Module) hat einige Instanzen nicht korrekt erkannt
Fix: Z-Wave: Dialog für unterstützte Kommando-Klassen hat unterstützte Version falsch angezeigt
Fix: KNX: Profil bei DPT 1.x korrigiert

Fix: KNX: Profil bei DPT 9.028 korrigiert
Fix: MQTT: Problem bei der Kommunikation mit MQTT Client auf ESP Basis korrigiert
Fix: Discovery Instanzen zeigen nun auch eingerichtete Konfiguratoren (welche in IP-Symcon sind, jedoch nicht im Netzwerk)
Fix: Reaggregation aller Variablen konnte manchmal zwischendurch abbrechen
Fix: Z-Wave Geräte, die defekte SerienNummern liefern, sollten die webbasierte Konsole nicht mehr beeinflussen (Neu „Laden“ ist erforderlich)
Fix: Potenzieller VoIP Crash korrigiert
Fix: Potenzieller Connect-Dienst (Reconnect) Crash korrigiert
Fix: KNX Device mit EIS10, 16 Bit, Signed kann wieder eingestellt werden
Fix: KNX XML Import nutzt nun " Status", "-Status" und ".Status" in den GAs als Trigger für Status GAs
Fix: Der Windows Installer lässt bei Problemen kein Abbrechen zu und verweist auf unseren Support
Fix: Der Windows Installer stellt sicher, dass das IP-Symcon Tray und IP-Symcon Verwaltungskonsole vor der Migration beendet wurden
Fix: Archiv: Bei IP-Symcon Neustarts konnte bei der Rekonstruierung der Zähleraggregation in seltenen Fällen eine fehlerhafte Berechnung auftreten
Fix: Archiv: Werte, die im selben Zeitstempel passieren wurden nicht korrekt im Minimum/Maximum berücksichtig
Fix: Z-Wave: "Werte auslesen" Button bei "Benutzerdefinierten Parametern" funktioniert wie vorgesehen
Fix: Z-Wave: Sporadisch wurde beim Hinzufügen/Entfernen von Geräte nicht angezeigt, dass das Gateway bereit ist
Fix: Z-Wave: Z-Wave Instanzen, welche nicht mehr auf dem Gateway vorhanden sind, können direkt gelöscht werden (ohne Test-Pakete zu versenden)
Fix: Der Update-Kanal konnte aus dem sources.list.d Ordner nicht korrekt ermittelt werden
Fix: Die Migration der Daten unter Windows ignoriert die setup.exe, welche bis IP-Symcon 4.2 erstellt wurde
Fix: HomeMatic: Wenn eine "fremde" HomeMatic Instanz keine "Protocol" Property besaß konnte der Konfigurator nicht geöffnet werden
Fix: M-Bus: Suchfunktion im Konfigurator weiter verbessert
Fix: M-Bus: Der vorgeschlagene Name beinhaltet die Sekundäradresse
Fix: Z-Wave: Sofern ungültige Secure-Pakete empfangen wurden konnte IP-Symcon abstürzen
Fix: KNX: ~Switch Profil wird überall konsistent verwendet
Fix: Fix für die Abstürze in Verbindung mit dem WebSocket Rückkanal / RTSP Streams
Fix: SSDP: Anfrage auf 5 Sekunden erweitert, da einige Geräte zu langsam antworten
Fix: DNS-SD: Wartezeit auf 500ms erhöht um Zuverlässigkeit zu verbessern
Fix: EnOcean EEP D2-01-xx mit Leistungsmessung aber ohne Energieausmessung wird nun korrekt ausgewertet
Fix: Multicast Socket sollte wieder wie erwartet funktionieren, auch wenn nur ein Interface in der BindIP gewählt ist
Fix: Query String wird in $_GET Variable korrekt kodiert
Fix: Connect: Verbindungsabbrüche sollten nun schneller erkannt werden
Fix: PHP Module: Type Hints zu "array/object" sind erlaubt und werfen keine Warning
Fix: Sys_GetMemoryInfo unter Linux korrigiert (Available Memory wurde falsch berechnet)
Fix: M-Bus: Negative Werte werden nun nach ISO EN13757-3 ausgewertet
Fix: HasAction war nicht korrekt vorhanden
Fix: KNX: DPT 11 hat beim Senden vom Datum den Tag um 1 versetzt versendet
Fix: PHP Inkludepfad enthälte immer den Pfad zum "scripts" Ordner, damit auch in Ereignissen Dateien korrekt per include geladen werden können

Fix: Z-Wave: Empfange von einigen Kommandoklassen (z.B. CRC_16) auch Nachrichten im Untergerät, wenn nur das Hauptgerät diese als unterstützt melden.
Fix: Z-Wave: Bessere Überprüfung, ob die empfangenen Daten vollständig sind
Fix: Z-Wave: Erkennung von "fehlerhaften" Knoten verbessert
Fix: Z-Wave: Abfrage der Manufacturer Specific Klasse reduziert (fast alle Geräte antworten nicht korrekt und wir brauchen die Informationen eigentlich nicht)
Fix: Z-Wave: Fehlermeldungen im Debug bzgl. falscher Länge (sofern Security genutzt wurde) korrigiert
Fix: Abstürze bei RTSP Streams über eine SSL Verbindung korrigiert
Fix: Aktualisierung-Widget zeigt neue Versionen wieder korrekt an
Fix: DNS-SD Modul limitiert die Anzahl der Einträge auf 60.

WEBFRONT

Fix: WebFront Editor zeigt unsichtbare Elemente an, sodass diese auch wieder sichtbar geschaltet werden können
Fix: WebFront Editor konnte nicht mehr geöffnet werden

KONSOLE

Neu: Verbesserte Performance bei vielen Nachrichten in Meldungen oder im Debug Log
Neu: Google Maps wird erst geladen, wenn erforderlich
Neu: Fonts werden lokal geladen
Neu: Im Skript-Editor werden PHP-Schlüsselwörter als Vorschläge für die Autovervollständigung angeboten
Neu: Suchen/Ersetzen-Dialog sortiert die angezeigten Skripte
Neu: Dialog um in allen Skripten zu ersetzen fragt vorm Ersetzen, ob ungespeicherte Skripte gespeichert werden sollen
Neu: Mitteilungen können markiert und somit auch kopiert werden
Neu: Fortschrittsdialog beim Löschen von Objekten
Neu: Die Variablenaktionen "Addieren" und "Subtrahieren" verwenden für den Summanden/Subtrahenden nun die Nachkommastellen des Profils
Fix: Ladenachrichten beim Start der Konsole werden nun wie vorgesehen auf englisch übersetzt
Fix: Ladeicon ist wieder animiert ohne den Chrome einzufrieren
Fix: Ladeicon ist nun auch im Firefox durchgängig animiert
Fix: SVG Animation entfernt - Dies sollte das Einfrieren von Chrome verhindern
Fix: "Befehle testen"-Dialog gibt nun wie vorgesehen Fehlermeldungen aus
Fix: Typo im Adventskalender
Fix: Status Widget war seit der letzten Beta defekt
Fix: Schnelles Duplizieren von Konsolen (Tabs im Browser) konnte die Verbindung abbrechen lassen
Fix: Fonts wurden seit der letzten Beta nicht korrekt geladen
Fix: Speichern von Änderungen in HID und KNX Device (EIS) Instanzen funktioniert wieder
Fix: Prüfe Kompatibilität bei Upgrade vom Module Control zum Module Store
Fix: Sortiere Statusdialog wie vorgesehen
Fix: Speichere Login-Passwort wie vorgesehen
Fix: Modulbeschreibungen mit langen Wörtern werden umgebrochen und blenden keine Scrollbar mehr ein

Fix: Prüfe ob der zurückgegebene Link eines Buttons Zeilenumbrüche enthält und gebe ggf. passende Fehlermeldungen
Fix: Lade Icons für Icon-Auswahl iterativ, dies verhindert mögliche Verbindungsabbrüche beim Laden von sehr vielen Icons
Fix: Korrekte HEX-Darstellung im Debug-Log - Es wird nicht versucht gelegentlich Zeichen zusammenzuziehen, da diese als UTF-8 interpretiert werden
Fix: Module Control filtert die installierten SymconMisc-Module wieder wie vorgesehen
Fix: Wird in einem Konfigurator eine Instanz mit falscher ModulID geprüft kommt wie vorgesehen eine passende Meldung

V5.1->V5.2 (Q3/2019)

WESENTLICHE ÄNDERUNGEN

Voice over IP (VoIP)
Dynamische Instanzkonfiguration
Dokumentation direkt öffnen
Performanceverbesserung der Verwaltungskonsole

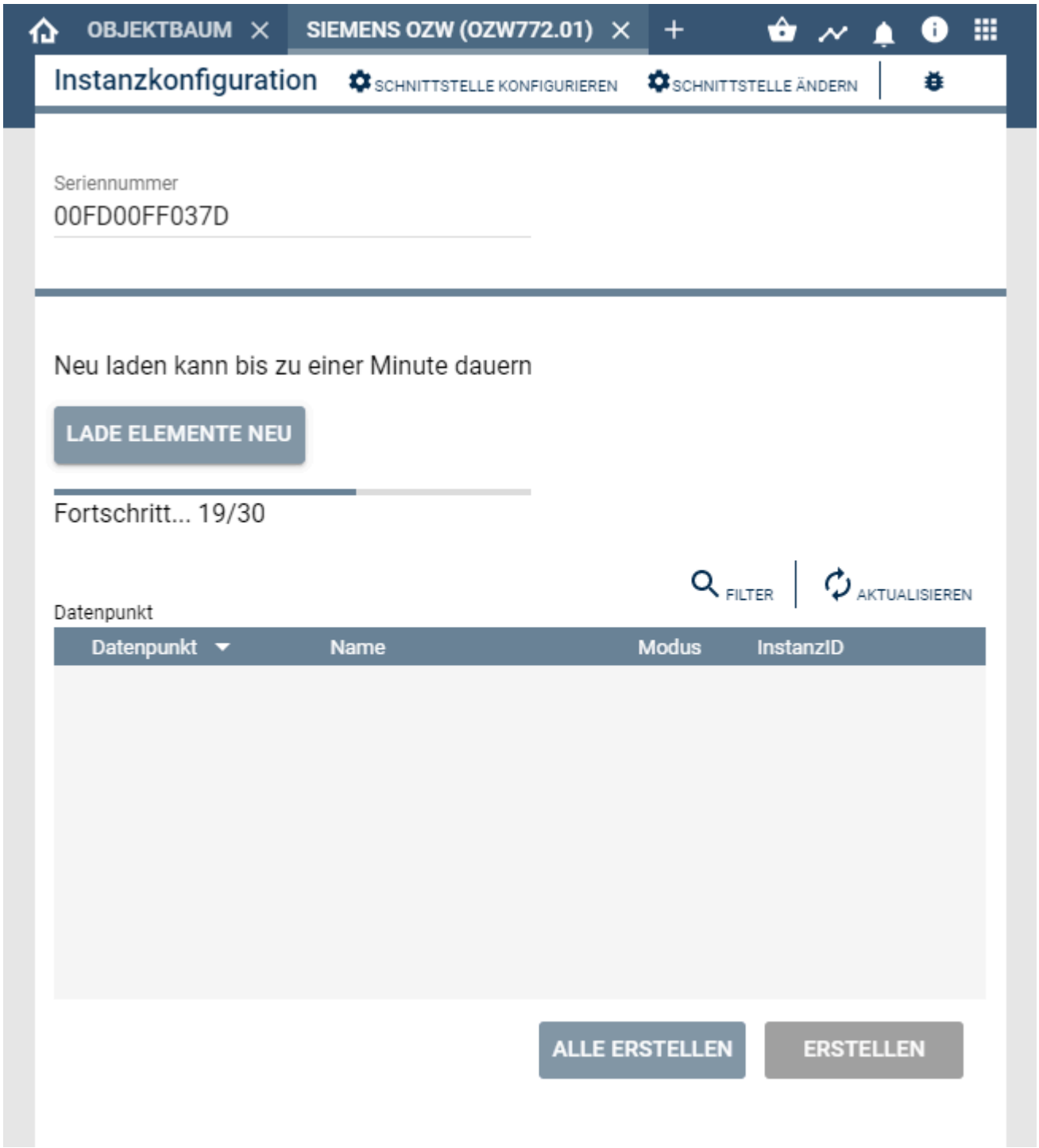
VOICE OVER IP (VOIP)

Das VoIP Modul kann Telefonverbindungen aufbauen und entgegennehmen. Es können DTMF Signale verarbeitet und Sounds abgespielt werden. So ist es möglich sich beispielsweise in einem Alarmfall automatisch durch einen Telefonanruf informieren zu lassen. Bei einem Anruf können die Zahlentasten ausgewertet und so beispielsweise Geräte geschaltet werden.



DYNAMISCHE INSTANZKONFIGURATION

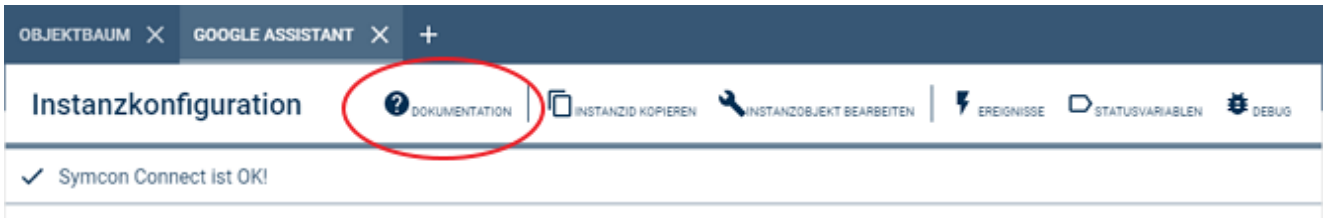
Die Konfigurationsseiten von Instanzen können sich nun dynamisch anpassen. So können unter anderem längere Ladezeiten eines Gerätes durch einen Ladebalken dargestellt oder individuell auf diverse Eingaben reagiert werden. Verwendung der Dynamik kann beispielsweise in den Z-Wave-Modulen, dem Notification Control und verschiedenen Konfiguratoren gesehen werden.



All diese Funktionen stehen selbstverständlich auch Modulentwicklern zur Verfügung. Zur dynamischen Anpassung wurden die Funktionen UpdateFormField und ReloadForm eingebaut und die möglichen Formularfelder um eine Vielzahl von Parametern erweitert.

DOKUMENTATION DIREKT ÖFFNEN

Die Dokumentation eines Moduls kann nun direkt aus der Instanzkonfiguration geöffnet werden. Kommen also während der Konfiguration eines umfangreichen Moduls Fragen auf, so können diese mit einem einzelnen Klick auf den Dokumentations-Button beantwortet werden.



PERFORMANCEVERBESSERUNG DER VERWALTUNGSKONSOLE

Die Performance der webbasierten Verwaltungskonsole wurde weiter verbessert. Durch die Verwendung von WebSockets werden Änderungen an IP-Symcon nun nahezu sofort in der Verwaltungskonsole wiedergespiegelt. Da verschachtelte Inhalte nun erst bei Bedarf geladen werden, ist die initiale Ladezeit der Verwaltungskonsole sowie das Öffnen einer Instanz beschleunigt.

MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 5.1 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 5.2 auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Es wird Raspbian Buster für Raspberry Pi erfordert! (Generelle Versionenübersicht)
Es wird Ubuntu 18.04 für Linux 64 Bit erfordert! (Generelle Versionenübersicht)
Für Microsoft Windows steht ausschließlich eine 64-Bit Version zur Verfügung.

MIGRATION (FÜR ENTWICKLER)

WebSocket Rückkanal von /api/wfc/{5-stellige-ID} zu /wfc/{5-stellige-ID}/api/ geändert
RTSP Proxy von /proxy/wfc/{5-stellige-ID} zu /wfc/{5-stellige-ID}/proxy/ geändert

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP 7.3.8 (Löst insbesondere unter Windows 10 (1903) Abstürze beim Starten)
Neu: VoIP Modul (SIP)
Neu: EnOcean: Konfigurator für die Geräteübersicht
Neu: EnOcean: Geräte, die eine eindeutige, verfügbare Geräte-ID zum Anlernen brauchen, können diese direkt per Knopfdruck bestimmen lassen. (Es wird automatisch die nächste freie ID gewählt)
Neu: Z-Wave: Kommandoklasse BARRIER_OPERATOR

Neu: Siemens OZW: Laden der verfügbaren Datenpunkte wird im Formular als Fortschrittsbalken angezeigt
Neu: Siemens SPS: Konfigurator für Siemens Logo 7 / Siemens Logo 8 VM-Adressen
Neu: LCN: Konfigurator lädt Module und Modulnamen mit einem schicken Fortschirttsbalken
Neu: LCN: Schwellwerte 4x4 können empfangen werden
Neu: LCN: Schwellwerte 1x5 und 4x4 können umgerechnet werden (wie bereits von den Variablen 1-12 gekannt)
Neu: xComfort: Neue Heizungsaktoren werden unterstützt
Neu: xComfort: Konfigurator unterstützt alle Kanäle vom 12-Kanal Heizungsaktor
Neu: WebSocket Support für WebHooks (Für Entwickler: WC_PushMessage zum Senden)
Neu: IRTrans Konfigurationsseite nutzt die neue Dynamik
Neu: Sys_GetNetworkInfo, Sys_GetBattery, Sys_GetProcessInfo sind jetzt überall verfügbar
Neu: Dynamik wird im Notification Control und WebFront Konfigurator genutzt
Fix: LCN: Konfigurator hat erstellte LCN Host-Kommando und Fingerprint Instanzen nicht korrekt erkannt

KONSOLE

Neu: Performance Verbesserungen bei Instanzseiten. Insbesondere bei z.B. Alexa, HomeKit oder Z-Wave merkbar
Neu: Das Aktualisierungswidget prüft nun auf neue IP-Symcon-Versionen
Neu: Dauert das Laden der Parameter eines Musters zur Variablenbearbeitung (in Ereignissen oder "Befehle testen") länger, kommt eine Ladeanimation
Neu: Gebe eine schöne Meldung aus, wenn in einem Ereignis oder in "Befehle testen" eine Funktion mit Parameter vom Typ Variant oder Array gewählt wird
Neu: Schöner Beschreibungstext für Aktionen im Ereignisdialog von Instanzen
Neu: Bei Konfiguratoren kann 'position' und 'info' mit angegeben werden
Neu: Bei Konfiguratoren wird eine Warnmeldung angezeigt, wenn die 'location' an der falschen Stelle definiert wurde
Neu: Wochenplan und Ereignisdialog von Instanzen zeigt in den Aktionen eine schönere Beschreibung an, sofern möglich
Neu: Tabs der letzten Sitzung werden erst bei Bedarf geladen. Dies sollte die initiale Ladezeit, insbesondere bei vielen Tabs, deutlich reduzieren
Neu: Aktualisierungswidget wurde überarbeitet und prüft nun auch auf Aktualisierungen von IP-Symcon
Neu: Sind ein oder mehrere Typen im Objektbaum ausgeblendet, so wird der "Typen"-Button eingefärbt
Neu: Konsole kommuniziert über WebSockets
Neu: Ist ein Kontextmenü offen wird bei Betätigung von Escape das Menü und nicht zusätzlich der darunterliegende Tab/Dialog geschlossen
Neu: Enthält ein Modul im Module Store keine Discovery- oder Konfigurator-Instanzen, so biete die Erstellung von Kern-Instanzen an bzw. Geräte-Instanzen, falls auch keine Kern-Instanzen existieren
Neu: Dialog zum Stream erstellen wurde erweitert. Nun können Benutzername und Passwort in seperaten Feldern eingegeben werden, außerdem kann der Streamtyp (RTSP oder MJPEG) ausgewählt werden
Neu: Die Position eines Objektes kann im Objektbaum durch einen Doppelklick auf die Positionsspalte bearbeitet werden
Neu: Dialog zum Instanz erstellen enthält nun die Auswahl "Kein Hersteller" und "(Sonstige)" wurde zu "(Gerät)" umbenannt
Neu: Verbesserte Performance bei vielen Einträgen pro Sekunde ins Meldungsfenster
Neu: Instanz Debug kann ins Logfile weitergeleitet werden
Neu: Im Instanz-Erstellen Dialog gibt es nun eine Auswahl für den Modultyp
Neu: Die Liste im Status-Widget erlaubt das Expandieren von langen Nachrichten

Neu: Module-Control schlägt Alexa als Standard-Modul vor
Neu: In Konfiguratoren können nun mehrere Instanzen zum Erstellen ausgewählt werden
Neu: Im RowLayout wird ein Zeilenumbruch gemacht, wenn zu viele Elemente enthalten sind
Fix: Alle erlaubten Dokumenttypen können nun wie vorgesehen im Dokument-Dialog ausgewählt werden
Fix: Lange Labels mit Links werden nun korrekt formatiert
Fix: Browsererkennung funktioniert nun auch bei iOS 13
Fix: Einige verwendete Bibliotheken wurden aktualisiert
Fix: Im Demo-Widget konnte "Abgelaufen" stehen, obwohl die Demo noch lange gültig ist
Fix: Konfiguratoren die keine Namensspalte haben funktionieren nun korrekt
Fix: Warte mit dem Springen zu neu erstellten Objekten bis diese korrekt einsortiert wurden
Fix: Aufräumen im Archiv erfolgt nun iterativ und nicht für alle Variablen parallel, was bei größeren Archiven zu Problemen führen konnte
Fix: Kann ein HTML-Link nicht direkt geöffnet werden, so öffne einen Dialog, der dies anbietet
Fix: Nachrichten vom Typ EM_CHANGESCHEDULEGROUPPOINT werden nun wie vorgesehen ausgewertet
Fix: Wird im Archiv eine Zeitspanne gelöscht, so erfolgt eine schöne Fehlermeldung, wenn die "Von"-Zeit nach der "Bis"-Zeit liegt
Fix: Slider in der Instanzkonfiguration als HorizontalSlider oder als Teil des Testcenters skalieren nun in Ihrer Schrittgröße, damit beispielsweise Variablen mit dem Profil ~Intensity.1 gut schaltbar sind
Fix: Es wird verhindert, dass durch die Verwendung der Browserhistorie Tabs zu entfernten Objekten geöffnet werden
Fix: Wird ein Skript von ausserhalb des Skripteditors ausgeführt, wird dies wie vorgesehen nicht mehr im Skripteditor angezeigt
Fix: Bei Konfiguratoren kann die 'location' korrekt auch bei Ketten verwendet werden
Fix: Ereignisse auf Statusvariablen werden wie vorgesehen erlaubt und nicht mit einer Nachricht kommentiert
Fix: Lange Beschreibungen von Parametern im Skripteditor können nicht mehr überlaufen
Fix: Das verwendete Framework Polymer wurde auf die neueste Version aktualisiert
Fix: Uhrzeiten für Aktionen im Wochenplan werden korrekt ausgewertet
Fix: Aktualisiere, die Aktionsauswahl, wenn wiederholt das gleiche Ziel für ein Ereignis gewählt wird, da sich die Schaltbarkeit von Variablen ändern könnte
Fix: Skins über das Aktualisierungswidget zurücksetzen funktioniert nun wie vorgesehen
Fix: Module aus dem Module Store können wie vorgesehen über das Aktualisierungswidget aktualisiert werden
Fix: Farbauswahl von Transparent aus funktioniert wieder wie vorgesehen in allen Browsern
Fix: Kompaktmodus hat den Switch nun links vom Text (wurde manchmal mit aktivierbaren Spezialschaltern verwechselt)
Fix: Dialoggröße von Auswahlbäumen passt sich dynamisch an die Anzahl dargestellter Objekte an
Fix: Fehlende Systemvariablen wurden der Autovervollständigung im Skripteditor hinzugefügt
Fix: Bei Modulen ohne Hersteller wird der Typ im Instanz-erstellen-Dialog wieder wie vorgesehen mit Klammern dargestellt
Fix: Die Größe von Spalten mit prozentualer Breite, z.B. im Module Control, kann wieder wie vorgesehen angepasst werden
Fix: Ausklappmenüs im Skripteditor werden wieder wie vorgesehen geschlossen, wenn daneben geklickt wird

WEBFRONT

Neu: Graphen in einem Multi-Chart können aus-/eingeblendet werden
Neu: Medien vom Typ "Dokument" (z.B. PDFs) können direkt heruntergeladen werden
Neu: Editor kann nun auch korrekt auf Tablets verwendet werden

Fix: Sichtbarkeit der einzelnen Elemente konnte nicht verändert werden im Gegensatz zur Legacy Konsole
Fix: Im WebFront Konfigurator kann HD/Extrema nicht mehr als Standard aktiviert werden, solange Rohdaten aktiv ist
Fix: Beim Öffnen eines WebFront über den Connect Dienst welcher ein Kennwort benötigt gibt es keine unnötige 10 Sekunden Wartezeit mehr
Fix: WebFront: Schaltbefehle konnten beim Nutzen von Slidern doppelt versendet werden
Fix: WebFront: HD/Extrema wurden ggf. aktiviert, obwohl die Rohdaten Ansicht aktiv sein sollte
Fix: Sortierung wurde für neue Objekte nicht korrekt übernommen
Fix: Sortierung wurde für neue Kategorien nicht korrekt übernommen
Fix: Kategorie wurde nicht korrekt abgewählt wenn diese versteckt/gelöscht wurde

PHP-SDK

Neu: Formular-Dynamik
Neu: In der module.json gibt es ein neues "url" Feld, welches einen String annimmt. Dies kann ein Link direkt zur Moduldokumentation sein und wird in Zukunft in der Instanzkonfiguration verfügbar sein, sofern angegeben.
Neu: In der library.json im Feld Versionen können nun beliebige Versionen (z.b. 1.0b) angegeben werden
Neu: Die IPS_GetLibrary Funktion liefert im Feld Version nun einen String anstatt einem Integer Wert

SONSTIGES

Neu: Erweiterte Bedingungen unterstützen den Vergleich auf einen Wochentag
Neu: Im Location Control kann das Interval für Azimuth/Altitude eingestellt werden
Neu: Im Location Control können für "IstTag" auch die Offsets verwendet werden
Neu: EnOcean Gateway liest das Chipset aus (z.B. TCM310 oder TCM515)
Neu: Duplizieren von Objekten übernimmt jetzt Konfiguration der Status-Variablen
Neu: Web-Server kann nun z.B. aus dem /webfront/user/ Ordner mp4 Dateien korrekt in Safari abspielen
Neu: IPS_EnableDebugFile($id) aktiviert das Schreiben vom Debug-Logs in eine Datei (z.B. Linux: /var/log/symcon/debug_id.log)
Neu: DiscoveryServer versendet Connect-Adresse (sofern verfügbar), sodass die Apps direkt den Zugriff über Connect einrichten können *
Neu: Util Control erlaubt das automatische Konvertieren von PHP Short Tags (<?) in die normalen Tags (<?php)
Neu: Ereignisse welche per IPS_SetScriptTimer erstellt werden sind initial versteckt
Fix: WebHook Aufruf wird an den "besten" gefundenen Hook weitergeleitet
Fix: KNX Dicovery setzt auch die BindIP korrekt
Fix: Authentifizierung für RTSP Streams war unvollständig (benötigt Symcon Mobile 5.2)
Fix: Bei der Weiterleitung von z.B. /console auf /console/ wurden ggf. vorhandene Parameter verworfen
Fix: Es konnte ein bad_variant_access Fehler auftreten, sofern intern nicht alle Variablentypen unterstützt wurden
Fix: Modifizierte Dateirechte von Git Submodulen werden nicht mehr als Lokale Änderung betrachtet

ÄNDERUNGEN 5.2.X

Neu: PHP 7.3.5
Neu: IPS_ConnectInstance prüft, ob die Verbindung sinn ergibt (z.B. Kern Instanzen haben keinen Datenfluss) und ob die Datenfluss GUIDs passen
Neu: Konsole: In Ereignissen gibt es ein neues Template zum Umschalten von Boolean Variablen
Neu: Konsole: Aktualisierungs-Widget berücksichtigt nun auch Skins und den Store
Neu: Konsole: Falsch konfigurierte Ereignisse werden im Baum markiert
Neu: WebFront: Per Parameter ?password=xxxx kann ein Kennwort-geschütztes WebFront direkt geöffnet werden
Neu: WebFront: "Nicht unterstützt" Meldung wird angezeigt, sofern iOS 9.x oder älter verwendet wird
Neu: WebFront: Reverse-Proxy mit Suffixen im Pfad sollte nun korrekt funktionieren
Fix: Docker: DBus und Avahi wurden nicht automatisch gestartet, wodurch das HomeKit Modul nicht genutzt werden konnte
Fix: KNX: Shutter wirft bei aktivierter Lamellensteuerung keine Fehlermeldung, wenn Stop mehrmals betätigt wird
Fix: KNX: Wenn im Shutter falsche EIS Typen empfangen wurden, wurde eine kryptische Fehlermeldung generiert. Die neue orientiert sich an der EIB Geräte Instanz
Fix: MQTT Server wirft bessere Fehlermeldung wenn die noch nicht unterstützten QoS > 0 Pakete empfangen werden
"Memory" Option nicht berücksichtigt wurde
Fix: Z-Wave: Control Klasse "SCENE_ACTIVATION" wird mit zu den unterstützten Klassen gewertet (Löst Probleme mit Scenen Activiation)
Fix: WebFront: CircleSlider und Farbrad können wieder genutzt werden, wenn vorher Werte gespeichert wurden
Fix: Konsole: Abstände der Elemente beim Test-Center verbessert
Fix: Konsole: Module Store überspringt beim Prüfen fehlerhafte alte Module
Fix: WebFront: IMAP/POP3 Postfächer werden wieder geladen
Fix: SymconMisc korrekt in "Misc Modules" beim Migrieren umbenannt
Fix: Skriptausgabe ist im Firefox wieder kopierbar
Fix: Module Store prüft auf Updates in der korrekten Sprache
Fix: Konsole: Einige Ereignisse konnte nicht bearbeitet werden
Fix: Konsole: Im Store kann nach Begriffen mit Leerzeichen gesucht werden
Fix: WebFront: Wenn ein ReverseProxy verwendet wird, versuchen wir den Pfad in der WebSocket Verbindung korrekt abzubilden
Fix: location kann nun wie vorgesehen auf für Ketten im Configurator-Element verwendet werden
Fix: Uhrzeiten für Aktionen im Wochenplan werden korrekt ausgewertet* Fix: Skins über das Aktualisierungswidget zurücksetzen funktioniert nun wie vorgesehen
Fix: Module aus dem Module Store können wie vorgesehen über das Aktualisierungswidget aktualisiert werden
Fix: dS Expert Instanz konnte nicht erstellt werden
Fix: WebFront: RTSP Streams werden zentriert und skaliert
Fix: WebFront: Skin Builder ignoriert alle @eaDir Verzeichnisse
Fix: Konsole: Sofern falsche Elemente genutzt werden, wird nun eine bessere Fehlermeldung geworfen
Fix: Konsole: Hintergrund bei "Befehl ausführen" Dialog wird korrekt vergrößert
Fix: Konsole: Kontextmenü auf Firefox unter MacOS funktioniert nun korrekt
Fix: Konsole: Wird ein Skript von ausserhalb des Skripteditors ausgeführt, wird dies wie vorgesehen nicht mehr im Skripteditor angezeigt

Fix: Abstürze unter Linux/Mac beim Aktualisieren von Modulen über das Module Control/Store gelöst
Fix: Es wird geprüft, ob eine Datenfluss-Verbindung eine Endlosschleife ergibt. Sollte dies der Fall sein, wird die Verbindung nicht erlaubt.
Fix: PHP-SDK: HasActiveParent liefert false, wenn es keinen Parent gibt.
Fix: WebFront Konfigurator zeigt wieder korrekt die Geräte in der Geräteliste an
Fix: _generated.inc.php sollte beim Aktualisieren von Modulen nicht mehr kaputt gehen
Fix: Beim Entladen der Module werden die Funktionsverweise auch gelöscht, sodass diese nicht mehr doppelt und dreifach in "Befehle testen" auftauchen
Fix: Konsole: Revisionsnummer war 11 statt 12 Zeichen lang
Fix: Konsole: Farbauswahl von Transparent aus funktioniert wieder wie vorgesehen in allen Browsern
Fix: WebFront: Bessere Fehlermeldung, wenn ein falscher Platzhalter verwendet wird
Fix: "Gerät reaktivieren" ist für Z-Wave auch in der webbasierten Konsole verfügbar
Fix: xComfort Konfigurator kann wieder die neuen "Schalter" hinzufügen
Fix: Z-Wave: Absturz beim Empfang von defekter BATTERY_REPORT Pakete wird abgefangen
Fix: Legacy Konsole: Öffnen des neuen WebFront Editors ist jetzt auch über Connect Dienst möglich
Fix: Wenn ein Modul in den "modules" Ordner initial kopiert wurde, konnte es nicht per MC_ReloadModule geladen werden
Fix: Beim Duplizieren von vielen Objekten über die Verwaltungskonsole konnte IP-Symcon hängen bleiben
Fix: Unterkanäle von Z-Wave Geräten können in der webbasierten Konsole korrekt bearbeitet werden
Fix: M-Bus: "stol" Fehler beim Auswerten von M-Bus Zählern korrigiert
Fix: Legacy Konsole: Der neue Editor kann korrekt geöffnet werden, wenn der Connect Dienst genutzt wird
Fix: Konsole: Beim Hinzufügen und Bearbeiten von Gruppen im Wochenplan betreffen Änderungen der neuen Gruppe wie vorgesehen keine anderen Gruppen
Fix: Konsole: Meldungen zu Statusvariablen ohne Aktion bei Ereignissen angepasst
Fix: Konsole: Wird eine Variable, die nie geändert wurde, im Archiv geöffnet, dann wird eine entsprechende Meldung angezeigt
Fix: Konsole: Die Bearbeiten-Dialoge im Archiv können nun wie vorgesehen erneut geöffnet werden, nachdem geloggte Werte hinzugefügt wurden
Fix: Race Condition beim Timer Pool gelöst, welcher ggf. endlose "Warte auf Timer" Fehler auslösen konnte
Fix: Duplizieren beachtet jetzt auch deaktivierte Standardaktionen
Fix: Suchen/Einrichten von D2-* EnOcean Geräten wieder möglich
Fix: MQTT Server (Broker) hat Eingabe für Benutzername/Passwort
Fix: Z-Wave: Wenn direkt beim Senden ein Fehler aufgetreten ist, wurden die Retries nicht berücksichtigt
Fix: KNX Shutter: Wenn auf der GA 0/0/0 Pakete gesendet wurde hat der Shutter diese fälschlicherweise ausgewertet
Neu: Module Store informiert über Module, welche eine neuere IP-Symcon Version erfordern
Fix: Fehlermeldung beim UDP Socket, sofern der Host leer gelassen wurde
Fix: Weiteres Absturzproblem bei unvollständigen Z-Wave Battery Nachrichten gelöst
Fix: Z-Wave: Schalter für "Erzwinge Simple Assoziationen" fehlte in der Legacy Konsole
Fix: LCN Modul Konfigurator: Instanzen für Fingerprint/Host Kommando werden korrekt erkannt
Fix: WebFront: Schaltbefehle konnten beim Nutzen von Slidern doppelt versendet werden

Fix: WebFront: HD/Extrema wurden ggf. aktiviert, obwohl die Rohdaten Ansicht aktiv sein sollte
Fix: VoIP: Erlaube Anrufer Liste wurde nicht korrekt ausgewertet
Fix: VoIP: VoIP_Disconnect kann nun "klingelnde" (=nicht angenommene) Anrufe abbrechen
Fix: Z-Wave: Einige Befehle sollte nicht als gesendete Pakete zählen
Fix: Z-Wave: Für einige Befehle sollte es keine Retries geben (insbesondere NIF-Anfragen)
Fix: Konsole: Erweiterte Bedingungen können nicht mehr unbeabsichtigt gelöscht werden
Fix: Beim Beenden konnte IP-Symcon Abstürzen, sofern OPcache aktiviert war
Fix: Konsole erkennt Safari auf iOS 13 korrekt und zeigt keinen Warnhinweis mehr an
Fix: Spalten gehen nicht mehr kaputt, wenn man die Positionsspalte einblendet und dann neu Lädt (Dies ist der Webinar Bug!)
Fix: Neu angelegte Ereignisse konnten unter bestimmten Bedingungen nicht mehr geöffnet werden

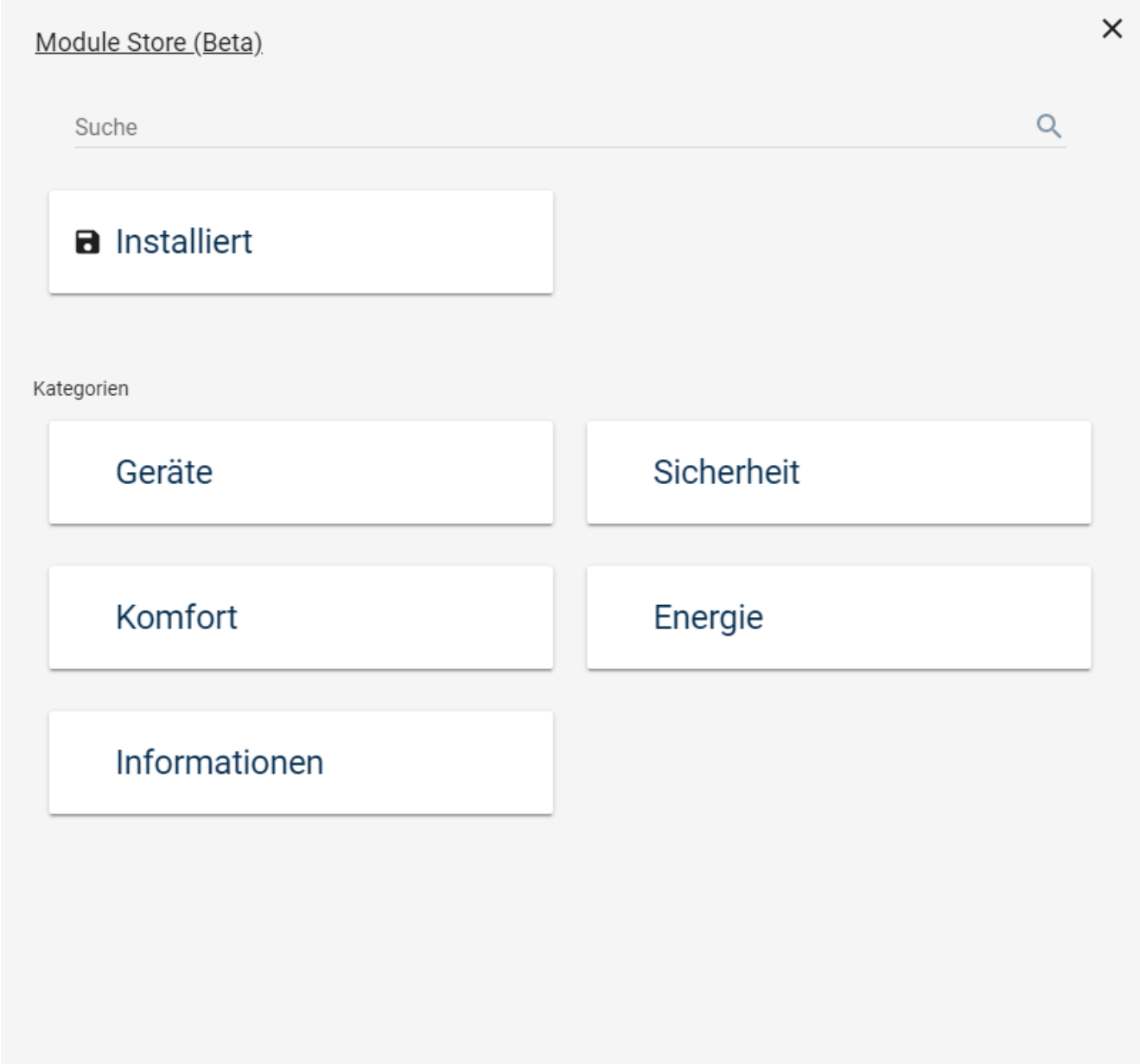
V5.0->V5.1 (Q2/2019)

WESENTLICHE ÄNDERUNGEN

Module Store (Beta)
RTSP-Streams
Z-Wave überarbeitet
MQTT (Beta)
Datenimport im Archiv
PHP 7.3
Performance der Konsole

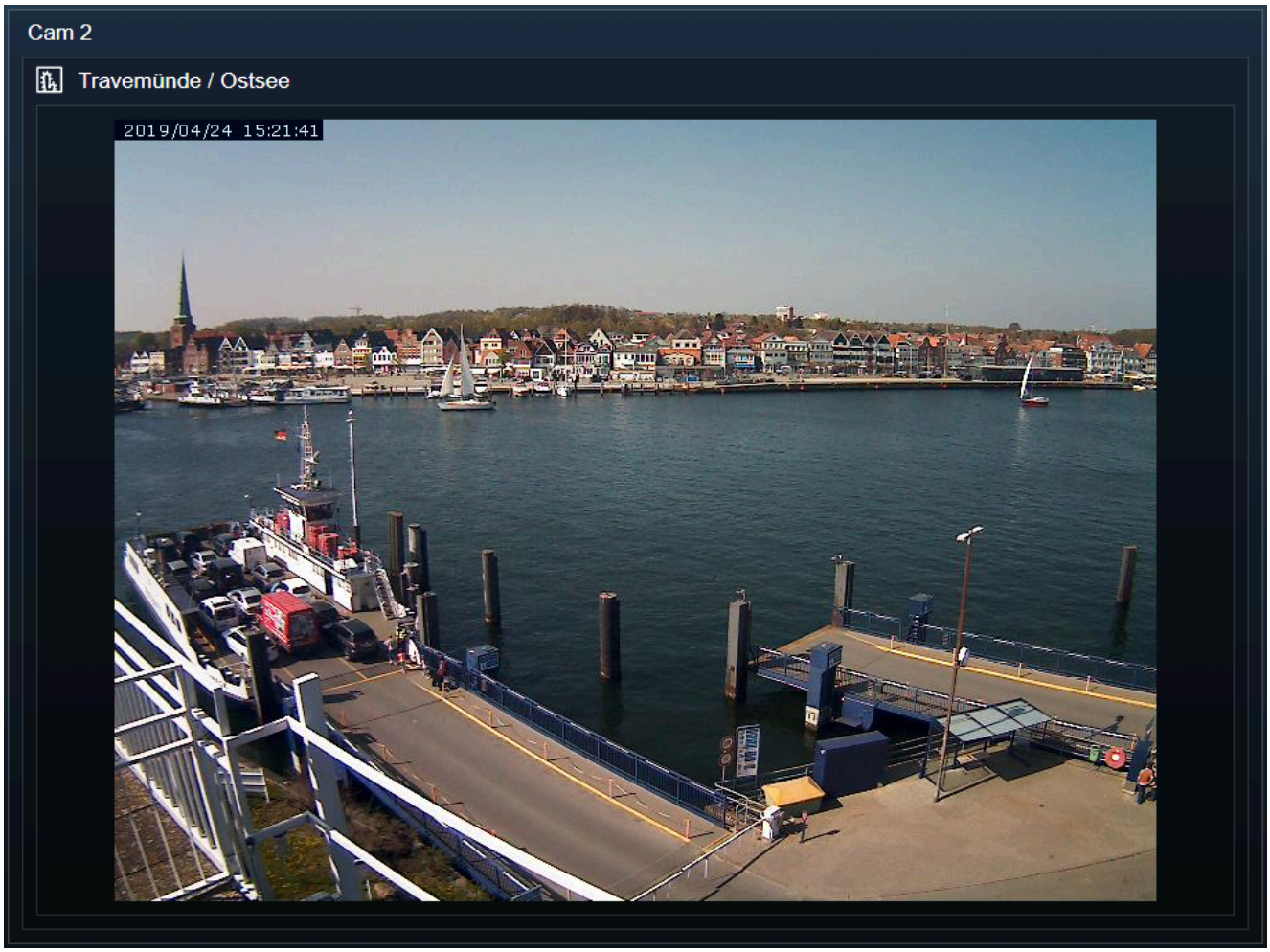
MODULE STORE (BETA)

In der webbasierten Konsole kann nun auf den Module Store zugegriffen werden. Hier können angebotene Module von anderen Benutzern komfortabel durchsucht und installiert werden. Aktuell ist der Module Store als Beta verfügbar und wird im Laufe der kommenden Versionen noch um diverse spannende Features erweitert.



RTSP STREAMS

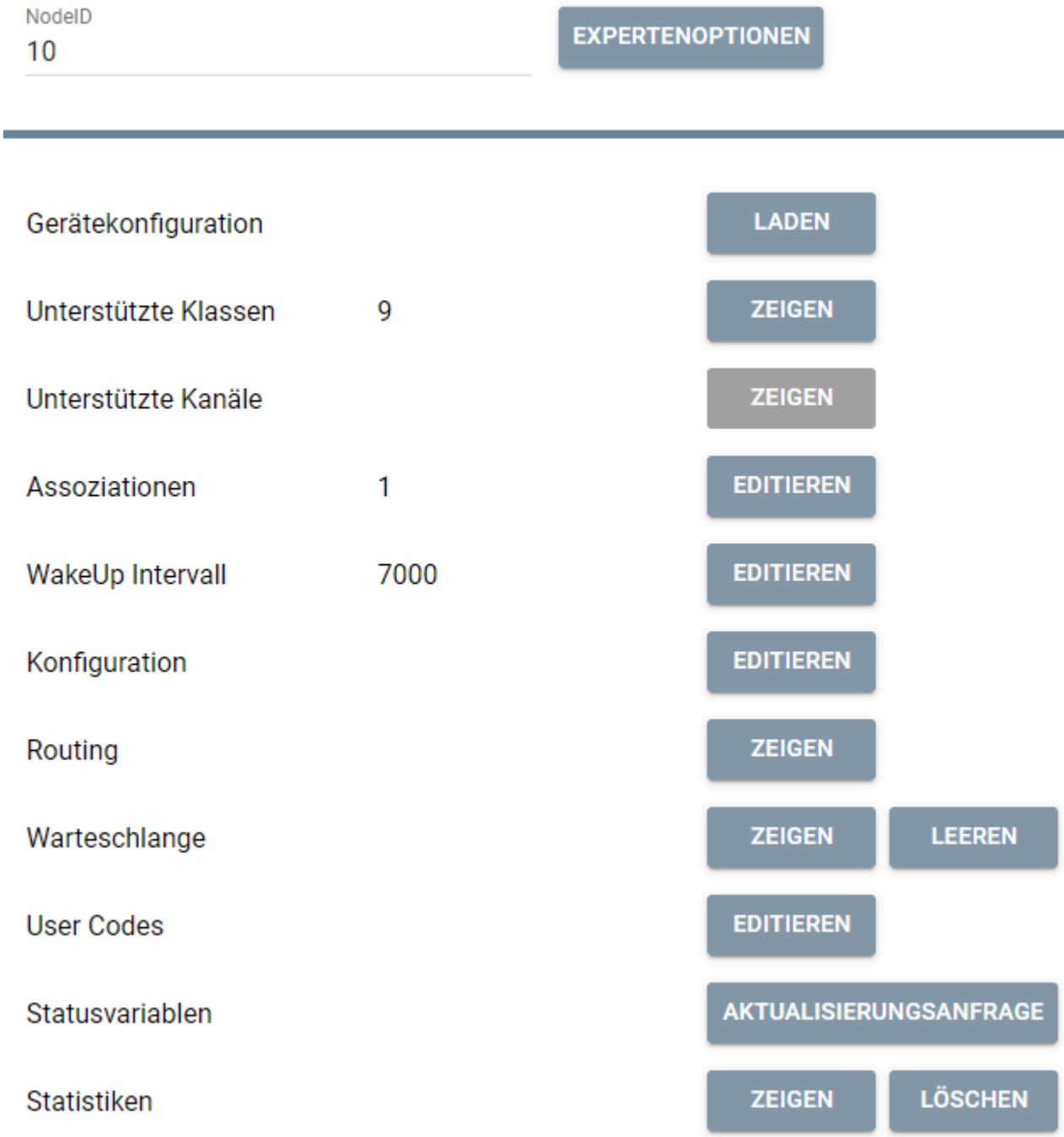
Über RTSP Streams können nahezu alle handelsüblichen Kameras in IP-Symcon eingebunden werden. Da IP-Symcon hier als Verteiler agiert, können auch bei Kameras, die eigentlich nur einen Empfänger zur Zeit unterstützen, mehrere Benutzer gleichzeitig auf den Stream zugreifen.



Aktuell funktionieren die RTSP Streams im Chrome, Firefox, Opera und Android (inkl. Apps ab Version 5.1). Zusätzlich müssen RTSP Streams aktuell mit h264 kodiert werden. Die Anzahl von RTSP Streams, die gleichzeitig angeboten werden können, hängen von der verwendeten Version ab, siehe Funktionsübersicht .

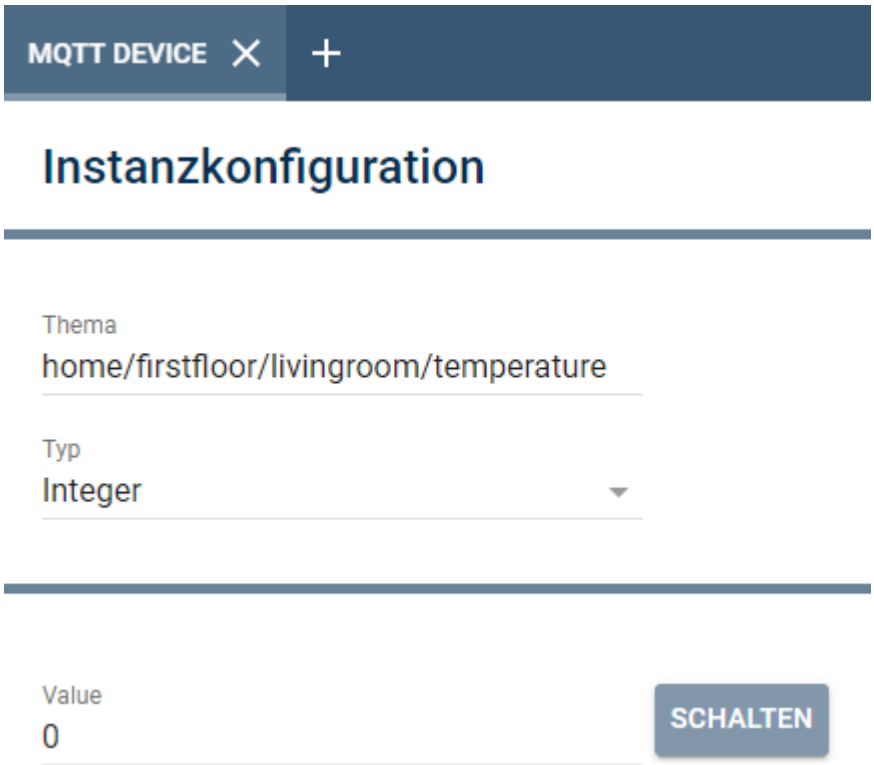
Z-WAVE ÜBERARBEITET

Die Unterstützung von Z-Wave wurde komplett überarbeitet. Beim Laden der Gerätekonfiguration werden nun alle Daten des Gerätes eingelesen und stehen direkt zur Verfügung. Zusätzlich kann nun eine noch größere Vielfalt von Z-Wave-Geräten eingebunden werden. Mehr Details zu den unterstützten Geräten finden Sie hier .



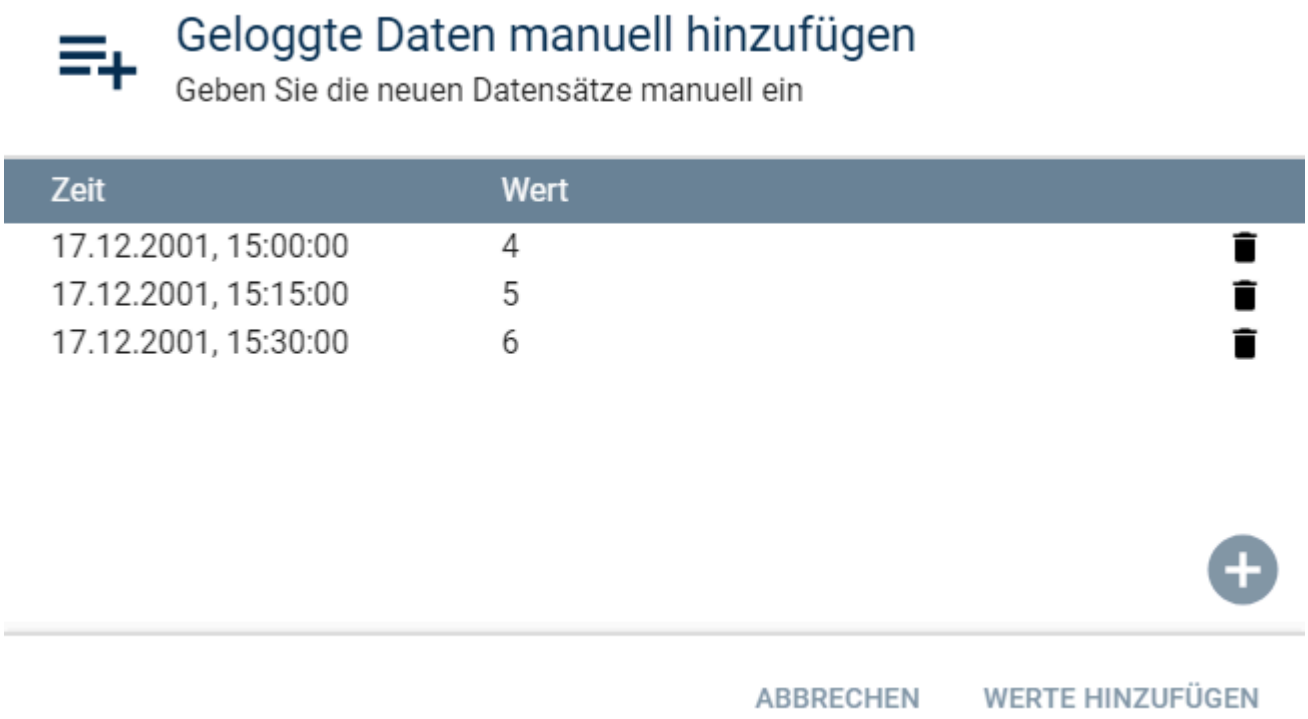
MQTT (BETA)

MQTT wird nun von IP-Symcon nativ unterstützt. Mit Version 5.1 startet die Open Beta der MQTT Anbindung, welche die grundlegenden Funktionen von MQTT anbietet und damit die meisten MQTT-Geräte unterstützt. Die vollständige Anbindung von MQTT kommt mit Version 5.2.



DATENIMPORT IM ARCHIV

Es ist nun möglich, die archivierten Werte einer Variablen im Nachhinein zu erweitern. So können beispielsweise Verbräuche eingetragen werden, welche ermittelt wurden bevor sie als Variable in IP-Symcon verwaltet wurden oder Daten nachgetragen werden, falls die Verbindung zwischen Sensor und IP-Symcon kurzzeitig gestört wurde. Es ist sowohl möglich die neuen Daten als Datei einzulesen als auch die Werte direkt in IP-Symcon einzutragen.



PHP 7.3

Die verwendete PHP-Version wurde von 7.2 auf 7.3 aktualisiert. Dies bietet verschiedene neue Funktionen. Details zur neuen Version können hier gefunden werden.

PERFORMANCE DER KONSOLE

Die Performance vom Objektbaum und vergleichbaren Elementen wurde deutlich verbessert. Nun können auch Kategorien mit mehr als 1000 Kindern in weniger als einer Sekunde ausgeklappt werden und auch das wiederholte automatisierte Löschen und Erstellen von Objekten stört nicht beim flüssigen Arbeiten.

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 5.0 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 5.1 auftreten.



Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Es wird Raspbian Stretch für Raspberry Pi erfordert! (Generelle Versionenübersicht)
Es wird Ubuntu 18.04 für Linux 64 Bit erfordert! (Generelle Versionenübersicht)
Für Microsoft Windows steht ausschließlich eine 64-Bit Version zur Verfügung.
Upgrade von PHP 7.2 auf PHP 7.3. (Upgrade auf 7.3)

MIGRATION (FÜR PHP-MODUL ENTWICKLER)

Das WebFront benötigt mindestens iOS 10.x aufgrund der Migration auf Dart 2.
Mit Version 5.1 werden die neuen Funktionen GetStatus und HasActiveParent angeboten. Hier kann es bei gleichnamigen Funktionen zu Konflikten kommen.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP 7.3.0
Neu: MQTT Server (ehemals Broker genannt) (Beta)
Neu: RTSP Streams können hinzugefügt werden
Neu: Z-Wave komplett überarbeitet
Neu: M-Bus: Es werden viele neue Geräte (insbesondere mit weiteren Telegrammen) unterstützt
Neu: Archiv kann Daten importieren

Neu: HomeMatic Konfigurator liest die Gerätenamen / Räume per HomeMatic Script aus (Bitte Firewall in der CCU anpassen!)
Neu: HomeMatic unterstützt Authentifizierung der CCU3
Neu: HomeMatic unterstützt Verschlüsselung der CCU3
Neu: KNX DPT 249.*
Neu: WebServer unterstützt WebSockets
Neu: digitalStrom (Expert Modul)
Neu: Presence Control
Neu: Connect Dienst Backend von libssh2 auf libssh gewechselt (Mehr Performance, Löst Probleme mit DS-Lite Anschlüssen)
Neu: Connect Dienst überprüft periodisch, ob das Eigenheim wirklich über ipmagic.de verfügbar ist.
Neu: Wir kompilieren fast alle Abhängigkeiten direkt in IP-Symcon rein, damit diese von der Versionen für alle IP-Symcon Betriebssysteme gleich sind. Damit sollte übrigens die Installation auf andere Linux-Derivate einfacher werden (auch wenn wir es nicht offiziell unterstützen)
Neu: Event Control verbindet fehlerhafte Instanzen nicht mehr jede Minute sondern mit Abstand (2^Fehlversuche Minuten). Mindestens jedoch einmal am Tag.
Neu: Profil für pH Werte
Neu: Status Code für Instanzen geändert: 201 wurde auf 105 geändert, da der 200+ Raum für benutzerdefinierte Fehler reserviert war
Neu: .well-known Ordner wird in den beschreibbaren Bereich gemappt (IPS_GetKernelDir())
Neu: Undokumentierte Funktion IPS_GetOptionList
Neu: IPS_GetReferenceList für alle Instanzen
Neu: Spezialschalter für Basis Authentifizierung des WebFront User Ordners, welche auch für den 3777 Server (inkl. Connect) gelten.
Neu: Skripte haben neue Felder: ScriptUpdated, ScriptCRC, ScriptSize (Neue Nachricht SM_UPDATE)
Neu: Neue Spezialschalter: ServerMaxPostSize, MaxLoginAttemptsBeforeSlowdown, MaxLoginAttemptsBeforeLockdown, MaxLoginAttemptsSlowdownDuration, MaxLoginAttemptsLockdownDuration, MaxLoginAttemptsSlowdownWaitTime (Doku)
Neu: Interne Übersetzungen sind nun aufgeteilt in einzelne Dateien im "locales" Ordner
Neu: LCN Module-Konfigurator legt Instanzen in einer passenden Kategorie an
Neu: Systemvariablen $_IPS['OLDUPDATED'] / $_IPS['OLDCHANGED'] für Ereignisse hinzugefügt

Fix: KNX DPT 13.002 korrigiert (verhält sich jetzt wie die ETS)
Fix: KNX Shutter: Stop konnte nicht immer ausgeführt werden
Fix: KNX DPT 242: Bad Variant Access korrigiert
Fix: KNX wirft keine Fehler mehr, wenn "leere" cEMI Pakete empfangen werden
Fix: DPT_WriteDPT17 lässt IP-Symcon nicht mehr abstürzen
Fix: Leerzeichen um eingefügte Adressen im Module und Skin Control können keine Fehler mehr verursachen
Fix: Multicast Modul zeigt direkt nach Erstellen den Wert für 0.0.0.0 zur Auswahl
Fix: INF/NaN wird bei SetValue* mit Fehlermeldung bemängelt
Fix: Probleme mit ALL2x Modul korrigiert
Fix: EIB/KNX-Formulare in der webbasierten Konsole speichern zusätzliche Adressen nun korrekt als Zahlen ab
Fix: Legacy Konsole zeigt im Location Control korrekt die Offset-Namen an
Fix: Wenn der Editor aktiviert war, konnte Objekte nicht geschaltet werden
Fix: Alte XML-Settings Relikte von IP-Symcon 2.5 und älter werden entfernt

Fix: HomeMatic KeepAlive differenziert je nach Rückkanal und kann somit Abbrüche für jeden Kanal einzeln erkennen
Fix: HomeMatic Geräte Instanzen speichern nun einige Meta-Daten, um die Aktion einer Variable nach dem Neustart korrekt widerherzustellen
Fix: Siemens OZW hat bei Datenpunkten mit Schreibmöglichkeit diese nicht korrekt eingerichtet
Fix: LCN Gateway ist nicht mehr im Fehlerzustand, sofern der I/O geschlossen ist
Fix: GUIDs werden bei Funktionsaufrufen auf Gültigkeit geprüft
Fix: GUIDs von KNX RGB/KNX RGBW korrigiert (Es fehlten die {})
Fix: LibraryID/ModuleID werden automatisch nach "UpperCase" konvertiert. (Wird mit einer netten Warning bestätigt)
Fix: Noch nicht unterstützte EnOcean Paket Typen werfen keine Fehler mehr, sondern werden im Debug Log protokolliert
Fix: xComfort: Der Anzeige der Firmware fehlen führende Nullen

KONSOLE

Neu: Der Module Store (Beta)
Neu: Umbau des Objektbaums und anderen Listen-/Baumstrukturen für deutlich bessere Performance
Neu: Im Archive Control können die Einstellungen zu einer Variablen alternativ durch einen Doppelklick geöffnet werden
Neu: Abbrechbarer Fortschrittsdialog beim Erstellen von allen Instanzen in einem Konfigurator oder beim Reaggregieren aller Variablen
Neu: Sicheres Löschen (Prüfen von Referenzen vorm Löschen) ist nun auswählbar
Neu: Variablen mit ungültiger Aggregation werden entsprechend im Archiv markiert
Neu: Lokale Sprachen wie de_AU werden nun bei der Übersetzung von Modulen beachtet und primär angewandt, falls vorhanden
Neu: Bessere Fehlermeldungen, wenn es beim Verbinden Probleme gibt, insbesondere bei deaktiviertem Fernzugriff
Neu: Von der Instanzkonfiguration und dem Skripteditor aus kann das entsprechende Objekt bearbeitet werden
Neu: Stelle Anzahl an gelöschten Datensätzen nach Löschen einer Zeitspanne im Archiv dar
Neu: Initialisiere den "In allen Skripten suchen"-Dialog bei Betätigung von Strg+Shift+F mit dem aktuell ausgewählten Text
Neu: Beim Sprung zum Skript aus dem "In allen Skripten suchen"-Dialog wird der gesuchte Text ausgewählt
Neu: Mitteilung beim Abschluss des Aufräumens im Archiv
Neu: Spaltengrößen sind nun auch in der webbasierten Konsole änderbar.
Neu: "Tab hinzufügen" Button ist jetzt immer sichtbar - auch bei vielen Elementen
Neu: Zeige SkriptID in PHP Information an
Neu: Wird im Wochenplan eine Aktion für ein nicht schaltbares Objekt hinzugefügt oder eine mit leerem Skript geöffnet, wird automatisch PHP ausgewählt
Neu: Wochenpläne wurden umgebaut, so dass beim Hovern die Uhrzeit angezeigt wird und das Verhalten bei Klicks dem WebFront entspricht
Neu: Ladebildschirme wurden an das neue Homepagedesign angepasst. Es wird nun auch vor dem Login-Dialog ein Ladebildschirm gezeigt
Neu: Der aktuell ausgewählte Tab wird hervorgehoben
Neu: Füllen die Tabs die Leiste, so werden sie dünner skaliert. Die Pfeile zum Scrollen erscheinen erst, wenn die Tabs sonst eine gewisse Mindestbreite unterschreiten würden

Neu: Autovervollständigung im Skripteditor unterstützt nun auch Standard-PHP-Funktionen
Neu: Enthält eine Variable über ihr Profil Assoziationen, hat aber einen Wert unter allen Assoziationswerten, wird nun als Wert '-' anstatt eines Fehlers angezeigt
Neu: Links werden (wieder) standardmäßig ohne Namen erstellt, da diese nun in allen Anwendungen mit dem Namen des Ziels dargestellt werden
Neu: Mittels Shift + Pfeiltasten können mehrere Knoten in Bäumen ausgewählt werden
Neu: Neues Konfigurationselement "TestCenter" zum Schalten aller Statusvariablen
Neu: Icon editieren Dialog zeigt Fortschritt beim Laden der Icons und Cached diese.
Neu: Skripteditor zeigt nach dem Ausführen die Laufzeit an
Neu: Beim Öffnen des Module Controls wird geprüft, ob installierte Bibliotheken durch Module aus dem Module Store ersetzt werden können
Neu: Im Demo-Widget wird nun beim Text zwischen Demos mit und ohne Lizenz differenziert

Fix: Konsole: Im Prüfen-Dialog des Konfiguratorelements wird nicht mehr "undefined" angezeigt
Fix: Der Tab öffnen Dialog und der Archiv Operations Dialog können per Escape geschlossen werden
Fix: Fehler in Suchfeldern (beispielsweise ID nicht vorhanden) werden korrekt dargestellt
Fix: Die Darstellungsoption im Debug Log wurde umgestaltet und sollte die Funktionalität nun besser verdeutlichen
Fix: Variablen im Archiv werden nun nach Position gruppiert
Fix: Prüfe ob aktuelle Filter ein Sprungziel bei "Zu Referenz springen" im Referenzdialog oder "Zum Quellobjekt springen" bei Links verbergen und frage, ob diese deaktiviert werden sollen
Fix: In Zwischenablage kopieren kann auch mit iOS benutzt werden
Fix: Titelzeile der Instanzkonfiguration auf kleinen Displays optimiert
Fix: In ausgelösten Ereignissen Instanzaktionen laden funktioniert nun auch mit Strings und wechselt auf PHP Code, sofern die Aktion nicht geparst werden kann
Fix: Sollte eine Demoversion geöffnet werden, so erscheint das Demowidget nun als letztes, nicht als erstes, Widget
Fix: Gesperrte Variablen über dem Variablenlimits werden korrekt markiert
Fix: Benutzernamen mit E-Mails, welche einen Suffix von mehr als 4 Zeichen haben, werden nun korrekt unterstützt

WEBFRONT

Neu: WebFront auf Dart 2 migriert (Entfernt Unterstützung von iOS 9.x)
Neu: WebFront fügt die IDs im DOM hinzu, sodass Skins gezielter greifen können
Neu: WebFront nutzt den neuen WebSocket Rückkanal
Neu: WebFront mit deaktiviertem Editor lädt bei fehlenden Objekten nicht mehr neu, sondern lädt diese passend nach
Neu: WebFront WYSIWYG Editor kann die in der Basic Version vorhandenen Elemente bearbeiten (wie zur Zeit in der Legacy Konsole möglich)
Neu: WebFront WYSIWYG Editor kann per Knopfdruck sicher aus der Verwaltungskonsole geöffnet werden
Neu: UnixTimestamp* Profile können im Info-Widget verwendet werden
Neu: Button im Konfigurator um WebFronts neu zu laden wurde entfernt. Stattdessen werden die WebFronts bei Änderungen automatisch neu geladen
Neu: Timeout Parameter für Logout Widget zum Automatischen Logout nach Inaktivität
Fix: UTF-8 kodierte Zeichen konnten in der WebFront Legende nicht angezeigt werden
Fix: Wenn RAW bei Graphen aktiviert ist, werden die Zeiten korrekt im Titel angezeigt
Fix: WebFront kann nun String-CSV Daten anzeigen

Fix: WFC_OpenCategory funktioniert wieder korrekt
Fix: Wenn das Laden einer Medien Datei fehlschlägt wird die Ladeanimation korrekt versteckt
Fix: Der Slider springt auf die alte Position zurück, wenn das Setzten des neuen Wertes fehlschlägt
Fix: Slider mit % korrigiert, wenn Min > 0 war (Fix für die Apps kommt)
Fix: WebFront Skin-Builder ignoriert .DS_Store Dateien
Fix: WebFront Konfigurator nutzt bei Skins den Namen und nicht mehr den Titel
Fix: Scrollen funktioniert bei vielen Konfiguratoren wieder korrekt
Fix: Die linken Achsen bei Multigraphen werden genauso sortiert wie die Legende
Fix: Achsen von Bool Graphen innerhalb von Line Charts werden dargestellt
Fix: WebFront: Bei UnixTimestampTime konnte exakt 01:00 nicht als Wert angezeigt/ausgewählt werden

PHP-SDK

Neu: Attribute, welche direkt ohne ApplyChanges gespeichert werden
Neu: Konstanten sind nun definiert
Neu: Funktionen zum Registrieren der benutzten/referenzierten IDs
Neu: Parameter "link" für das Konfigurationselement Button ermöglicht die Angabe, ob die Ausgabe als Link interpretiert werden soll oder nicht
Neu: Das Formularelement "Image" kann durch das Flag "center" nun zentriert werden
Neu: GetStatus
Neu: HasActiveParent

ÄNDERUNGEN 5.1.X

Neu: PHP 7.3.5
Neu: IPS_ConnectInstance prüft, ob die Verbindung sinn ergibt (z.B. Kern Instanzen haben keinen Datenfluss) und ob die Datenfluss GUIDs passen
Neu: Konsole: In Ereignissen gibt es ein neues Template zum Umschalten von Boolean Variablen
Neu: Konsole: Aktualisierungs-Widget berücksichtigt nun auch Skins und den Store
Neu: Konsole: Falsch konfigurierte Ereignisse werden im Baum markiert
Neu: WebFront: Per Parameter ?password=xxxx kann ein Kennwort-geschütztes WebFront direkt geöffnet werden
Neu: WebFront: "Nicht unterstützt" Meldung wird angezeigt, sofern iOS 9.x oder älter verwendet wird
Neu: WebFront: Reverse-Proxy mit Suffixen im Pfad sollte nun korrekt funktionieren
Neu: Module Store informiert über Module, welche eine neuere IP-Symcon Version erfordern
Fix: DBus und Avahi wurden nicht automatisch gestartet, wodurch das HomeKit Modul nicht genutzt werden konnte
Fix: KNX: Shutter wirft bei aktivierter Lamellensteuerung keine Fehlermeldung, wenn Stop mehrmals betätigt wird
Fix: KNX: Wenn im Shutter falsche EIS Typen empfangen wurden, wurde eine kryptische Fehlermeldung generiert. Die neue orientiert sich an der EIB Geräte Instanz
Fix: MQTT Server wirft bessere Fehlermeldung wenn die noch nicht unterstützten QoS > 0 Pakete empfangen werden
"Memory" Option nicht berücksichtigt wurde
Fix: Z-Wave: Control Klasse "SCENE_ACTIVATION" wird mit zu den unterstützten Klassen gewertet (Löst Probleme mit Scenen Activiation)
Fix: WebFront: CircleSlider und Farbrad können wieder genutzt werden, wenn vorher Werte gespeichert wurden

Fix: Konsole: Abstände der Elemente beim Test-Center verbessert
Fix: Konsole: Module Store überspringt beim Prüfen fehlerhafte alte Module
Fix: WebFront: IMAP/POP3 Postfächer werden wieder geladen
Fix: SymconMisc korrekt in "Misc Modules" beim Migrieren umbenannt
Fix: Skriptausgabe ist im Firefox wieder kopierbar
Fix: Module Store prüft auf Updates in der korrekten Sprache
Fix: Konsole: Einige Ereignisse konnte nicht bearbeitet werden
Fix: Konsole: Im Store kann nach Begriffen mit Leerzeichen gesucht werden
Fix: WebFront: Wenn ein ReverseProxy verwendet wird, versuchen wir den Pfad in der WebSocket Verbindung korrekt abzubilden
Fix: location kann nun wie vorgesehen auf für Ketten im Configurator-Element verwendet werden
Fix: Uhrzeiten für Aktionen im Wochenplan werden korrekt ausgewertet* Fix: Skins über das Aktualisierungswidget zurücksetzen funktioniert nun wie vorgesehen
Fix: Module aus dem Module Store können wie vorgesehen über das Aktualisierungswidget aktualisiert werden
Fix: dS Expert Instanz konnte nicht erstellt werden
Fix: WebFront: RTSP Streams werden zentriert und skaliert
Fix: WebFront: Skin Builder ignoriert alle @eaDir Verzeichnisse
Fix: Konsole: Sofern falsche Elemente genutzt werden, wird nun eine bessere Fehlermeldung geworfen
Fix: Konsole: Hintergrund bei "Befehl ausführen" Dialog wird korrekt vergrößert
Fix: Konsole: Kontextmenü auf Firefox unter MacOS funktioniert nun korrekt
Fix: Konsole: Wird ein Skript von ausserhalb des Skripteditors ausgeführt, wird dies wie vorgesehen nicht mehr im Skripteditor angezeigt
Fix: Abstürze unter Linux/Mac beim Aktualisieren von Modulen über das Module Control/Store gelöst
Fix: Es wird geprüft, ob eine Datenfluss-Verbindung eine Endlosschleife ergibt. Sollte dies der Fall sein, wird die Verbindung nicht erlaubt.
Fix: PHP-SDK: HasActiveParent liefert false, wenn es keinen Parent gibt.
Fix: WebFront Konfigurator zeigt wieder korrekt die Geräte in der Geräteliste an
Fix: _generated.inc.php sollte beim Aktualisieren von Modulen nicht mehr kaputt gehen
Fix: Beim Entladen der Module werden die Funktionsverweise auch gelöscht, sodass diese nicht mehr doppelt und dreifach in "Befehle testen" auftauchen
Fix: Konsole: Revisionsnummer war 11 statt 12 Zeichen lang
Fix: Konsole: Farbauswahl von Transparent aus funktioniert wieder wie vorgesehen in allen Browsern
Fix: WebFront: Bessere Fehlermeldung, wenn ein falscher Platzhalter verwendet wird
Fix: "Gerät reaktivieren" ist für Z-Wave auch in der webbasierten Konsole verfügbar
Fix: xComfort Konfigurator kann wieder die neuen "Schalter" hinzufügen
Fix: Z-Wave: Absturz beim Empfang von defekter BATTERY_REPORT Pakete wird abgefangen
Fix: Legacy Konsole: Öffnen des neuen WebFront Editors ist jetzt auch über Connect Dienst möglich
Fix: Wenn ein Modul in den "modules" Ordner initial kopiert wurde, konnte es nicht per MC_ReloadModule geladen werden
Fix: Beim Duplizieren von vielen Objekten über die Verwaltungskonsole konnte IP-Symcon hängen bleiben
Fix: Unterkanäle von Z-Wave Geräten können in der webbasierten Konsole korrekt bearbeitet werden
Fix: M-Bus: "stol" Fehler beim Auswerten von M-Bus Zählern korrigiert
Fix: Legacy Konsole: Der neue Editor kann korrekt geöffnet werden, wenn der Connect Dienst genutzt wird
Fix: Konsole: Beim Hinzufügen und Bearbeiten von Gruppen im Wochenplan betreffen Änderungen der neuen Gruppe wie vorgesehen keine anderen Gruppen
Fix: Konsole: Meldungen zu Statusvariablen ohne Aktion bei Ereignissen angepasst

Fix: Konsole: Wird eine Variable, die nie geändert wurde, im Archiv geöffnet, dann wird eine entsprechende Meldung angezeigt
Fix: Konsole: Die Bearbeiten-Dialoge im Archiv können nun wie vorgesehen erneut geöffnet werden, nachdem geloggte Werte hinzugefügt wurden
Fix: Race Condition beim Timer Pool gelöst, welcher ggf. endlose "Warte auf Timer" Fehler auslösen konnte
Fix: Duplizieren beachtet jetzt auch deaktivierte Standardaktionen
Fix: Suchen/Einrichten von D2-* EnOcean Geräten wieder möglich
Fix: MQTT Server (Broker) hat Eingabe für Benutzername/Passwort
Fix: Z-Wave: Wenn direkt beim Senden ein Fehler aufgetreten ist, wurden die Retries nicht berücksichtigt
Fix: KNX Shutter: Wenn auf der GA 0/0/0 Pakete gesendet wurde hat der Shutter diese fälschlicherweise ausgewertet
Fix: Fehlermeldung beim UDP Socket, sofern der Host leer gelassen wurde
Fix: Weiteres Absturzproblem bei unvollständigen Z-Wave Battery Nachrichten gelöst
Fix: Z-Wave: Schalter für "Erzwinge Simple Assoziationen" fehlte in der Legacy Konsole
Fix: LCN Modul Konfigurator: Instanzen für Fingerprint/Host Kommando werden korrekt erkannt
Fix: WebFront: Schaltbefehle konnten beim Nutzen von Slidern doppelt versendet werden
Fix: WebFront: HD/Extrema wurden ggf. aktiviert, obwohl die Rohdaten Ansicht aktiv sein sollte

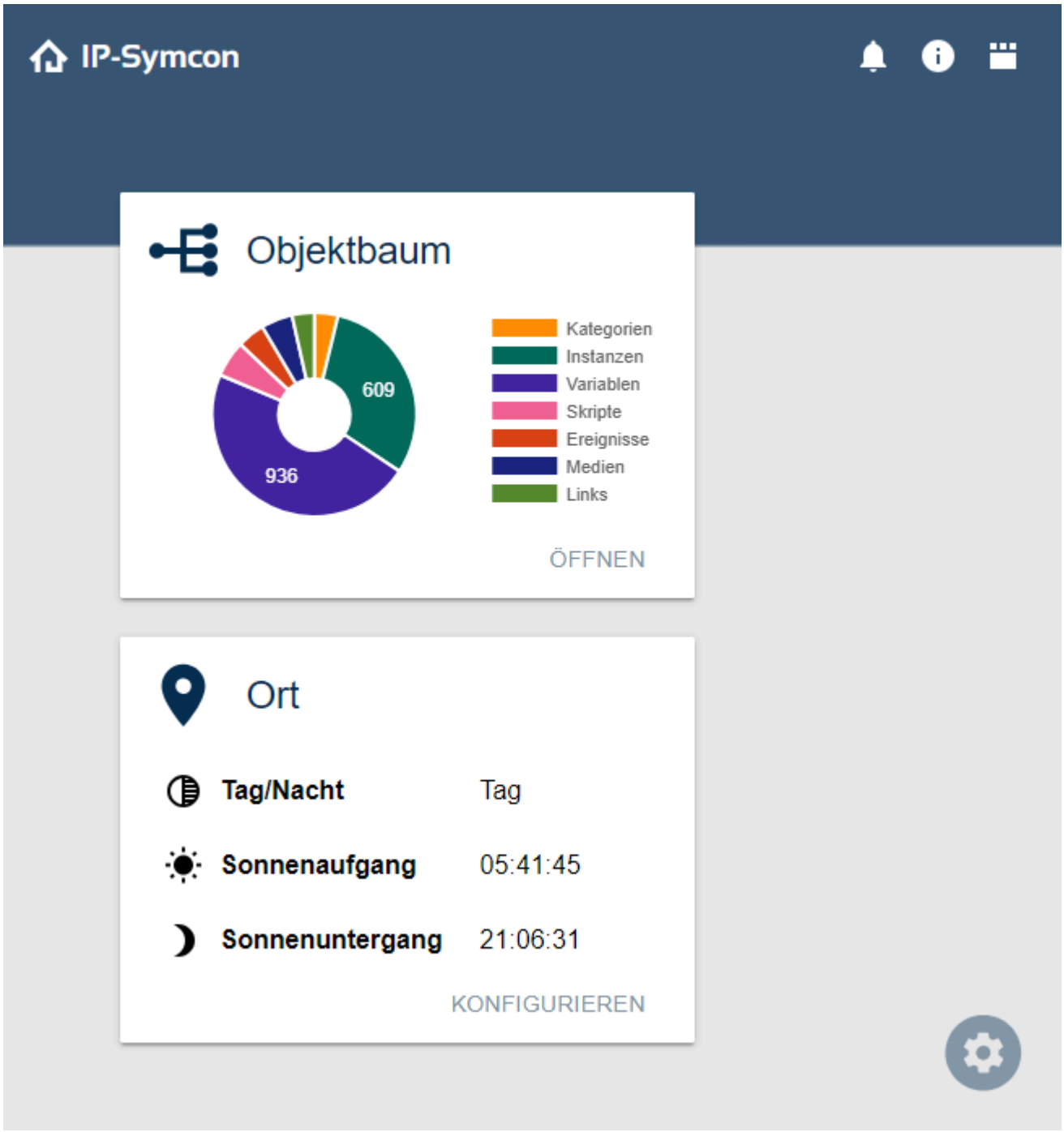
V4.4->V5.0 (Q3/2018)

WESENTLICHE ÄNDERUNGEN

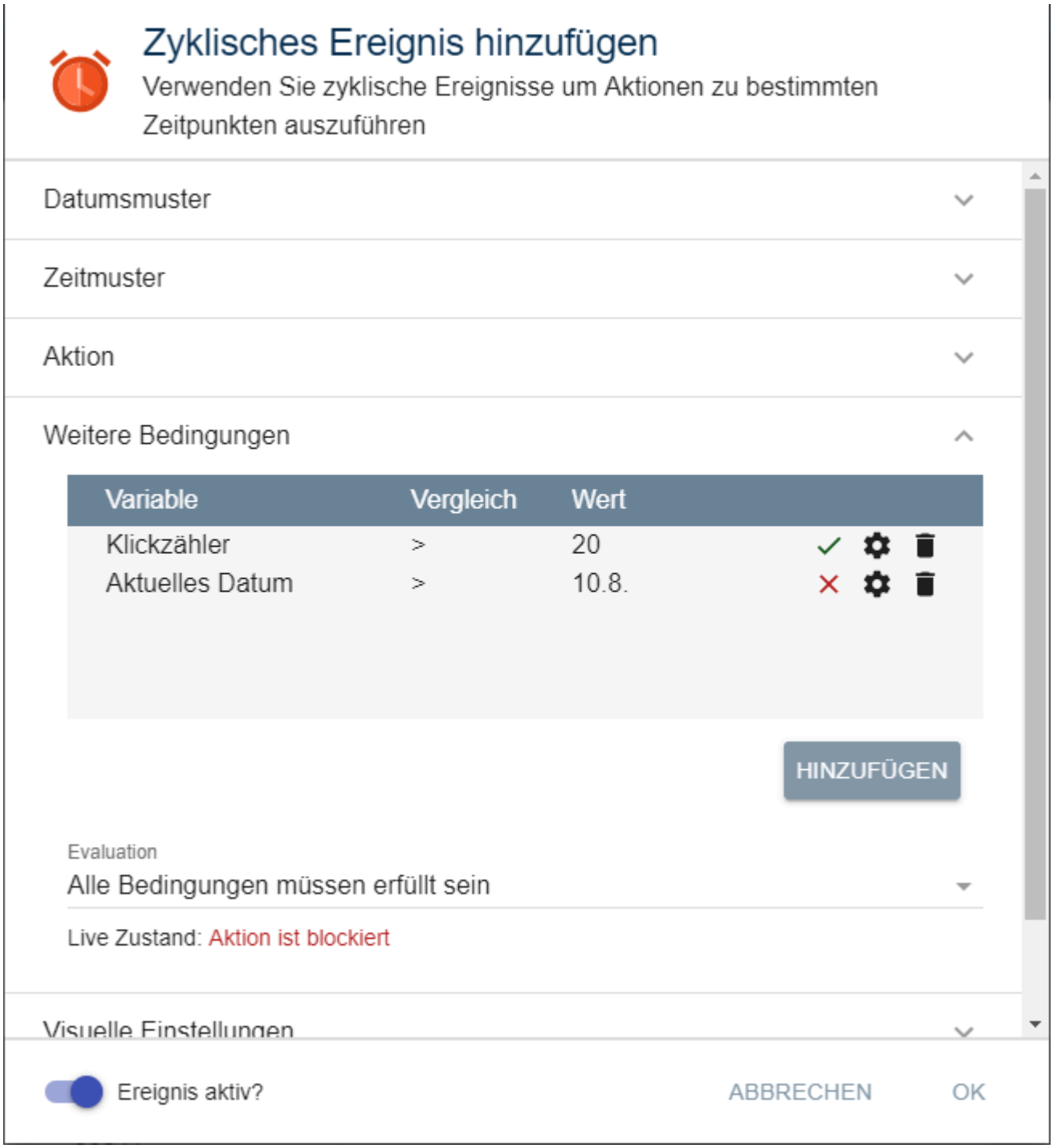
Webbasierte Verwaltungskonsole
Amazon Alexa und Google Home
WebFront: Neuer Light Skin
PHP 7.2
KNX: Erweiterung für alle DPTs
Hardwareunterstützung für ekey Fingerprint Lösungen
Docker Support
Verbesserung der mobilen Apps
Erweiterung vom Location Control
Neue Funktionen im WebFront

WEBBASIERTE VERWALTUNGSKONSOLE

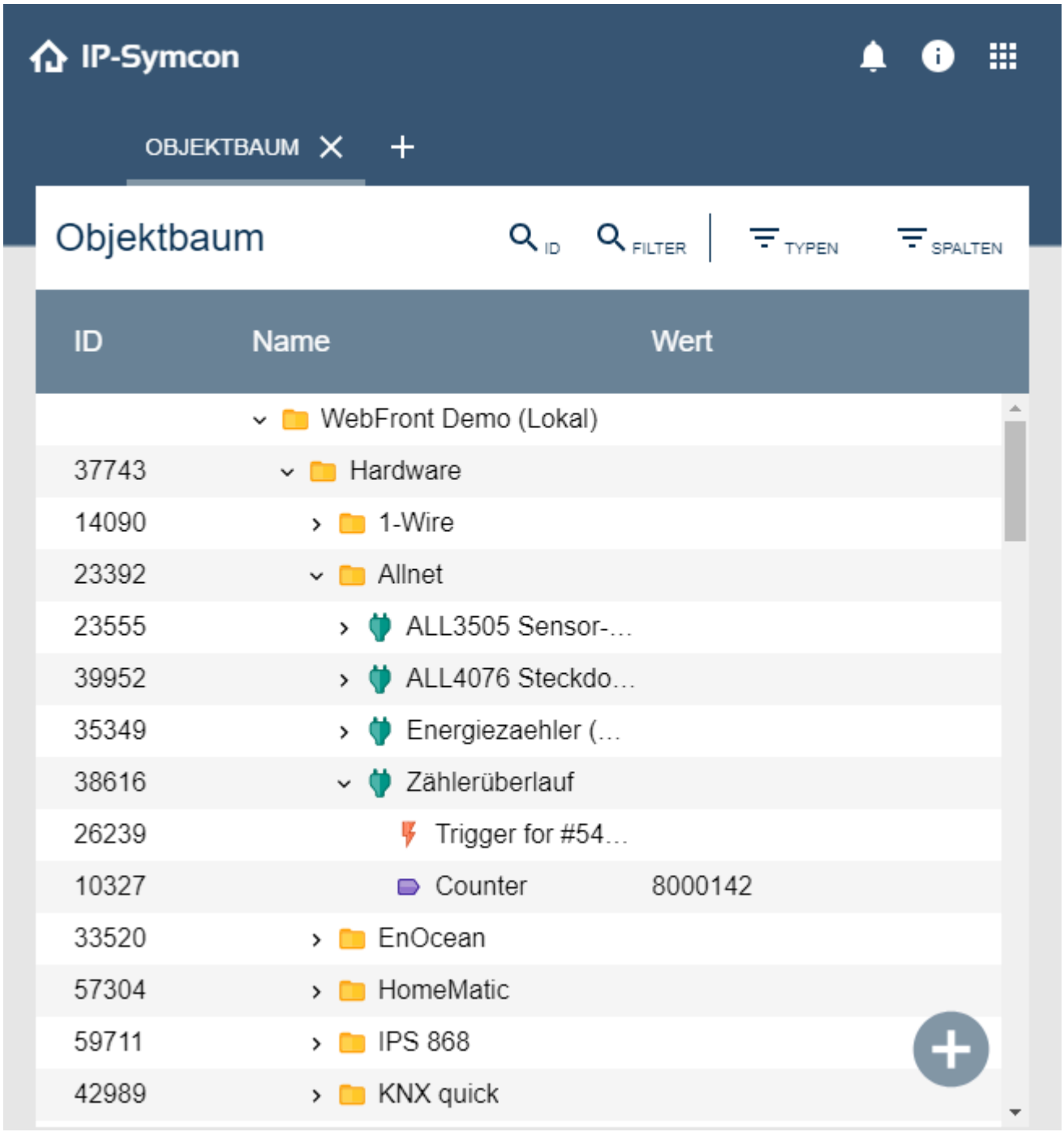
Die webbasierte Verwaltungskonsole ersetzt zu IP-Symcon 5.0 standardmäßig die alte Verwaltungskonsole. Sie bietet diverse Vorzüge und macht die Konfiguration von IP-Symcon um einiges komfortabler. Unterstützt werden die Browser Chrome, Opera, Safari, Firefox und Edge.
So bietet die neue Startseite der Verwaltungskonsole diverse individuell einstellbare Widgets, welche die verschiedenen Aspekte des Systems darstellen können.



Die Dialoge zur Erstellung von Objekten wurden komplett überarbeitet und bieten nun alle Einstellungen mit ausklappbaren Reitern auf einer Seite. So ist alles an einem Ort zu finden und bleibt dennoch übersichtlich.

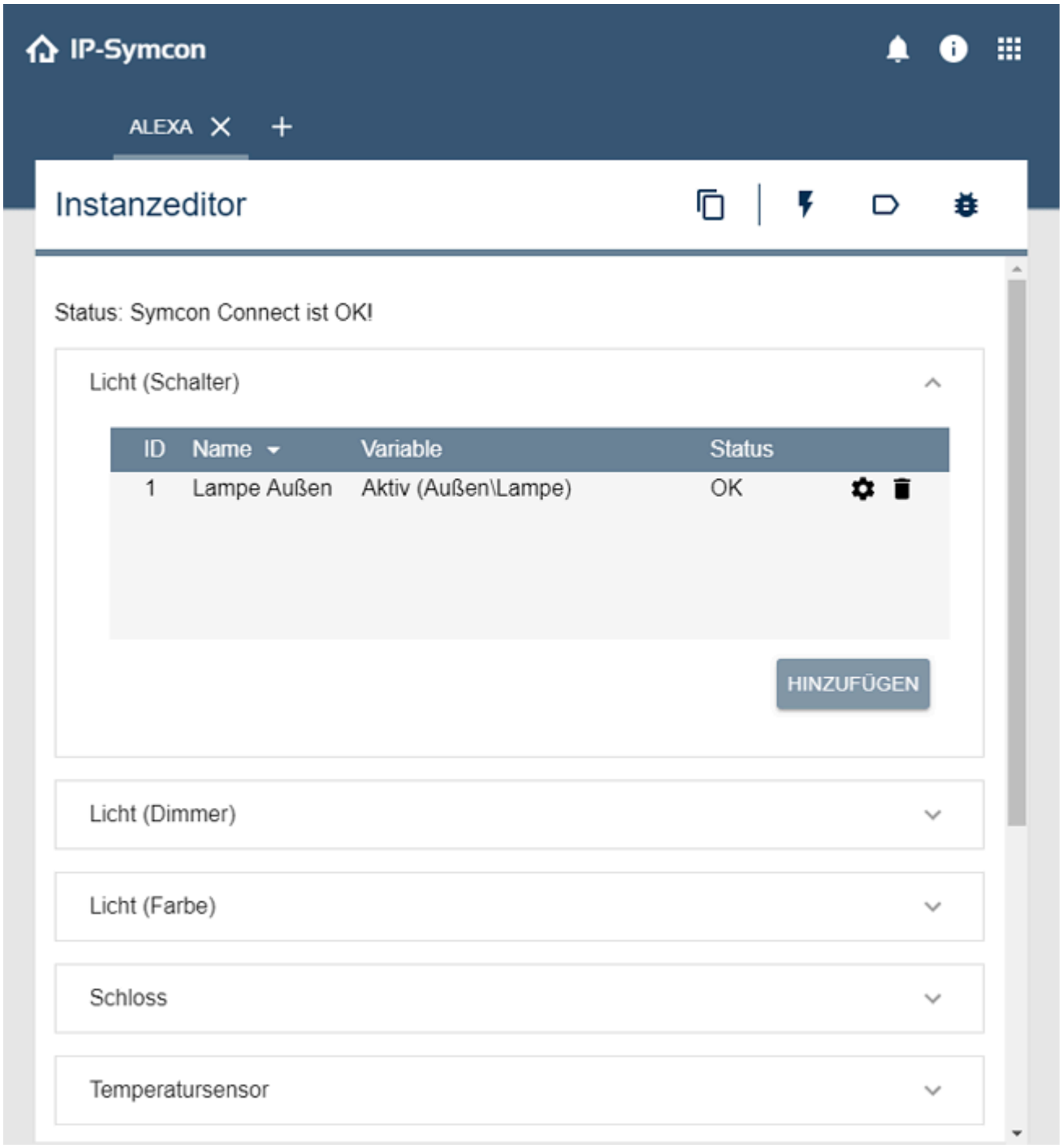


Darüber hinaus ist die Verwaltungskonsole betriebssystemunabhängig. So kann die Konfiguration von IP-Symcon auf jedem üblichen Betriebssystem vorgenommen werden. Zusätzlich muss die Konsole nicht mehr installiert oder heruntergeladen werden, denn sie ist immer Bestandteil von der IP-Symcon Installation. Über Symcon Connect ist die Verwaltungskonsole ebenfalls erreichbar, sodass ein Zugriff jederzeit und von jedem Ort leicht möglich ist.



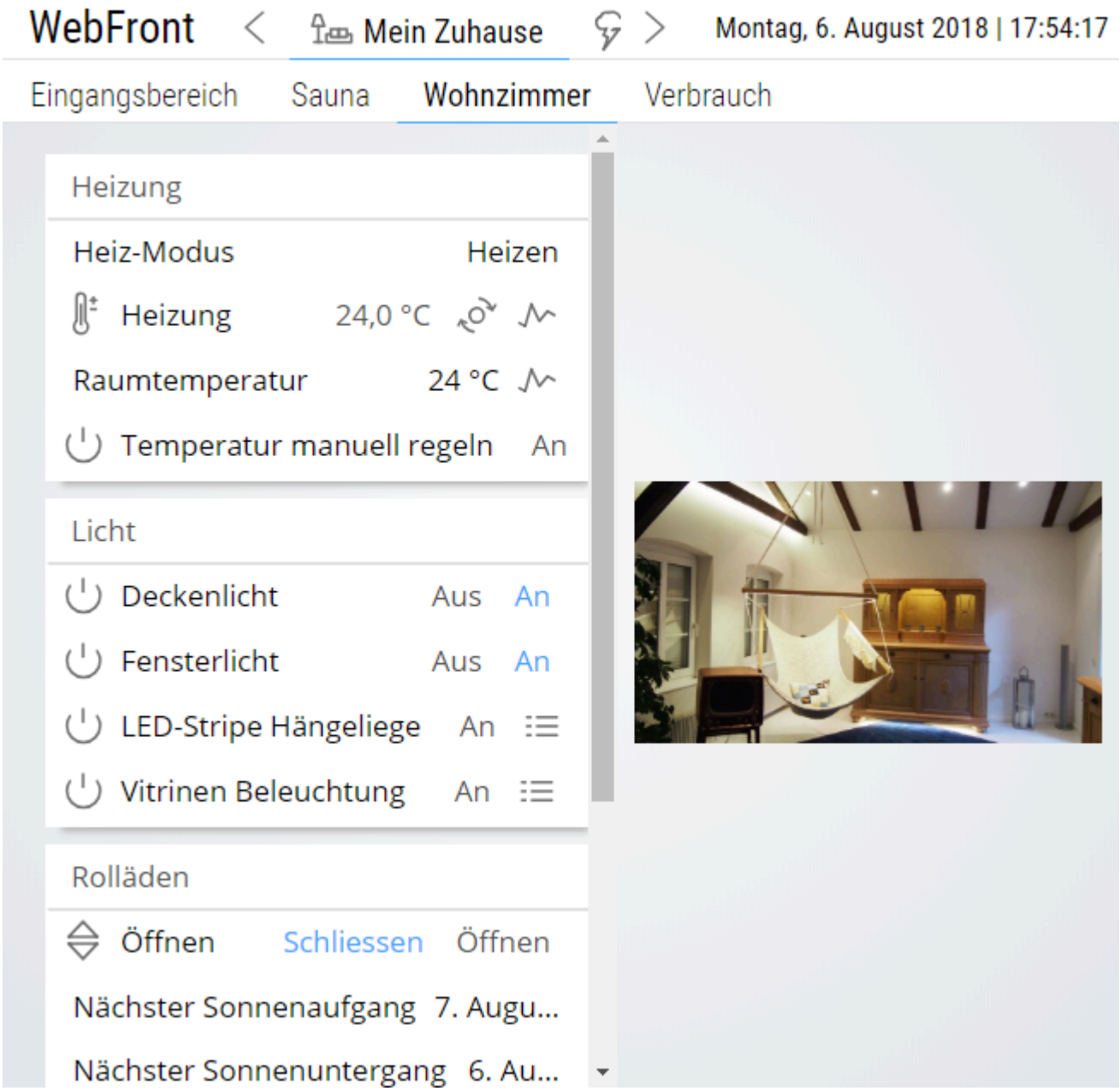
AMAZON ALEXA UND GOOGLE HOME

Mit IP-Symcon 5.0 kommen zwei neue Module, welche die Sprachsteuerung via Amazon Alexa oder Google Assistant ermöglichen. Wir möchten uns hier noch einmal bei unserem Communitymitglied Dave bedanken, welcher mit uns das Alexa-Modul IQL4SmartHome entwickelt hat. Das neue Alexa-Modul verwendet die Alexa Skill API in der aktuellen Version 3, welche einige neue Features ermöglicht, wie beispielsweise die Szenensteuerung.



WEBFRONT: NEUER LIGHT SKIN

Ein neuer moderner Skin findet Einzug. Neben dem “Standard” und “DarkSkin” bieten wir zusätzlich den “LightSkin” an. Dieser ist in seinem Aufbau und Optik bewusst modern und hell gehalten um neusten Designs zu entsprechen und stellt ein farbiges Komplement zu den vorhanden Skins dar. Ihr könnt den Light Skin ab sofort auf unserer WebFront Demo ausprobieren: IP-Symcon WebFront. Zur Installation könnt ihr euch die Beispiele im Skin Control ansehen.



PHP 7.2

Die verwendete PHP-Version wurde von 5.6 auf 7.2 aktualisiert. Dies bietet verschiedene neue Funktionen und einen spürbaren Performancegewinn. Details können hier gefunden werden.

KNX: ERWEITERUNG FÜR ALLE DPTS

Zusätzlich zu den jeher unterstützten 15 EIS Klassen wird mit der Version 5.0 eine Unterstützung von über 370 Datentypen in 77 DPT Klassen integriert. Dadurch wird die Integration von allen KNX Geräten ermöglicht.

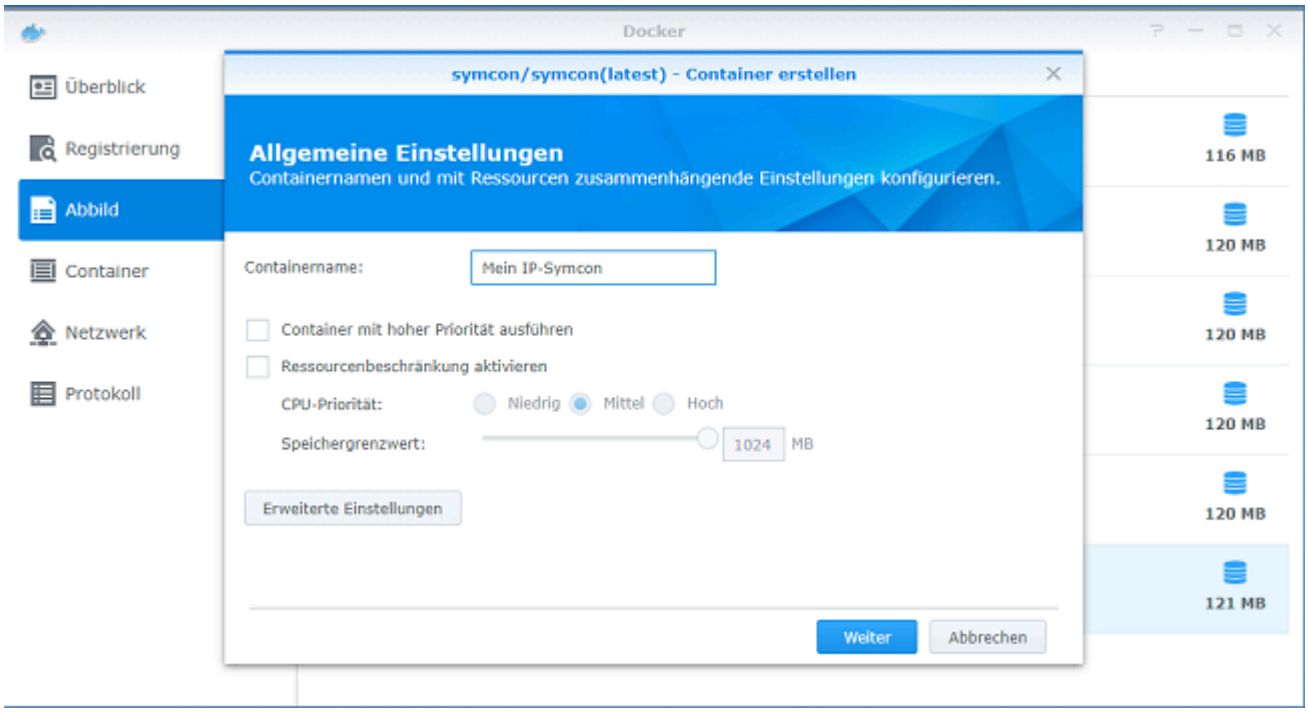
HARDWAREUNTERSTÜTZUNG FÜR EKEY FINGERPRINT LÖSUNGEN

Es wurden neue Module entwickelt, welche nativ die ekey home/multi/net Zutrittslösungen einbinden. Die eingebundenen Geräte stellen die gerätespezifischen Werte in IP-Symcon zur

Verfügung. Dazu gehören z.B. die bekannten Fingerscanner.

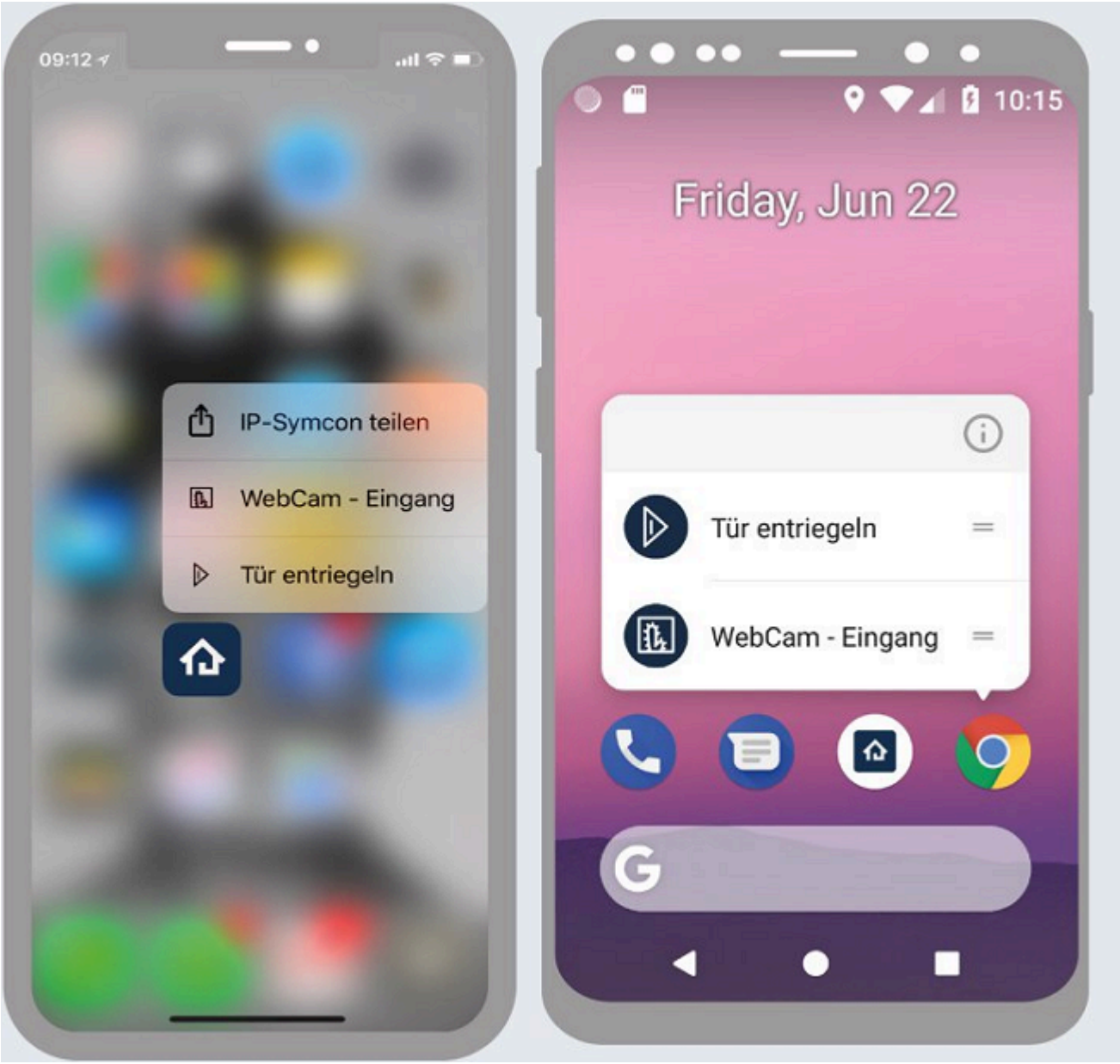
DOCKER SUPPORT

Zusätzlich zu den bisherigen unterstützten Systemen Windows, MacOS, Ubuntu, RaspberryPi und unserer SymBox wird IP-Symcon “Docker” fähig.
Dies bedeutet, dass IP-Symcon auf allen x86 64bit Systemen lauffähig wird, auf denen Docker funktioniert. Dies ist im Rahmen von NAS Systemen wie Qnap und Synology interessant, welche zum Großteil “Docker” unterstützen. Das Docker Image basiert auf der Alpine Distribution und ist somit schlanke 40MB groß.



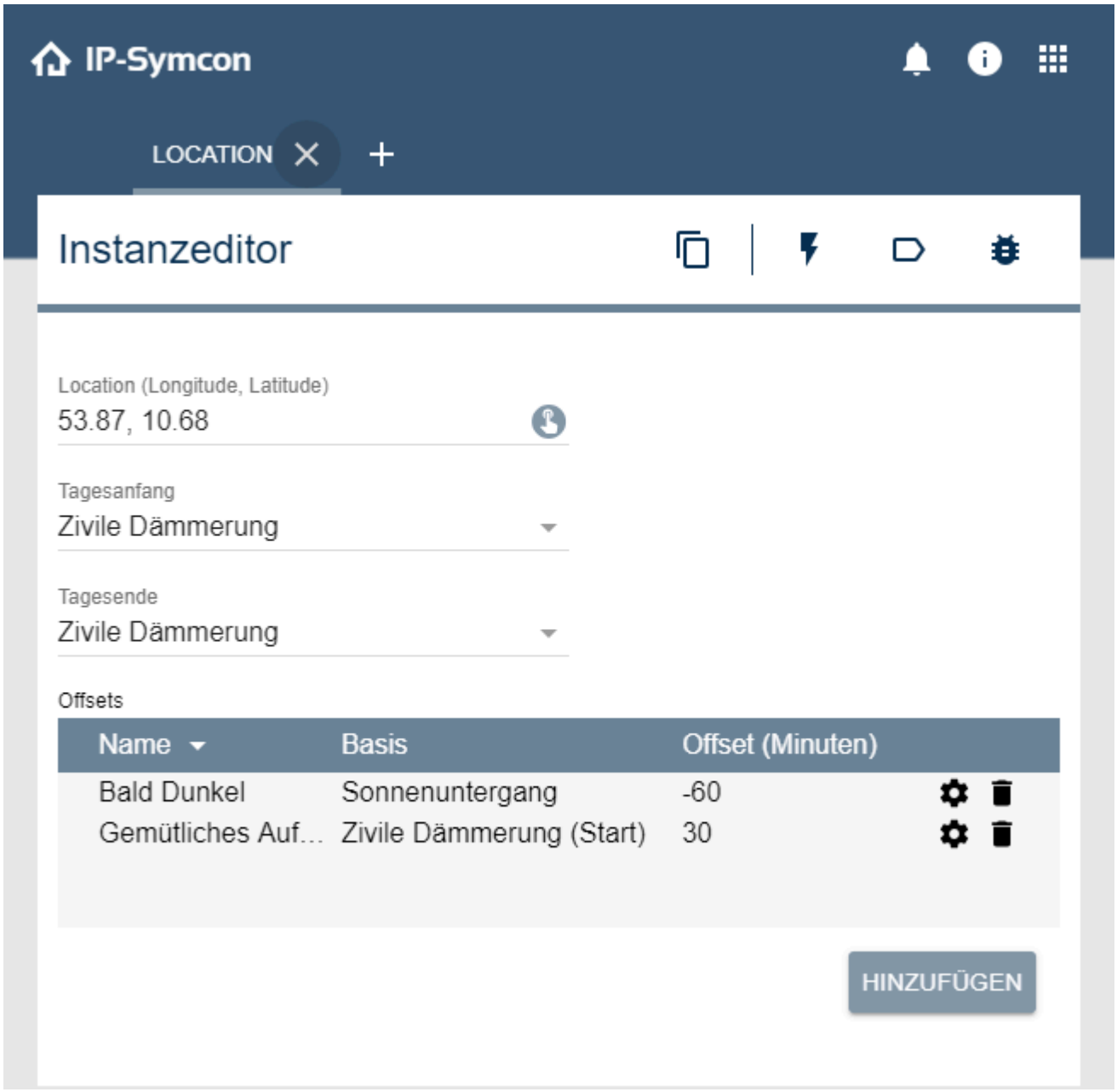
VERBESSERUNG DER MOBILEN APPS

Mit der Version 5.0 von IP-Symcon bekommen auch die mobilen Apps weitere Features. Beim Öffnen von Push-Nachrichten können nun beliebige objektbezogene Funktionalitäten aufgerufen werden. Sei es das Öffnen einer Kategorie, das Anzeigen eines Bildes oder Skripte die gestartet werden sollen. Durch die “Force Touch Geste” (iOS) oder “Long Press” (Android) kann ein Schnellzugriffsmenü geöffnet werden, wodurch ebenfalls Aktions ausgeführt werden können.



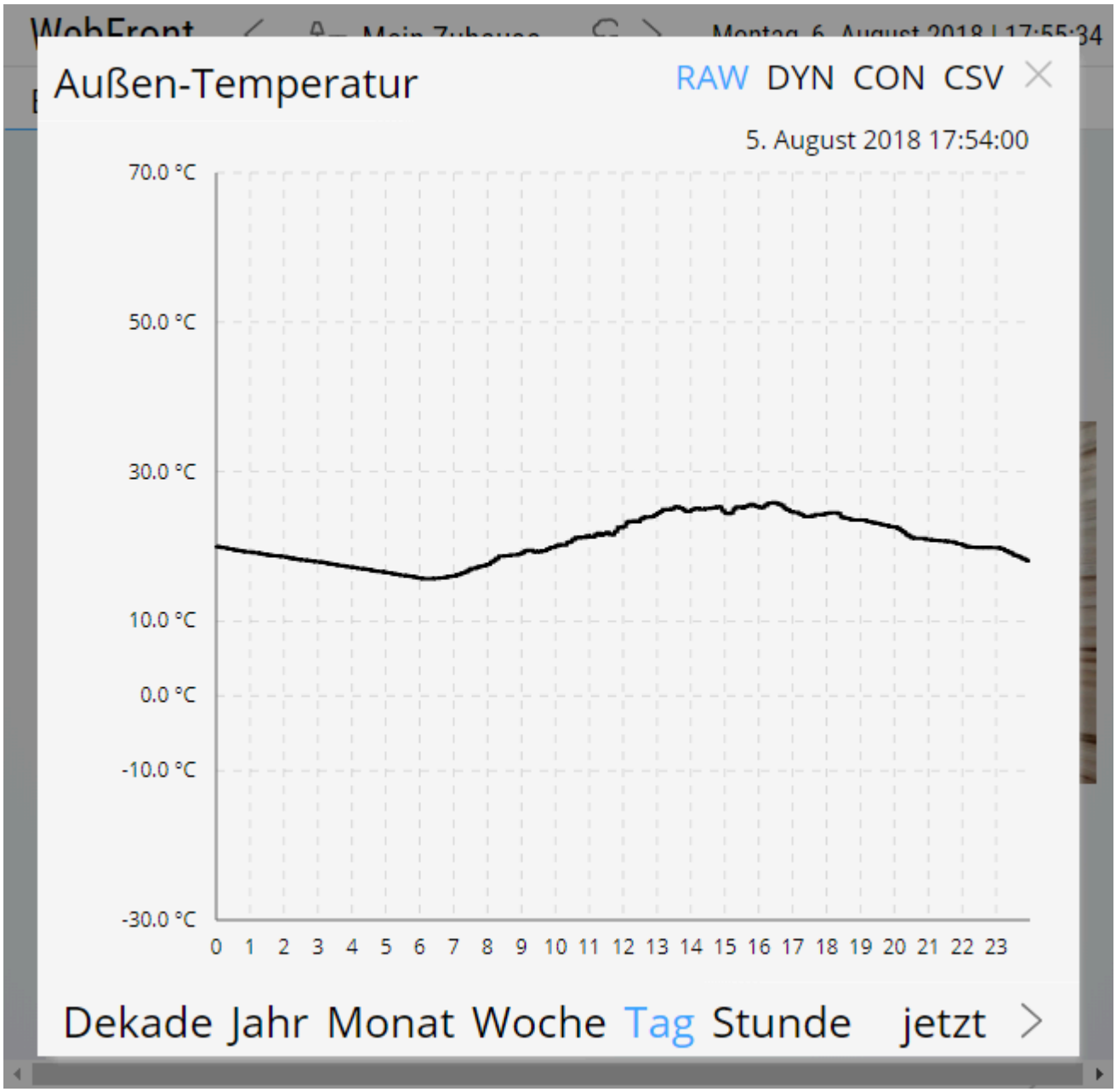
ERWEITERUNG VOM LOCATION CONTROL

Mit der neuen Version des Location Control ist es möglich zu jedem zur Verfügung gestellten Zeitpunkt, sei es Sonnenaufgang, Beginn der Dämmerung, etc., einen Offset in Minuten hinzuzufügen um so individuelle Wünsche und persönliche Vorlieben zu verwirklichen. Somit können z.B. immer 30 Minuten vor Sonnenuntergang die Rollläden heruntergefahren werden



NEUE FUNKTIONEN IM WEBFRONT

Das WebFront bekommt einige neue Features.
Die angezeigten Zeiten beziehen sich ab der IP-Symcon Version 5.0 standortunabhängig immer einheitlich auf die Serverzeit. Ist man beispielsweise im Urlaub in New York und möchte schon einmal seinen Wecker für den ersten Arbeitstag nach dem Urlaub einstellen, so kann man sich darauf verlassen, dass man an diesem Tag zur richtigen Zeit geweckt wird und nicht etwa um 8:00 nach der Zeitzone in New York.
Ebenso bekommen Graphen die Möglichkeit Rohdaten anzuzeigen, um eine absolut authentische Darstellung zu verwirklichen. Desweiteren können Multi-Graphen jetzt auch Bool- und Liniengraphen in einem Graphen vereinen und Übersichtlich darstellen.



MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 4.4 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 5.0 auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Es wird Raspbian Stretch für Raspberry Pi erfordert! (Generelle Versionenübersicht)
Es wird Ubuntu 18.04 für Linux 64 Bit erfordert! (Generelle Versionenübersicht)
Für Microsoft Windows steht ausschließlich eine 64-Bit Version zur Verfügung.
Upgrade von PHP 5.6 auf PHP 7.2. (Upgrade auf 7.0 , Upgrade auf 7.1 , Upgrade auf 7.2)

Profilnamen sind Case Sensitive. Ggf. müssen Skripte die Profile automatisch zuweisen korrigiert werden.
Ziele für Links werden beim Setzen auf Vorhandensein geprüft
Eigene Profile und eigene Aktionen für Variablen werden beim Setzen auf Vorhandensein und Gültigkeit geprüft
~String Profil entfernt, da es keinerlei nützliche Funktion hatte. Einfach kein Profil setzen ergibt die selbe Darstellung
RegisterVariable unterstützt für Server Sockets ein neues TYPE Feld, welches Verbunden/Getrennt Zustände angibt. Diese werden ggf. bei alten Skripten als leere Datenpakete empfangen
Skins werden aus dem /webfront/ anstatt aus dem /webfront/user/ Ordner geladen. Dies löst ein Problem mit der Basis Authentifizierung bei eigenen WebServern, bei der man denken könnte das WebFront wäre Passwort-Geschützt, obwohl nur der Skin per Passwort gesichert war und dieser die Abfrage bewirkt hat.
Property im Location Control von "Latitude" und "Longitude" zu einer "Location" Property zusammengefasst.
Property im WebFront Konfigurator von IPAutoStart zu AutoStart umbenannt und vom Format angepasst, sodass es Kompatibel zum "List" Element ist.
Property im WebFront Konfigurator von LANPasswordNeeded zu IgnorePasswordOnLAN umbenannt, sodass es passend zur vorhandenen Checkbox genutzt werden kann
Bei dS Shutter, xComfort Shutter und Z-Wave Shutter wurde das Profil zur Ansteuerung auf ~ShutterMoveStop geändert
PHP-Funktion setlocale wurde aufgrund von Nebenwirkungen auf das Restsystem (siehe Warnung in PHP-Dokumentation) deaktiviert

MIGRATION (FÜR PHP-MODUL ENTWICKLER)

In Konfigurationsformularen ist das "label" Feld mit IP-Symcon 5.0 veraltet und sollte für neue Module nicht mehr verwendet werden. Stattdessen gibt es überall das "caption" Feld, welches Konsistent verfügbar ist.

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: Webbasierte Konsole zur plattformübergreifenden Konfiguration
Neu: PHP 7.2.5 inkl. GMP und Sodium Erweiterung
Neu: Zusätzliche Icons: Bath, Bike, Book, Cloudy, Doll, Door, Download, EnergyProduction, EnergySolar, EnergyStorage, Favorite, Fitness, FloorLamp, Gas, Handicap, Heart, Help, Link, Menu, Pants, Party, People, Presence, Raffstore, Remote, Sink, Sleet, Sunny, Teddy, Tee, Thunder, Umbrella (Übersicht)
Neu: Zusätzliche Profile: Euro, Dollar, ShutterMoveStop, ShutterMoveStep
Neu: Ereignisse mit ungültiger Variable werden mit rotem Ausrufezeichen dargestellt
Neu: Limit für Profil-Assoziationen auf 128 erhöht
Neu: Genutzte Zeitzone wird automatisch erkannt und in der php.ini angepasst
Neu: WebServer leitet Links ohne Slash am Ende automatisch weiter, sofern möglich
Neu: Skins können nun Titel, Favicon und Geräte Icons (img) Ordner überschreiben
Neu: Summary bei I/O Instanzen wieder hinzugefügt (Fehlendes Feature von IP-Symcon 3.4)

Neu: Ereignis Port bei HomeMatic wird automatisch gesetzt (relevant bei mehreren CCUs)
Neu: Auslesen vom Subskriptionsdatum kann im Utils Control aktiviert werden
Neu: Multicast Socket kann auf allen Interfaces gleichzeitig lauschen (0.0.0.0)
Neu: UDP Socket/Multicast Socket können nun über die Server Socket GUIDs beim Datenfluss auf ClientIP/ClientPort lauschen und auf explizite Zieladressen senden.
Neu: Spezialschalter "ServerLogging", welcher die access.log auch für den internen WebServer (Port 3777) aktiviert
Neu: access.log zeigt bei Zugriffen über Proxy die Remote-IPs an
Neu: WebFront Konfigurator liefert bei der Suche nach Referenzen alle genutzten Objekte (z.B. Startkategorien)
Neu: Windows Installer ist bei Upgrades wesentlich kompakter strukturiert
Neu: Spezialschalter OPcache wird unterstützt (Experimentell)

SYSTEME & GERÄTE

Neu: KNX DPTs für fast alle von der KNX Association definierten Typen
Neu: Modul für eKey (ASCII Protokolle von Home, Multi, Net)
Neu: Eltako FHK14 kann die Soll Temperatur direkt übermitteln
Neu: EnOcean EEP A5-14-07
Neu: EnOcean EEP A5-14-08
Neu: EnOcean EEP A5-14-09
Neu: EnOcean EEP A5-14-0A
Neu: EnOcean FGW14 kann GeräteIDs bis 255 nutzen
Neu: xComfort: Unterstützung für xComfort Room Controller Touch (CRCA-00/08, CRCA-00/09, CRCA-00/10, CRCA-00/11)
Neu: Location Control unterstützt Offsets
Neu: Location Control hat Variablen für Sonnenhöhe und Sonnenrichtung, die alle 5 Minuten aktualisiert werden
Neu: Location Control kann mit Längengrad/Breitengrad auf 0 deaktiviert werden, um Variablen zu sparen
Neu: Neues Shutter Control, welches bei manuellem Fahren die Position nachführen kann
Neu: Neues Heating Control, welches auch Kühlen kann
Neu: ALLNET Geräte mit Analogausgang (z.B. RGBW) werden unterstützt
Neu: DNS-SD Control für Bonjour/Avahi
Neu: NAT Support (insbesondere für Docker wichtig!)
Neu: Für Web/Hook/OAuth Requests sind in der $_SERVER Variable alle Header verfügbar
Neu: Legacy Konsole zeigt Listen in voller Breite an
Neu: LCN Shutter, EnOcean Shutter, KNX Shutter nutzen die neuen Profile
Neu: Z-Wave Shutter, dS Shutter, xComfort Shutter an die neuen Profile angepasst
Neu: digitalStrom: Neue Funktionen DS_ShutterStepDown und DS_ShutterStepUp

MOBILE APP (IOS)

Neu: Favoriten, welche per 3D-Touch direkt über den Home-Screen geöffnet werden können (ab iPhone 6s, iOS 10)
Neu: Neue Icons für IP-Symcon 5.0 hinzugefügt
Neu: Unterstützung für Rohdaten-Darstellung in Graphen
Neu: Push-Nachrichten können Ziel (TargetID) enthalten
Fix: Watch App hat beim Betreten von Kategorien Aktionen ausgeführt

MOBILE APP (ANDROID)

Neu: Bessere Unterstützung für Android Oreo
Neu: Favoriten, welche direkt über den Home-Screen geöffnet werden können (ab Android 7.1)
Neu: Neue Icons aus IP-Symcon 5.0 hinzugefügt
Neu: Unterstützung für Rohdaten-Darstellung in Graphen
Neu: Push-Nachrichten können Ziel (TargetID) enthalten

WEBFRONT

Neu: Datum/Uhrzeit bei Graphen ist auf der rechten Seite für eine bessere Darstellung bei kleineren Auflösungen
Neu: Boolean/Integer/Float Variablen ohne Profil, jedoch mit Aktionen können über Freitext verändert werden
Neu: unterstützt Links ohne Namen, wodurch der Name vom Zielobjekt geerbt wird
Neu: zeigt in den Ereignis Dialogen den vollständigen Namen an. (Funktioniert nun auch mit verlinkten Ereignissen)
Neu: erkennt nun eine Abweichung in der Zeitzone (zwischen Browser + Server) und zeigt eine entsprechende Meldung
Neu: zeigt alle Zeiten und Graphen (inkl. CSV Export) immer in der Server-Zeitzone an
Neu: Logout Widget (zur Zeit nur über WebFront Editor hinzufügbar)
Neu: Graphen können in der Aggregationsstufe Raw, also Rohdaten dargestellt werden
Neu: Idle Widget (Erlaubt das zurückspringen auf eine Seite nach Inaktivität)

SYMOS

Neu: Kernel auf 4.9.73 aktualisiert
Neu: Willkommensseite bietet direkten Link zur neuen webbasierten Verwaltungskonsole an

PHP-MODULE

Neu: Neue Elemente für Eigenschaftenseiten (ausschließlich für neue webbasierte Verwaltungskonsole)
Tree
Configurator
ExpansionPanel
Image
PopupButton
RowLayout
SelectTime
SelectDate
SelectDateTime
Neu: Kernel-Konstanten (Nachrichten) sind nun automatisch definiert
Neu: FM_CHILDADDED, FM_CHILDREMOVED um zu erkennen, wenn neue "Kinder" am Datenfluss angemeldet wurden
Neu: SetValue, GetValue Funktion zum einfacheren Lesen/Schreiben von Werten
Neu: GetMessageList Funktion zum Auslesen der registrierten Nachrichten

Neu: GetBufferList Funktion zum Auslesen der verfügbaren Buffer
Neu: ServerSocket unterstütztein neues "Type" Feld mit Wert (0=Data, 1=Connect, 2=Disconnect)
Neu: LogMessage zum Senden von benutzerdefinierten Nachrichten ins Log

FIXES

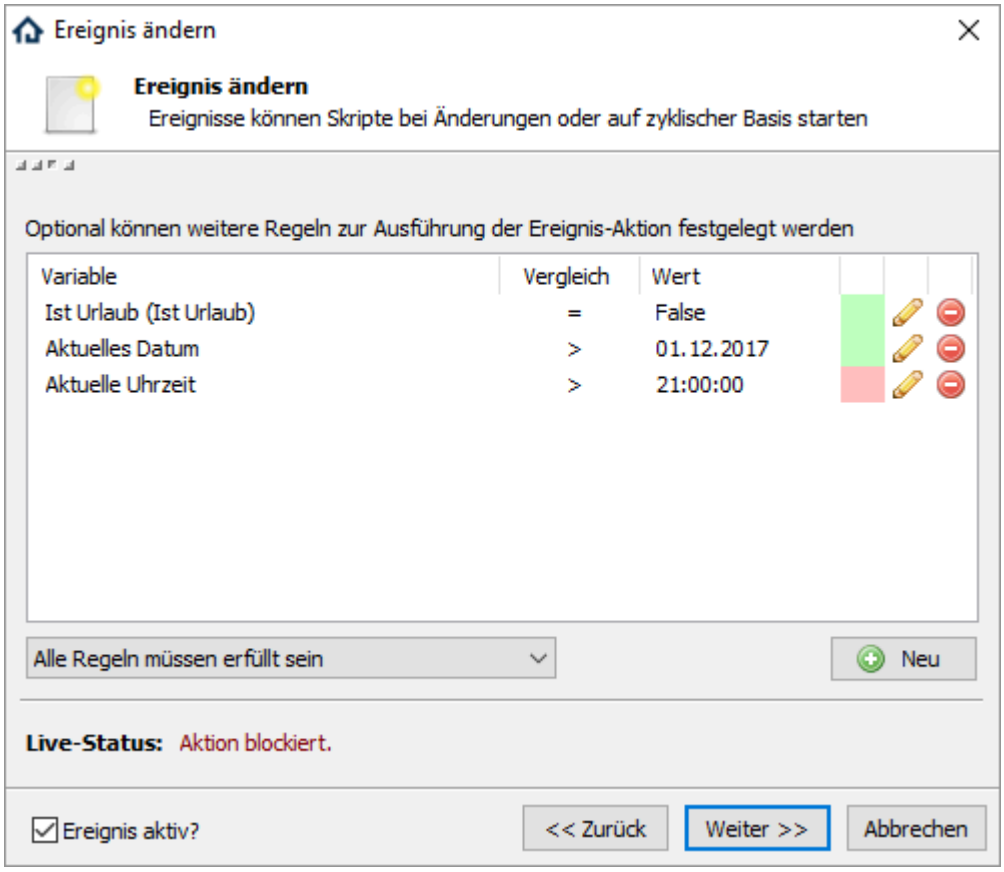
Fix: Funktion setlocale ist in PHP komplett deaktiviert
Fix: ML_LOAD/ML_UNLOAD Nachrichten werden korrekt versendet
Fix: Sofern in der Demo-Version das WebFront modifiziert wurde, wurden die Basic-Lizenz Restriktionen auf das WebFront nach dem Einspielen der Lizenz nicht korrekt angewandt.
Fix: Gleichzeitige Updates von Modules über das Module Control könnten einen Deadlock verursachen
Fix: WebFront: Circle Slider Wert wurde bei kleinen Auflösung nicht korrekt zentriert
Fix: WebFront: Circle Slider Editier-Animation korrigiert
Fix: WebFront: Color Wheel wird besser zentriert bei kleinen Auflösungen
Fix: Beim Duplizieren werden ungültige Inhalte (z.b. ungültiges Profil bei einer Variable) nicht dupliziert
Fix: Fehlermeldung bei JSON-RPC Anfragen gegen Funktionen von PHP Modulen korrigiert
Fix: Absturz bei LCN_SetDisplayText mit leerem Inhalt korrigiert
Fix: DMX: 16-Bit Farbwerte wurden bei 100% im Colorwheel falsch angezeigt
Fix: DMX Instanzen konnten bei Kanal < 0 abstürze verursachen
Fix: Dashboard: Einige Speicherlecks korrigiert
Fix: Legacy Konsole: Einige Speicherlecks korrigiert
Fix: M-Bus Werte, welche ungültige Werte enthielten konnte weiteres Auslesen verhindern
Fix: Fernzugriff konnte unter MacOS nicht korrekt gesetzt werden
Fix: Dashboard konnte in der Demo-Version nicht getestet werden
Fix: Links auf Links sind nicht möglich und werden nun korrekt verhindert
Fix: IPS_GetVariableEventList gibt nun die korrekte aktuelle Liste von Ereignissen zurück
Fix: Wenn die settings.json korrupt war und eine aus dem backup Ordner geladen wurde, wurden die Änderungen in die Backup-Settings gespeichert
Fix: Wenn bei PHP Module im Destroy eine Fehlermeldung geworfen wurde, konnte sich IP-Symcon aufhängen
Fix: Skripttext wurde in PHP Informationen ohne Limitierung angezeigt (Nun auf 100 Zeichen beschränkt)
Fix: MacOS: Fehlende SSL-Zertifikate hinzugefügt für cURL/PHP
Fix: MacOS: Beim Beenden des Dienste wurde ggf. die Settings nicht korrekt geschrieben
Fix: S7-Anbindung wertet vorhandene Fehlerinformationen besser aus
Fix: IMAP/POP3 Mail mit "kaputten" Headern werden besser eingelesen
Fix: Falsche Größenangaben in Sys_GetHardDiskInfo() korrigiert
Fix: Legacy Konsole: Standardwert beim Inhaltswechsler für Scaling war falsch
Fix: SE_EXECUTE Fehlermeldung, wenn ein Ereignis sich selbst gelöscht hat

V4.3->V4.4 (Q4/2017)

WESENTLICHE ÄNDERUNGEN

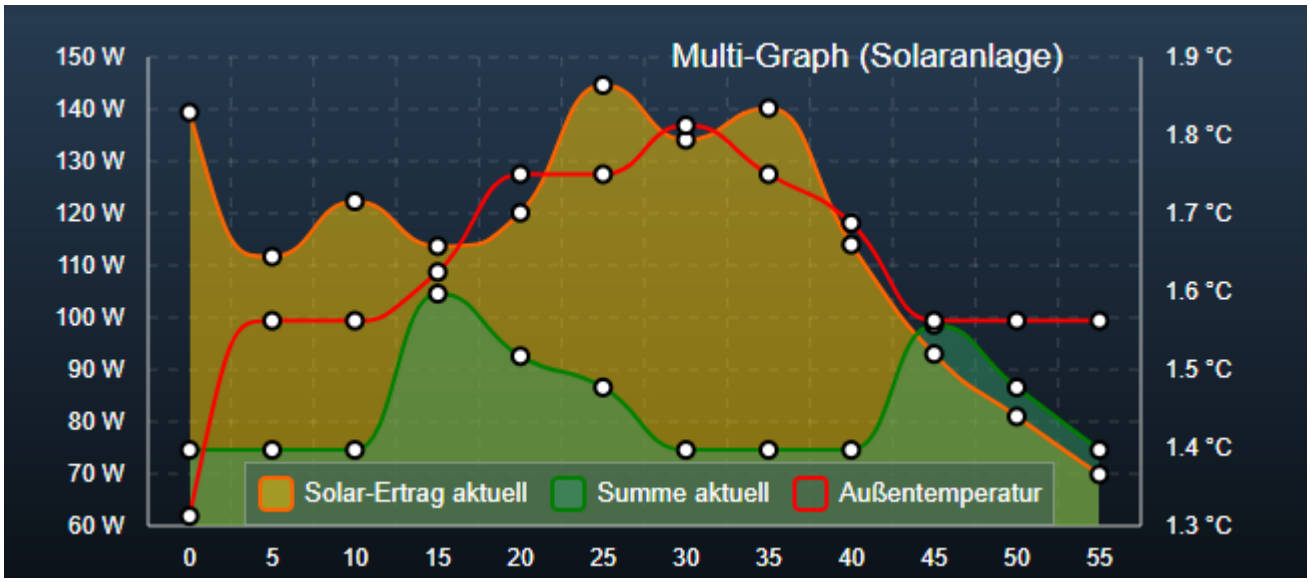
Bedingte Ereignisse

Ereignisse können noch einfacher mit Bedingungen versehen werden ohne auch nur eine Zeile Code zu schreiben. Soll beim Betreten eines Raumes das Licht angeschaltet werden, aber nur wenn es draußen schon dunkel ist? Oder soll die Heizung automatisch auf den Komfortmodus wechseln, aber nicht wenn Sie im Urlaub sind? Diese Beispiele und mannigfaltige andere Möglichkeiten sind nun durch bedingte Ereignisse komplett ohne Skripte darstellbar.



Graphen mit mehreren Achsen im WebFront

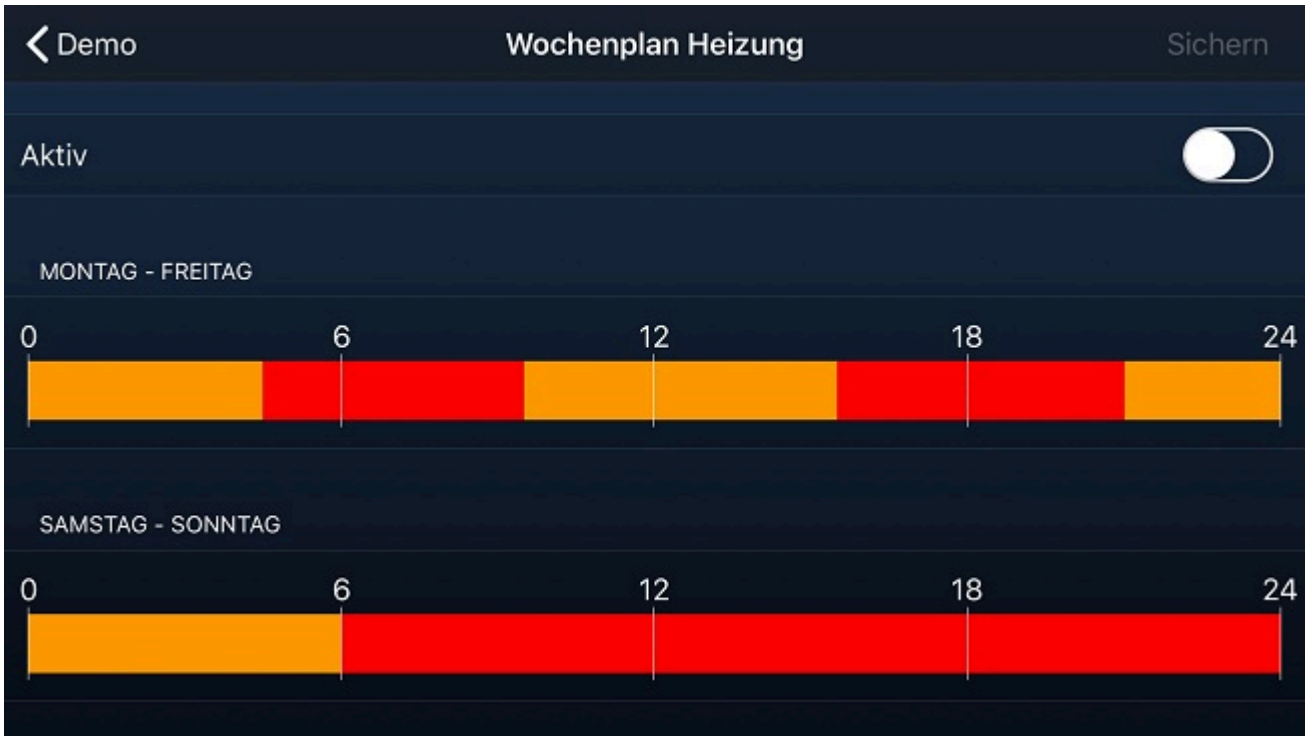
Graphen können nun mehreren Achsen dargestellt werden. So können komfortabel Variablen mit verschiedenen Profilen verglichen werden, wie beispielsweise Außentemperatur und gewonnene Solarenergie.



Graphen-Optionen in den mobilen Apps
Die mobilen Apps unterstützen ebenfalls die neuen Graphen mit mehreren Achsen. Zusätzlich sind alle Optionen wie dynamische Skalierung, hohe Dichte, Extrema, ect. verfügbar



Verbesserter Wochenplan in den mobilen Apps
Die Wochenplandarstellung wurde überarbeitet, sodass die Wochenpläne wie im WebFront intuitiv nutzbar sind. Durch Ziehen und Wischen kann der gewünschte Modus im Handumdrehen angepasst werden.



Verbesserte Geräteunterstützung
Verbesserungen bei EnOcean, HomeMatic, KNX, Siemens OZW, Technische Alternative CMI, Siemens S7, Z-Wave

MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 4.3 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 4.4 auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Es wird Raspbian Stretch für Raspberry Pi erfordert! (Generelle Versionenübersicht)
Die 32-Bit Version für Microsoft Windows ist abgekündigt und wird zu IP-Symcon 5.0, welches 2018 erscheint, nicht mehr angeboten.
Die 64-Bit Version wird automatisch installiert, sofern möglich.

GEÄNDERTE FUNKTIONEN

S7_WriteRegisterShortInt umbenannt zu S7_WriteRegisterChar
S7_WriteRegisterSmallInt umbenannt zu S7_WriteRegisterShort
ModBus_WriteRegisterShortInt umbenannt zu ModBus_WriteRegisterChar
ModBus_WriteRegisterSmallInt umbenannt zu ModBus_WriteRegisterShort
Rückgabeparamteter von AC_DeleteVariableData auf Integer geändert, welcher die Anzahl der betroffenen Datensätze zurückgibt.
IPS_SetEventTriggerValue validiert den Wert, sodass dieser zur Trigger-Variable passt

LISTE DER ÄNDERUNGEN

ALLGEMEIN

Neu: PHP 5.6.32
Neu: Multi-Charts können mehrere Achsen (Links/Rechts) mit verschiedenen Profilen haben
Neu: Bedingungen (Variable, Datum, Uhrzeit) für ausgelöste Ereignisse
Neu: Brute-Force Attacken auf den Fernzugriff werden nach 5 Versuchen verlangsamt und nach 15 Versuchen blockiert
Neu: Raspberry Pi Version benötigt/basiert auf Raspbian Stretch
Neu: Erstellen-Button beim Erstellen/Editieren von Variablen für einfaches anlegen von Standardaktionsskripten
Neu: Rekursives duplizieren von Objekten
Neu: Sollte die settings.json defekt sein, wird nach einer funktionierenden settings.json im Backup Ordner gesucht
Neu: Standardprofile für Energie, Wasser, Gas haben nun Icons zugewiesen bekommen
Neu: "Im Lizenz anzeigen" Dialog wird die genutzt Plattform angezeigt

SYSTEME & GERÄTE

Neu: HomeMatic Anbindung sollte nun korrekt funktionieren, wenn zwei IP-Symcon auf die selbe CCU zugreifen
Neu: KNX: Unterstützung für Weinzierl kBerry
Neu: Siemens Instanzen unterstützt nun Strings (Array of Char)
Neu: Siemens OZW unterstützt nun Verbindungen per SSL
Neu: LCN GFPS wird unterstützt
Neu: LCN: Limit Variable für LCN Shutter Motor
Neu: EnOcean D2-32-*
Neu: EnOcean Eltako FGW14 kann bis zu 255 Geräte ansprechen
Neu: AC_DeleteVariableData liefert als Rückgabe die Anzahl der betroffenen Datensätze
Neu: Z-Wave: ZW_DimUp, ZW_DimDown, ZW_DimSetEx, ZW_DimUpEx, ZW_DimDownEx
Neu: W&T ThermoHygro unterstützt nun Geräte mit mehr als drei Kanälen

PHP-MODULE

Neu: PHP Module dürfen nun auch einen "tests" Ordner für automatisierte Tests enthalten.

FIXES

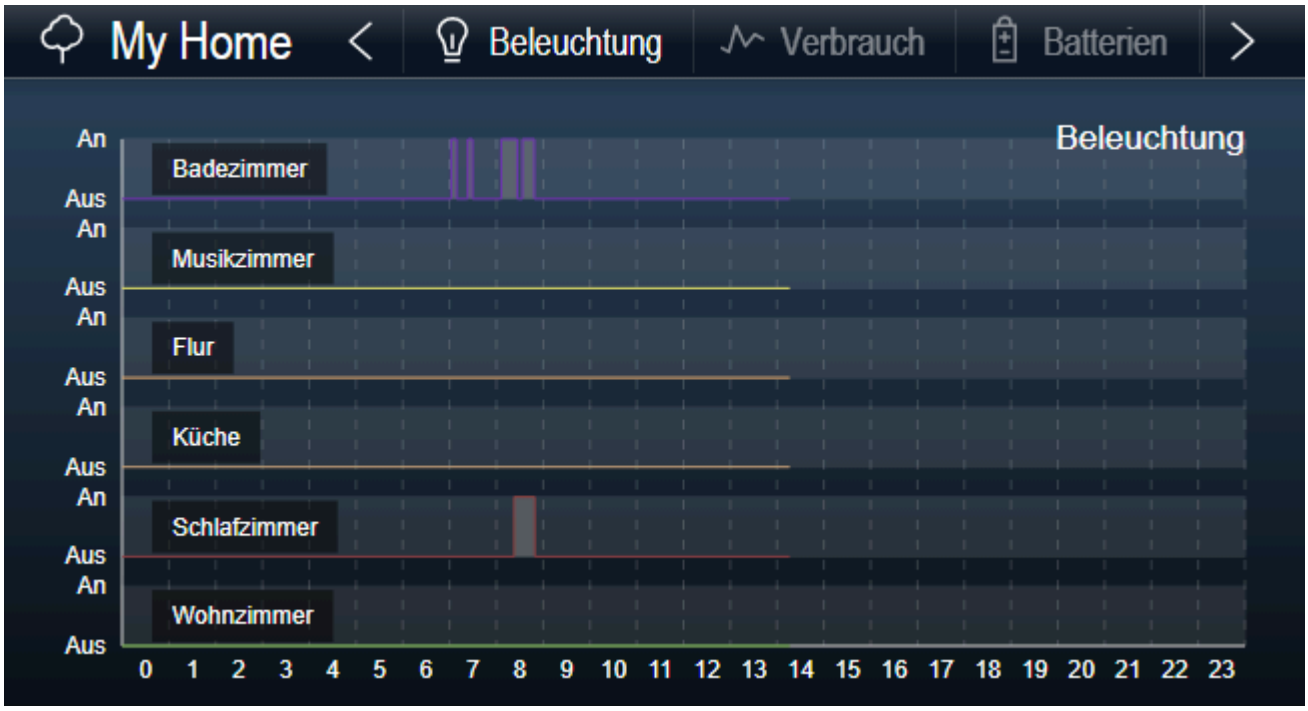
Fix: Unter frischen Windows 10 Systemen konnte ggf. SSL nicht korrekt genutzt werden (insbesondere Push-Nachrichten/Connect ect.)
Fix: Funktionen in Siemens und ModBus Instanzen umbenannt.* Siehe Migration
Fix: Die settings.json wird in den Backup Ordner kopiert anstatt verschoben, sodass zu jedem Zeitpunkt garantiert ist, dass eine settings.json vorhanden ist.
Fix: CSV Export konnte nicht im Firefox korrekt kopiert werden
Fix: Problem unter MacOS mit Auto-Update gelöst

Fix: Verwaltungskonsole kann auf dem Mac wieder über CrossOver gestartet werden (über das Menü)
Fix: Charts werden nun in die Referenzsuche einbezogen
Fix: Sollte bei einem Multichart die Variable nicht mehr existieren, kann der Chart trotzdem editiert werden
Fix: CMI Abfrage der Ausgänge konnte nur deaktiviert werden, wenn auch die Eingänge deaktiviert waren
Fix: CMI Profile hatten kein Leerzeichen bei den Suffxen, wodurch diese "geklebt" haben
Fix: WebFront: Wochenplan Aktionen konnten nicht direkt von 0:00 bis 0:00 gesetzt werden
Fix: Access Violation beim Rechtsklick im Message-Log sofern die ObjektID nicht vorhanden ist
Fix: dSS11 braucht bei größeren Gebäuden mehr Zeit, um die Informationen zu liefern (nun max. 30 Sekunden)
Fix: Auf der SymBox wurden nicht alle Festplatten für Sys_GetHardDiskInfo angezeigt
Fix: Heating Control hat Ausgänge von LCN Unit Instanzen nicht korrekt angeschaltet
Fix: Fehlerhafte fortlaufende Zähleraggregation, wenn Werte nur einmal pro Stunde auf fast genau der vollen Stunde geloggt wurden

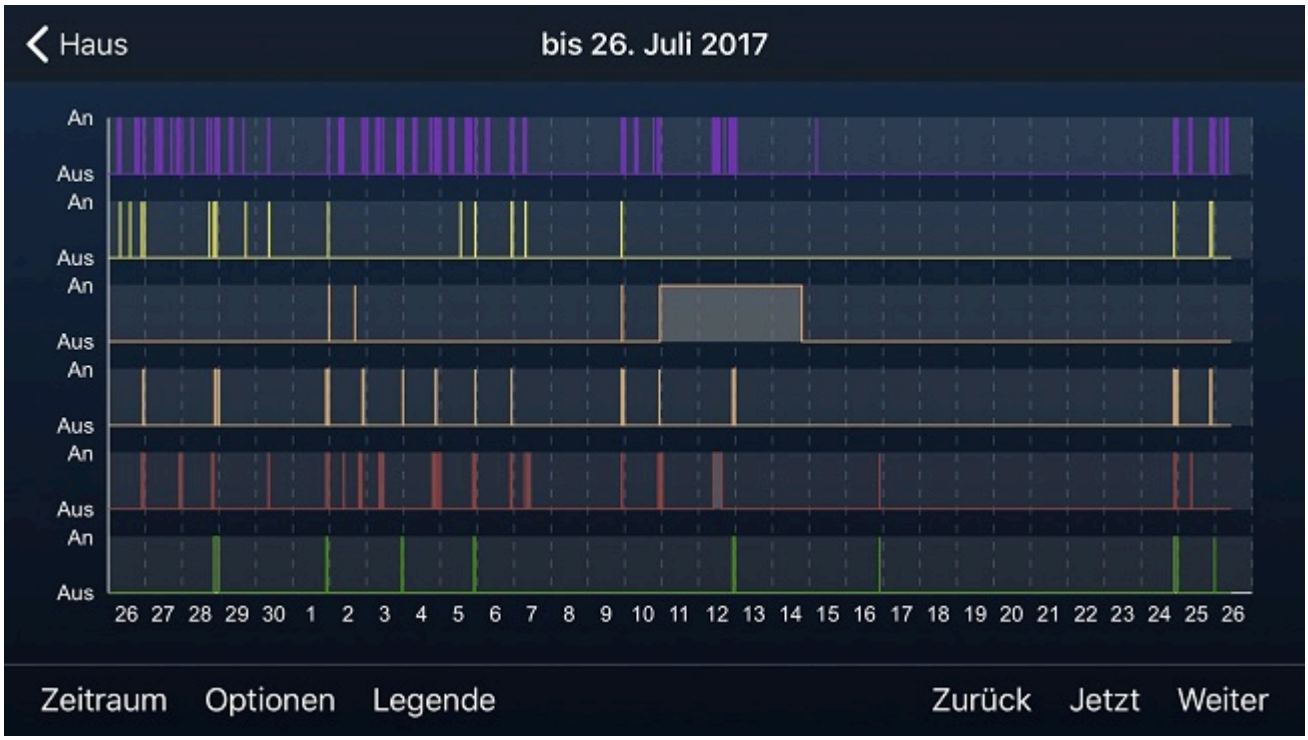
V4.2->V4.3 (Q3/2017)

WESENTLICHE ÄNDERUNGEN

Graphen im WebFront
Die Graphen im WebFront wurden komplett überarbeitet, sodass diese direkt vom IP-Symcon Server erzeugt werden. Das WebFront ist dadurch um ~15% kleiner, wodurch es über das Internet schneller lädt. Zusätzlich wurde die Darstellung verbessert, sodass die Graphen weniger S-Kurven-artig aussehen.

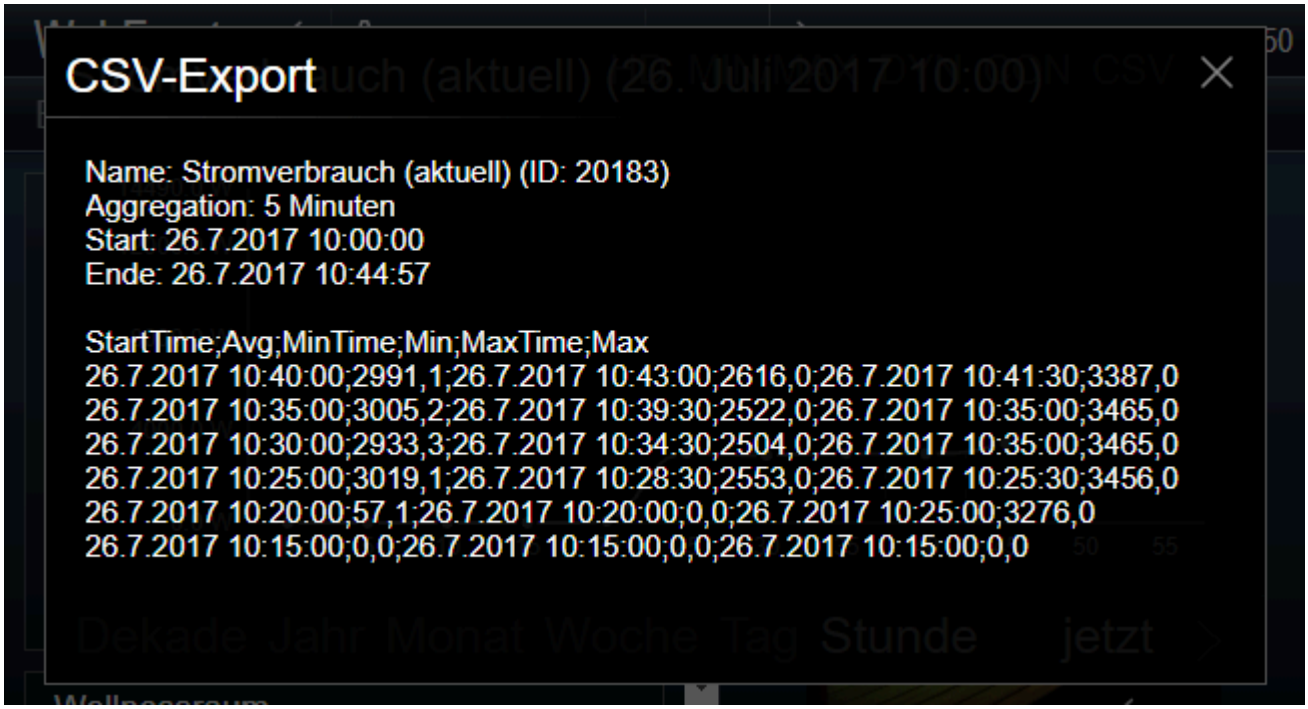


Graphen in den mobilen Apps
Die mobilen Apps unterstützen alle Optionen für Graphen wie die dynamische Skalierung, hohe Dichte und das Anzeigen von Extrema. Zusätzlich werden auch die Multi-Graphen unterstützt, sodass Auswertungen direkt auf dem mobilen Gerät durchgeführt werden können.



CSV-Export im WebFront

Der CSV-Export unterstützt nun Multi-Graphen und zeigt die einzelnen Datenreihen nacheinacher an. Es wird dabei jeweils Name, Datum, Uhrzeit und Aggregationsstufe angezeigt. Die Daten sind in einem Microsoft Excel kompatiblen Format und können direkt kopiert werden.



Bessere Unterstützung der Geräte

- Module für EnOcean D2-01-, *D2-06-*
- Module für Eltako FGW14
- Module für LCN Display und LCN RGBW

Verbesserung für PHP-Module

- Beliebige Strings können in Modulen übersetzt werden
- Module überprüfen vor dem Update auf Kompatibilität
- Module können lokale Änderungen automatisch entfernen vor einem Update

(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Edition nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Edition von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 4.3 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen und dem Symcon Connect Dienst eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 4.2 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 4.3 auftreten.



Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die 32-Bit Version für Microsoft Windows ist abgekündigt und wird zu IP-Symcon 5.0, welches 2018 erscheint, nicht mehr angeboten.
Die 64-Bit Version wird automatisch installiert, sofern möglich.

MIGRATION (FÜR PHP-MODUL ENTWICKLER)

Für Übersetzungen sollte ab sofort die neue Translate Funktion eingesetzt werden.

LISTE DER ÄNDERUNGEN

GRAPHEN IM WEBFRONT

Neu: WebFront nutzt die neuen Graphen, welche auch die Apps nutzen
Neu: WebFront ist nun ~100kb kleiner, da die Graphen vom Backend erzeugt werden
Neu: Splines swingen nicht mehr so sehr (Vermeidet die typischen S-förmigen Kurven)
Neu: CSV-Export verbessert, sodass die Daten besser kopierbar sind
Neu: Kontinuierliche Graphen haben im Titel das "bis" wie die mobilen Apps
Neu: Legende bei Multi-Graphen ist kompakter, sofern kein Offset benutzt wird

GRAPHEN IN DEN MOBILEN APPS

Neu: Multi-Graphen werden in den mobilen Apps angezeigt

SYSTEME & GERÄTE

Neu: Unterstützung für EnOcean Gateway "Eltako FGW14"
Neu: Unterstützung für EnOcean EEP D2-01-*
Neu: Unterstützung für EnOcean EEP D2-06-*
Neu: Technische Alternative CMI Modul unterstützt die Netzwerk Eingänge/Ausgänge
Neu: LCN_SwitchRelayTimer
Neu: LCN Display Modul
Neu: LCN RGBW Modul
Neu: Z-Wave: Zipato Mini Keypad zeigt nun in einer Variable die ID vom Benutzer an, welcher die Tür zuletzt geöffnet/geschlossen hat

PHP-MODULE

Neu: Module prüfen die Kompatibilität zur IP-Symcon Version vor einem Update
Neu: Translate Funktion für Module
Neu: Sofern das Modul keine HTTP/HTTPS Adresse hat wird gefragt, ob die aktuelle Adresse in die Zwischenablage kopiert werden soll
Neu: Module können aktualisiert werden, auch wenn lokale Änderungen vorgenommen wurden (Änderungen ggf. werden verworfen)
Neu: Module unterstützen das Überschreiben von gleichen Funktionen mit dem selben Prefix durch andere Module bei gleichen Parametern
Fix: Module: Editierbare Listen im Aktionsbereich funktionieren korrekt
Fix: Modulliste wird nun ohne Beachtung der Groß- und Kleinschreibung sortiert
Fix: Module: Select* Elemente können im Aktionsbereich ausgewertet werden
Fix: Wenn bei Listen kein gültiges "values" Array geschickt wurde, gab es eine AccessViolation
Fix: Fehlende Farben bei Modulen mit Listen, sofern die Liste im "elements" Bereich war und vorhandene Einträge überschrieben hat
Fix: Module mit sauberem Type Hinting warfen einen Fehler sofern diese aus ProcessHookData genutzt wurden
Fix: Leere Strings wurden für rowColor nicht korrekt ausgewertet
Fix: Module: Rot+Blau waren beim Wert für SelectColor vertauscht
Fix: Module welche beim Erstellen Fehler hatten wurden nicht sauber aufgeräumt wenn bereits Unterobjekt erstellt wurden
Fix: Module: Access Violation bei einigen Buttons im Aktionsbereich

SONSTIGES

Neu: Wir würden gerne eine paar nicht personalisierte Benutzerdaten über Ihr System sammeln (Details)
Neu: Neuer Medientyp: Dokument, welcher zur Zeit txt, doc, xml und pdf Uploads erlaubt
Neu: Bibliothek zum Formatieren von Strings (z.B. %f, %d) wurde gegen eine schnellere getauscht
Neu: Instanz hinzufügen Dialog zeigt den Author des Moduls an
Neu: Konsole: WebFront Konfigurator wählt beim Duplizieren das selektierte Element automatisch vor
Neu: DWD Wetterdaten werden über neues CDN geladen

FIXES

Fix: Z-Wave Multisensor wählt korrekte Profile aus

Fix: Z-Wave Parameterliste liest "Nur Schreiben" Parameter nicht mehr aus
Fix: Z-Wave: Besseres Handling von vielen gleichzeitigen verschlüsselten Nachrichten
Fix: Z-Wave: Konsole: WAKE_UP wurde nicht korrekt erkannt, wenn vorher ein Timeout beim Auslesen eines Parameters passiert ist
Fix: Z-Wave: Beim Laden von Parametern kann dies korrrekt erneut versucht werden, sofern eine Unterbrechung stattgefunden hat
Fix: Parameterinformationen für IPSModule Klasse
Fix: Filter im Instanz-Debug wurde für neue Nachrichten nicht angewendet
Fix: KNX Konfigurator überprüft ESF Datei auf dreistufige GAs
Fix: KNX Repeat Flag wird nicht mehr gesetzt
Fix: Rundungsfehler im WebFront
Fix: WebFront hat mit einer Fehlermeldung quittiert, sofern ein eingebetteter Graph mit einer nicht existenten Variable vorhanden war
Fix: Für EnOcean Thanos konnte die Temperatur aus dem WebFront nicht korrekt eingestellt werden
Fix: Kommentare zu Ereignissen werden im SkriptEditor korrekt erstellt
Fix: WebFront: CSS Spezialklassen wurden nicht korrekt an verlinkte Objekte angefügt
Fix: Bei Ereignissen musste ein Ziel angegeben werden, obwohl dies gar nicht notwendig war
Fix: Objektnamen waren Leer wenn diese per Skript erstellt wurden und kein IPS_SetName folgte
Fix: Seltenes Problem beim Auslesen von dS Joker Klemmen korrigiert
Fix: Problem gelöst, wenn Gerätetaster von dS Klemmen im SceneResponder genutzt wurden
Fix: Für nicht existente Variablen konnte Logging aktiviert werden
Fix: Multicast Pakete konnten auf Linux, Raspberry Pi und MacOS nicht empfangen werden
Fix: Berechnung von LCN Motor war noch falsch
Fix: Ein Problem beim Starten mit der Aggregation gelöst
Fix: Wir löschen curl.cacert Einträge in der php.ini, wenn diese für die Plattform unnötig sind
Fix: Die Vorschläge im Circle Slider beachten nun die Schrittweite
Fix: Beim Öffnen von Variablen im Archiv wurde der Baum neu geladen und die Scrollposition wurde zurückgesetzt
Fix: LCN_SwitchRelayTimer hat auf dem ersten Kanal falschen Wert gesendet
Fix: Multicast Socket sendet unter Windows nun korrekt über die ausgewählte Schnittstelle (BindIP)
Fix: Speicherleck bei PHP-Modulen, welche den Datenfluss verwendet haben
Fix: Kleines Speicherleck beim Aktualisieren von PHP-Modulen
Fix: Potentielle Race Condition im Discovery Server
Fix: Potenzieller Deadlock beim Laden von Modulen
Fix: Cutter kann HID als I/O auswählen
Fix: %d funktionierte in GetValueFormatted nicht bei Assoziationen
Fix: Windows Installer erstellt keine neuen Einträge pro Version
Fix: Graphen: Bei Graphen, welche Boolean oder HD sind, wird die X-Achsen-Beschriftung korrekt links am Startpunkt gezeichnet
Fix: Graphen: Boolean-Graphen mit Wert-Wechseln auf der selben Sekunde werden korrekt angezeigt
Fix: Graphen: Boolean-Graphen bei dem der erste Wert exakt auf den Startwert fiel, werden korrekt angezeigt
Fix: Graphen: Einzelne Boolean-Graphen haben keinen "hellen" Hintergrund mehr, um den Kontrast zu verbessern
Fix: WebFront: Internet Explorer 11 funktioniert wieder korrekt
Fix: WebFront: CSV-Export zeigt bei Boolean Variablen Min/Max nicht mehr
Fix: LCN RGBW kann über WebFront angesteuert werden

Fix: LCN Shutter hat bei Ausgängen nicht korrekt geschaltet
Fix: LCN Shutter Motor verändert nun auch die Aktionsvariable je nach Richtung
Fix: Konsole: Shortcut (STRG+C) zum Kopieren von Objekt / ObjektID ist nun sichtbar
Fix: Konsole: Sortierung im Calendar Control korrigiert
Fix: KNX Group: Checkboxen waren nicht breit genug für deutsche Übersetzung
Fix: Instanz für eine Test-Modul wird nicht mehr erstellt, wodurch Fehlermeldungen im Log auftraten
Fix: Archiv war defekt für Zeitzonen westlich vom Nullmeridian
Fix: Archivgröße wurde bei >2GB Größe als negativ angezeigt
Fix: Sofern das Archiv defekt ist, wird dies beim Editieren der Variable angezeigt
Fix: OZW Instanzen konnten nicht korrekt erstellt werden (seit IP-Symcon 4.3 Release)
Fix: Module mit lokalen Änderungen konnten teilweise nicht zurückgesetzt/aktualisiert werden, obwohl die Sicherheitsabfrage bestätigt wurde
Fix: Problem mit EnOcean D2 Geräten, welche nur einen Kanal hatten und nicht auf KanalID 0 angesprochen wurden
Fix: IPS_GetLiveUpdateVersion ist wieder kompatibel zu IP-Symcon 4.0-4.2
Fix: Möglicher Absturz nach Allnet-Update verhindert und durch entsprechende Fehlermeldung ersetzt, wenn Remote Control nicht aktiviert wurde
Fix: IPS_Execute hat unter Linux den geforkten Prozess nicht abgebrochen, wenn die neue Applikation nicht gestartet werden konnte
Fix: EIS5 bei KNX konnte nicht abgefragt werden
Fix: Viele Fehlermeldungen wenn unter Linux Locales vom Typ de_DE@euro (ohne .utf-8) verwendet wurden
Fix: Es sollte nun zuverlässig möglich sein mehrere IP-Symcon Systeme an einer CCU anzumelden
Fix: WebFront: Bei Boolean-Variablen mit Aktion und zu wenig Platz konnte der Button für das List-Popup nicht angeklickt werden
Fix: WebFront: Bei Variablen mit Aktion und zu wenig Platz wurde manchmal ein Scrollbalken im Element angezeigt
Fix: Fehlerhafte Daten von der ELV FHZ konnten den Dienst zum Absturz bringen
Fix: Z-Wave: Zipato Light Bulb funktioniert nun korrekt
Fix: Z-Wave Regression seit IP-Symcon 4.2 bei der Kommunikation mit FLIRS Geräten (Bitte Instanz nach dem Update neu "Laden", um Problem zu lösen!)
Fix: Bei 2 Kanal W&T Geräten wurde der Status nicht korrekt aktualisiert
Fix: Wenn bei HomeMatic Werte ohne Datentyp gesendet werden, loggen wir diese nicht mehr als Fehlerhaft im Log (z.B. SUBMIT)
Fix: Clickatell SMS funktioniert wieder korrekt
Fix: Seit dem Clickatell Fix konnte der digitalStrom dSS nicht mehr korrekt angesprochen werden
Fix: Fehlerhafte FHZ Pakete konnte IP-Symcon zum Absturz bringen
Fix: CMI Abfrage der Ausgänge konnte nur deaktiviert werden, wenn auch die Eingänge deaktiviert waren
Fix: CMI Profile hatten kein Leerzeichen bei den Suffxen, wodurch diese "geklebt" haben

V4.1->V4.2 (Q2/2017)

WESENTLICHE ÄNDERUNGEN

IP-Symcon 4.2 gibt es für Microsoft Windows zusätzlich als 64-Bit Version. Dadurch können größere Projekte mit mehr Variablen und Archivdaten realisiert werden
PHP wurde auf die aktuellste Version (5.6.30) aktualisiert
Neues Modul für das CMI der Technischen Alternative
Neues Modul für den LCN-BS4 (LCN Shutter Motor)
Verbesserung vom Eltako Shutter, sodass die Fahrzeiten angegeben werden können
Viele Verbesserungen beim SDK für PHP Module wie z.B. native Timer, bessere Validierung und GetConfigurationForParent
Viele Verbesserungen bei den Konfigurationsmöglichkeiten von PHP Modulen wie z.B. List, SelectColor, SelectFile
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Version nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Version von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 4.2 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen und dem Symcon Connect Dienst eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 4.1 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 4.2 auftreten.



Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die 32-Bit Version für Microsoft Windows ist abgekündigt und wird zu IP-Symcon 5.0, welches 2018 erscheint, nicht mehr angeboten.
Die 64-Bit Version steht nach dem Update auf IP-Symcon 4.2 direkt über "Auf Update suchen..." zur Verfügung.

MIGRATION (FÜR PHP-MODUL ENTWICKLER)

Es stehen viele neue Funktionen zur Verfügung. z.B. GetConfigurationForParent, List, SelectColor, SelectFile
Die IPSModule Klasse ist jetzt nativ in IP-Symcon implementiert.
Breaking Change: Es stehen nur bestimmte Unterordner zur freien Verfügung. Siehe: Struktur

Breaking Change: Die Datentypen von RegisterProperty *und ReadProperty* müssen übereinstimmen.

LISTE DER ÄNDERUNGEN

NEUE FUNKTIONEN

64 Bit Version für Microsoft Windows verfügbar!
PHP 5.6.30
Modul für das CMI der Technischen Alternative (erfordert Firmware 1.25.1 oder neuer)
Modul für LCN Shutter (Motor)
MacOS Tray-Anwendung überprüft korrekt die Subskription beim Updatevorgang
Eltako Shutter kann direkt die Fahrzeit angeben über ENO_ShutterMoveUpEx, ENO_ShutterMoveDownEx

NEUE FUNKTIONEN (PHP-MODULE)

SelectColor in Eigenschaftenseiten für PHP Module (Doku)
SelectFile in Eigenschaftenseiten für PHP Module (Doku)
List in Eigenschaftenseiten für PHP Module (Doku)
GetConfigurationForParent für PHP Module (Doku)
Die internen Timer (Millisekundenauflösung) werden in PHP Module genutzt
Überprüfung, ob bei PHP Modulen die library.json / module.json korrekt vorhanden ist und geparst werden kann (Ausnahme sind die Ordner: docs, libs, imgs)
Im Modul Control werden fehlerhafte Module markiert

BEHOBENE FEHLER

Fix: OS X: Beim Neustart des Rechners wurde ggf. bemängelt, dass IP-Symcon nicht gestartet ist, obwohl es am Starten war
Fix: Bei PHP Modulen wurden JSON Daten (SendData*) nicht korrekt einbezogen, welche ein Array waren
Fix: Bei PHP Modulen wird jetzt korrekt das Standardprofil/Standardaktion gesetzt (Löst Problem mit eigenen Actionen/Profilen)
Fix: Bei PHP Modulen werden alle PHP Magic Funktionen nicht nach IP-Symcon exportiert
Fix: Bei PHP Modulen kann auf IPS_KERNELSTARTED/IPS_KERNELSHUTDOWN reagiert werden
Fix: Bei PHP-Modulen kann RegisterTimer nur noch im Create aufgerufen werden
Fix: Multigraphen welche Graphen ohne Datensätze anzeigen sollten, haben diese gar nicht angezeigt (auch nicht in der Legende)
Fix: Color-Wheel und Circle-Slider Presets werden nun pro ObjektID gespeichert
Fix: Lernen Button bei EnOcean Geräten sind nur nach dem Übernehmen anklickbar
Fix: M-Bus unterstützt korrekt Real32 Werte
Fix: LCN Module Eigenschaften Seite kann nun auch korrekt Instanzen löschen
Fix: RegisterVariable konnte hängenbleiben, wenn die Skriptausführung z.B. wegen zu vielen Skriptausführungen abgebrochen wurde
Fix: DMX_FadeRGBDelayed fadete bei vielen nacheinanderfolgeden Fadings falsch
Fix: ModBus 64Bit Real schreiben funktioniert nun korrekt

Fix: Im SkriptEditor wurde von Wochenplänen der Name nicht korrekt als Kommentar eingefügt
Fix: ALT+R funktionierte im Objektbaum nicht
Fix: Start-/Enddatum und Start-/Endzeit werden bei zyklischen Ereignissen korrekt überprüft
Fix: Z-Wave zeigt nun alle unterstützten Klassen an (auch die, die eher kryptisch/lowlevel sind)
Fix: Beim Ändern der Farbe im Multi-Chart wird die Transparenz automatisch deaktiviert
Fix: Speichern Button im Skript-Editor wurde beim Ändern einer ObjektID über STRG+O nicht aktiviert
Fix: Edit-Typen werden korrekt validiert, ob diese erlaubt sind
Fix: HomeMatic Socket hat KeepAlive durchgeführt, obwohl der Socket Inaktiv war

V4.0->V4.1 (Q4/2016)

WESENTLICHE ÄNDERUNGEN

Funktionserweiterung für diverse Hardware-Systeme (Z-Wave, HomeMatic IP, digitalStrom, EnOcean, DMX, ModBus, LCN, Siemens OZW)
Vollständig überarbeitete Verwaltungskonsole mit verbessertem Skript-Editor und Suche nach Querverweisen für beliebige ObjektIDs
Viele neue Funktionen für die Entwicklung der beliebten PHP Module
Neue PHP Module aus der Community u. a. Module von tommi, FireTV, Möhlenhoff Alpha 2, Alarmierung, Anwesenheitssimulation u.v.m. (Link zur Übersicht)
Skins für die Login Seite vom WebFront
Dialoge für Circle Slider und Auswahllisten im WebFront
Dialoge für Datum/Uhrzeit im WebFront
SDKs für Microsoft Excel und WebFront Skins
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Version nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Version von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 4.1 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen und dem Symcon Connect Dienst eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 4.0 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 4.1 auftreten.



Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Es wird Raspbian Jessie für Raspberry Pi erfordert! (Generelle Versionenübersicht)
Es wird Ubuntu 16.04 für Linux 64 Bit erfordert! (Generelle Versionenübersicht)

LISTE DER ÄNDERUNGEN

NEUE FUNKTIONEN

PHP 5.6.26 (löst insbesondere das microtime() Speicherleck und enthält offiziell den cURL Crashfix)

Z-Wave unterstützt mehr Klassen (SECURITY, COLOR , CENTRAL_SCENE, DOOR_LOCK , USER_CODE, TIME, SWITCH_ALL)
Z-Wave zeigt List der Geräteparameter an
Z-Wave hat Möglichkeit die interne WAKE_UP Warteschlange einzusehen
Z-Wave optimiert jede Nacht alle Nodes automatisch (zur Zeit im 5 Minuten Abstand)
Z-Wave kann die Erreichbarkeit der (auch batteriebetriebenen) Geräte direkt aus dem Konfigurator prüfen
Z-Wave Funktion ZW_ConfigurationSetValueEx kann wieder für batteriebetriebene Geräte ausgeführt werden
HomeMatic IP (Vollständiger Umbau von Bin-RPC auf XML-RPC)
digitalStrom: Unterstützung für App-Taster über Joker Klemmen
digitalStrom: Unterstützung für Sensoren an z.B. iSens200
digitalStrom: Unterstützung für binäre Eingänge an z.B. Automatisierungsklemmen
KNX-Shutter unterstützt weitere Rückmeldeadressen und Stepping für Lamellen
KNX-RGB/KNX-RGBW unterstützt weitere Rückmeldeadressen
DMX unterstützt 16 Bit Adressen
xComfort Dimmaktoren (Neue Gerätegeneration mit speziellen Rückmeldungen wird unterstützt)
ModBus Module unterstützt explizite Angabe des Funktionscodes (01, 02, 03, 04, 05, 06, 15, 16)
ModBus RTU/TCP Gateway-Instanzen synchronisieren sich korrekt, sofern mehrere an einer I/O Instanz dran sind
Unterstützung weiterer Eltako EnOcean Geräte (FKC, FKF, FRW, FTR65DS, FTR65HS, FUTH65D, FZS)
Unterstützung von EnOcean EEP F6-10-00
LCN unterstützt die Einstellung für 200 Dimmschritte
ServerSocket mit ClientIP/ClientPort Adressierung
MulticastSocket neu hinzugefügt (erlaubt Empfangen von z.B. UPNP/SSDP)
Siemens OZW unterstützt den Typ RadioButton

VERWALTUNGSKONSOLE

Upgrade auf neuste Delphi Version (Mit korrekter Unicode Unterstützung)
Verbesserter Skripteditor
Suchen & Ersetzen wird direkt im Editor angezeigt
Besseres Highlighting bei der Suche
Code Folding
Nach Querverweisen suchen (Im Kontextmenü eines Objekts und im Skript-Editor)
Konsistentere Icon-Auswahl
Assistent zum Validieren auf UTF-8 Konformität (über Utils Control) (Bitte Settings + Skripte sichern!)

WEBFRONT

Login-Seite kann einen Skin laden (über Skin Control einstellbar) (Beispiel auf GitHub)
Profile mit Schrittweite werden als Circle Slider (wie in den mobilen Apps) dargestellt
Profile mit sehr vielen Aufzählungen werden reduziert und können über einen Dialog eingestellt werden

Dialoge für UnixTimestamp, UnixTimestampDate, UnixTimestampTime, sofern Aktionsskript definiert ist
WFC_OpenCategory($id);

PHP MODULE

Auswahl von Branches im Module Control
Anzeige der Repository URL im Module Control
Übersetzung von PHP-Module (Beispiel auf GitHub)
Dynamische Konfigurationsformulare (Beispiel auf GitHub)
Liste der verfügbaren Nachrichten im MessageSink
$this->MessageSink($TimeStamp, $SenderID, $Message, $Data); (Beispiel auf GitHub)
$this->SetSummary($Summary);
$this->RegisterMessage($SenderID, $Message); (Beispiel auf GitHub)
$this->UnregisterMessage($SenderID, $Message); (Beispiel auf GitHub)
$this->SetBuffer($Name, $Data); (Beispiel auf GitHub)
$this->GetBuffer($Name); (Beispiel auf GitHub)
$this->SetReceiveDataFilter($RequiredRegexMatch); (Beispiel auf GitHub)
$this->SetForwardDataFilter($RequiredRegexMatch);
$this->Destroy();
MC_ReloadModule($module_control_id, "ModulName"); (Lädt das PHP Modul neu und erstellt auch die Instanzen neu)

CHANGELOG

Neu: OAuth Control für die neue Symcon Connect + OAuth2 Brücke
Neu: Ein Klick auf die rote "Verbindung verloren" Meldung startet die Konsole neu und verbindet automatisch
Neu: PHP Module können direkt Hook/OAuth Daten empfangen
Neu: Konsole maskiert Kennwörter im Module Control
Neu: Z-Wave unterstützt korrekt NOTIFICATION V4
Neu: Z-Wave Optimierung kann im Gateway für alle Geräte und zusätzlich für jedes Gerät einzeln konfiguriert werden
Neu: Z-Wave: Im Konfigurator kann das Einlernen ohne SECURITY erzwungen werden
Neu: Konsole überprüft, ob mindestens Wine 1.8 genutzt wird
Neu: Z-Wave Parameterdaten werden online abgefragt, anstatt auf die Offline Daten zuzugreifen. (Dadurch ist die ZWDB täglich aktuell)
Neu: Clickatell Modul für neue REST API
Neu: EnOcean EEP A5-04-02
Neu: EnOcean EEP A5-04-03
Neu: Mehr Debug Informationen, wenn SendNonce bei Z-Wave fehlgeschlagen ist
Neu: Profil und Aktion wird für HMIP-eTRV korrekt gesetzt
Neu: WebServer unterstützt Cookies (z.B. Adminer läuft dadurch im /user/ Ordner korrekt)
Fix: Sortierung der Info Spalte ist möglich
Fix: Variable ändern Dialog vergrößert das Eingabefeld mit
Fix: Korrekte Erkennung von abgesteckten Serial Ports (z.B. USB)
Fix: Datum wird bei SMTP E-Mail-Versand korrekt mitgesendet
Fix: Arbeitsverzeichnis ist nun auch bei Skript-Ausführung von Ereignissen korrekt
Fix: DiscoveryServer sollte nicht mehr die set_option Probleme bemängeln

Fix: OnClick bei Formularbuttons konnte keine Selects auswerten (Beispiel: GitHub - symcon/SymconLJ: Lingg&Janke KNX quick)
Fix: Sofern ganz bestimmte ungültige UTF-8 Zeichenketten in String Variablen gespeichert wurden, konnte die Konsole nicht mehr geöffnet werden
Fix: Sofern beim Starten von IP-Symcon keine Internet Verbindung vorhanden ist, markiert das Connect Control dies korrekt als Fehler und versucht eine neue Verbindung aufzubauen.
Fix: WebFront zeigt überlange String korrekt mit "..." an
Fix: WebFront nutzt für Selector/Circle im CSS keine !important Attribute, um besser Skins zu erlauben (Beispiel)
Fix: Cursor im Editor springt bei Mausklick zum Zeilenende
Fix: Hohe-CPU-Last der Konsole verbessert (ist noch nicht perfekt!)
Fix: Wine Version Überprüfung erkennt auch korrekt RC Versionen
Fix: In Ereignissen können wieder Fließkommawerte bei "Bei Wert" eingestellt werden
Fix: Für einige Aktionen darf nur ein Element im Baum ausgewählt sein
Fix: EnOcean TCM120 Gateways wurden seit dem letzten Update nicht korrekt ausgewertet
Fix: LCN Load/Save Scene Befehle waren seit 4.x defekt
Fix: Eltako FWS61 Modul berechnet korrekt die Nachkommastellen
Fix: UnixTimeStampDate/UnixTimeStampTime wird auch in GetValueFormatted unterstützt
Fix: digitalStrom Module haben sich nicht an Sendelimits gehalten (max 1 Request/Sekunde)
Fix: UVR1611 hat für einige Sensortypen den Wert falsch berechnet
Fix: Beim Ausführen von Skripten das Timeout entfernt (wie bei der 3.4/4.0)
Fix: Beim Ausführen von Skripten den "Es wird ausgeführt" Dialog wiederhergestellt
Fix: Übersetzungsfehler im Skript-Editor beim Ersetzen korrigiert
Fix: Vorhandene Enums wurden beim Hinzufügen von Funktionen bei Ereignissen nicht angezeigt
Fix: Im Meldungsfenster bringt ein Doppelklick auf Einträge von gelöschten Objekte eine korrekte Fehlermeldung
Fix: Abbrüche beim HomeMatic Rückkanal sollten nun zu 100% erkannt werden (Nur CCU2)
Fix: Absturz unter nicht Windows-Systemen bei Zeitumstellung korrigiert
Fix: Absturz beim Setzen von Variablen-Profilen, wenn die Groß-/Kleinschreibung falsch war
Fix: Verzögerung beim Starten bzgl. DiscoveryServer korrigiert
Fix: WebFront: Rundungsfehler
Fix: EnOcean Eltako Dimmer sendet korrektes Lern-Telegramm
Fix: WebFront Retro/Mobile hat manchmal einen Redeclare Fehler angezeigt
Fix: AccessViolation in der Konsole, wenn ein Tab ohne Fokus geschlossen wurde und dann wieder geöffnet wurde
Fix: Importieren von Skripten funktioniert im Editor wieder
Fix: Editieren durch langsamen Doppelklick geht in der Baumansicht wieder
Fix: Bei Redo/Undo wurde der Speichern-Button manchmal fälschlicherweise ausgegraut, obwohl Änderungen vorhanden waren
Fix: Der Z-Wave Testen Dialog hat die WAKE_UP Klasse bei SECURITY Geräten nicht ausgewertet.
Fix: Das Connect Control hat sich selber zu schnell deaktiviert bei Fehlern
Fix: Image Grabber konnte Dateityp nicht automatisch erkennen, wenn der Charset mitgesendet wurde
Fix: Bei einigen falsch kodierten Strings (nicht UTF-8 kompatibel) konnten die Konsole den Snapshot nicht laden
Fix: Visual Studio 2015 Redistributable wurde nicht in der neusten Version installiert
Fix: Bei PHP Modulen wurden JSON Daten (SendData*) nicht korrekt einbezogen, welche ein Array waren
Fix: Anpassung des Working Dir für Ereignisse, sollte nicht bei WebServer-Ausführung greifen

V3.4->V4.0 (Q1/2016)

Um die Migration auf 4.0 oder neuer durchführen zu können ist ein Upgrade auf die letzte IP-Symcon 3.4 Version (Stand: 17.07.2016) erforderlich. Erst danach kann das Upgrade auf 4.0 oder neuer ausgeführt werden:

JETZT IP-SYMCON 3.4 (SETUP) FÜR WINDOWS HERUNTERLADEN



Sofern das Setup nicht korrekt installiert werden kann ist es möglich eine vorhandene Installation manuell auf die 3.4 zu aktualisieren. Dazu muss der IP-Symcon Dienst benendet werden und alle Dateien im IP-Symcon Order mit denen aus der ZIP-Datei überschrieben werden.

JETZT IP-SYMCON 3.4 (ZIP) FÜR WINDOWS HERUNTERLADEN

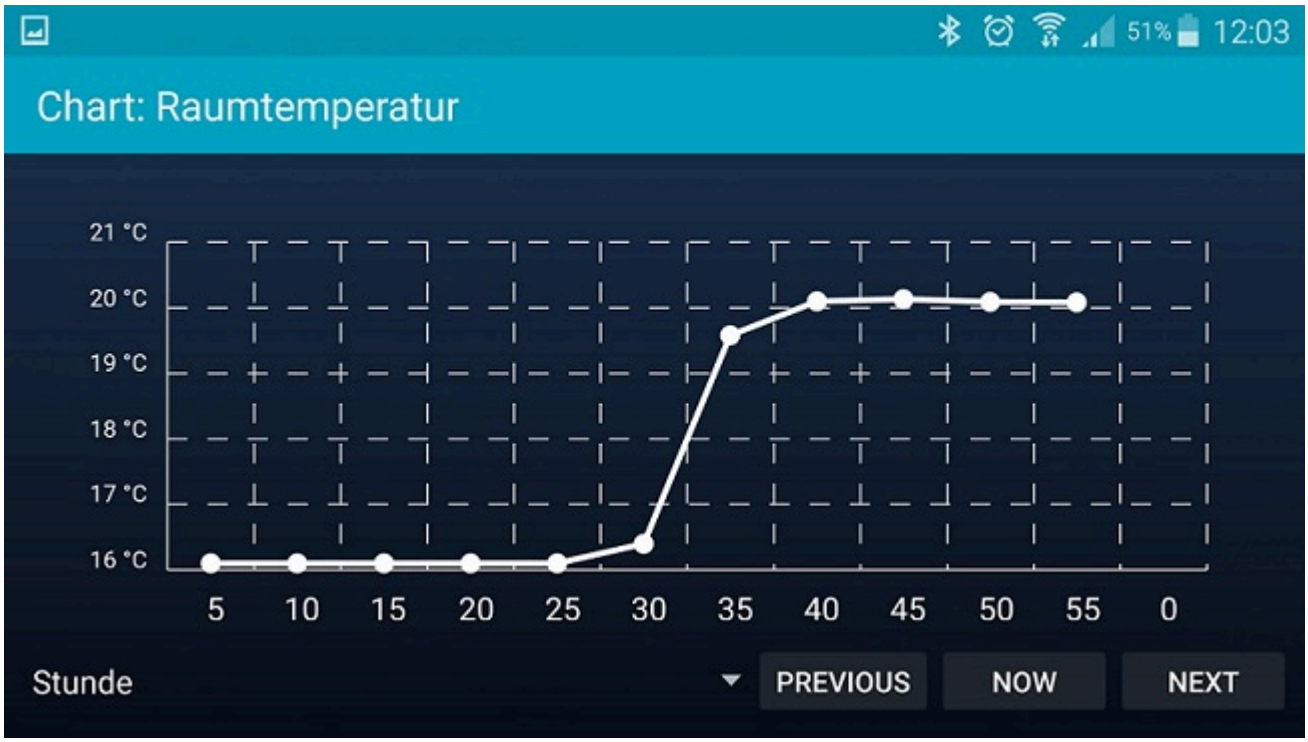
WESENTLICHE ÄNDERUNGEN

Für IP-Symcon 4.0 wurde insbesondere Wert auf die Unterstützung unterschiedlicher Betriebssysteme gelegt. Dadurch steht ab sofort der IP-Symcon Dienst für den Raspberry Pi, Ubuntu, MacOS und Windows zur Verfügung. Der Einstieg in die Gebäudeautomation wird somit effizienter, flexibler und kostengünstiger.
IP-Symcon 4.0 enthält zudem ein SDK auf PHP Basis. Dadurch kann die Community plattformunabhängige Module bereitstellen, welche jeder verwenden und einfach in seine Hausautomation installieren kann. Bis heute sind bereits über 50 PHP-Module z.B. für die Integration von SONOS, Philips Hue oder Netatmo entstanden.

Weitere neue Funktionen im Überblick

Neues Datenbankformat mit bis zu 70% Platzeinsparung. Dank dem einfachen CSV-Format gibt es bessere Möglichkeiten beim Backup/Restore und Modifikation der geloggten Daten. Ein einfacher Import in Microsoft Excel ist ebenfalls möglich.
WebFront Visualisierung mit fortlaufenden Graphen, deaktivierbaren Objekten, Popup Instanzen, neuen Icons mit Zuständen uvm.
Mobile Graphen/Charts in den mobilen Apps
Verwaltungskonsole wurde optisch für Windows 10 überarbeitet und mit vielen neuen Details verbessert.
Symcon Connect erlaubt die kinderleichte Verbindung zwischen IP-Symcon und der mobile App ohne extra den Router zu konfigurieren. Zum Einrichten genügt das Abscannen eines QR-Codes.
Neue Module für astronomische Ereignisse , Kalendereinträge , WebHooks , 1-Wire ohne TMEX, RGB/RGBW für KNX, native Anbindung von Siemens S7 Ethernet, verbesserte

Zuverlässigkeit und Funktionalität bei Z-Wave, Unterstützung weiterer EnOcean Eltako Geräte , uvm.
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Edition nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Edition von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 4.0 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen und dem Symcon Connect Dienst eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))



MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 3.4 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 4.0 auftreten.

Ein direktes Update ist ab Version 3.4 möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Kritisch: PHP nutzt jetzt bei Ausgaben das lokale Dezimaltrennzeichen (Für DE/AT/CH: 3.4 -> Punkt. 4.0 -> Komma)
Objektnamen (Strings) werden als UTF-8 gespeichert, sodass z.B. Skripte mit Umlauten welche IPS_GetObjectIdByName nutzen ins Leere laufen können. Es wird empfohlen auf die neuere IPS_GetObjectIdByIdent Funktion umzusteigen, die keine Umlaute enthält und besser für den gewünschten Zweck geeignet ist.
Kleinere Änderung bei IPS_GetVariable (VariableType anstatt VariableValue.ValueType)

Kleinere Änderung bei IPS_GetScript (ScriptIsBroken anstatt IsBroken, ScriptExecuted anstatt LastExecute)
Kleinere Änderung bei IPS_GetLink (TargetID anstatt LinkChildID)
Kleinere Änderung bei IPS_GetMedia (MediaUpdated anstatt LastUpdate, MediaIsAvailable anstatt IsAvailable, IsLinked/SendEvent entfällt)
Kleinere Änderung bei Sys_GetHardDiskInfo (Indexed Array anstatt HDD0 ... HDDn)
Kleinere Änderung bei Sys_GetCPUInfo liefert nun die aktuelle Durchschnittlast, nicht den Mittelwert der letzten 60 Sekunden
SMS-Modul benötigt APIID der HTTP/S API anstatt der SOAP API
Siemens SPS Module basieren von der Idee auf Snap7 und unterstützt ausschließlich Ethernet basierte PLCs
FTDI/USBXpress Instanzen werden zu SerialPort Instanzen konvertiert und müssen ggf. neu konfiguriert werden
Bei FHT_* Befehlen wird eine Fehlermeldung erzeugt, wenn der FHT Buffer im FHZ Splitter aufgebraucht wurde
SOAP Server nicht vorhanden. Am besten gleich auf JSON-RPC wechseln
PHP 5.6 unterstützt Windows XP/Windows 2003/Windows Home Server 2007 nicht mehr. IP-Symcon somit auch nicht.
Image Grabber nutzt die neue Cache Funktion für Medien, und speichert diese nur noch beim Beenden auf die Festplatte/Flash (Dadurch wird die Lebensdauer des Flash-Speichers enorm verbessert!) Ggf. auf IPS_GetMediaContent umstellen.
WebServer unterstützen keine eigenen Ordner mehr. Es wird nur noch webfront/user Ordner unterstützt.
Der "media" Ordner wird nicht mehr überwacht. d.h. neu hinzugefügte oder geänderte Dateien werden nicht bemerkt. Sofern neue Medien über IP-Symcon eingelesen/aktualisiert werden, kann IP-Symcon über eine Veränderung mit IPS_SendMediaEvent aufmerksam gemacht werden. Bei zyklischen Abfragen, die nicht über den ImageGrabber von extern erfolgen, kann ein Skript, welches zyklisch aufgerufen wird, ebenfalls die IPS_SendMediaEvent Funktion aufrufen.
Für Neuinstallationen wird kein WebServer für Port 82 erstellt. Der interne WebServer ist jetzt unter Port 3777 erreichbar!
Die Datenbank wird während der Installation in ein neues Format konvertiert. Dabei werden die Nachkommastellen mit einer Genauigkeit von 6 Nachkommastellen importiert. Sofern die volle Genauigkeit von 15 Nachkommastellen beim Importieren gewünscht wird (höhere Datenmenge), kann dazu das externe DBConvert Tool genutzt werden.
Einige Module sind entfallen: ISDN, OneWire DS2490, Siemens S7 MPI/PPI, MF420-IR-CTF Klimawächter, EZControl, RFReader, CGEM24
Externe Module/Delphi SDK (*.dll) aus IP-Symcon 3.4 sind nicht mehr kompatibel.
Nach der Installation muss im Archiv unter Kern Instanzen eine Re-Aggregation gestartet werden, damit alle historischen Daten wieder korrekt sichtbar sind.
Skripte in Wochenplan-Aktionen müssen sauber mit einem Semikolon (;) beendet sein.

LISTE DER ÄNDERUNGEN

DIENST IST FÜR VERSCHIEDENE PLATTFORMEN VERFÜGBAR

Verfügbar für Windows (ab Vista), RaspberryPi (alle Modelle), Linux (Ubuntu 14.04 x64), MacOS (ab 10.8)

WEBFRONT ENTHÄLT FOLGENDE NEUE FUNKTIONEN

Fortlaufende Graphen (Link zum Wunsch)
Deaktivierbare Objekte (Link zum Wunsch)
Öffnen von eingebetteten Graphen im Popup (Wurde auch gewünscht)
Voreinstellungen für Graphen (Link zum Wunsch)
Popup-Instanzen (Link zum Wunsch)
Variablenprofil CSS-Hooks zur besseren Individualisierbarkeit über Skins (Link zum Wunsch)
Vereinfachte Ansicht für ~HexColor Profil (Link zum Wunsch)
Neue Vektorbasierte-Icons für noch bessere Retina Kompatibilität (RetroSkin für die alten Icons)
Adaptive Icons (Icons mit Statuszuständen), welche bisher für folgende Icons realisiert wurden: Clock, Garage, Hourglas, Jalousie, Light, Lock, Power, Speaker, Speedo, Temperature, WindSpeed und Window
WebFront Retro optimiert, sodass Blinde dies über JAWS besser bedienen können

VERWALTUNGSKONSOLE ENTHÄLT FOLGENDE NEUE FUNKTIONEN

Optisch für das Windows 10 Design überarbeitet
Beim "Objekte bearbeiten" Dialog ist die Unterstützung für Aktiv/Inaktiv hinzugekommen
Im WebFront Konfigurator können die Voreinstellungen für Graphdialoge konfiguriert werden
"Suche ID" im Objektbaum enthält die ID aus der Zwischenablage falls vorhanden
Im Debug für Instanzen werden Steuerzeichen entsprechend kodiert. z.B. <STX>
Im Debug werden beim Speichern in eine Datei Text und HEX exportiert
Icons für Skripte, Variablen und Wochenpläne getauscht
Alle "Vollbild" Dialog haben nun eine obere Toolleiste mit den relevanten Optionen (Mehr Konsistenz in der Bedienung)
Ereignis/Statusvariablen/Debugfenster werden beim Editieren von Instanzen in eigenen, dockbaren Fenstern geöffnet
Spezialschalter sind aus der Konsole änderbar
Sofern keine Lizenz vorhanden ist, wird diese beim Öffnen der Konsole abgefragt
"Ansicht hinzufügen" ist nun auf der Willkommensseite ganz unten unter "Expertenansichten hinzufügen" platziert
Dokumentation für "Symcon Connect" und "Keine Server gefunden" im Verbindungsassistenten verknüpft
Reaggregation kann im Archiv bequem für mehrere Variablen in Auftrag gegeben werden

PERFORMANTERES ARCHIV

Die Daten im CSV-Format verbrauchen ca 70% weniger Speicherplatz
Vorteile beim Backup, Restore und Löschen
Hinzufügen von eigenen Datensätzen ist möglich
Öffnen der CSV-Dateien mit Microsoft Excel möglich

SONSTIGES

Symcon Connect Dienst erspart lästige DynDNS/Routerkonfiguration (benötigt gültige Subskription) und erhöht über die standardmäßig aktivierte SSL-Verschlüsselung die Sicherheit

Neues Modul für Astronomische Ereignisse
Neues Modul für Kalendereinträge
Neues Modul für WebHooks
Native PHP Erweiterungen (Übersicht der verfügbaren PHP Module)
Native 1-Wire Module (Kein TMEX Treiber wird mehr benötigt)
1-Wire DS2490/DS9490R (blauer USB-Stick). Dieser benötigt noch immer die TMEX Treiber.
KNX RGB/RGBW Modul
KNX Shutter unterstützt Rückmeldegruppenadressen
digitalStrom Konfigurator kann komfortabel Räume anlegen
digitalStrom Klemmen können über DS_RequestStatus nach aktuellem Zustand abgefragt werden
DMX4ALL Gateway wird nun auch über TCP unterstützt
PHP 5.6.17

ENTFALLENE MODULE

ISDN
Siemens S7 MPI/PPI (Es werden nur noch Ethernet PLCs unterstützt)
MF420-IR-CTF Klimawächter, EZControl, RFReader, CGEM24 (Nur noch als PHP-Module: symcon/SymconBC Â· GitHub)
Alle externen Module der Version 3.x sind nicht mehr mit 4.0 verwendbar. Alternativ gibt es nun die platformunabhängigen PHP-Module.

V3.3->V3.4 (Q2/2015)

WESENTLICHE ÄNDERUNGEN

Verbesserte Nutzbarkeit bei batteriebetriebenen Z-Wave-Geräten
Die Unterstützung von batteriebetriebenen Geräten wurde wesentlich verbessert, sodass die Einrichtung und Veränderung von Einstellungen leichter möglich ist. Intern wurde die Warteschleife für Nachrichten verbessert, sodass keine gleichen, doppelten Anfragen an das Gerät gesendet werden, wodruch die Lebensdauer der Batterie profitiert. Zusätzlich kann das Abfrageintervall in der Instanz definiert werden, wodurch weitere Einsparungen im Batterieverbrauch möglich sind.
Benutzerdefinierte Sounds können an mobile Apps gesendet werden
Push-Benachrichtigungen der mobilen Apps können mit einem benutzerdefinierten Sound ausgeben werden. Ein Blick auf das Smartphone kann somit oftmals entfallen. Weitere Informationen zu den Verfügbaren Sounds sind sind in unserer Dokumentation unter WFC_PushNotification zu finden.
ALLNET Module aktualisiert
Die ALLNET Module bevorzugen ab IP-Symcon 3.4 die neue JSON Schnittstelle der Geräte, welche ab Firmware 3.0 angeboten wird. Dadurch wird die Anbindung der Geräte nochmals vereinfacht. Eine aktuelle Liste der unterstützten Geräten finden Sie unter ALLNET .
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Edition nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Edition von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 3.4 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 3.3 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 3.4 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Sofern batteriebetriebene Z-Wave-Geräte genutzt werden, muss nach dem Update das Abfrageintervall explizit angegeben werden! Andernfalls wird der Status nicht vom Gerät abgefragt, sondern nur die vom Gerät gesendeten Werte aktualisiert.

LISTE DER ÄNDERUNGEN

Neue Funktionen

-Neu: Verbesserte Nutzbarkeit bei batteriebetriebenen Z-Wave-Geräten
-Neu: Digitale Signaturen (Löst Probleme mit Fehlalarmen in Virenscannern)
-Neu: Im ModBus Modul kann ein Umrechnungsfaktor angegeben werden
-Neu: Im Browser wird der Titel des WebFront in der Titelleiste angezeigt
-Neu: xComfort Konfigurator kann die neuen Geräte verwalten (DeviceID 74)
-Neu: Real64 Datentyp in ModBus hinzugefügt
-Neu: Sound kann an mobile Apps gesendet werden (App-Updates werden benötigt...)
-Neu: Push-Nachrichten können nun 256 Zeichen lang sein (plus zusätzlich 32 Zeichen im Titel) (Erfordert ggf. iOS8)
-Neu: OneWire Module wurden aufgeteilt, um kompatibel zur Linux, RaspberryPi und MacOS Version zu sein
-Neu: Animationen für Graphen können im WebFront über Skins deaktiviert werden
-Neu: ALLUniversal Modul umbenannt zu "alte Firmware 2.x"
-Neu: ALL5000 Modul umbenannt zu ALLUniversal (für Firmware 3.x)

Behobene Fehler

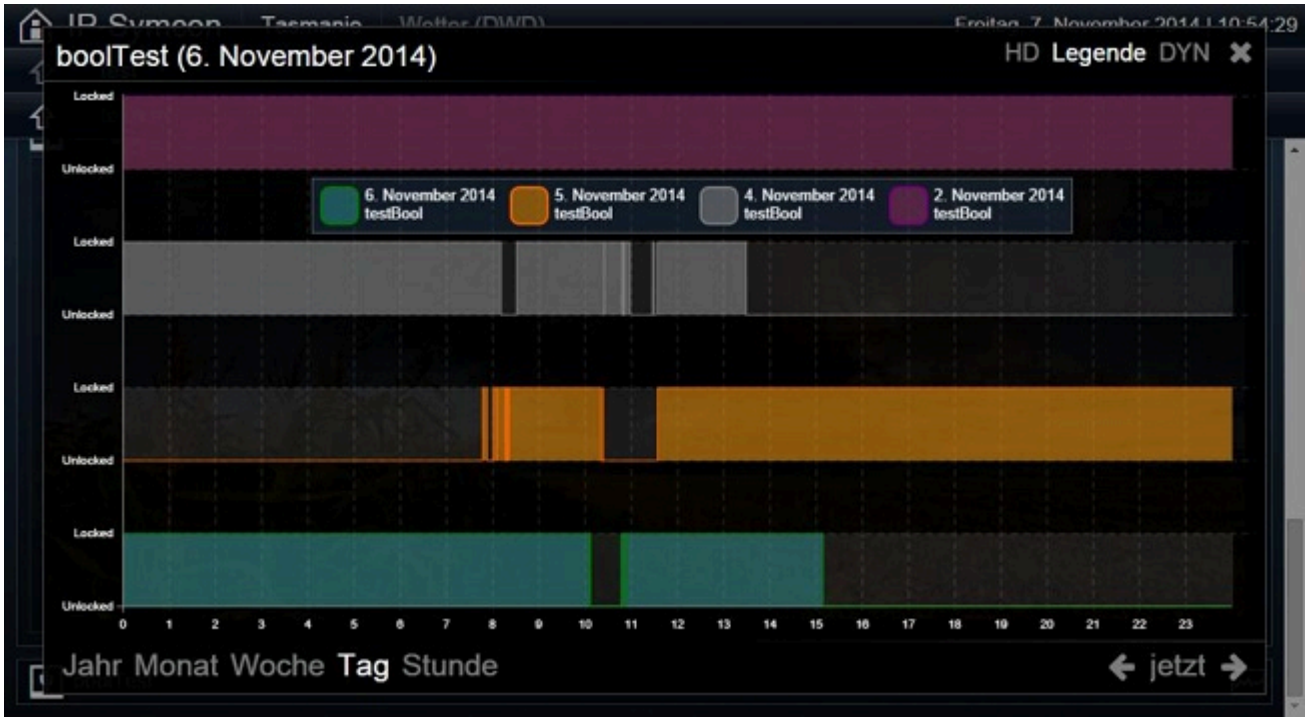
-Fix: Beim Wochenplan wurde unter bestimmten Umständen die Eingabe von Quellcode nicht erlaubt
-Fix: Problem mit Zwischenablage beim Kopieren von ObjektIDs
-Fix: JSON-RPC Klasse wirft nun Exceptions, damit diese korrekt abgefangen werden können
-Fix: Fehlende 1-Wire Kompatibilitätsfunktionen ergänzt
-Fix: Geänderte Idents bei den Modulen werden bei IPS_GetStatusVariableID berücksichtigt
-Fix: Link zum Shop springt direkt zu den Subskriptions
-Fix: Datum wird bei Boolean-Graphen jetzt immer korrekt angezeigt
-Fix: Velleman Board Modul sollte mit VM110N laufen
-Fix: Z-Wave Parameter Dialog überarbeitet
-Fix: WinLIRC kommt mit fragmentierten Paketen zurecht
-Fix: LCN Host Kommando wird korrekt ausgewertet
-Fix: Shortcuts gehen im Objektbaum Popup korrekterweise nicht mehr
-Fix: Ereignisse, die per PHP erstellt werden, werden korrekt als einmalige Ereignisse erstellt
-Fix: Vollbild-Boolean-Charts wurden nicht angezeigt, wenn Extrema aktiviert wurden
-Fix: Kleines Usability-Problem beim Erstellen von Wochenplanaktionen im WebFront gelöst
-Fix: Problem mit EventLimit bei Ereignissen gelöst
-Fix: IPS_VariableProfileExists hat sich bei der Prüfung inkonsistent zu IPS_CreateVariableProfile verhalten
-Fix: Problem mit Symcon-Tray nach Windows-Start gelöst
-Fix: Access.log schreibt bei Vorhandensein die X-Forwarded-IP in Logfile
-Fix: Problem mit xComfort Rückmeldungen gelöst
-Fix: Problem mit Setzen von DS2405 Pins über PHP gelöst (TMEX_F05_SetPin)

V3.2->V3.3 (Q1/2015)

WESENTLICHE ÄNDERUNGEN

Neuer Graphentyp Bool-Chart für Bool-Variablen

Auf vielfachen Wunsch und aufgrund hoher Funktionalität ermöglicht IP-Symcon nun die präzise Darstellung von Bool-Variablen innerhalb von Graphen.
Es ist möglich einen Bool-Chart sowohl im Einzelgraphen als auch Multigraphen darzustellen. Es werden ausschließlich die Rohdaten genutzt, wodurch eindeutige Flanken in den Graphen dargestellt werden. Dies führt zu einer präzisen Veranschaulichung der aufgenommen Daten. Eine weitere Besonderheit zum bisherigen Linien- oder Bar-Multi-Chart ist die Nutzung des jeweils eigenen Profils der Bool-Variablen innerhalb eines Multi-Charts. Dies führt zu einer hohen Übersichtlichkeit.



Unterstützung für neue EnOcean Geräte und EnOcean EEPs

Neben neuen Geräten wie z.B. Eltako FWS61, Eltako FSB14, Eltako FSB61, Eltako FSB71 können nun auch eine Vielzahl von Geräten eingebunden werden, welche die offiziellen "EnOcean Equipment Profiles" (kurz EEPs) unterstützen. Dadurch sind fast alle neuen EnOcean Geräte mit IP-Symcon kompatibel sein. Zusätzlich wurde auch die Unterstützung für BiDirektionale Geräte, wie zum Beispiel Stellantriebe verbessert. Somit können die Geräte SAB02 von "Thermokon", B2 Drive von "alphaEOS" und das MD15 von "Kieback & Peter" nun genutzt werden. Eine vollständige Liste der aktuell unterstützen Geräte und EEPs kann der EnOcean Geräteliste entnommen werden.

Verbesserungen der Verwaltungskonsole

Zur Einrichtung von IP-Symcon ist die Verwaltungskonsole besonders wichtig. Aus diesem Grund wurde in diesem Update ein besonderes Augenmerk auf dessen Anwenderfreundlichkeit gelegt. Es wurden, um eine Einrichtung und Bedienung zu beschleunigen, viele Tastenkombinationen hinzugefügt. Beispielsweise ist nun das Hinzufügen, Duplizieren und Verlinken von Objekten mit einem einfachen Tastendruck möglich. Durch das Update des Schnellfilters ist die integrierte Suche im Objektbaum noch

einfacher zu bedienen. Damit die Konsole sich konsistent verhält, lassen sich schnell alle Dialoge per ESC schließen und die Dialogfelder per Tabulator durchschalten.
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Edition nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Edition von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 3.3 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 3.2 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 3.3 auftreten.

Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

LISTE DER ÄNDERUNGEN

Neue Funktionen
-Neu: Boolean-Multigraphen (siehe Screenshot)
-Neu: Boolean-Variablen werden in Graphen besser dargestellt
-Neu: EnOcean EEP Module (Nur Empfangen!)
-Neu: Eine Menge Shortcuts für die logische Baumansicht und den Willkommen-Reiter
-Neu: Duplizieren von Objekten (Noch nicht rekursiv!)
-Neu: "Instanz erstellen" Dialog überarbeitet
-Neu: "Variablen erstellen/editieren" Dialog schlanker gestaltet
-Neu: Reaggregieren wird automatisch nach Änderung des Aggregationstyps gestartet
-Neu: "Objekt bearbeiten" Dialog verbessert
-Neu: QuickFilter in der Baumansicht ist nun Case-Insensitive
-Neu: Alle Dialoge sind korrekt mit ESC schließbar
-Neu: Navigieren per Tab wurde überall korrekt korrigiert
-Neu: Alle Zusammenfassungsdialoge (Summary) entfernt
-Neu: Logos angepasst
-Neu: IPS*GetModuleListByType*
*-Neu: COMPort heißt jetzt SPRT_* und WWWReader** heißt jetzt WWW**
-Neu: Neue Module für EnOcean Enegergiezähler (EEP A5-12-00, A5-12-01, A5-12-02, A5-12-03)
-Neu: Neues Modul für Eltako FWS61
-Neu: Eltako Shutter Modul für FSB14, FSB61, FSB71
-Neu: Bool-Charts können nun unterschiedliche Profile bei Variablen anzeigen
-Neu: Auswahl für ModBus TCP über UDP
-Neu: EnOcean EEP A5-20-01 für z.B. Thermokon SAB02, alphaEOS B2 Drive oder Kieback&Peter MD15
-Neu: EnOcean Modul für alphaEOS SENSE TF-H
-Neu: Debugging verlängert sich automatisch

-Neu: TextParser, HeatingControl, ShutterControl, EventControl auf IPS_SetProperty/IPS_GetProperty umgestellt
-Neu: Geloggte Strings können über den Graphdialog angezeigt werden
-Neu: Geloggte String Variablen können auch als Vollbild "Graph" angezeigt werden

Behobene Fehler

-Fix: Sortierung im WFC "Element hinzufügen" Dialog
-Fix: AccessViolation bei der Timer-Übersicht, wenn InstanzID Null war
-Fix: Probleme mit Eltako FSS12 Modul gelöst
-Fix: Gelöschte, geöffnete Instanzen werden automatisch als Reiter entfernt
-Fix: Neue Konfiguratoren können aus dem "Konfiguratoren verwalten" Dialog erstellt werden
-Fix: QuickFilter merkt sich die letzte Eingabe
-Fix: Bessere Exception bei fehlerhaften JSONRPC Requests
-Fix: Behebt Sicherheitslücke in Verbindung mit Proxies/ReverseProxies
-Fix: Behebt Fehlermeldung bei JSONRPC Klasse
-Fix: Fehlerhafter Punkt bei Float-Werten im WebFront bei englischsprachigen Systemen
-Fix: Probleme mit leeren HD-Graphen
-Fix: Ventilprosition bei FHT wird bei der Entkalkungsfahrt korrekt ausgewertet
-Fix: Z-Wave Protection Klasse wird nun in Version 2 unterstützt
-Fix: Problem mit Z-Wave SensorAlarm Klasse gelöst
-Fix: Profilmanager kann besser mit Pfeiltasten bedient werden
-Fix: Lade-Dialog ist nun mittig positioniert
-Fix: Fehlende $_IPS Variablen im Skript Editor hinzugefügt
-Fix: Problem mit nicht angezeigten, ersten Kategorien im WebFront korrigiert
-Fix: Kleine Usability-Probleme mit Wochenplan korrigiert
-Fix: Namensänderungen werden auch in Reitern berücksichtigt
-Fix: Fehler beim Schalten mit dem Shutter Control korrigiert

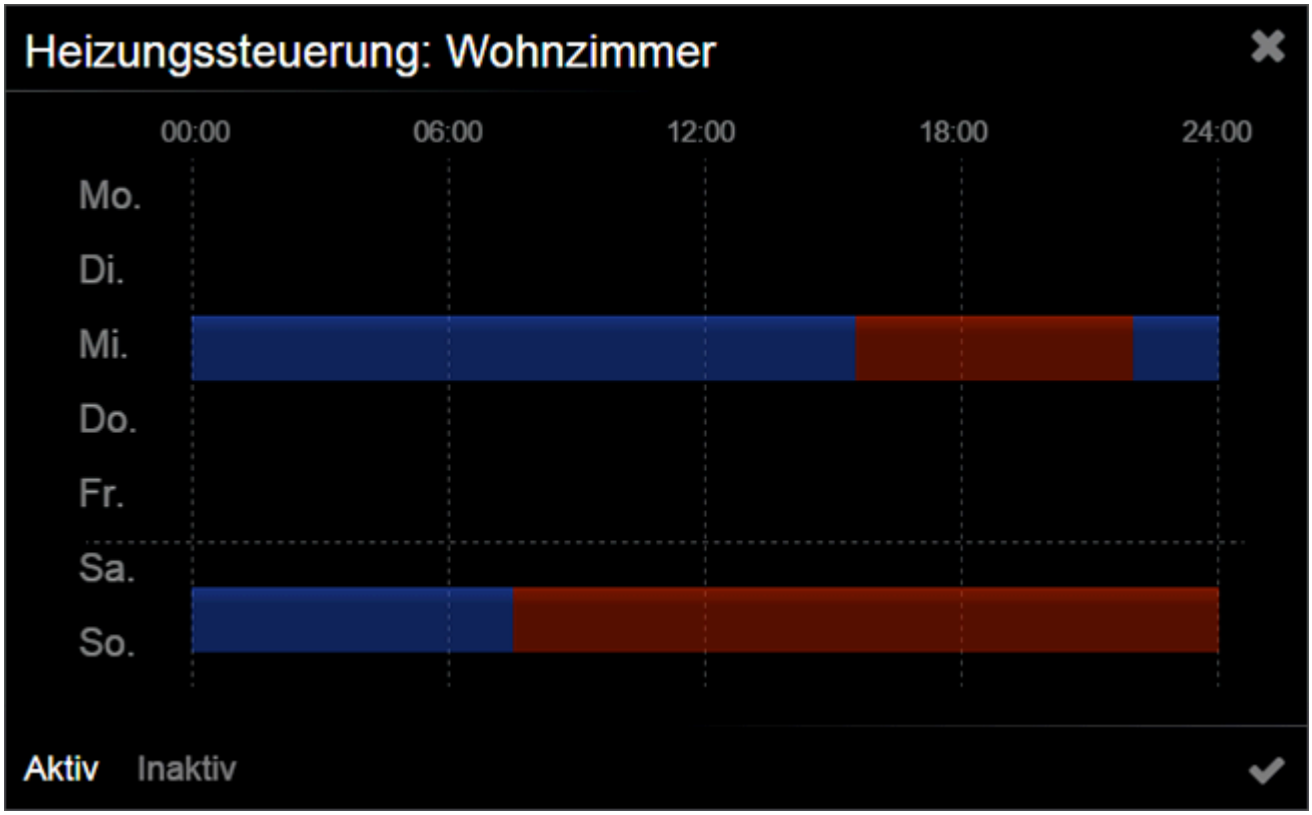
Änderung Dokumentation

Die Doku befindet sich in einem Status der Generalüberholung und wird stetig aktualisiert.
-Neu: [Schnelleinstieg]
-Neu: Bool Chart
-Neu: Links
-Neu: Gerätelisten (1-Wire , ALLNET , DMX , Eaton xComfort , KNX , EnOcean/EEP , Z-Wave)
-Neu: Befehlsliste vervollständigt (Z-Wave)
-Update: Instanzen
-Update: Ereignisse
-Update: Fernzugriff
-Update: Geräte einbinden
-Update: Links verwenden
-Update: Email Benachrichtigung

V3.1->V3.2 (Q3/2014)

WESENTLICHE ÄNDERUNGEN

Neuer Ereignistyp: Wochenplan
Eine der Neuerungen ist der Ereignistyp Wochenplan, mit dem es nun noch schneller möglich ist,
einzelne Zustände z.B. von Licht-Szenarien oder der Heizungssteuerung für jeden Wochentag anzulegen oder zu ändern. Im WebFront werde geplante Zustände durch farbige Balken übersichtlich dargestellt und sind intuitiv anpassbar.



HD für Multi-Charts
Die Multi Charts im WebFront bieten die viel gewünschte HD-Funktion an, mit der es nun möglich ist, die einzelnen Graphen detailliert in hoher Dichte anzuzeigen.
Updates für Systeme
Neue Module für digitalSTROM erlauben das Setzen von Stimmungen direkt aus dem WebFront heraus und endlich den viel gewünschten Rückkanal, der Änderungen der Raumstimmung in IP-Symcon wiederspiegelt. Zusätzlich wurde die Eaton xComfort Unterstützung überarbeitet, sodass die neuen Geräte mit gerouteten Statusmeldungen eingebunden werden können.
Skin-Manager in der Verwaltungskonsole
Der neue Skin-Manager erlaube die bequeme Verwaltung aller Skins und ist nun direkt in der Verwaltungskonsole unter Kern-Instanzen verfügbar.
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Version nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Version von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version

3.2 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 3.1 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 3.2 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

LISTE DER ÄNDERUNGEN

Neue Funktionen
- Neu: Wochenplan Ereignisse können erstellt werden
- Neu: IPS_SetEventScheduleAction
- Neu: IPS_SetEventScheduleGroup
- Neu: IPS_SetEventScheduleGroupPoint
- Neu: HD für MultiCharts
- Neu: dSRoom und dSApartment Module für digitalStrom zur Ansteuerung der Stimmungen
- Neu: Skins werden über die Konsole verwaltet
- Neu: ZW_ConfigurationSetValueEx für Z-Wave Parameter mit variabler Größe (IPSD-86)
- Neu: WinLIRC wertet Rückmeldungen aus und wirft somit auch Fehlermeldungen (IPSD-116)
- Neu: WebFront: Inhaltswechsler reagiert auf Änderungen von Links
- Neu: WebFront: Inhaltswechsler reagiert auf Verstecken/Sichtbar machen von Objekten

Behobene Fehler
- Fix: Aus aktuellem Anlass (POODLE) wird SSLv3 nicht mehr erlaubt
- Fix: IPS_SetEventCyclicTimeFrom/IPS_SetEventCyclicTimeTo überprüft das angegebene Datum
- Fix: Icons der WebFront Menüleiste wurden unter iOS7 leicht verschoben angezeigt
- Fix: WebFront: CSV auch im FF und IE auswählbar
- Fix: WebFront: CSV mit besserer Formatierung
- Fix: WebFront Retro/Mobile können korrekt separat aktiviert/deaktiviert werden
- Fix: Link zur Doku innerhalb der Konsole korrigiert
- Fix: Probleme mit Ereignissen bei Zeitzonen ohne Sommerzeit korrigiert
- Fix: Reloads im WebFront durch z.B. ungültige Float-Werte von ModBus Instanzen korrigiert
- Fix: OneWire DS2413 überprüft komplementär Bits (IPSD-111)
- Fix: Archivhandler Ansicht Zeitraum Auswahl fehlerhaft (IPSD-38)
- Fix: WebFront-Konfigurator überschreibt existierendes Objekt mit gleichem Namen (IPSD-121)
- Fix: Beim Wechsel der Listenansicht des Objektbaums wird ein gesetzter Filter nicht berücksichtigt (IPSD-123)
- Fix: Unnötige Scrollbar bei Inhaltswechslern ohne Navigationsleiste (WEBFRONT-58)
- Fix: Name beim Duplizieren von Profilen werden auf korrektheit überprüft
- Fix: DMX hat ggf. nach 49,7 Tagen ohne Windows Neustart nicht mehr korrekt funktioniert
- Fix: Kleine Änderung die bei KNX evtl. die "Not connected to interface" Probleme löst

V3.0->V3.1 (Q2/2014)

WESENTLICHE ÄNDERUNGEN

Multi-Charts
Eine der Neuerungen sind die Multi Charts , mit denen es nun einfach möglich ist mehrere Graphen in einer Ansicht darzustellen. Es können so z.B. die Verbrauchskosten von dem laufendem Jahr mit dem des Vorjahrs graphisch gegenübergestellt werden, um so schnell Unregelmäßigkeiten zu entdecken. Dabei kann jeweils ein beliebiger Titel, Linien und Füllfarbe ausgewählt werden. Die Charts können nach der Konfiguration direkt im WebFront angesehen oder über das Graph-Element im Vollbild Modus eingebettet werden.
Fernzugriff auf Konfiguration mit Kennwortschutz
Eine weitere neue Funktion ist der Fernzugriff. Mit ihm ist es nun leicht möglich die Verwaltungskonsole von einem anderen Rechner aus zu nutzen. So lassen sich von überall Änderungen der Konfiguration zu Hause oder beim Kunden durchführen. Durch den neuen, integrierten Kennwortschutz ist dies auch über das Internet sicher möglich.
Komfortabler Profilmanager
Variablenprofile können nun noch schneller und einfacher mit Hilfe des Profil-Manager bearbeitet werden. Dabei ist dieser in einem eigenem Fenster mit viel Platz ausgestattet worden. Änderungen an den Assoziationen sind nun direkt möglich ohne ein zusätzliches Fenster öffnen zu müssen. Die Sortierung nach Variablentyp vereinfacht die Suche nach dem Profil. Zusätzlich können Profile schnell dupliziert und deren Nutzung in den jeweiligen Variablen untersucht werden.
Updates für Systeme
Des Weiteren wurde viele Module überarbeitet. So wurden z.B bei LCN neue Umrechnungsfaktoren, die Unterstützung des S0-Counter vom BU4L hinzugefügt oder bei diversen Z-Wave Geräten die Kompatibilität verbessert. Alle Details können der Liste der Änderungen entnommen werden.
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Version nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Version von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 3.1 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 3.0 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 3.1 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Sofern die Verwaltungskonsole von einem anderen Rechner aus genutzt wird, muss der Fernzugriff aktiviert werden. Fernzugriff aktivieren
Sofern die Verwaltungskonsole von einem anderen Rechner aus genutzt wird, muss diese einmalig erneut aus unserem Download-Bereich geladen werden
Die Verwaltungskonsole kommuniziert über den WebFront Port (82) anstatt dem SOAP Port (3773)
Die JSON-RPC API ist lokal immer aktiv, und ist von extern ausschließlich über das Fernzugriff-Kennwort und eurem Lizenz-Benutzernamen zugreifbar.
SOAP ist standardmäßig deaktiviert, kann jedoch jederzeit aktiviert werden. Spezialschalter
Die Felder CyclicDateFrom/CyclicDateTo/CyclicTimeFrom/CyclicTimeTo von IPS_GetEvent wurden geändert. Bitte die Dokumentation beachten.

LISTE DER ÄNDERUNGEN

Neue Funktionen
Neu: Multi-Charts
Neu: Profile werden über den neuen Profilmananger bearbeitet
Neu: Konsole arbeitet über die neue JSON-RPC API (=Fernzugriff, Passwortgeschützt)
Neu: Konsole benötigt für Fernzugriff einen Benutzernamen/Kennwort
Neu: Proxy Support für Verwaltungskonsole
Neu: API-Vorbereitungen für neue App-Versionen
Neu: Automatische Suche von IP-Symcon Servern via UPNP/SSDP
Neu: Variablenprofil-Einstellungen aufgeräumt
Neu: Upgrade auf PHP 5.4.24
Neu: Unterstützung für Farben bei der Werteauswahl (WebFront)
Neu: Retina-Support für den Farbauswahl-Dialog
Neu: Unterstützung für umgerechnete Werte beim W&T Counter Modul
Neu: Startdatum/Enddatum/Startzeit/Endzeit werden über neue Funktionen gesetzt (Alte Funktionen weiterhin vorhanden)
Neu: LCN - Neue Umrechnungsfaktoren (wie auch in der LCN Pro vorhanden)
Neu: LCN - S0-Counter des BU4L werden unterstützt
Neu: LCN - Unterstützung für Ausgang 4
Neu: Z-Wave - Profile/Namen werden für METER Klasse besser gesetzt
Neu: Z-Wave - Battery-Low Anzeige für Z-Wave BATTERY Klasse
Neu: Z-Wave - Konfigurator kann Untergeräte korrekt einsortieren
Neu: DMX - RGB Modul für bessere WebFront Integration
Neu: Module für SERVO-868
Neu: IsCharging, BatteryRemainingTime, BatteryMaxTime für Sys_GetBattery
Neu: Unterstützung für "letzter Wochentag im Monat und letzter Tag im Monat" bei Ereignissen
Neu: Graph-Dialog im WebFront zeigt nun nur gültige Variablen/Charts zum auswählen an
Neu: Sortierung wird im Objektbaum pro Ansicht gespeichert
Neu: Suchen/Ersetzen in Dateien merkt sich letzte Begriffe
Neu: Spalten bei Timer Informationen und Ereignis Informationen können sortiert werden
Neu: Intern alles auf UnixTime umgestellt, anstatt der proprietären TDateTime von Delphi
Neu: Modul für ALLNET 5000 / ALLNET 3500 (neue 3.x Firmware)
Neu: Status wird bei ALL3073 ausgelesen
Neu: Empfang von Push-Nachrichten kann direkt im Konfigurator/Notification-Control aktiviert/deaktiviert werden
Neu: Transparente Füllfarbe bei Multi-Charts

Neu: dSS-Splitter kann nun auch mit einem alternativen Port umgehen
Neu: DMX_SetRGB, um die RGB Instanz per PHP anzusteuern
Neu: FS20, FS20EX und FHT intern auf IPS_SetProperty/IPS_GetProperty umgestellt
Neu: HID und MediaPlayer Modul auf IPS_GetProperty/IPS_SetProperty umgebaut-Fix: Update auf OpenSSL 1.0.1g
Neu: Status von ALL3075V2 wird eingelesen
Neu: Status von ALL3075 wird eingelesen
Neu: HTTP_X_FORWARDED_FOR wird ausgewertet
Neu: Lizenz kann direkt über die Verwaltungskonsole geändert werden
Neu: DLL vom MediaPlayer aktualisiert
Neu: ModBus unterstüzt 64bit Werte
Neu: Konsole kann mit Parameter /locationview gestartet werden und zeigt sofort die logische Baumansicht
Neu: DMX_FadeRGB und DMX_FadeRGBDelayed (respektive wurde DMX_SetValue zu DMX_SetChannel ect. umbenannt. Die alten Funktion sind weiterhin als Kompatibilitätsfunktionen verfügbar)
Neu: Basis Authentizifierung greift beim WebFront nur noch für den "user"-Ordner (Sicherheit)
Neu: Mobile/Retro Reiter im WebFront Konfigurator umstrukturiert
Neu: Sicherheitshinweis beim Deaktivieren des Mobil-Zugriffs
Neu: Unterstützung für Pool-Sensor bei Oregon Wetterstationen
Neu: WAC_Reset($id) - Befehl um den MediaPlayer bei einem Ausfall wieder zum Leben zu erwecken. Feedback ist erwünscht!
Neu: Module werden im "Über" Dialog nach Namen sortiert

Behobene Fehler

Fix: Probleme mit der Berechnung von Ereignissen korrigiert
Fix: Mehr Abstand beim Speichern Button im Farbkreis
Fix: EIS8 bei KNX hat nur eine 0 gesendet
Fix: Probleme mit IPS_RunScriptEx/IPS_RunScriptWaitEx über JSON-RPC gelöst
Fix: IMAP/POP3 Mails werden korrekt sortiert
Fix: Probleme mit Schrittweiten < 1 behoben
Fix: Bei RGB Slidern kann nun auch die Prozentzahl geklickt werden
Fix: Konsole stürzte ab, wenn Übergeordnete Instanz auf sich selbst gerichtet wurde
Fix: Übergeordnete Instanz kann nicht mehr auf sich selbst gerichtet werden
Fix: TextBox zeigte fälschlicherweise HTML an
Fix: SMTP funktioniert nun korrekt mit iCloud/MobileMe
Fix: EnOcean Thermokon Instanz kann korrekt konfiguriert werden
Fix: Problem beim Löschen vieler Objekte
Fix: Problem beim Löschen von Objekte, wenn Unterobjekte mit ausgewählt wurden
Fix: Countdown im Skripteditor durch absoluten Zeitpunkt ersetzt.
Fix: Z-Wave - Problem mit fehlerhaften Fibaro FGR222 umgangen
Fix: Z-Wave - Probleme mit SENSOR_ALARM Klasse gelöst
Fix: Z-Wave - Probleme mit SENSOR_MULTILEVEL Klasse gelöst
Fix: Z-Wave - Probleme mit ALARM Klasse gelöst
Fix: Z-Wave - Probleme mit Batterieanzeige korrigiert
Fix: Skalierungsprobleme bei sehr kleinen Graphen gelöst
Fix: Konfiguratoren können nun korrekt per Tastatur/Pfeiltasten bedient werden (#IPSD-61,IPSD-64)
Fix: ArtNet sendet eine korrekte Versionsnummer
Fix: Z-Wave Philio 4-1 Sensor erstellt die Variable nun wirklich korrekt
Fix: Ersetzen in Dateien funktioniert korrekt

Fix: EIS5 wird korrekt abgerundet
Fix: Alle Gruppen Funktionen bei KNX können korrekt ausgewählt werden
Fix: WebFront zeigte keine Fehlermeldung, wenn Aktionsskript einen Fehler geworfen hat
Fix: Probleme beim Erstellen von Instanzen mit KNX Konfigurator behoben
Fix: Skin Manager wird auf dem korrekten Port geöffnet
Fix: WebFront Editor konnte Kategorie nicht auswählen, wenn diese zwischenzeitlich ungültig geworden ist
Fix: AccessViolation beim Laden der Geräte im WebFrontConfigurator, wenn Token/Lizenz unpassend war
Fix: EIS6 wird korrekt gesendet/interpretiert
Fix: xComfort 12-Kanalaktor wird korrekt unterstützt
Fix: Stop Kanal für KNX Shutter wird korrekt gesetzt
Fix: Automatischer Reconnect für Dashboard reagiert nun auch /form Parameter
Fix: IPS_SetScriptTimer kann wieder korrekt mit Werten >=24h umgehen
Fix: Sekündliche Fehlermeldung, wenn ArtNet Gateway nicht verbunden war
Fix: Legende kann nun über den WFC aktiviert/deaktiviert werden
Fix: Konfiguration von monatlichen Ereignissen korrigiert
Fix: Z-Wave: Inclusion/Exclusion sollte nun mit Razberry und Z-Wave.ME Stick korrekt funktionieren.
Fix: StartTime in IPS_GetScriptThread war falsch
Fix: Problem mit Sommerzeitumstellung korrigiert
Fix: FS10 kann wieder konfiguriert werden
Fix: Problem mit OZW Variablenprofilen gelöst
Fix: Fehler bei ausgelösten Ereignisse, welche Werte vom falschen Variablentyp enthielten
Fix: Skriptzuweisungen in Instanzkonfigurationen können entfernt werden
Fix: Bei alten KNX Instanzen konnten zusätzliche GAs nicht gespeichert werden
Fix: Profilmanager hatte verschobene Buttons
Fix: ModBus: Int64 sollte nun korrekt kodiert werden
Fix: Problem mit WAC_* Kompatibilitätsfunktionen
Fix: Alte iOS App sollte wieder korrekt funktionieren
Fix: Profilnamen werden korrekt auf Groß/Kleinschreibung geprüft

V2.7->V3.0 (Q3/2013)

WESENTLICHE ÄNDERUNGEN

IP-Symcon WebFront inkl. Editor
Eine besondere neue Funktion ist der WYSIWYG Editor, über den die Einrichtung von komplexen Visualisierungen zu einem Kinderspiel wird. Natürlich ist euer altes WebFront aus IP-Symcon 2.x vollständig dazu kompatibel und kann direkt mit dem neuen Editor bearbeitet werden. Zusätzlich kann das neue WebFront über Skins z.B. farblich umgestaltet werden, welche entweder direkt bei uns im Forum heruntergeladen, oder bei entsprechendem Know-How selbst erstellt werden können. Das neue WebFront bietet außerdem noch Unterstützung für MJPEG-Streams, Audio-Benachrichtigungen, komfortable Konfiguration von zyklischen Ereignissen und einen neuen Farbkreis für RGB-Geräte. Die vollständige Liste mit allen Änderungen ist unterhalb der Migration zu finden.
Neue unterstütze Geräte und Funktionen
Im Bereich DMX unterstützt IP-Symcon nun sämtliche Art-Net-Controller mit verschiedenen Universen.
Im Bereich LCN werden die neuen LCN-Module unterstützt, welche bis zu 12 Variablen zur Verfügung stellen.
Web-Schnittstelle (JSON-RPC)
Die neue, offzielle JSON-RPC Schnittstelle erlaubt eine einfache Anbindung von Dritthersteller-Software, eigenen mobilen Anwendungen und der einfachen Kommunikation zwischen IP-Symcon Servern. Wer Daten zwischen mehreren IP-Symcon Servern austauschen wollte, kann dies jetzt mit wenigen Handgriffen tun. Die Dokumentation liefert dazu unter dem Punkt Datenaustausch einige Beispiele.
(Um den WebFront-Editor nutzen zu können ist mindestens eine IP-Symcon Professional Version nötig. Alle anderen Funktionen sind in der Basic, Professional und Unlimited Version von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 3.0 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop)

VIDEO ZUM WEBFRONT-EDITOR

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.7 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 3.0 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Benutzerdefinierte Icons werden in IP-Symcon 3.0 über WebFront-Skins eingebunden. (Dokumentation)
Die Funktion AC_GetLoggedValues ist nun offiziell dokumentiert, hat sich jedoch leicht in der Funktionsweise geändert. Sofern genutzt, kann das in der Dokumentation zur Verfügung gestellt Mini-Skript die alte Funktionsweise vollständig wiederherstellen.
Der Typ von eigenen Profilen muss für das neue WebFront zum Variablentyp passen. Sofern eigene Profile nur über die Konsole erstellt wurden, ist dieser Hinweis nicht relevant.
Wenn im WebFront Instanzen unterhalb von Instanzen angezeigt werden sollen, dann muss in jedem Konfigurator die Option "Verschachtelung" aktiviert werden.
Um das On-Screen-Keyboard im WebFront unter Windows nutzen zu können, kann folgende Extension für Google Chrome installiert werden: Virtual Keyboard
Für die Graphen im Dashboard gibt es eine Änderung in der Nutzungsweise: Forum

LISTE DER ÄNDERUNGEN

Komplett für HTML5-Browser neu entwickelt
Der integrierte WebFront-Editor ersetzt den unbequemen WebFront-Konfigurator
Eigene Skins können mit den Skin-Manager erstellt oder fertige direkt aus der Community heruntergeladen werden!
RGB-Slider haben nun zusätzlich einen HSL-Farbkreis
Audio-Notificatations über WFC_AudioNotification
MJPEG-Stream können über das neuen Stream-Mediaobjekt angezeigt werden. Medien
Graphen-Rohdaten können direkt nach CSV exportieren werden
Graphen im WebFront können für jede Variable deaktiviert werden (trotz aktiviertem Logging)
Alle Icons und Grafiken sind für Retina-Displays optimiert.
Das WebFront sieht nun auf dem iPad3+ und hochauflösenden Display einfach toll aus!
Lädt schneller - besonders bei langsamen Internetleitungen. Die schicke Ladeanimation versüßt die Wartezeit
Natives Scrolling auf iOS & Android Geräten
Zusatz Scrollleisten können für ältere Touchscreens alternativ per Skin aktiviert werden (Download)
Das Editieren von Text-Boxen nutzt nun das integrierte OSK des Touch-Geräts
DWD-Wetter benötigt keinen Benutzernamen/Passwort mehr
Inhaltswechsler kommt nun auch mit sehr vielen Objekten zurecht (z.B. Webcam Archiv Funktion)
Ereignisse werden nun direkt untereinander inklusive Konfiguration und nächster Ausführung angezeigt
Die neue JSON-RPC API arbeitet direkt im WebServer ohne kostbare PHP-Threads zu belegen
Über die neue JSON-RPC API können Sie mit wenigen Handgriffen Daten austauschen. Datenaustausch
Für alle Technikbegeisterten: Das neue WebFront ist mit Google Dart erstellt. Kein Dojo mehr.
MIN/MAX Anzeige bei Graphen
Farbauswahl mit persönlich anpassbarem Farbspeicher!
Verbesserungen beim dynamischen Skalieren von Graphen
Skin-Editor hat Option zum Einfärben der Icons
Skin-Icons können in der Konsole ausgewählt werden
Medien (Streams) können editiert werden
Es wird ein Ladekringel beim Klick auf ein Info-Widget angezeigt (Wenn ein OnKlick Skript definiert wurde) (#WEBFRONT-19)
Wenn mehrere Popups per WFC_SendPop geöffnet werden, gibt es einen "Alle Popups schließen" Button (#WEBFRONT-2)

Unterstützung für Datenpunkte vom neuen HomeMatic Stellantrieb

Neue Systeme/Module

Neu: DMX Splitter unterstützt nun ArtNet (Unicast)
Neu: LCN untersützt die neuen Module mit 12 Variablen
Neu: Native JSON-RPC Schnittstelle

Neue Funktionen

Neu: GZIP-Komprimierung für den WebServer
Neu: X-Forwarded-For wird im WebServer ausgewertet und an PHP weitergeleitet
Neu: Netzwerkbibliothek aktualisiert. Behebt hoffentlich Probleme mit POP3/IMAP
Neu: Silent-Eigenschaft für TIPSWebBrowser um alle Javascript Fehler und Popups zu unterdrücken

V2.6->V2.7 (Q1/2013)

WESENTLICHE ÄNDERUNGEN

Jetzt direkt auf dem Smartphone oder Tablet benachrichtigen lassen!
Mit der Einführung von IP-Symcon 2.7 und der Apps in der Version 2.0 gibt es endlich die Möglichkeit direkt Push-Benachrichtigungen an Ihre Smartphones und Tablets zu versenden. Lassen Sie sich zum Beispiel bei einer fertigen Waschmaschine, beim Klingen der Tür oder bei Alarmzuständen sofort informieren. Viele weitere Ideen finden Sie auch in unserem Community-Forum

Neue unterstütze Geräte und Funktionen
Zur besseren und komfortableren Einrichtung haben wir die Konfiguratoren überarbeitet, sodass diese die volle Größe der Konsole nutzen können. Dadurch haben Sie bei vielen vorhandenen Geräten oder Adressen einen vollständigen Überblick. Zusätzlich haben wir einen Konfigurator für OneWire integriert und den KNX-Konfigurator so erweitert, dass er die Rückmeldeadresse direkt in IP-Symcon importiert.
Ein neues Modul ist für die TermoJET und LevelJET Systeme der Fa. ProJet verfügbar. Die Geräte können in das IPS-868 Funksystem über ein neues Funkgateway (JKM-868) angebunden werden. Dadurch lassen sich die empfangenen Temperaturen oder Füllstände in IP-Symcon auswerten. Alle Geräte des IPS-868 Funksystems können über den ProJET Shop erworben werden.
Im Bereich Z-Wave haben wir die Klasse MULTI_INSTANCE komplett überarbeitet, sodass die neuen Geräte von Fibaro, wie z.B. der 2-fach Aktor oder der vielseitige Binärsensor einfach und mit allen Funktionen genutzt werden können.
Ferner wurden wieder neue Geräte unseres Vertriebspartners ALLNET in IP-Symcon integriert:

- ALL3073WLAN - Netzwerksteckdose zum Schalten über WLAN (16A)
- ALL3075V2 - Netzwerksteckdose zum Schalten über LAN (8A)
- ALL4075 - Netzwerk / WLAN Relais 4-fach 230V/10A
- ALL4076 - IP-Steckdosenleiste 6-fach, schaltbar per Netzwerk (6*10A)
- ALL3418V2 - LAN / WLAN IP Thermometer (ALL3418v2)

(Alle Funktionen sind in der Basic, Professional und Unlimited Version von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 2.7 updaten. Bitte beachten Sie, dass das Nutzen der Push-Benachrichtigungen eine Verbindung zum Internet und eine gültige Subskription voraussetzt. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.6 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 2.7 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Falls Sie von Z-Wave einen der Mehrfachaktoren (z.B. Fibaro Dual Switch) verwenden, so müssen Sie die Untergeräte über eigene Instanzen einbinden, da die Unterstützung entsprechend verbessert wurde. Beachten Sie auch, dass die ZW_SwitchModeEx Funktion entfernt wurde, und jedes Untergerät korrekt per ZW_SwitchMode angesprochen wird.

LISTE DER ÄNDERUNGEN

Jetzt direkt in den Mobilanwendungen benachrichtigen lassen!
- Neu: Push-Benachrichtigungen (Über Kern-Instanzen -> Notification) (Dokumentation Notification Control)
- Neu: WFC_PushNotification($wfcid, $title, $message, $icon, $targetID); (Dokumentation: (link: page://ByMmKKFK2unaOdIE text: WFC_PushNotification))

Neue Systeme/Module
- Neu: DMX Module komplett erneuert. Es werden intern die nativen IP-Symcon I/O Module genutzt
- Neu: Neue ALLNET Geräte werden unterstützt (ALL3073 WLAN, ALL3075V2, ALL3418V2, ALL4075, ALL4076)
- Neu: KNX-Konfigurator importiert zusätzliche "Empfangs-Gruppenadressen" aus der ESF-Datei
- Neu: OneWire besitzt einen komfortablen Konfigurator. (Dies ersetzt die Suche innerhalb der Instanz!)
- Neu: Z-Wave unterstützt jetzt korrekt Mehrfachgeräte mit der MULTI_INSTANCE Klasse
- Neu: Module für das IPS-868 System (IPS-JKM-868 ThermoJET-, LevelJET-Transmitter)
- Neu: WebFront Konfigurator umgestaltet. Gruppiert. Es gibt jeweils mehr Platz.
- Neu: Alle Konfiguratoren nutzen die komplette Bildschirmgröße

Neue Funktionen
- Neu: Fehlerbehandlung der I/Os wird zentral vom Event Control übernommen
- Neu: WWWReader kann deaktiviert werden. (Praktisch für Systeme wie digitalStrom/OZW)
- Neu: Proxy-Support ist nun IPS-Systemweit einheitlich. Übernimmt Konfiguration aus dem LiveUpdate.
- Neu: Module nutzen nun zu 90% die neuen IPS_SetProperty Befehle zur konfiguration. (Dokumentation: (link: page://pGgwGLaGoWzdLcTX text: IPS_SetProperty))
- Neu: Warnung, wenn neues Dashboard-Formular mindestens 10% kleiner ist, als das Original. (Workaround für Teilweise fehlerhafte Uploads)
- Neu: Alte, seit 2010 abgekündigte MBT_* Funktionen entfernt
- Neu: Sortierung der Spalten im ArchiveControl
- Neu: Velleman Board liest die AnalogInputs direkt in Variablen ein
- Neu: M-Bus unterstützt nun weitere Spezialcodierungen für Zählerüberlauf/Zählerunterlauf
- Neu: Unterstützung für 2-fach Bidi-Aktoren von Eltako. (Neue Option im Eltako Switch Modul)
- Neu: Modul für Thermokon Thanos (EnOcean)

Behobene Fehler
- Fix: Timeout bei Abfrage einiger M-Bus Geräte
- Fix: WebFront Ladeanimation bleibt hängen
- Fix: Überprüfung der maximalen WebFront Konfigurationen funktioniert nun korrekt
- Fix: Zeitüberschreitungsfehler bei KNX Sendeanfrage
- Fix: Potenzieller Division durch Null bei der Graphengenerierung

Fix: AccessViolation beim Erstellen des Objektbaumes über Ansicht hinzufügen
Fix: AccessViolation beim Hochladen von Dashboard-Formularen
Fix: ALLNET3691 Phasen 4-6 werden korrekt angezeigt
Fix: Potenzielle Sicherheitslücke im WebFront Retro/Retro-Mobile
Fix: Cutter mit #0
Fix: Zeilenumbrüche werden in der Objektbeschreibung korrekt angezeigt
Fix: KNX: Fehler beim Empfangen von Rückmeldungen beim Siemens N146/02 IP-Gateway

V2.5->V2.6 (Q3/2012)

WESENTLICHE ÄNDERUNGEN

IP-Symcon startet, lädt und agiert schneller
In der Version 2.6 haben wir besonders die Performance verbessert. Die Startzeit des Dienstes konnte minimiert werden, so dass auch bei großen Projekten ein Aktualisieren der Software nur wenige Minuten in Anspruch nimmt. Da die Konsole zusätzlich nahezu komplett überarbeitet wurde, sind auch dort Start- und Ladezeiten minimal. Das WebFront wurde optimiert und den neusten Sicherheitsanforderungen der aktuellen Browser angepasst. Durch die neuen Verbesserungen wird außerdem die CPU weniger belastet, was sich besonders auf älteren Systemen schnell bemerkbar macht. Auch beim WebFront Konfigurator haben wir auf Feedback reagiert und dessen Bedienung verbessert.
Neue Hardware-Systeme
Darüber hinaus haben wir neue Systeme und Module integriert. Problemlos können nun in IP-Symcon u.a. das gesamte digitalStrom- und das innovative Siemens-Synco™-living-System eingebunden werden, wodurch unsere breite Palette an unterstützter Hardware perfekt ergänzt wurde. Das macht unsere Hausautomationssoftware noch flexibeler, als sie bereits ist.
Noch zuverlässiger
Wir haben zusätzliche die Version unserer integrierten Skriptsprache PHP aktualisiert, um auch in IP-Symcon die neusten Sicherheitspatches, die die PHP-Community veröffentlicht hat, zu nutzen. Wir haben weitere gewünschte Optimierungen vorgenommen, damit IP-Symcon noch besser und leistungsfähiger sein kann. Die vollständige Liste dazu kann unterhalb des Kapitels Migration eingesehen werden.
(Alle Funktionen sind in der Basic, Professional und Unlimited Version von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles "Upgrade" muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die Version 2.6 updaten. (Shop))

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.5 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 2.6 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Sofern der IPSLogger von Brownson verwendet wird, ist ein Update dieser Komponente für den fehlerfreien Betrieb der 2.6 zwingend erforderlich. Bei etwaigen Problemen mit Abstürzen empfiehlt es sich den IPSLogger temporär zu deaktivieren, um dessen Funktion als Fehlerquelle auszuschließen.

Falls auf Ihrem Windows Server 2008 R2 die Installation des VCRedist 2008 in einer Endlosschleife feststecken sollte, überprüfen Sie bitte, ob Ihr System die korrekte Version vom VCRedist 2008 installiert. Es muss die Version ohne SP1 installiert werden, da nur diese mit PHP korrekt funktioniert!
Falls Sie die COM/.NET Wrapper Funktionen von PHP nutzen, so ist mit Version 2.6 #2554 (und damit verbunden PHP 5.4.8) eine manuelle Installation der Extension erforderlich, da PHP sich entschieden hat diese auszulagern.

LISTE DER ÄNDERUNGEN

Schneller und besser
- Neu: Performance verbessert, so dass die CPU weniger belastet wird. IP-Symcon sollte sich auf Low-End Systemen wesentlich agiler anfühlen.
- Neu: Performance verbessert, so dass IP-Symcon viel schneller startet. (Bei sehr vielen Objekten reduziert sich die Ladezeit auf <1% der ursprünglichen Ladezeit)
- Neu: Neues Format zum Speichern der Settings, welches wesentlicher schneller lädt und die sporadisch auftretenden Probleme mit der "settings.xml" lösen soll.
- Neu: Konsole wurde komplett überarbeitet. Sie lädt schneller und reagiert besser.
- Neu: WebFront Konfigurator wurde optimiert
- Neu: Upgrade auf die aktuellste PHP-Version 5.4.4

Neue Systeme/Module
- Neu: Module für digitalStrom (Firmware 1.4.0 oder neuer erforderlich)
- Neu: Module für Siemens OZW (OZW Firmware 4.0.4 erforderlich.
- Neu: Z-Wave Klasse: COMMAND_CLASS_SENSOR_ALARM
- Neu: Z-Wave Fibaro Dual Switch
- Neu: Z-Wave Alarm Klasse unterstützt mehr Typen
- Neu: Modul für ALLNET 3691, 4500, 5000

Neue Funktionen
- Neu: Spezialoptionen können nun über die Tray-Anwendung eingestellt werden. (Siehe: (link: page://s6UcBMxs0S8FHhPA text: Spezialschalter))
- Neu: Bessere Fehlermeldung bei Ausführungsfehlern von PHP-Code innerhalb von Ereignissen
- Neu: Restlaufzeit-Anzeige bei Ereignissen innerhalb vom Skript-Seitenmenü
- Neu: Alle hauptsächlich genutzten $_IPS[*] Systemvariablen werden bei STRG+SPACE angezeigt.
- Neu: SerialPort: DTR/RTS FlowControl Funktionen
- Neu: Das Deaktivieren von Autoscroll gibt einen Hinweis und deaktiviert das Limit der Einträge.
- Neu: PHP Memory-Limit ist jetzt default auf 32MB. Maximal jedoch 64MB. (Siehe php.ini)
- Neu: Maximale PHP-Threadanzahl staffelt sich je nach installiertem RAM (max 40 bei 4GB, 30 bei 3GB, 20 bei 2GB, 10 bei weniger).
- Neu: PHP-Skript Queue meldet sich schon ab zehn Skripten in der Schleife.

Behobene Fehler
- Fix: Benutze Standardaktion kann wieder korrekt genutzt werden
- Fix: WebFront: Timing-Problem bei Ausführung vieler gleichzeitiger IPS_SetHidden Befehle behoben
- Fix: WebFront: Lade-Problem mit aktueller Chrome Version bei aktiviertem SSL und externem Zugriff
- Fix: VCRedist2008 wurde auf dem Server2008 R2 inkorrekt als installiert erkannt.
- Fix: Neue $_IPS Schreibweise in den Ereignis-Templates wird nun genutzt.
- Fix: LCN: Sonderzeichen bei Modulnamen wird korrekt gefiltert.
- Fix: Siemens S5: Verbindungsprobleme gelöst

Fix: Eaton xComfort: Signalstärke in Verbindung mit dem ECI LAN Gateway
Fix: Eaton Analogeingang: Fehler beim Auswerten von Prozentwerten
Fix: Fehler, dass unter Umständen $IPS_* nicht korrekt definiert war
Fix: WebFront Startkategorie wurden u.U. nicht korrekt übernommen
Fix: Reihenfolge (Objekt-Position) im ContentChanger wird korrekt ausgewertet.
Fix: LCN: R2 Regler wurde nicht korrekt ausgelesen.
Fix: Timing Problem/Umschaltzeiten für LCN Shutter
Fix: Sporadisch auftretendes Floating-Point Problem bei SetValue
Fix: Zu viele Datensätze bei einer AC_* Abfrage konnten das System negativ beeinträchtigen. Limitation auf 10000 Datensätze pro Abfrage. (Siehe: (link: page://CJ7CQ655yiILr2pO text: Limitationen))
Fix: Unendlich viele Fehlermeldungen (z.B. korrupte for-Schleife) konnten den kompletten RAM des Systems aufbrauchen.
Fix: Widgets konnten in SplitPanes platziert werden.
Fix: False-Positives bei Sys_Ping
Fix: Problematische Werte bei Sys_GetCPUInfo
Fix: Link unter Link konnte fälschlicherweise erstellt werden.
Fix: Alle FHZ Module (FS20, HMS, FHT, KS300) werfen nun direkt Fehlermeldungen, wenn die I/O Instanz nicht verfügbar ist
Fix: OSK übernimmt Zeilensprünge nicht.

Keine Abwärtskompatibilität für folgende Änderungen:

Geändert: IPS_GetFunctionList (Gibt nur noch den Namen zurück. Bitte mit IPS_GetFunction kombinieren.)
Entfernt: FHZ_GetQueue, FHZ_GetFHTQueue
Entfernt: IPS_SendMediaEventEx entfernt

Vollständige Abwärtskompatibilität für folgende Änderungen:

Neu: IPS_FunctionExists, IPS_GetFunction hinzugefügt
Neu: IPS_SetLinkChildID wurde zu IPS_SetLinkTargetID umbenannt. IPS_SetLinkChildID wird zur 3.0 entfernt.
Neu: IPS_GetLink liefert das Feld TargetID, welches identisch zu LinkChildID ist. LinkChildID wird zur 3.0 entfernt.
Neu: IPS_HasInstanceParent, IPS_GetInstanceParentID veraltet. Kann direkt über IPS_GetInstance und ConnectionID abgefragt werden.
Neu: IPS_HasInstanceChildren, IPS_GetInstanceChildrenIDs veraltet. Kann direkt über IPS_GetInstance und ConnectionID abgefragt werden. Ein Beispiel dazu befindet sich auf IPS_GetInstanceChildrenIDs
Neu: LCN_GetStatus wird nur noch emuliert. Auf GetInstance($id)['InstanceStatus'] ausweichen.

Folgende Konfigurationsfunktionen werden nur noch emuliert:

CSCK_SetOpen, CSCK_SetHost, CSCK_SetPort, CSCK_GetOpen, CSCK_GetHost, CSCK_GetPort, SSCK_SetOpen, SSCK_GetOpen, SSCK_SetPort, SSCK_GetPort

V2.4->V2.5 (Q4/2011)

WESENTLICHE ÄNDERUNGEN

Mit IP-Symcon 2.5 wurde auf die neue PHP 5.4 Version aktualisiert, die viele Verbesserungen bei Skripten und die Sicherheit dieser mitbringt. Desweiteren wurden neue Funktionen für eine Vielzahl von Hardware-Systemen hinzugefügt und bekannte Fehler beseitigt. Ein Auszug der neuen Funktionen:

Unterstützung der neuen bidirektionalen EnOcean TCM310 Geräte mit dem EnOcean Serial Protocol 3.
Unterstützung der neuen bidirektionalen EnOcean Aktoren von Eltako
Unterstützung des neuen Eaton Xcomfort Thermostats
Unterstützung für das neue Xcomfort LAN-Gateway (ECI) und deren spezielle DPL Datenpunktliste
Unterstützung vom OneWire Baustein DS2406
Komfortables LCN Modul für Rollladen/Jalousien bei Ansteuerung über Ausgänge/Relais
Unterstützung für das neue LAN-868 Gateway für das IPS Funksystem
Unterstützung für neue Z-Wave Klassen
- COMMAND_CLASS_CLOCK
- COMMAND_CLASS_PROTECTION

Im Kern wurden die Importier- und Kompatibilitätsfunktionen aus IP-Symcon 1.x entfernt, so dass viele Funktionen schneller und effizienter arbeiten können. Desweiteren wurde eine neue Funktion (IPS_SetIdent) geschaffen, mit der Variablen eindeutig per Name identifiziert werden können. Dadurch können Skripte portabel erstellt und mit Nutzern besser ausgetauscht werden. Ein Beispiel dafür ist das Skript zum Auslesen von Miele@Home Geräten (Link).
Alle Funktionen sind in der Basic, Professional und Unlimited Edition von IP-Symcon verfügbar und können mit einer gültigen Subskription bezogen werden. Ein spezielles “Upgrade” muss nicht erworben werden. Sofern Ihre Subskription abgelaufen ist, können Sie diese einfach erneuern und direkt ohne zusätzliche Mehrkosten auf die 2.5 updaten. (Shop)

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.4 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 2.5 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Die "auto_prepend_file"-Funktion in der "php.ini" wird von IP-Symcon intern genutzt. Falls eine PHP-Datei automatisch geladen werden soll, so muss diese im Skript Ordner liegen und "__autoload.php" heißen.

Die Extension "php_http.dll" ist in PHP 5.4 nicht mehr verfügbar. Falls diese genutzt wurde bitte auf die cURL-Extension ausweichen.
Durch eine Inkompatibilität von PHP 5.4 und OpenSSL 1.0 können über die "file_get_contents" (u.ä.) Funktion keine Anfragen per SSL gestellt werden. Die IP-Symcon interne Funktion Sys_GetURLContent bietet dort passenden Ersatz.
Einige spezielle Funktionen zum Handling von StatusVariablen werden zur 3.0 entfernt werden. Sobald Sie auf 2.5 aktualisiert haben, können Sie dieses Skript zur Überprüfung nutzen: IP-Symcon 2.5 – Hilfsskripte
Falls Skripte im "webfront/user"-Ordner erstellt und genutzt wurden, müssen in PHP 5.4 alle Parameter per "$_GET[...]" abgefragt werden.
Kompatibilitätsfunktionen aus alten IP-Symcon 1.0 Zeiten und die Kompatibilität mit Namen statt ID bei alten Funktionen wurde entfernt. Für alle die mit IP-Symcon 2.x angefangen haben, hat dies keine Bedeutung.

LISTE DER ÄNDERUNGEN

Neu: PHP 5.4
Neu: PHP Extensions werden per LiveUpdate **aktualisiert** (nur offizielle von PHP. Kein PECL!).
Neu: EnOcean Module können mit dem neuen ESP3 Protokoll umgehen.
Neu: “No configuration” Fehlermeldung verbessert
Neu: Modbus TCP noch robuster gemacht
Neu: xComfort Thermostat
Neu: LCN Shutter Modul
Neu: Neuer Lichtsensor (LCN-LSA) wird unterstützt.
Neu: Typ kann für T, R1, R2 gewählt werden.
Neu: BiDi Auswertung für Eltako Schalter/Dimmer
Neu: Eltako Dimmer kann direkt angesteuert werden. (Muss erneut angelernt werden!)
Neu: xComfort Thermostat kann Variable als Ist-Temperatur verwenden.
Neu: Ports für Wired/RF/Eventkanal können beim HomeMatic Socket eingestellt werden.
Neu: OneWire DS2406
Neu: Z-Wave Rückkanal bei Fibaro Geräten (Über BASIC Kommando Klasse)
Neu: LAN Version vom USBT-868 wird komplett unterstützt.
Neu: $IPS_ *ist nun per $_IPS[]* verfügbar-Neu: RegisterGlobals ist off. Das bedeutet, dass alle Variablen vom WebServer nur noch per $_GET abrufbar sind. (PHP5.4 erfordert dies!)
Neu: OpenSSL 1.0 wird genutzt. Löst Probleme mit SSL-Zugriff von Mobilgeräten
Neu: Importer für V1 Settings wurden entfernt.
Neu: IPS_SetValue*/IPS_GetValue*, IPS_SetScriptTimer, IPS_GetScriptTimer, IPS_RunScript* akzeptieren keine Strings mehr (V1 Kompatibilität).
Neu: IPS_GetVariableID, IPS_GetScriptID entfernt (V1 Kompatibilität). IPS_GetVariableIDByName/IPS_GetScriptIDByName als Ersatz nutzen
Neu: IPS_StatusVariableExists, IPS_GetStatusVariable, IPS_GetStatusVariableID, entfernt. Ersatz: IPS_GetObjectIDByIdent
Neu: IPS_GetStatusVariableIdents entfernt. Ersatz: IPS_GetChildrenIDs mit IPS_GetObject (ObjectIdent)
Neu: IPS_SetStatusVariableUseAction entfernt. Ersatz: Noch keiner.
Neu: IPS_SetIdent. Erlaubt einen internen, auf Kategorieebene eindeutigen, Namen zu vergeben, der dann wieder über IPS_GetObjectIDByIdent abgefragt werden kann. Dieses Feld kann für universelle Skripte hervorragend genutzt werden, um in Skripten auf Variablen per Name/Identifikator zuzugreifen und dem User trotzdem Änderungen an den Namen zu erlauben.
Neu: Links können über das Kontextmenü schneller erstellt werden. Objekt verlinken anklicken. Ziel auswählen. Fertig. (Die ursprüngliche Idee mit den Standard-Windows-Shortscuts erwies sich als viel zu kompliziert in der Verwendung.)

Neu: Sortieren nach Werten in der Listenansicht möglich
Neu: Unterstützung für neue Z-Wave Klassen
  COMMAND_CLASS_CLOCK
  COMMAND_CLASS_PROTECTION
Fix: W&T: Teilweise waren Kanäle nicht auslesbar.
Fix: IPS_Semaphore* Fix. Danke an tgusi74 fürs finden!
Fix: HomeMatic Wired Komponenten wurden nicht korrekt automatisch erstellt.
Fix: ALL3690: Auslesen der Summen wartet nun max. 5min auf die Berechnung
Fix: Konsole konnte per Remote nicht aktualisiert werden, wenn auf dem Server (Win7) die Konsole gestartet war.
Fix: Strings wurden in AC_GetLoggedValues nicht zurückgegeben.
Fix: Werte bei % Profilen wurden bei “MinValue < 0” falsch berechnet
Fix: LCN Modul Konfigurator zeigte TVar/RVars Instanzen nicht korrekt an
Fix: Profil beim KS300 korrigiert
Fix: Installation unter Windows 7 (x86) ist nun im Programme Ordner möglich
Fix: GIF Bilder sollten Dashboard nicht mehr zum Absturz bringen.
Fix: AssertionViolation bei TeamViewer 6 QuickConnect behoben
Fix: FTDI Instanz wurde als fehlerhaft markiert, wenn diese deaktiviert war, aber man schalten wollte.
Fix: Werte bei Boolean Assoziationen können per PHP nur noch modifiziert werden.
Fix: IP-Symcon stürzte ab, wenn der media Ordner nicht existierte.
Fix: Sortieren nach Wert in Listenansicht funktionierte nicht.
Fix: Xcomfort Feuchtigkeit wurde im Sende AN/AUS Modus nicht empfangen.
Fix: DS2438 VAD/VDD wurden unter Umständen nicht korrekt gelesen.
Fix: GetValue versagte nach Aufruf externer Programme oder RRD-Execute.
Fix: LCN Limit auf 5 Telegramme/Sekunde (PCHK 2.1 erfordert dies)
Fix: DS18S20/DS18B20 – Startwert 85 wird nicht mehr gelesen.
Fix: LiveUpdate überprüft, ob CRT Bibliotheken installiert sind.

V2.3->V2.4 (Q2/2011)

WESENTLICHE ÄNDERUNGEN

Der Fokus von IP-Symcon 2.4 liegt auf unserer bekannten und einzigartigen Visualisierung – dem WebFront. Die neuste Version hat ein neues, frisches Design und wurde "unter der Haube" nahezu komplett neu entwickelt. IP-Symcon verwendet die neusten Web-Technolgien und ist für die kommenden IP-Symcon-Versionen gerüstet. Wer es noch nicht live gesehen hat, kann jederzeit auf www.webfront.info gehen und sich selbst ein Bild von dem Erscheinungsbild und den neuen Funktionen machen. Einzelne Highlights werden nachfolgend dargestellt.

Mehrere WebFront-Konfigurationen*

Wer kennst das Problem nicht? Es soll ein eigenes Panel für das Wohnzimmer, für den Service-Techniker oder für das Kind, welches nicht an allen Knöpfen und Reglern drehen soll, geben. Mit diesem Feature bietet IP-Symcon demnächst die volle Kontrolle für verschiedene Bereiche oder Gruppen. Die Auswahl der Konfiguration geschieht, sofern es mehrere gibt, über einen Login-Dialog. Alternativ kann für festgelegte IP-Adressen eine bestimme Konfiguration automatisch geladen werden. Ohne komplizierte Passwort-Eingabe. Die oft nachgefragte Funktion, im sicheren Heimnetzwerk keine Passwörter eingeben zu müssen, gibt es per Mausklick dazu.

Farbige Zustände

Die schaltbaren Rot/Grün-Buttons haben in IP-Symcon 2.4 ausgedient! Stattdessen können für alle Assoziationen benutzerdefinierte Farben angeben werden. IP-Symcon bedient sich dabei standardmäßig bei Grün für alle Geräte, die aktuell Strom verbrauchen, Rot für akute Warnmeldungen und Blau für Geräte, die sich gerade in Aktion befinden. Neutrale Zustände werden gräulich und dadurch unauffällig dargestellt. Selbstverständlich kann auch diese Funktionsweise völlig umgestaltet werden über die altbekannten Variablenprofile.

Wecker-Funktion

Alle verfügbaren zyklischen Ereignisse können nun direkt über das WebFront konfiguriert werden. Sei es der Wecker, die Gartenbewässerung oder die zeitabhängige Nachtabsenkung. In IP-Symcon 2.4 kann dies direkt über das Touch-Panel ohne Mehraufwand eingestellt werden.

Automatische Aktualisierung aller Objekte im WebFront

WFC_Reload? Nicht mehr notwendig. Alle Objekte im Objektbaum werden vollständig im WebFront aktualisiert. Sei es der Name, der Zustand, das Profil, das Icon oder einfach nur die Sichtbarkeit. Neue Objekte werden selbstverständlich auch hinzugefügt und gelöschte entsprechend entfernt.

Volle Konfigurationsvielfalt für Unterebenen**

Dank der neuen TabPanes (Menüleisten mit Reitern und Widgets) lassen sich bequem neue Unterebenen erstellen. In Kombination mit den SplitPanes (ehemals Container) können so eigene Menüleisten (TabPanes) links und rechts erstellt werden, die jeweils auf neue Kategorien, Inhaltwechsler oder Graphen verweisen können. Ein Beispiel dafür ist unsere dreistufige Gebäude-Navigation im SmartHome Paderborn.

Balkendiagramme

Für die Darstellung der Zähler-Graphen werden wir in IP-Symcon 2.4 eine Balkenansicht, statt der bekannten Kurven, verwenden, um den Verbrauch besser verständlich darzustellen.

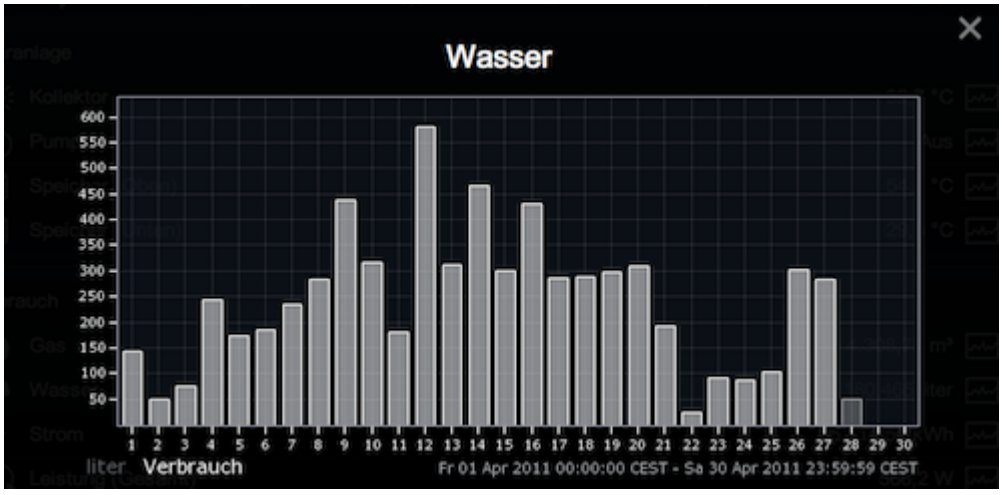
* Diese Funktion ist nur für IP-Symcon Professional und IP-Symcon Unlimited Kunden verfügbar. In der IP-Symcon Professional Version können maximal fünf (5) WebFront Konfiguratoren erstellt werden. IP-Symcon Unlimited beinhaltet, wie der Name schon sagt, keine Limitierung.

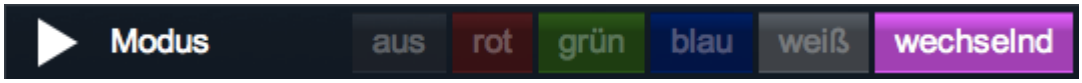
** Diese Funktion ist in IP-Symcon Basic nicht verfügbar.

SCREENSHOTS











MIGRATION

Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.3 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 2.4 auftreten.

Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Der WebFront-Ordner wird automatisch von unbekannten und veralteten Dateien gereinigt. Alle eigenen Dateien, die nicht gelöscht werden sollen, müssen in den user-Unterordner verschoben werden!
Variablen ohne Profil werden angezeigt. Um alle Variablen ohne Profil auszublenden, kann das folgende Skript genutzt werden. (Link)
Das iFront ist auf Version 1.1 zu aktualisieren, um mit IP-Symcon 2.4 kompatibel zu sein.
Die Module RRDTool (Link) und XBee Device (bereit 2009 abgekündigt) wurden entfernt.

LISTE DER ÄNDERUNGEN

Neu: iFront 1.1 wird unterstützt
Neu: Diverse neue Icons
Neu: Module (z.B. Chromoflex) aus IP-Symcon 2.3 können weiterverwendet werden.
Neu: Link auf Ereignis wird erlaubt.
Neu: Sortierung im WebFront Konfigurator / Archiv Modul
Neu: Auswerten von empfangenen Intensity-Werten bei FS20
Neu: Letzter Start von Ereignissen wird gespeichert.
Neu: Entpackvorgang beim LiveUpdate wird angezeigt.
Neu: Dynamische Skalierung bei Graphen jetzt zuschaltbar
Neu: XBee Device Modul entfernt (XBee Splitter als Ersatz verwenden)
Neu: WebFront-Konfigurator mit direkter Konfiguration für Retro, Retro Mobile und iFront
Neu: IPS/ProJet WDT-868 Modul
Neu: ~HM.Motion / ~Presence mit Icon
Neu: Profilliste wird sortiert.
Neu: .Background wird nicht mehr unterstützt.
Neu: Ausklappbare Boxen wurden endgültig entfernt.
Neu: . (Punkt) Notation im WebFront wird nicht mehr unterstützt.
Neu: retro.php/retromobile.php/ifront.php in index.php zusammengeführt
Neu: Parameter für dashboard.php überarbeitet (dynamicScaling=true/false)

Neu: IPS_SetVariableProfileAssociation wurde geändert (neuer Color Parameter).
Neu: Profilassoziationsanzahl auf maximal 32 beschränkt
Neu: KNX: Auslesen von Adressen vom Bus. (EIB_RequestStatus)
Neu: KNX: KNX Geräte können Leseanforderungen an IP-Symcon schicken.
Neu: KNX: FT1.2 kann über TCP/IP getunnelt werden.
Neu: KNX: Spezialmodul für Rollläden/Jalousien
Neu: Eltako FAFH60 (Temp-/Feuchtigkeitssensor)
Neu: Eltako FAH60 (Helligkeitssensor)
Neu: Spezial Eltako Modul für Schalter/Dimmer die Invertiert sind
Neu: ALL3418 wird unterstützt.
Neu: ALL3421 wird unterstützt.
Neu: ALL3690 funktioniert korrekt mit der neuen Firmware 2.0 (Diese wird auch erfordert!).
Neu: Z-Wave: Neue Kommandoklassen werden unterstützt.
  COMMAND_CLASS_METER
  COMMAND_CLASS_ALARM
  COMMAND_CLASS_LOCK
  COMMAND_CLASS_THERMOSTAT
Fix: “Jetzt aktualisieren” sperren, wenn Konfiguration geändert wurde (ImageGrabber/WWWReader)
Fix: DWD FTP-URL angepasst (outgoing2)
Fix: ThermoJet wird korrekt bei RF-Übertragung ausgewertet.
Fix: Fehlermeldung, wenn Name verändert wurde (ALL3690)
Fix: %f in Assoziationen wurde falsch ausgewertet
Fix: Assoziationen wurden nicht überall korrekt nach dem Muster Präfix %2B Name %2B Suffix dargestellt.
Fix: KNX Zeitüberschreitungen
Fix: OneWire DS2450, Timing-Problem beim Auslesen behoben
Fix: Flackern des Volume-Sliders im WebFront beim MediaPlayer Modul
Fix: Sys_GetMemoryInfo() Limit von 4GB behoben

V2.2->V2.3 (Q3/2010)

WESENTLICHE ÄNDERUNGEN

HomeMatic CCU ohne TCPDump Erweiterung
Funktioniert auch mit dem LAN Adapter, wenn der BidCos Service installiert ist.
Container-Modul für das WebFront (Horizontale/Vertikale Aufteilung)
WebFront Container
Inhaltswechsler für das WebFront
Erlaubt die Anzeige von Media Dateien, TextBoxen, HTMLBoxen, die zeitlich umgeschaltet werden können in einem Reiter. (Besonders geeignet für WebCams, die durchgeschaltet werden wollen.) Alle Objekte müssen in einer Kategorie liegen und auf die muss dann die BaseID zeigen.
KNX Konfigurator
Bietet das einfache Importieren von Gruppenadressen aus dem ETS Export (.esf). Über einen Klick werden automatisch alle Gruppenadressen importiert.*
iPhone/iPad Verbesserungen für das WebFront
Splashscreen, Icons fürs iPad, …

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.2 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 2.3 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Spezielle Änderungen sind nicht erforderlich.
Falls Sie im WebFront die DWD Seite nutzen, müssen Sie einen Benutzernamen/Kennwort vom DWD anfordern und im Konfigurator angeben.

LISTE DER ÄNDERUNGEN

Neu: Container-Modul für das WebFront (Horizontale/Vertikale Aufteilung)
Neu: MediaChanger für das WebFront
Neu: HomeMatic ohne TCPDump
Neu: FS20: Weitere Adresse können angegeben werden. (Gruppen-/Masteradressen)
Neu: ProJet Input Modul
Neu: MediaPlayer Funktion (WAC_SetPosition)
Fix: WebFront DWD (Es muss ein Benutzername/Passwort beim DWD angefordert werden! Dokumentation)
Fix: Mail Modul gibt als Zeitstempel 0 aus, wenn keine Mails vorhanden

Fix: Modbus TCP kann nun auch die DeviceID nutzen.
Fix: Diverse WebFront/iFront fixes
Fix: Assoziationen mit %f funktionieren jetzt korrekt.
Fix: Thermokon: Berechnen der Temperatur war unter Umständen fehlerhaft.
Fix: MediaPlayer: Soundkarten wurden falsch zugeordnet, wenn z.B. ein USB Soundkarte entfernt wurde.
Für Entwickler:
Neu: HTTP Post für PHP/WebServer ($_POST, $_FILES)
Neu: Enumerations in der Rückgabe von IPS_GetFunctionList

V2.1->V2.2 (Q1/2010)

WESENTLICHE ÄNDERUNGEN

In der Version 2.2 von IP-Symcon wurde besonders an der grafischen Darstellung von Daten gearbeitet. So erlaubt das WebFront die Darstellung der in der SQLite-Datenbank aggregierten Daten in einer Stunden-, Tages-, Wochen-, Monats- und Jahresansicht. Eine Live-Demo kann unter webfront.info angesehen werden.
Weitere neue Funktionen sind native Module zum Empfangen von E-Mails über POP3 und IMAP, sowie das Versenden von E-Mails (SMTP) und SMS (Clickatel).
Seit der Version 2.2 ist es möglich, Meldungen (Notifications) an das WebFront zu schicken, um sich über Ereignisse im System, direkt auf der Visualisierung informieren zu lassen. Ein eingebautes On-Screen-Display ermöglicht im WebFront die direkte Eingabe von Textdaten in einer Touchscreen-kompatiblen Art und Weise.
Für eine komplette Auflistung aller neuer und geänderter Funktionen lesen Sie bitte die Liste der Änderungen am Ende des Dokuments.

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.1 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 2.2 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Bedingt durch den Wechsel von PHP 5.1.6 auf PHP 5.3.1 müssen Sie die ggf. genutzte Extensions aktualisieren. Aktuelle Extension DLLs finden sie in diesem Paket: PHP 5.3.1 Extensions
Um Graphen im WebFront sehen zu können, muss eine 'Graphen Seite' im WebFront Konfigurator erstellt werden.

LISTE DER ÄNDERUNGEN

Neu: Graphen im WebFront (inkl. Archiv Modul auf SQLite Basis)
Neu: Popups/Notifications im WebFront
Neu: Graphen im Dashboard über WebBrowser (http://localhost:82/dashboard.php)
Neu: On-Screen-Keyboard für das Editieren von Strings/TextBoxen
Neu: Schnellerer Rückkanal
Neu: E-Mail versenden (SMTP)
Neu: E-Mail empfangen (POP3&IMAP)
Neu: Image Grabber (Lädt ein Bild direkt in den Medienpool)
Neu: SMS Modul für Clickatel
Neu: M-Bus Modul

Neu: Heating Control Modul
Neu: Ereignisse direkt in der Instanzkonfiguration erstellbar
Neu: Geräte können direkt über Ereignisse angesprochen werden.
Neu: Module für die EKM-868 (Counter) und ViewJet (Display)
Neu: WebFront nativ auf Englisch %2B Deutsche Übersetzung
Neu: ProJet Display (Input Modul)
Neu: Statusveränderungen von I/O's abfangen (Kern Instanzen->Event Handler)
Neu: FS20 Adresse kann auf doppelte Verwendung geprüft werden
Neu: Filter HTML Checkbox bei Skriptausgabe
Neu: Sys_GetURLContentEx -> Erlaubt Zugriff mit BasicAuthentification/Proxy
Neu: Icons können über ein Auswahlfenster ausgewählt werden.
Neu: Locked Eigenschaft für den TIPSLockButton
Neu: AutoLocked Eigenschaft für den TIPSLockButton
Fix: Dimmer im WebFront bei HomeMatic nicht bedienbar
Fix: 32Bit/Real Werte bei ModBus
Fix: SetTargetValue bei LCN
Fix: n wird bei der TextBox im WebFront erkannt
Fix: Werte/Reaktionsbug im Dashboard
Fix: FHT/HomeMatic FHT im WebFront (haken für "Emulation" aktivieren!)
Fix: Leveljet wieder auslesbar
Fix: LCN_StartFlicker fix
Fix: EnOcean kann mit 32bit unsigned IDs umgehen.
Fix: 1-Wire negative Temperaturen bei manuellem polling falsch
Fix: FHT: Negative Ventilpositionen

V2.0->V2.1 (Q3/2009)

WESENTLICHE ÄNDERUNGEN

Die neue Version 2.1 von IP-Symcon fügt viele neue Funktion, aber auch Detailverbesserungen, ein, so dass ein Update auf diese Version empfohlen wird. Besonders im Bereich der Visualisierung ist seit der Version 2.0 viel verbessert worden. Aber auch am Kern von IP-Symcon wurde viel Zeit in eine bessere Stabilität und Speichernutzung investiert. Einige besondere Funktionen werden hier vorgestellt:

Variablenprofile, die eine Variable für die Visualisierung beschreiben, wodurch Sie ohne weiteren Aufwand in der Visualisierung (WebFront) angezeigt werden können. Weiterhin ist es möglich Aktionen zu definieren, um virtuelle Geräte zusammenzustellen. Die Dokumentation der Möglichkeiten finden Sie hier: Variablenprofile

Symbolische Links vereinfachen eine mehrfache Verwendung von Objekten in der Visualisierung. Sie müssen Ihr Panikschalter-Skript nur noch einmal anlegen und können es in beliebig vielen Kategorien verknüpfen.

Dadurch sind auch Übersichtsseiten möglich, um z.B. alle Stati der Fenster anzuzeigen.

Die WebFront Konfiguration erspart ihnen eine mühselige Anpassung irgendwelcher Konfigurationsdateien. Fügen Sie der Visualisierung (WebFront) Ihre eigenen Seiten als Reiter hinzu, verändern Sie den Ausgangspunkt Ihrer Visualisierung oder fügen Sie eigene Widgets hinzu, die Ihnen z.B. das Datum der nächsten Müllentsorgung anzeigen. Alles ein paar Mausklicks entfernt. WebFront konfigurieren

Neue Module für:

- den Energiezähler EM24 von Carlo Gavazzi
- generische Module für ModBus TCP und ModBus RTU
- viele neue Funktionen für das Z-Wave System

Als Vorbereitung für die in der Version 2.2 kommende graphische Darstellung von erfassten Daten können Sie bereits mit dieser Version Daten mitloggen und in einer SQLite-Datenbank abspeichern. Sobald die Version 2.2 verfügbar ist, können Sie ihre Daten sofort graphisch auswerten.

Für eine komplette Auflistung aller neuen und geänderten Funktionen lesen Sie bitte die Liste der Änderungen am Ende des Dokuments.

MIGRATION



Bitte erstellen Sie auf jeden Fall ein Backup Ihres gesamten IP-Symcon Ordners, um auf die Version 2.0 zurück zu wechseln, sollten wider erwarten Probleme mit der Version 2.1 auftreten.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Löschen Sie **vor** der Installation alle Module (*.dll) aus dem 'modules' Ordner von IP-Symcon, da diese nicht mehr gebraucht werden, oder falls Sie welche von Drittanbietern haben, diese

einfach nicht mehr kompatibel sind.
Falls Sie PHP Extensions nutzen, müssen Sie diese ab der Version 2.1 in den ‘ext’ Ordner von IP-Symcon kopieren. Alle Extensions in diesem Ordner werden automatisch von IP-Symcon geladen. Die "php.ini" muss dazu nicht mehr angepasst werden, sondern wird bei jedem IP-Symcon Start automatisch neu aktualisiert. Ein anderer Ort ist für Extensions nicht mehr möglich.
__Das Error Reporting ist seit der Version 2.1 standardmäßig auf "E_ALL" gesetzt. Das bedeutet, dass Fehler, wie z.B. nicht definierte Variablen, direkt als Fehler angezeigt werden. Außerdem müssen Arrayindexe als String angesprochen werden. z.B. $a['MeinIndex'] statt $a[meinIndex]. (Auslöser dieser Fehlermeldungen sind meist unsauber programmierte Skripte.) Sie können das Error Reporting in der "php.ini" verändern, wovon wir aber abraten.__
Fehler können Sie, wie immer, in der Projektverwaltung im IP-Symcon Forum melden.

LISTE DER ÄNDERUNGEN

Kernel
Keine Veränderung der PHP Version (5.1.6 wird benutzt)*
php.ini wird nur noch aus dem IP-Symcon Order geladen
Alle passenden Extensions werden automatisch aus dem ‘ext’ Ordner geladen.
Alle Module sind direkt in die ips.exe integriert, wodurch der Speicherbedarf sinkt.
Eigene Variablen Profile (Dokumentation)
Objekte können im WebFront versteckt werden ohne dem Punkt (.) Workaround (Objekt bearbeiten -> Optionen).
Ereignisse können direkt an Variablen/Instanzen erstellt werden.
Objekte können sortiert/positioniert werden.
Symbolische Links können für Instanzen, Variablen, Skripte und Medien erstellt werden.
Neue Funktion GetValueFormatted, die den Variablenwert als formatierten String zurück gibt. Z.B. 17,3°C
Suche/Ersetze in Dateien implementiert
Präfixe haben jetzt Leerzeichen, die entsprechend der DIN 5008 erstellt wurden.
**Ein Wechsel auf 5.2.x oder 5.3.x war geplant, aber aufgrund von Problemen wieder verworfen worden. Fixes für einige relevante PHP-Bugs wurden für PHP 5.3.1 angekündigt, welches für Version 2.2/2.3 erneut getestet wird.*
WebFront
file_get_contents gegen eine Funktion ausgetauscht (Sys_GetURLContent)
Wetterseiten werden gecacht, für schnellere Ladezeiten/Ausfallsicherheit
WebFront mit eigenen Profilen kompatibel
WebFront zeigt alle Variablen mit Variablenprofil, Skripte und Medien direkt unterhalb von Instanzen an.
Anzahl der Elemente in einem ‘enum’ / ‘selector’ kann in der Config definiert werden.
Slider für Prozentwerte (mit Möglichkeit zum Deaktivieren)
Media-Dateien können ausgeklappt werden.
Media-Dateien werden bei Aktualisierung automatisch im WebFront aktualisiert.
WebFront Retro, iFront den Neuerungen entsprechend angepasst
$REMOTE_ADDR, $REMOTE_HOST bei Ausführen einer Aktion verfügbar
Module
Dummy Modul als Platzhalter für eigene Variablen, die im WebFront angezeigt werden können
Wago: Native Implementation (Wago DLL wird nicht mehr benötigt)
ModBus: Modbus TCP, Modbus RTU, Modbus RTU over TCP wird unterstützt.
Carlo Gavazzi: EM24 Modul hinzugefügt

Virtual I/O Modul zum Testen eigener Device Module (nur für Entwickler interessant!)
Webserver: OpenSSL Update auf 0.9.8k
HomeMatic: Standardprofile für die Wetterstation
Z-Wave
- Weitere Geräteklassen werden unterstützt
- Assoziationen können konfiguriert werden.
- Parameter können an Geräten gesetzt werden.
- Routing Tabelle wird ausgelesen.

Archiv Modul auf SQLite Basis zum Loggen von Variablenwerten

Dashboard

Wenn der Parameter/cache angegeben wird, dann werden alle Bilder in einem cache-Ordner gespeichert, wodurch das erneute Laden wesentlich beschleunigt wird.
Neues Control: TIPSSpinEdit für Integer/Float Werte
Dashboard kann Variablenwerte mit Profilformatierung anzeigen.
$REMOTE_ADDR, $REMOTE_HOST, $IPS_FORM bei Ausführen eines Skriptes verfügbar

Geänderte Funktionen

IPS_SetEventScript (Nimmt nun den auszuführenden Text statt der ID des Skriptes.)
IPS_GetEvent (Index: TriggerVariable -> TriggerVariableID)
IPS_GetStatusVariableIDs -> IPS_GetStatusVariableIdents

Änderungen im SDK

KERNEL_VERSION = 0x020A
Konstanten sind jetzt in 100er Schritten definiert.
Kategoriefunktionen vom ObjectManager in den CategoryManager verschoben
LinkManager für die symbolischen Links
Variablenprofile werden über Strings verknüpft und müssen vorher im VariableManager registriert werden.

Mehr Informationen und das neue SDK finde sie im Entwicklerbereich

V1.0->V2.0 (Q1/2009)

FEATUREÄNDERUNGEN

Generelle Änderungen
- Server/Client Design, Windows Dienst + Netzwerkfähige Clientapplikation
- Alle Objekte (Instanzen, Variablen, Skripte, Events, Medien) haben eine eindeutige ID.
- Designer (jetzt Dashboard) direkt in die Clientapplikation eingebettet

Serverseitige Änderungen
- Mehr PHP Befehle zum Eingreifen/Konfigurieren von IP-Symcon
- Pro Skript können mehrere Timer (jetzt: Ereignisse) erstellt werden.
- Ereignisse können zu bestimmten Zeitpunkten gestartet werden (Timer Wizard Ersatz!, Kein Importer!).
- SOAP Schnittstelle (Entwicklerbereich)
- Bei Fehlern (IPS Funktionen) werden diese direkt in PHP Fehlermeldungen ausgegeben.

Clientseitige Änderungen
- Clientanwendung funktioniert über das Netzwerk/Internet
- Grafische Instanzenansicht durch Baumansicht ersetzt
- Multilingual (Deutsch/Englisch)
- Wizards zum Einrichten der Objekte
- Mehr Drag&Drop Möglichkeiten
- Docking der einzelnen Fenster
- Quickfilter an den meisten Stellen vorhanden

WebFront
- WebInterface, welches automatisch vorhandene Geräte visualisiert/schalten lässt
- Eine Variante im iPhone / iTouch Design erlaubt schnellen Zugriff

Modulspezifische Änderungen
- Native LCN Module
- HomeMatic Module
- ProJet/LevelJet Module
- SiLabs USBXpress I/O Modul
- Siemens S7 SPS Module
- FS20TX/RX zusammengelegt; Das FS20 Modul kann nun senden/empfangen und die Timer emulieren.
- Aus FS20RX wird FS20EX. Es erlaubt komfortabel eine Mehrkanalfernbedienung einzurichten.
- Alle WAGO Adressen werden als einzelne Instanzen erstellt. Ermöglicht vernünftige Namensvergabe und Anzeige im WebInterface
- I/O Module reconnecten sich automatisch solange, bis Sie eine Verbindunge bekommen.
- WebServer bietet SSL-Verschlüsselung
- MediaPlayer Modul als Ersatz für das WinampControl. Unterstützt mehrere Soundkarten mit direkter Soundkartenauswahl.
- ISDN Modul kann nun mehrere Anrufe gleichzeitig bedienen. Beachten Sie bitte die neue Funktionsweise. (Siehe ISDN Beispiel)
- Das FS20MS Modul ist jetzt nicht mehr an FS20 gebunden, sondern kann für fast alle Systeme verwendet werden.
- Link: Shutter Control

Wegefallende Module
- DirectX MediaPlayer (eignes MediaPlayer Modul als Ersatz)

COM/LAN (SerialPort/ClientSocket als Ersatz)
memPIO
M34 USB
RedLab
BTKit
IPPower

Weggefallende/Geänderte Funktionen (nur die wichtigsten Funktionen angegben)
TWZ_ItemExists (Ersatz: Ereignissfunktionen)
TWZ_DeleteItem (Ersatz: Ereignissfunktionen)
TWZ_CreateItem (Ersatz: Ereignissfunktionen)
IPS_UnregisterScript (Ersatz: Skriptfunktionen)
IPS_RegisterScript (Ersatz: Skriptfunktionen)
IPS_ScriptExists (Ersatz: Skriptfunktionen)
IPS_CreateVariable (Ersatz: Variablenfunktionen)
IPS_VariableExists (Ersatz: Variablenfunktionen)
IPS_GetVariableType (Ersatz: s.u.)
IPS_GetVariables (Ersatz: s.u.)
LOC_GetLocation_Variable (Ersatz: s.u.)
IPS_GetInstanceIDs (Ersatz: s.u.)
IPS_GetModuleInstance (Ersatz: s.u.)
SYS_Ping (2. Parameter entfällt)
IMGP_SendEvent (Ersatz: IPS_SendMediaEvent)
SNDP_SendEvent (Ersatz: IPS_SendMediaEvent)
TTS_Speak (Syntaxänderung + Hinweise: TTS_Speak)

MIGRATION

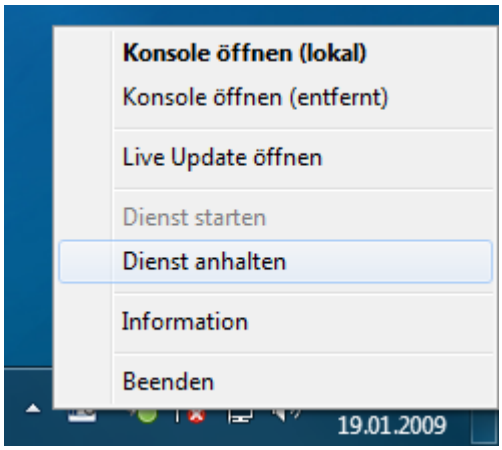


Führen Sie bitte ein Backup von Ihrem alten IP-Symcon durch.



Ein direktes Update ist immer möglich. Lesen Sie bitte vorher die passenden Migrationsanleitungen.

Installieren Sie bitte IP-Symcon 2.0 in einen **neuen** Ordner.
Stellen Sie sicher, dass der IP-Symcon Dienst beendet ist. Dies können Sie am grauen TrayIcon erkennen. Einen gestarteten Dienst können Sie durch die Funktion „Dienst anhalten“ beenden.



Sobald der Dienst gestoppt ist, müssen folgende Dateien aus Ihrem alten IP-Symcon Ordner in den neuen IP-Symcon Order kopieren.

scripts*
web* (für WIIPS)
web_data* (für WIIPS)
settings.xml
bricks*
php.ini (Pfade anpassen!)
Alle PHP Extension DLLs, wenn Sie welche installiert haben (z.B. GD2)

Nun können Sie IP-Symcon starten. Alle Variablen, Skripte, Instanzen sollten mit allen Einstellungen importiert werden. Zum Abschluss der Migration sollten Sie Ihre Skripte auf in der neuen Version von IP-Symcon nicht mehr vorhandene Funktionen überprüfen. Ferner müssen Sie alle Zugriffe über Namen durch Zugriffe über IDs ersetzen. Solange Sie aber keine Geräte umbenennen, können Sie diese Aktion auch bei Bedarf durchführen.

KOMPATIBILITÄTSFUNKTIONEN

Diese Funktionen sollten nur übergangsweise genutzt werden und so schnell wie möglich gegen die neuen, ID basierten Funktionen ausgetauscht werden, um die neuen Möglichkeiten ausnutzen zu können.

```
function IPS_GetVariableType( $varid )
{
if(!is_numeric($varid))
$varid = IPS_GetVariableID($varid);
$var = IPS_GetVariable($varid);
switch($var['VariableValue']['ValueType'])
{
case 0:
return "Boolean";
break;
case 1:
return "Integer";
break;
case 2:
return "Float";
break;
```

```
case 3:
return "String";
break;
default:
return "Unknown";
break;
}
}

function IPS_GetUpdateTime($id)
{
$v = IPS_GetVariable($id);
return $v['VariableUpdated'];
}

function FS20_SetPosition($id, $pos)
{
SC_Move($id, $pos);
}

function IPS_GetVariables()
{
$result = Array();

$varids = IPS_GetVariableList();
foreach($varids as $varid)
{
$var = IPS_GetVariable($varid);
$obj = IPS_GetObject($varid);
$result[] = Array(
"name" => $obj['ObjectName'],
"type" => IPS_GetVariableType($varid),
"value" => GetValue($varid),
"updated" => (int)$var['VariableUpdated']
); }

return $result;
}

function LOC_GetLocation_Variable($varid)
{
if(!is_numeric($varid))
$varid = IPS_GetVariableID($varid);

return 'IP-SYMCON\\'.IPS_GetLocation($varid);
}

function IPS_GetInstanceIDs()
{
return IPS_GetInstanceList();
}

function IPS_GetModuleInstance($instanceid)
{
$instance = IPS_GetInstance($instanceid);
return Array(
"instanceid" => $instance['InstanceID'],
"instancestatus" => $instance['InstanceStatus'],
"moduleguid" => $instance['ModuleInfo']['ModuleID'],
```

```
"packetsreceived" => 0,
"packetstransmitted" => 0,
"packetsaccepted" => 0,
"packetsdropped" => 0,
"packetsbuffered" => 0
);
}
function IPS_GetModuleInfo ( $mguid ) {
$marray = IPS_GetModule( $mguid );
$m = Array();
$m['name'] = $marray['ModuleName'];

switch ($marray['ModuleType']) {
case 0:
$m['type'] = 'Core';
break;
case 1:
$m['type'] = 'I/O';
break;
case 2:
$m['type'] = 'Splitter';
break;
case 3:
$m['type'] = 'Device';
break;
}
$m['guid'] = $mguid;
$larray = IPS_GetLibrary($marray['LibraryID']);
$m['version'] = ($larray['Version'] >> 8) .".". ($larray['Version'] &
0xFF);
return $m;
}
```

V0.0->V1.0 (Q1/2005)

Erste Version von IP-Symcon wurde veröffentlicht.

Security

IP-Symcon includes diverse security features that set a password or a special service for different sections. Read this article carefully to prevent illegal external access.



The factory settings are only secure while IP-Symcon cannot be accesses externally. If IP-Symcon should be accessible externally some settings need to be adjusted accordingly!

REMOTE ACCESS (PASSWORD)

Factory Settings: *deactivated* (secure)
The management of IP-Symcon can be accessed remotely via the management console. Direct access from the outside of the local network is only possible when a password is set.
Further informations: Set Remote Access

CONNECT SERVICE

Factory Settings: *activated* (secure)
The Connect service can access the management console and the visualization comfortably from outside when a subscription is active.
The visualization may only be accessed when a password was previously set or the visualization was explicitly marked as public.
The management console can access the configuration only when a password is set for remote access .
Further informations: Connect Control

VISUALIZATION (PASSWORD)

Factory Settings: *deactivated* (secure)
The factory settings allow access to the visualization without password in internal networks. A set password secures the webbased visualization and the mobile apps. A password or explicit clearance is required if the visualization should be accessible from the Connect service or be accessible from the outside.
Further informations:Tile-Visualizations Security / WebFront-Visualizations Security



Requests from the following networs are considered internal (Private IP Adresses). For internal networks no visualization password is required, if the option "Require password only for external connections" is enabled. If connections are being routed through a NAT it is recommended to leave this options disabled.

USER FOLDER (USER NAME/PASSWORD)

Factory Settings: *deactivated* (insecure)
If the User folder contains individual content it is recommended the secure the content.
Usage of the User folder is strongly discouraged.
Further informations: User Folder Security

SYMOS (PASSWORD)

Factory Settings: *activated* (secure)
SymOS is the operating system of the SymBox. It is possible to protect the access to its web surface via password. The web surface can set the password. If the SymOS surface should be available externally setting a password is urgently recommended.
Further informations: SymBox Tutorial PDF - Chapter 4.4.6. Security

SYMOS CONNECT

Factory Settings: *activated* (secure)
Via SymOS Connect it is possible to connect to the SymOS operating system. The external connection is in general only possible if a SymOS password is set.



Some functions require internet access and a properly configured Firewall.

Backup & Restore

Two different kinds of backups are possible.
Backup: A backup that stores all relevant user data like media files, scripts, databases, etc.
settings.json: The configuration file is the core element in every installation of IP-Symcon. The file is saved automatically.

BACKUP

A backup will include all relevant user data. A complete backup is not done automatically. Creating and loading a backup is explained in Create Backup and Restore Backup.

AUTOMATICAL BACKUP OF THE CONFIGURATION FILE (SETTINGS.JSON)

The file "settings.json" in the IP-Symcon folder contains all relevant settings of IP-Symcon. The file is the core element of the configuration. A copy of the file is stored in the folder "backup" every day at 0:00 and can be accessed in case of an error. A maximum of 25 files are stored in the folder. When the limit is exceeded, the oldest file is deleted.
The backup of the file is automatically stored with the name "settings1234567890.json", where "1234567890" is replaced with the current Unix timestamp. Thus, the highest number corresponds to the most current backup.



The automatical backup of the configuration file does not replace a regular complete backup. See Create backup

RESTORE CONFIGURATION FILE (SETTINGS.JSON) IN CASE OF AN ERROR

Different circumstances can cause errors, e.g.:
- Unwanted modifications (e.g., wrong instances, deleted variables, ...)
- The settings are unexpectedly "lost", "deleted", or "empty". (In most cases, this is caused by an unexpected termination of the software, e.g., in case of a power outage)

To restore the configuration file the following steps are necessary:
- Terminate the IP-Symcon service
- Copy any "settings1234567890.json" from the folder "backup" into the main folder (e.g., the file with the highest number in the file name)
- Delete the current "settings.json"
- Rename the copied "settings1234567890.json" into "settings.json"
- Start the IP-Symcon service again

If the chosen backup of the configuration file does not work, try another backup. If no backup works, ask for help in the forum. When you ask for help, you need to attach **at least one logfile**. Posts without logfiles cannot be processed.

HOW OFTEN IS A BACKUP FILE WRITTEN?

Since version 4.0, IP-Symcon has various mechanisms to minimize the write cycles.
This is required if the IP-Symcon is operated on storage media that are showing signs of wear.
This is the case, for example, with a Raspberry Pi with an SD card.

MEDIA AND SETTINGS.JSON

In general, the settings.json is saved cyclically every 10 minutes (see special switch) and when the service starts/stops.
Media files (e.g. from Image Grabber) are cached and are only written at start/stop.

VARIABLE RAW DATA AND AGGREGATION

IP-Symcon writes variable raw data, which is saved in the archive, to the hard disk every minute. Since these are constantly growing datasets this is intended. In turn, the aggregation data is only written if an aggregation period is complete. The respective incomplete aggregation periods are restored at startup from the variable raw data. For this reason, starting with many archived variables also takes a little longer.

LOGFILES

Log files, which cause the majority of write cycles, are handled differently depending on the system. Normally these end up on the hard drive. However, the SymOS on the SymBox maps /var/log/symcon to a RAM drive, which means that the storage medium is not worn out unnecessarily and the longevity is extended.

Restore Backup

If your installation of IP-Symcon is in a non-repairable state, it is possible to load a backup. Backups can also be used to move from one platform to another.

CHANGE TO ANOTHER PLATFORM

The structure of the backup is independent of the system and thus can be used to move IP-Symcon from one platform to another.



When changing from IP-Symcon <=3.4 to IP-Symcon 4.0 or higher, the database needs to be converted into the new CSV format by using the Windows installer. Afterwards, the backup can be migrated to the new platform.

LOAD BACKUP

SymOS (SymBox)
Windows (Desktop/Server)
MacOS
Linux (Ubuntu)
Raspberry Pi
Docker/Synology/QNAP
Catan C1

SYMOS - LOAD BACKUP

The backup can be chosen on the web interface in "Settings -> Backup". Afterwards, IP-Symcon (NOT SymOS) needs to reboot. For further informations, see 4.4.4 Backup

WINDOWS - LOAD BACKUP

The setup automatically installs the required redistibutables. If IP-Symcon was copied into the system some required redistibutables may be missing. The required redistibutables are:
Visual Studio 2015/2017/2019 C++ Redistributable x64 (Direct Download)
Bonjour64 Installer (Direct Download)
The following steps need to be executed to load the backup:
Stop the IP-Symcon service. (Right click on tray icon -> "Stop Service")
Extract the backup.zip into the folder C:\ProgramData\Symcon
The easiest way to access the folder is via "Right click on the tray icon -> Information -> Server Data -> Open"
Restart the IP-Symcon service. (Right click on tray icon -> "Start Service")

MACOS - LOAD BACKUP

Stop the IP-Symcon service.
Move the backup.zip file into the folder "/Library/Application Support/Symcon" and extract via double click.
Delete the archive and restart the IP-Symcon service.

LINUX - LOAD BACKUP

Stop the IP-Symcon service:

```
sudo /etc/init.d/symcon stop
```

Extract the file backup.zip in the home folder into the folder "/var/lib/symcon/":

```
sudo unzip backup.zip -d /var/lib/symcon/
```

Restart the IP-Symcon service:

```
sudo /etc/init.d/symcon start
```

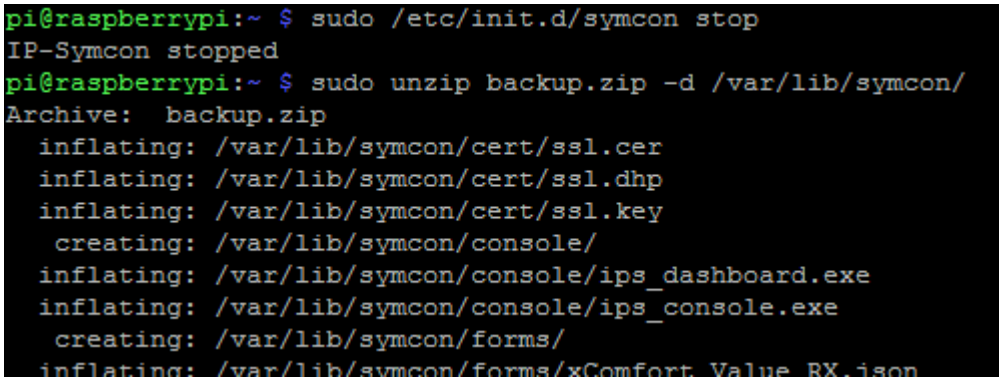
RASPBERRY PI - LOAD BACKUP

Stop the IP-Symcon service:

```
sudo /etc/init.d/symcon stop
```

Extract the file backup.zip in the home folder into the folder "/var/lib/symcon/":

```
sudo unzip backup.zip -d /var/lib/symcon/
```



Restart the IP-Symcon service:

```
sudo /etc/init.d/symcon start
```

DOCKER/SYNOLOGY/QNAP - LOAD BACKUP

First, the IP-Symcon container needs to be stopped. It can be stopped via the container management of the used system.
Afterwards, the mounted IP-Symcon folder which references the folder "/var/lib/symcon" needs to be opened.
The volume on which the folder was "mounted" can be seen in the settings of the image. According to the installation tutorial, the default is in the format "/Symcon/[MeinIPSymcon]/Data".
Extract the backup and copy it into the folder.
Restart the IP-Symcon container.

CATAN C1 - LOAD BACKUP

Stop the Symcon App. The app can be stopped in WBM under Settings -> App Management.
If a ZIP file from another system is available, extract it into a folder first.
Copy all files via WinSCP into the following folder:

```
/opt/plcnext/apps/home/data/60002171301055/data/
```

Make sure that settings.json is exactly in the data/ folder and not in a further subfolder.
Start the Symcon App.

HINTS

The following things should be checked and adjusted eventually:
The server needs to be reactivated for Push notifications
Absolute paths in scripts and media files
Occupied interfaces (e.g., serial ports) on the new system
IP adresses like the 'Event Server' for the HomeMatic CCU or 'WebServer'
Required additional programs, drivers, or services (e.g., *BidCOS*)

Create Backup

A backup stores all user files. It should be created within regular intervals.

CREATE BACKUP

SymOS (SymBox)
Windows (Desktop/Server)
MacOS
Linux (Ubuntu)
Raspberry Pi
Docker/Synology/QNAP
Catan C1

SYMOS - CREATE BACKUP

Start the web surface, enter the menu "Settings", and click "Backup". The folder structure is created automatically and backup.zip is downloaded. Further informations can be found within the manual of the SymBox under 4.4.4 Backup.

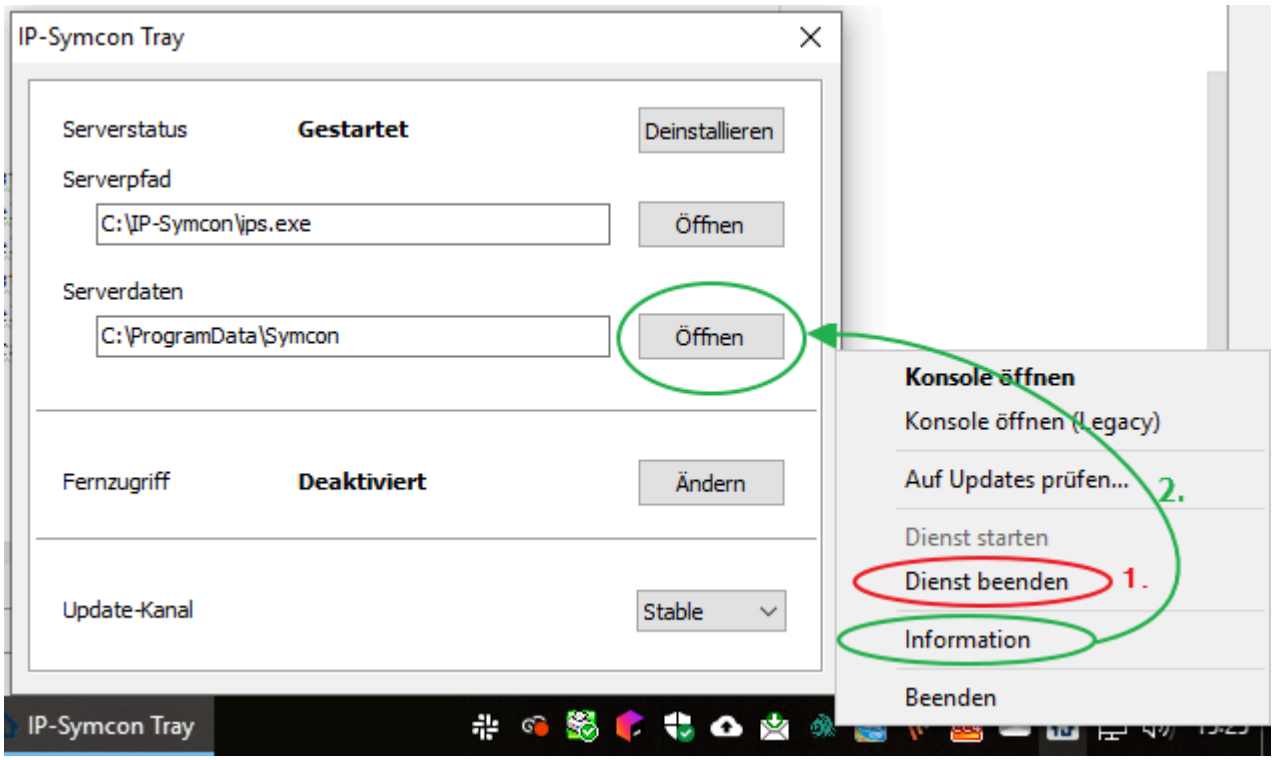
WINDOWS - CREATE BACKUP

The whole content of the folder C:\ProgramData\Symcon needs to be compressed as backup.zip.

Stop the Service via Tray Icon.
Open the Data folder via "Tray Icon -> Information -> Server Data -> Open" and select all files and folders.
Compress the files via "Right click -> Send to -> Compressed (zipped) folder" to create a .zip file. The file needs to be renamed to backup.zip.
The backup should be copied or moved to an appropriate medium.
Start the Service via Tray Icon.



If a message about empty folders appears while creating the Zip file, it can be ignored safely. Missing empty folders are created by IP-Symcon on demand.



MACOS - CREATE BACKUP

The whole content of the folder "/Library/Application Support/Symcon" needs to be compressed as backup.zip.
Stop the IP-Symcon Service.
Select "Finder -> Go to -> Go to folder" and enter the path "/Library/Application Support/Symcon".
Mark all required folders and files in the opened folder (usually all files and folders).
An Archive.zip is created on the desktop via "Clipboard -> Compress X objects".
Rename the file into backup.zip.
The backup should be copied or moved to an appropriate medium.
Start the IP-Symcon Service.

LINUX - CREATE BACKUP

The whole content of the folder "/var/lib/symcon" needs to be compressed as backup.zip.
zip may need to be installed.

```
sudo apt-get install zip
```

First, stop the IP-Symcon Service.

```
sudo /etc/init.d/symcon stop
```

Go to the Symcon folder and compress the folder.

```
cd /var/lib/symcon/ //Go to IP-Symcon folder
zip -r ~/backup.zip * //Compress the whole folder into backup.zip
```

The backup.zip is stored in the home folder. It can be copied or moved anywhere.
Start the IP-Symcon Service.

```
sudo /etc/init.d/symcon start
```

RASPBERRY PI - CREATE BACKUP

The whole content of the folder "/var/lib/symcon" needs to be compressed as backup.zip.
zip may need to be installed.

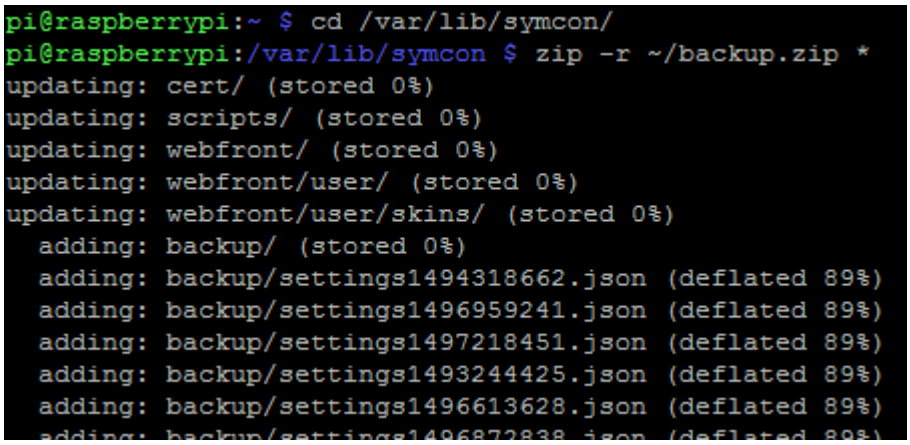
```
sudo apt-get install zip
```

First, stop the IP-Symcon Service.

```
sudo /etc/init.d/symcon stop
```

Go to the Symcon folder and compress the folder.

```
cd /var/lib/symcon/ //Go to IP-Symcon folder
zip -r ~/backup.zip * //Compress the whole folder into backup.zip
```

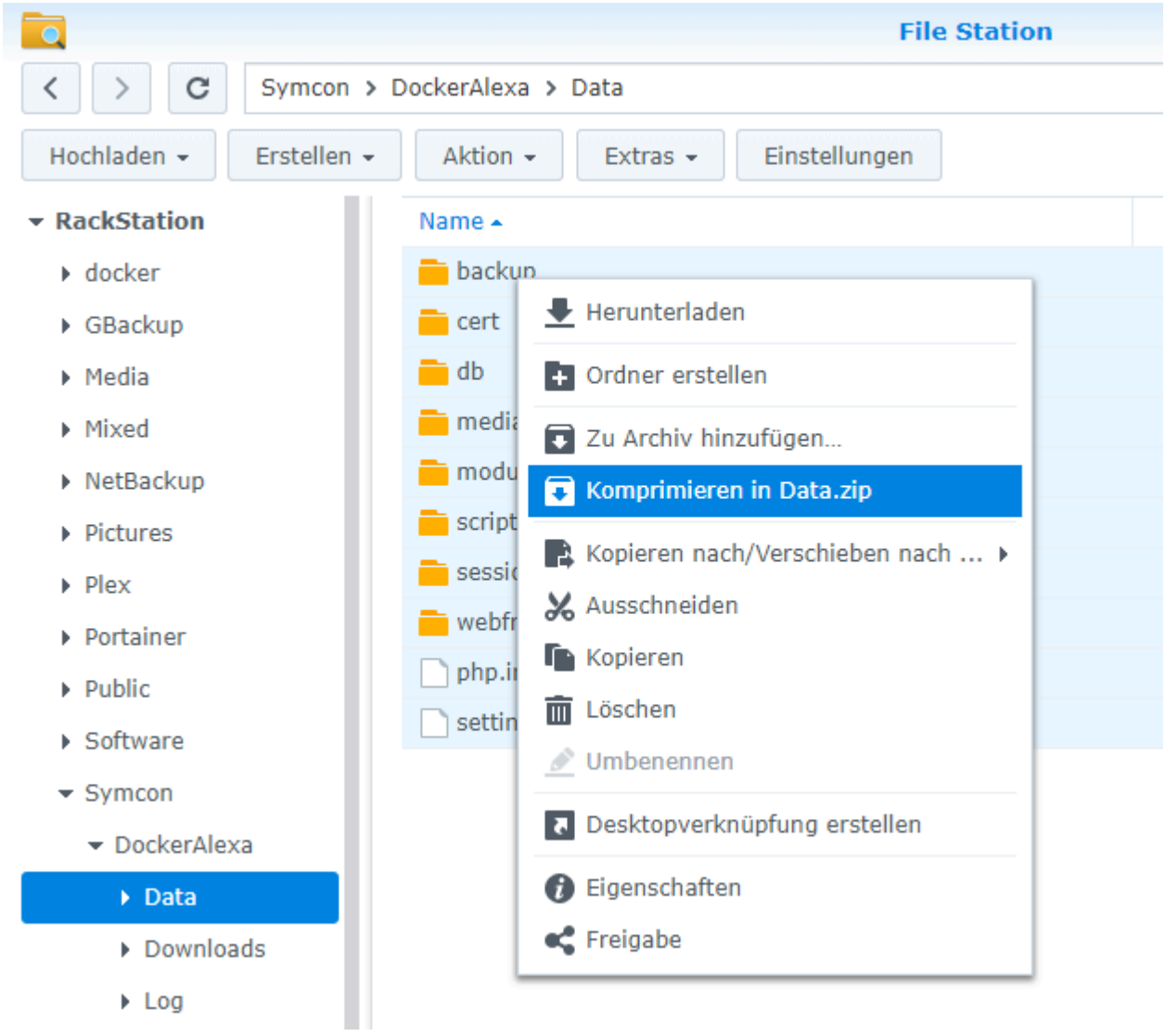


The backup.zip is stored in the home folder. It can be copied or moved anywhere.
Stop the IP-Symcon Service.

```
sudo /etc/init.d/symcon start
```

DOCKER/SYNOLOGY/QNAP - CREATE BACKUP

The whole content of the folder "/var/lib/symcon" needs to be compressed as backup.zip.
As an example we use a Synology system.
First, stop the IP-Symcon container.
The container needs to be stopped via the container management of the used system.
Afterwards, the mounted Symcon folder needs to be opened to pack the folder.
The volume on which the folder was "mounted" can be seen in the settings of the image.
According to the installation tutorial, the default is in the format "/Symcon/[MeinIPSymcon]/Data".
Finally, select all content and select "Compress" via right click.



The backup.zip is stored on the file system and can be downloaded.
Start the IP-Symcon container.

CATAN C1 - CREATE BACKUP

Stop the Symcon App. The app can be stopped in WBM under Settings -> App Management.
Copy all files via WinSCP from the following folder:

```
/opt/plcnext/apps/home/data/60002171301055/data/
```

Compress all files into a ZIP file to transfer them to another target system.
Start the Symcon App.



The "session" folder sometimes cannot be copied. This problem can be safely ignored

Basics

OBJECTS

IP-Symcon contains seven different types of objects:
Objects can be categories, instances, variables, automations, events, media, or links.

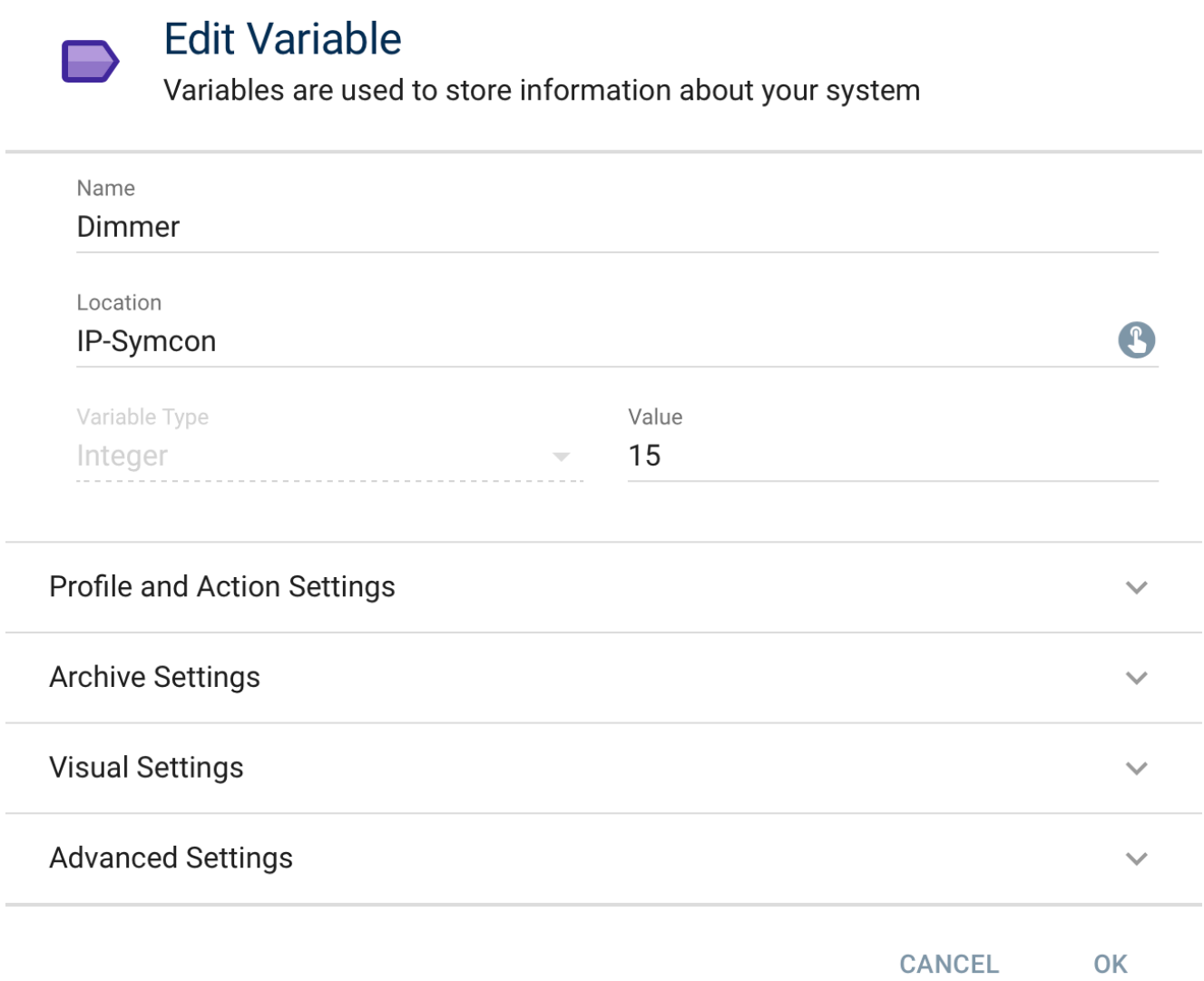


Every object is identified by a fixed and unchangable ID (identification number). This leads to multiple advantages:

- Every object is unique and distintively identifiable
- The name of an object can be changed at any time
- Bigger projects are processed faster
- Extensive functions of object management
- Performant links within the software

PROPERTIES OF AN OBJECT

The properties or settings of an object can be called via double click or "Right click" -> "Edit object".



ObjectID
Every object within IP-Symcon posesses a unique and unchangable identification number (ObjectID). Thus, every object is distinctively identifiable and referencable.

17275	Dimmer	Integer	15
43627	Save Energy	Flow Script	43627.ips.json
47150	Test Automation	Flow Script	47150.ips.json

Name
This is the name of the object. The name is also used when the object is shown in Visualizations.
Location
The location is the parent object in the object tree and defines its position in the object tree as well as the WebFront.

Certain instances, e.g., configurators, are positioned in specific special categories. These instances cannot be moved.

VISUAL SETTINGS

These settings define the behavior in the WebFront.
Icon
This is a symbol that is used in Visualizations. For further informations, see Icons.
Show object
The check box controls whether the object is shown in the Visualizations or remains hidden.
Show title (since Symcon 9.1)
Controls whether the title of the object's tile is shown or hidden in the visualization.
Show maximize button (since Symcon 9.1)
Controls whether the maximize button of the object's tile is shown or hidden in the visualization.
Activate object
Theis option controls if an object is active. If it is deactivated, the object is shown in grey in the Visualizations and cannot be controlled.

ADVANCED SETTINGS

Additional optional settings
Description
This is a description of the object. It can be used to store important notes and provide a direct description.
Ident
Objects possess an additional identifier Ident. Unlike a name, every value of Ident within a category of the logical tree view needs to be distinct.

CONTEXT MENU

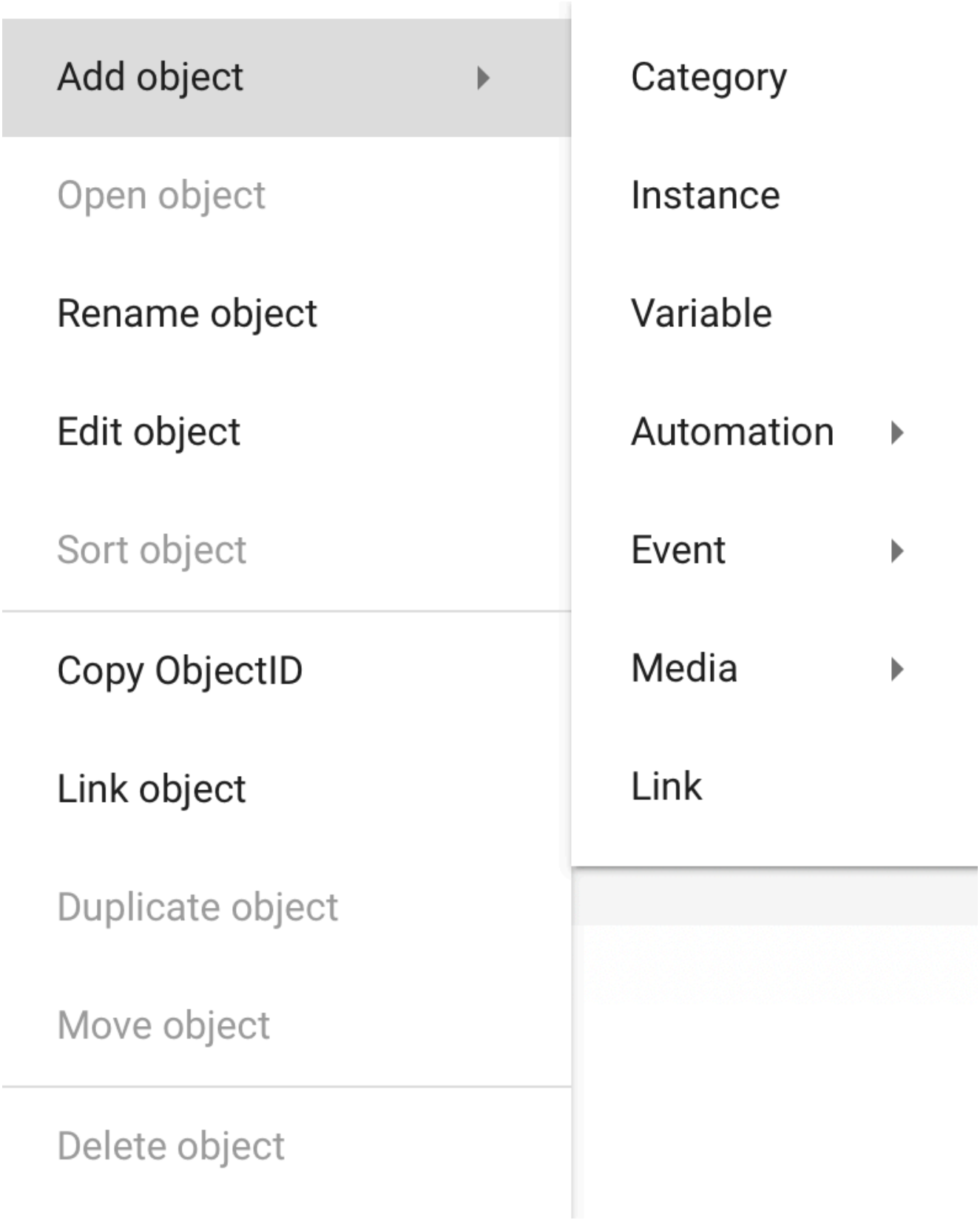
The context menu in the Object Tree provides a multitude of functions.



In addition to the described functions, each object type provides additional type specific options, see Object Type Specific Context Entries

ADD OBJECT

Add a new object. A dialog is opened that configures all relevant settings step by step.
For further information, see Add Object.



OPEN OBJECT

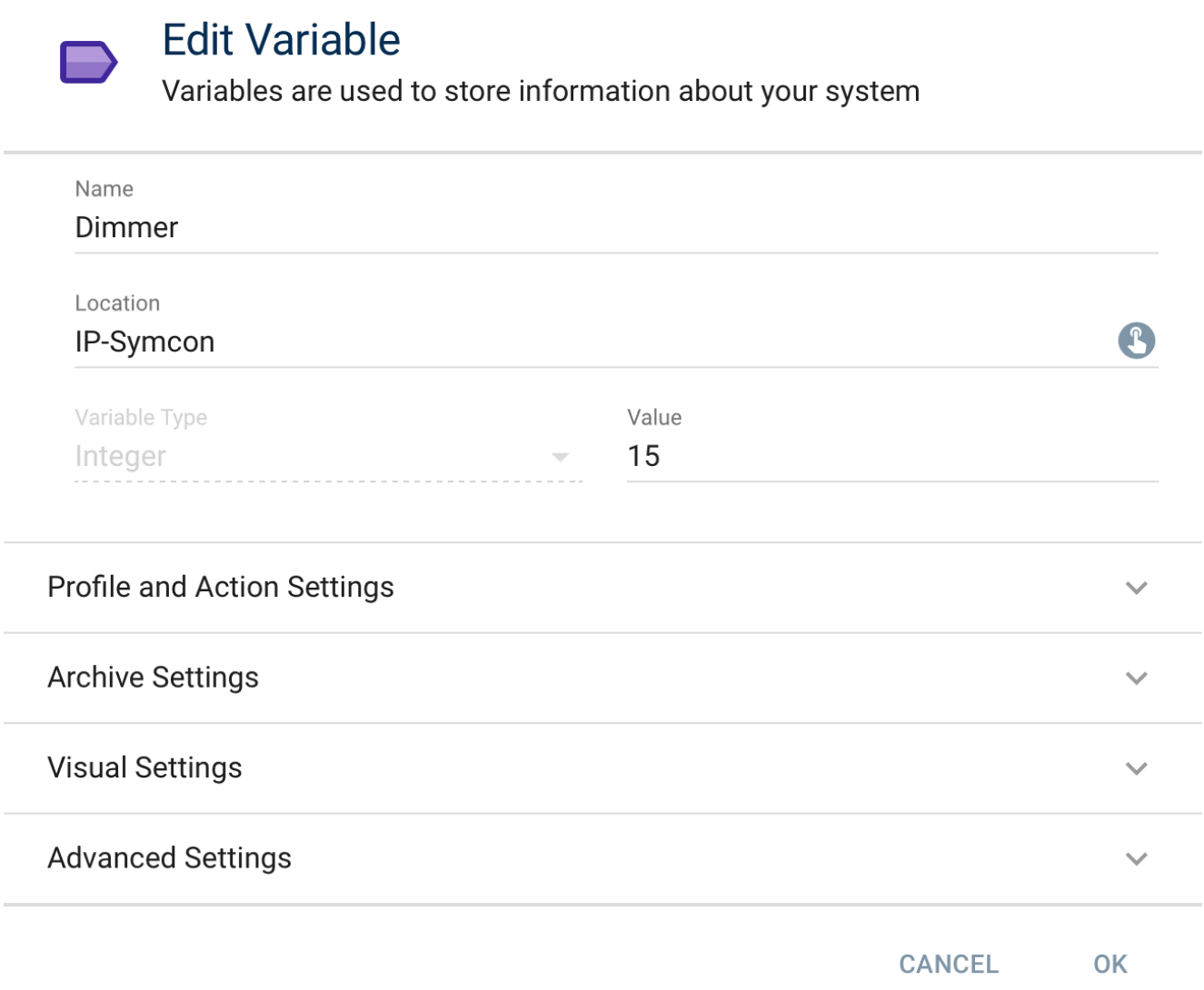
If the object has an object specific action, it is called. For instances, the configuration is opened, for automations the corresponding editor, media shows a preview. Otherwise, the option is deactivated. If possible a double click opens the object. Otherwise it edits the object.

RENAME OBJECT

Change the name of an object. The ObjectID and Ident remain unchanged. Multiple objects can have the same name.

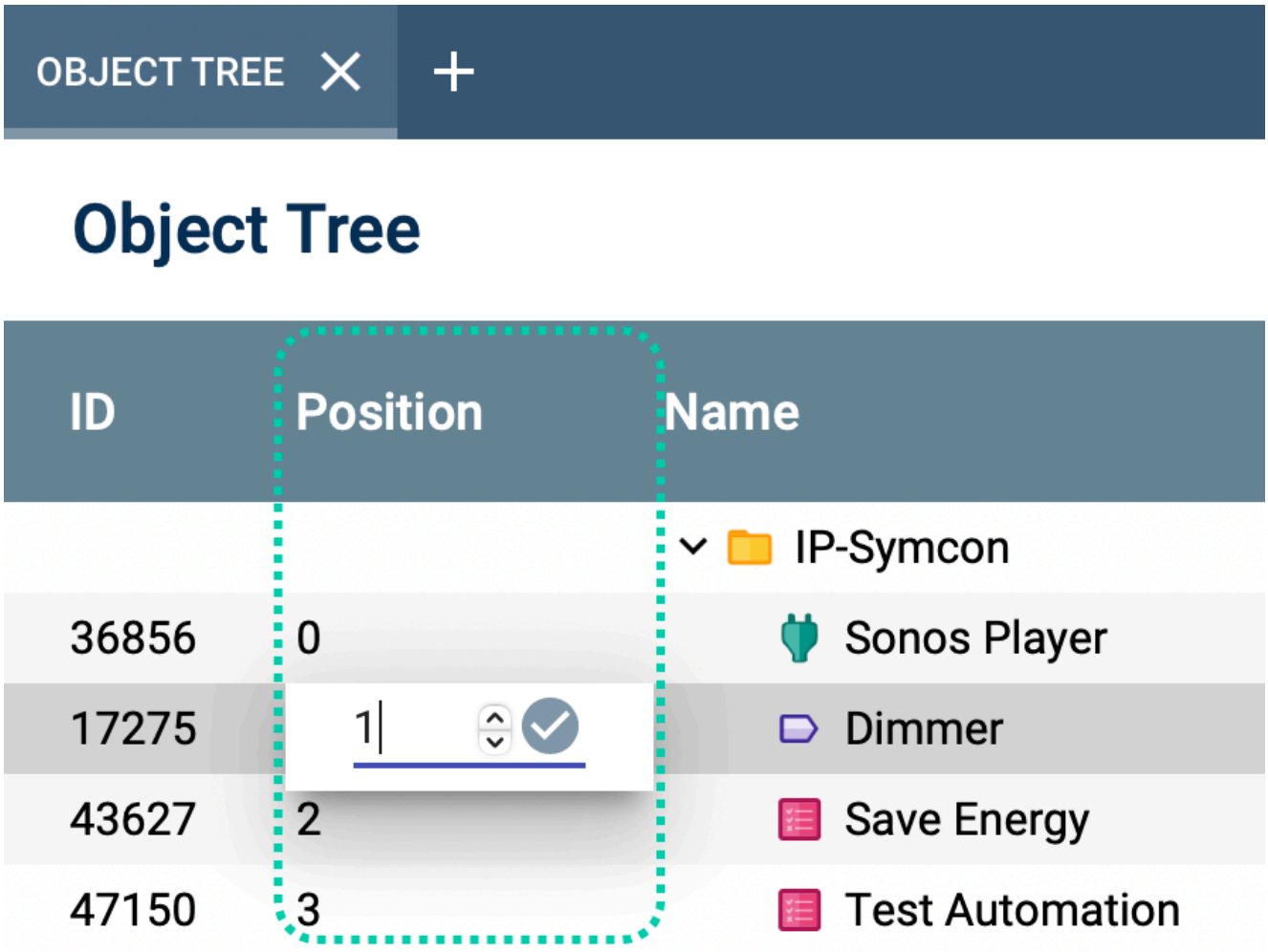
EDIT OBJECT

Every object offers the option "Edit Object". It can be used to read different informations and configure settings.



SORT OBJECT

Position the object at any position in the object tree. This option also controls the position in the Visualizations.



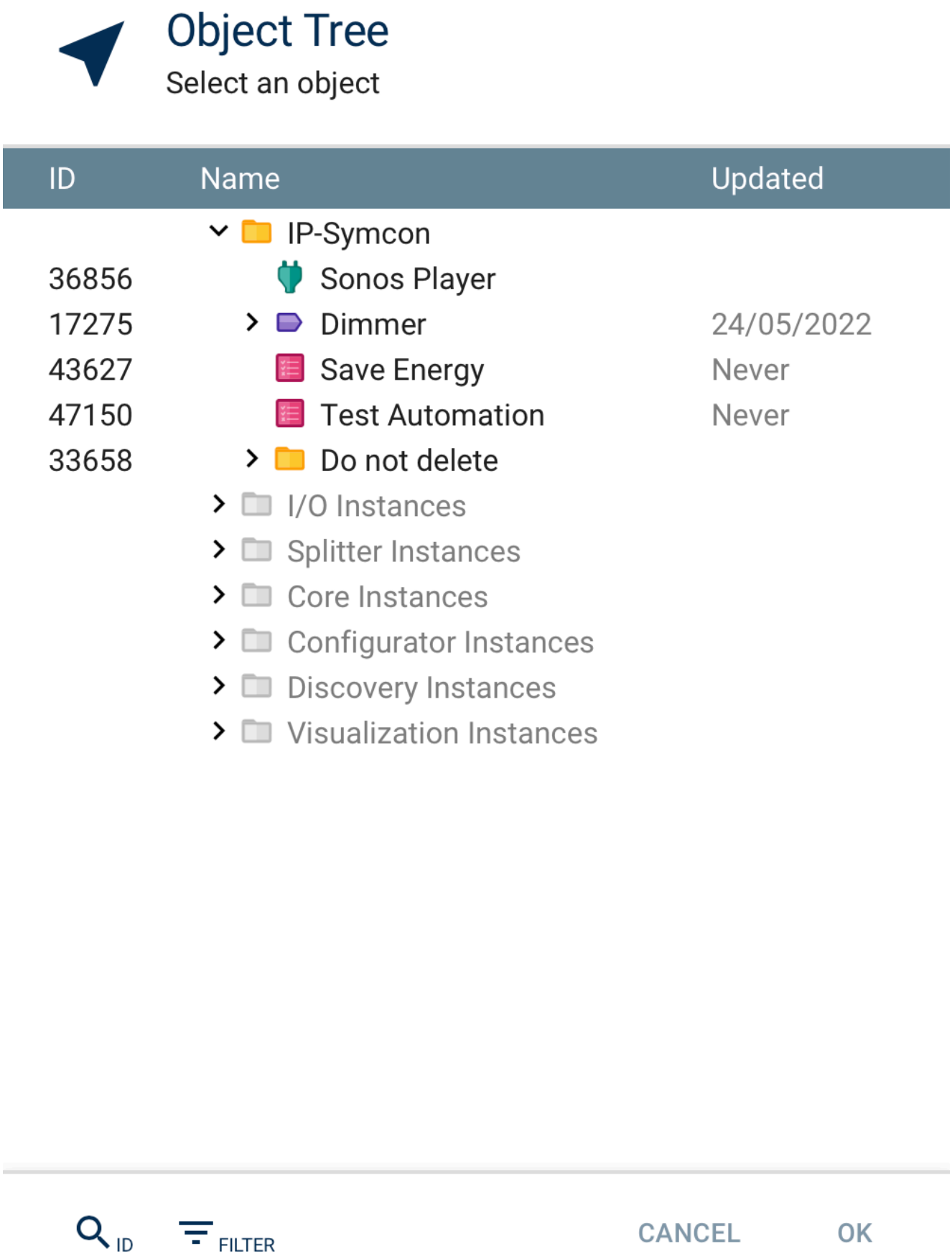
The "Position" column can be displayed permanantly with the "Columns" option. Thus, objects can be sorted faster via double click on the Position value instead of clicking "Sort object" for each object in the context menu.

COPY OBJECTID

Copy the ObjectID into the clipboard. It is useful to insert the IDs into scripts. ("Right click->Insert" or "Ctrl + V" in the editor)

LINK OBJECT

Create a link to the object. Multiple links can refer to the same object. The position of the link object is chosen in the dialog window.



DUPLICATE OBJECT

Create an identical object with its own distinct ObjectID.

MOVE OBJECT

Open a dialog to select the new selection of the object and place it there.

DELETE OBJECT

Delete the object and all its children, if applicable.

OBJECT TYPE SPECIFIC CONTEXT ENTRIES

SEARCH FOR REFERENCES (ALL EXCEPT FOR CATEGORY)

Open a dialog that either states that there are no references or lists all references.
The depending criteria are checked depending on the object type:

OBJECT TYPE	WHERE IS CHECKED FOR THE REFERENCE/ID?
Variable	The IDs for "VariableAction" and "VariableCustomAction" are checked.
PHP Script	The content of the PHP Script is checked for the ID.
Event	The content and the trigger are checked for the ID.
Link	The TargetID/Value is checked.
Instance	IDs of possible Objects that are used on the configuration page are checked.

TEST COMMANDS (INSTANCES ONLY)

Open a dialog to test the most relevant Actions of an instance.
For Experts: With the keyboard combination Ctrl + C, the currently set action can be copied as PHP Code and can be inserted into a script editor.

CHANGE VARIABLE (VARIABLES ONLY)

Open the dialog for setting a variable value.

EXECUTE AUTOMATION (AUTOMATIONS ONLY)

Executes the automation without opening it.
Eventual output is shown in a dialog.

JUMP TO SOURCE OBJECT (LINKS ONLY)

Jump to the source object within the object tree.

Categories

To simplify the handling of multiple active devices and keep an overview over a multitude of instances, categories can be used. The instances within the location tree can be assigned to different categories.

SETUP

A category can be created with the "+" at the bottom left or via "right click"->"**Add object**"->"**Category**". In a next step, the category can be configured. This includes the name and location. A suitable name could refer to the area where the according devices are localized.
For example, in a one family house, the categories could refer to the floors.



When a category is deleted, consider that all subcategories and devices within that category are removed from the location tree as well.

CONFIGURATION

The created category can be found in the Object Tree. Further categories can be created in the same way, e.g., multiple categories for different floors.
With a right click on the chosen main categorie and "Add Object -> Add Category", the main location is already selected as location. Afterwards, the subcategory can be configured, e.g., naming it "living room". This procedure can be repeated multiple times.
The order of the categories can be modified with "Right click -> Sort Objects".
Nesting categories within each other can also be done via Drag and Drop.



We recommend to use the structure
Category/Subcategory/Device as follows:
Floor / Room / Device

HINT

For a better overview, this naming should be done from the start as more devices are added over time. Every device name should explain itself und contain no previously named components.
E.g., Ground Floor / Living Room / Ceiling Lamp (North)

WEBFRONT

An additional advantage of this structure is the automatic direct visualization within the WebFront.
When a shown category contains visible subcategories, a navigation bar for categories is shown automatically.

When a subcategory is selected, the object area shows its content.
When the shown subcategory contains further subcategories, the navigation is extended automatically.
Categories that are linked with the "Link" object are shown in the navigation bar as well. The property "Show Navigation" can be used to hide the navigation bar when needed.

Instances

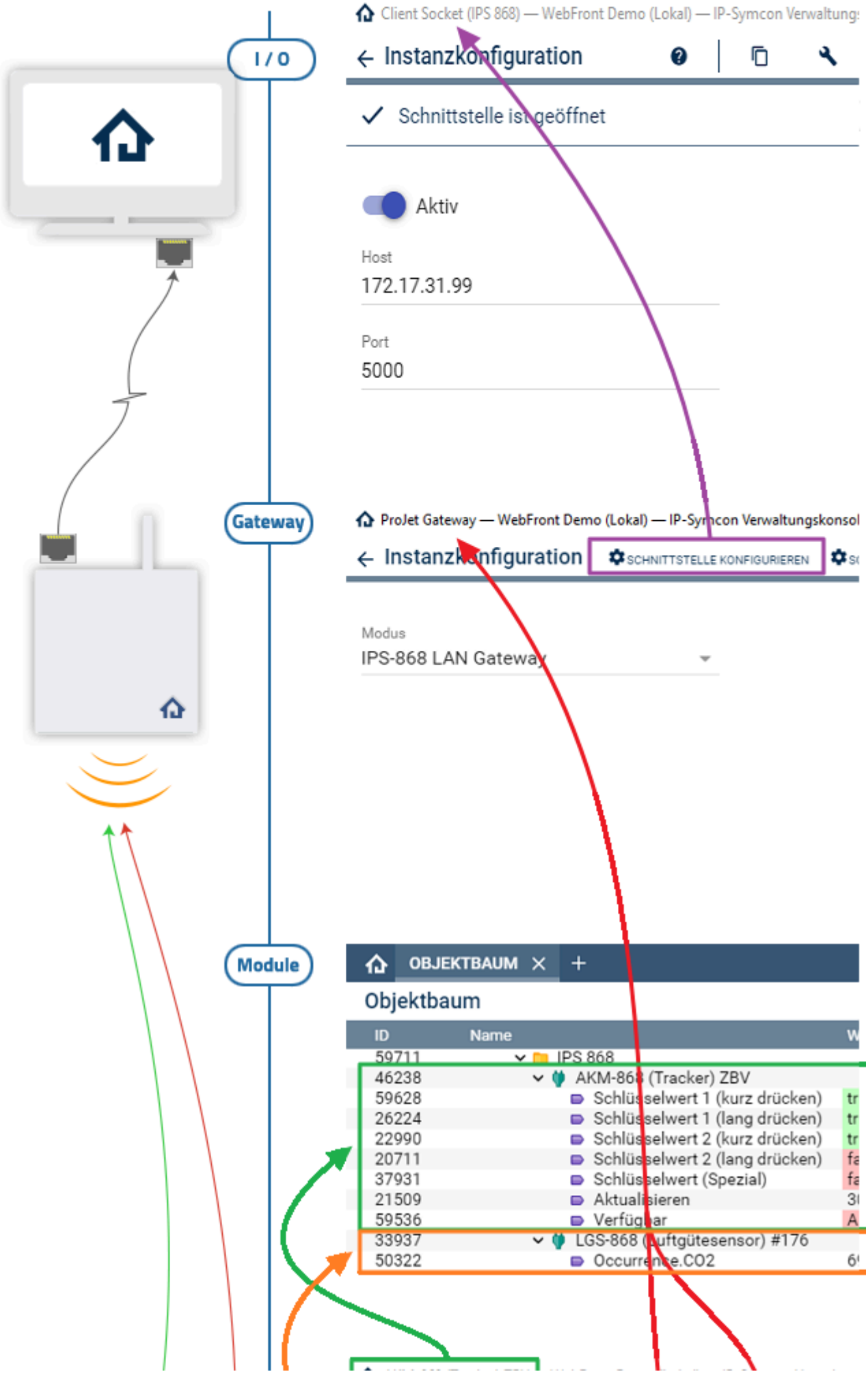
Instances usually represent devices that are connected to IP-Symcon. Other instances include non-physical modules like Text To Speech or Mediaplayer . The states of devices are represented via Variables in the Management Console and WebFront.

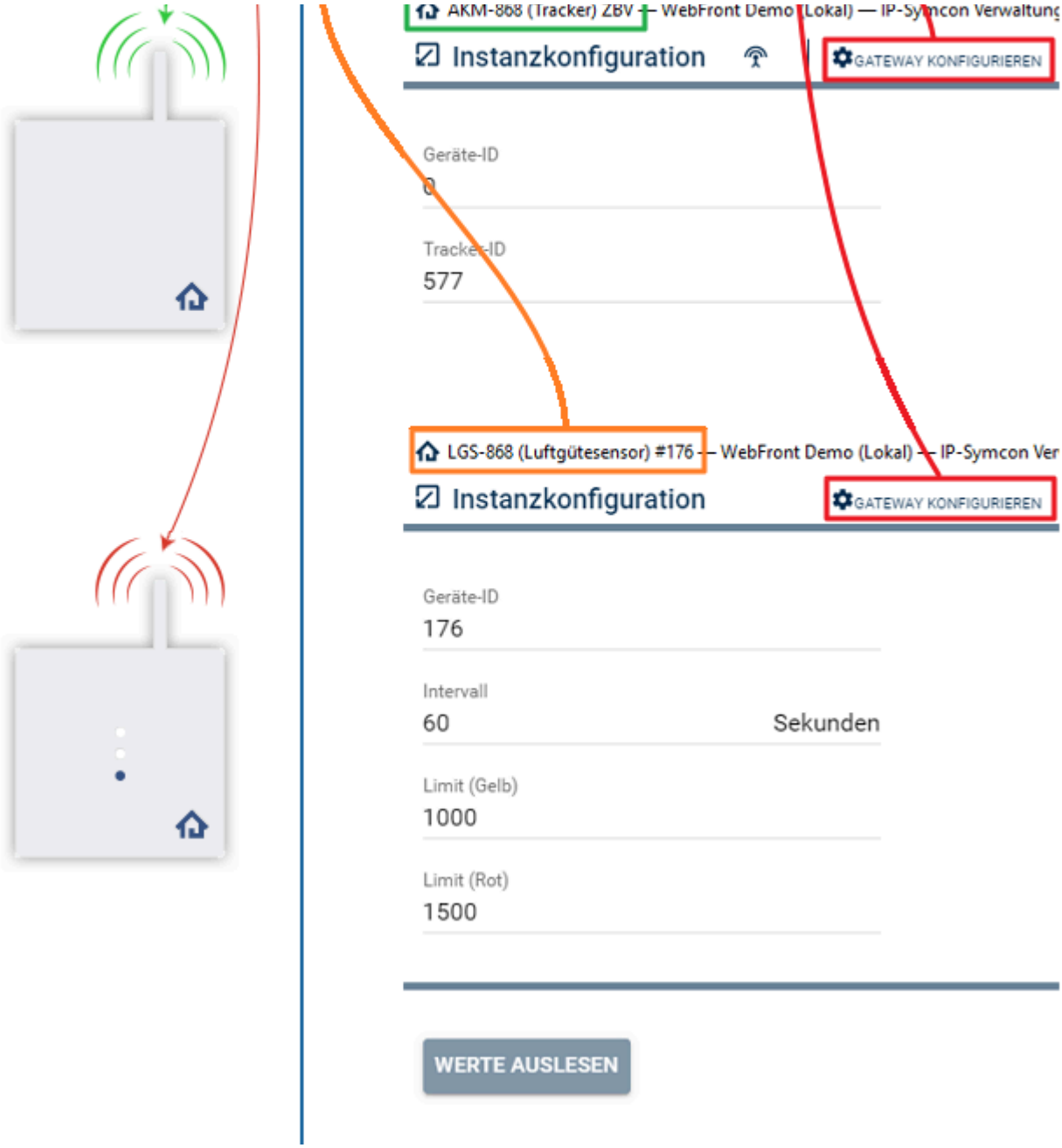
INCLUSION

New devices can be added via the "Add Instance" dialog or the configurators within the management console. In addition, IP-Symcon automatically creates further instances that establish the communication between devices and IP-Symcon when required. These parent instances are exchangable and reconfigurable without including the device another time. For example, if the communication protocol should be changed from radio to serial, it can be changed in the corresponding gateway or I/O instance. From then on, the devices that are connected via the corresponding gateway are handled accordingly. However, the state variables, the existing scripts, or the originally created instance for those devices are unaffected.

PARENT INSTANCES

Parent instances know and use the required protocols and interface properties to provide a connection between the devices and IP-Symcon. These instances need to be gateway or I/O instances.
If parent instances are required for a connection, IP-Symcon automatically creates and configures them. Eventually, further configuration is required, e.g., setting the IP address. The fastest way to access the parent instance of a device/module is to use the gear within the corresponding configuration tab.
As shown in the example below, the two devices "AKM-868" and "LGS-868" communicate via radio with their parent instance "ProJet Gateway". The gateway itself communicates with its parent I/O instance "Client Socket IPS 868" via LAN.
The communication of the I/O instance could easily be changed from LAN to serial (which only makes sense if the new gateway communicates with IP-Symcon via a serial cable). The change can be done by changing the mode within "ProJet Gateway" to "Connection via: Serial". IP-Symcon automatically creates a new serial port instance that is connected to the gateway. The parent instance "Serial Port IPS 868" could be accessed via the gear on the configuration tab of the gateway. The two devices AKM and LGS would be completely unaffected by this change.


I / O
Client Socket (IPS 868) — WebFront Demo (Lokal) — IP-Symcon Verwaltung
Instanzkonfiguration
Schnittstelle ist geöffnet
Aktiv
Host
172.17.31.99
Port
5000
Gateway
ProJet Gateway — WebFront Demo (Lokal) — IP-Symcon Verwaltungskonsol
Instanzkonfiguration
SCHNITTSTELLE KONFIGURIEREN
Modus
IPS-868 LAN Gateway
Module
OBJEKTBAUM
Objektbaum
ID
Name
59711
IPS 868
46238
AKM-868 (Tracker) ZBV
59628
Schlüsselwert 1 (kurz drücken)
26224
Schlüsselwert 1 (lang drücken)
22990
Schlüsselwert 2 (kurz drücken)
20711
Schlüsselwert 2 (lang drücken)
37931
Schlüsselwert (Spezial)
21509
Aktualisieren
59536
Verfügbar
33937
LGS-868 (Luftgütesensor) #176
50322
Occurrence.CO2



TYPES OF INCLUSION

Instances are always included into IP-Symcon with a similar pattern.
There are three types of inclusion.

TYPE	DESCRIPTION	EXAMPLE SYSTEMS
1:1:n	1 I/O - 1 Gateway - n Instances	EnOcean
1:m:n	1 I/O - m Gateway - n Instances	LCN
1:n	1 I/O - n Instances	Register Variable

CONNECTION COMPONENTS

I/O:
I/O describes the type of communication between the gateway and the server (IP-Symcon).

TYPE	DESCRIPTION
Client Socket	TCP client based communication
UDP Socket	UPD client based communication
Serial Port	serial based communication
Virtual I/O	emulation of a Serial Port, a Client Socket, or a UDP socket
WWW Reader	HTTP(Get) requests
HID	HID based communication (USB)
Server Socket	TCP server based communication

Gateway/Splitter:
One or more instances of devices can be connected to one gateway. The gateway handles the communication between the devices and I/O.



If a non-standard configuration is used, it needs to be controlled if the according I/O instance fits the corresponding configuration, e.g., LAN is used instead of a serial connection. In that case, the I/O instance needs to be changed.

GUID

Every instance has a unique GUID that determines the type, e.g., an AKM-868 or WDT-868. The GUID is a UUID and has the format 8-4-4-4-12. The numbers describe the amount of letters. Each letter is either 0-9 or A-F. Hypens and braces are required and only capital letters are allowed. (Example: {12345678-90AB-CDEF-1234-567890ABCDEF})



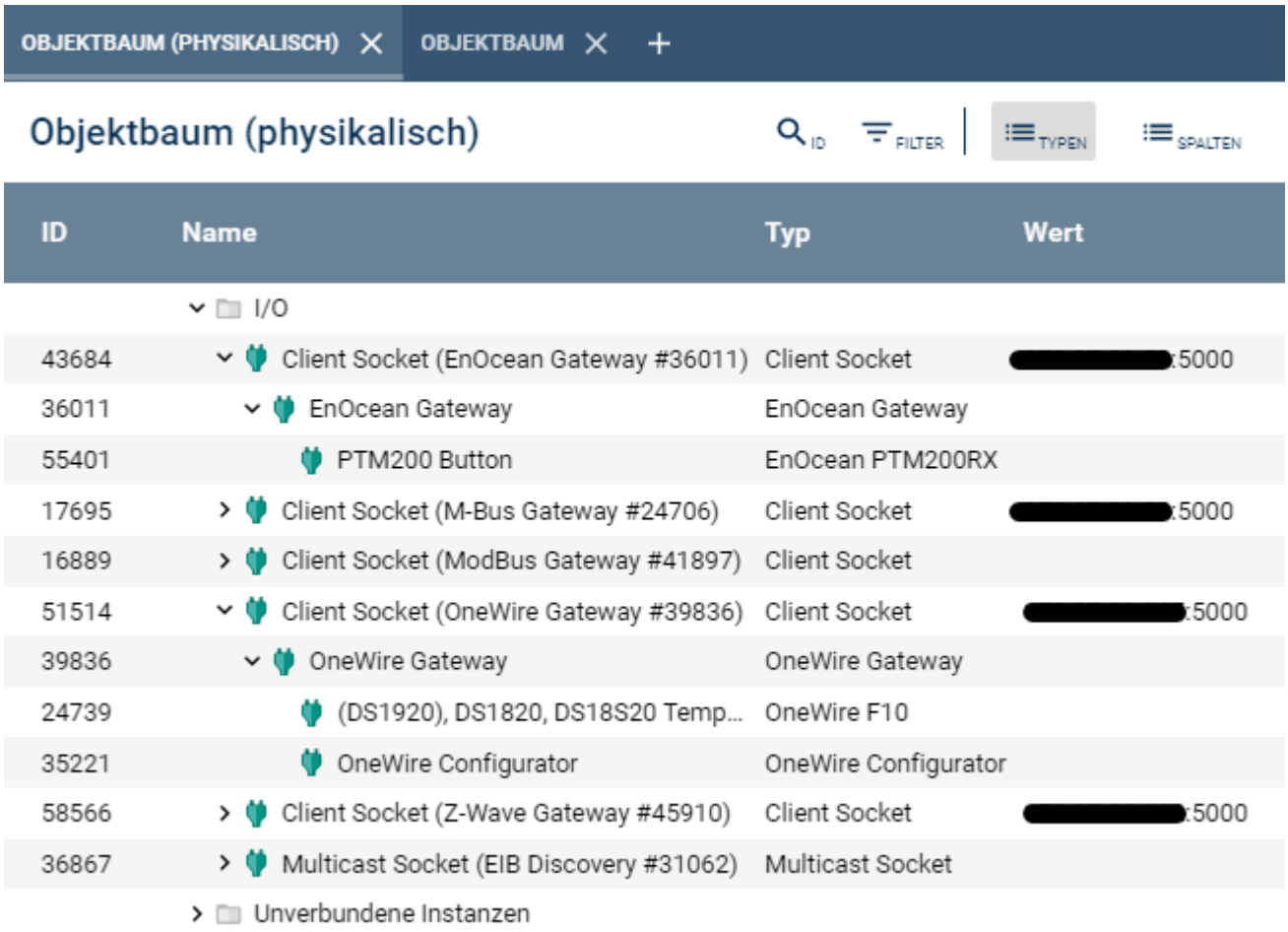
The GUID should not be confused with die 5-digit ObjectID, which distinctly identifies an individual object but not its type.

CREATE INSTANCE

The creation of an instance and the inclusion of devices is described here:
Include Device

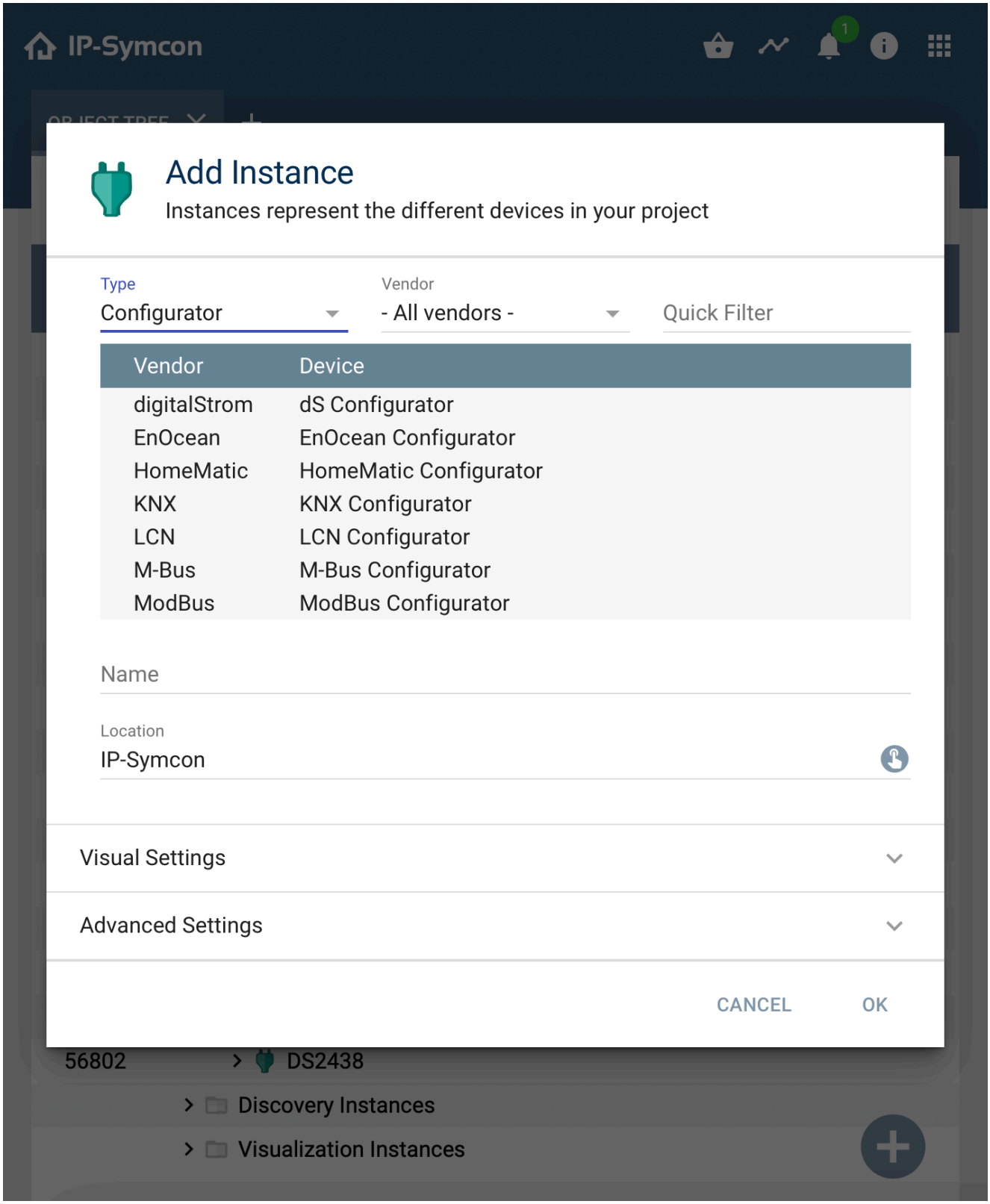
EXAMPLE

This is an example with the physical tree view.
Device instance "PTM200 Button"
-> (parent) gateway instance "EnOcean Gateway"
-> (parent) I/O instance "Client Socket (EnOcean Gateway #36011)"



Configurators

Configurators are a special type of instances. When a configurator is available for a system, it drastically simplifies the configuration of IP-Symcon. Configurators can be created for multiple strands/lines/gateways. A configurator can be created from the Welcome tab of the IP-Symcon management console.



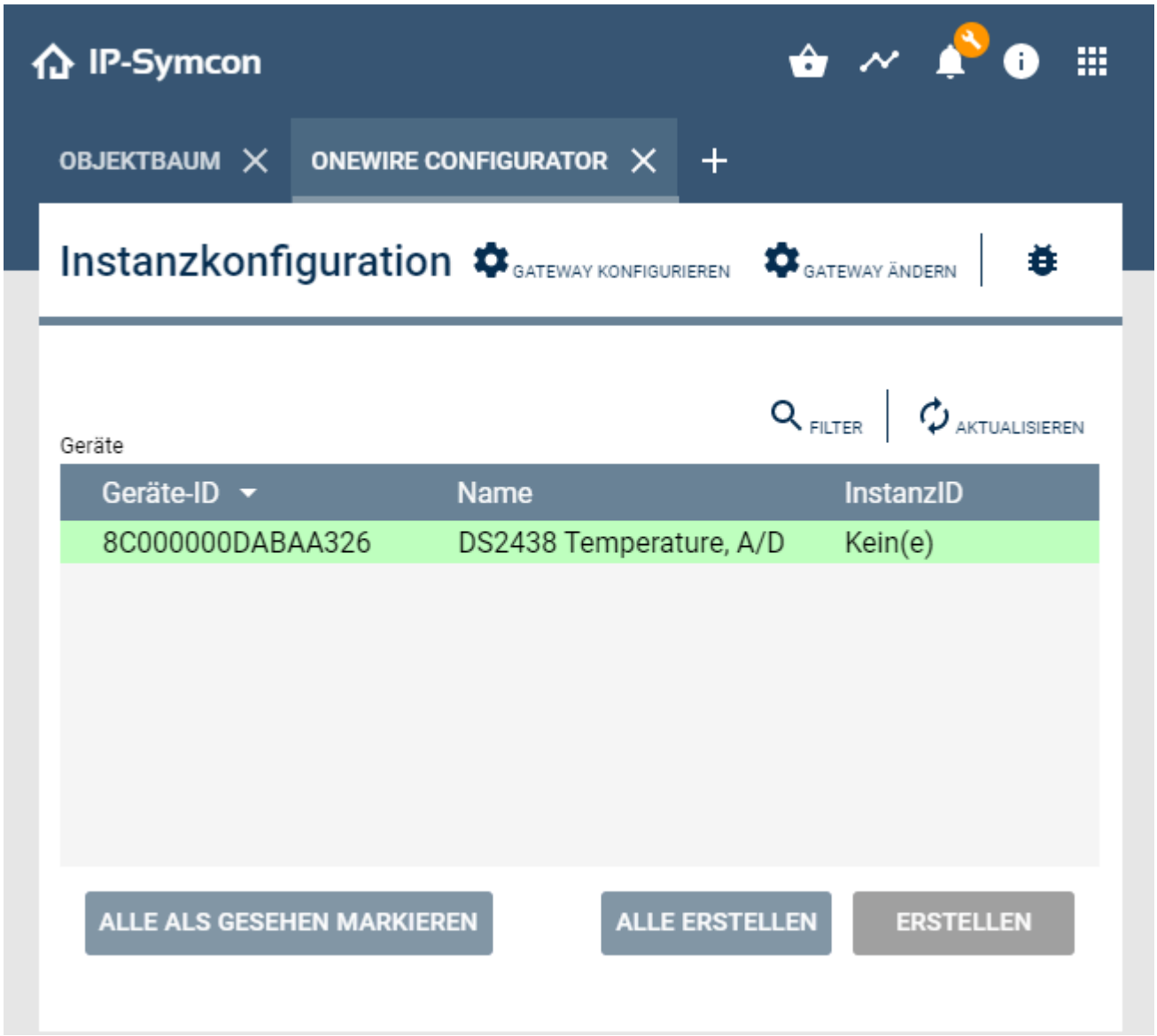
Currently, configurators are available for the following systems:

1-Wire
digitalStrom
Eaton xComfort
KNX
EnOcean
HomeMatic
IPS-868
LCN
M-Bus
MQTT
Siemens Logo
Siemens OZW
Z-Wave

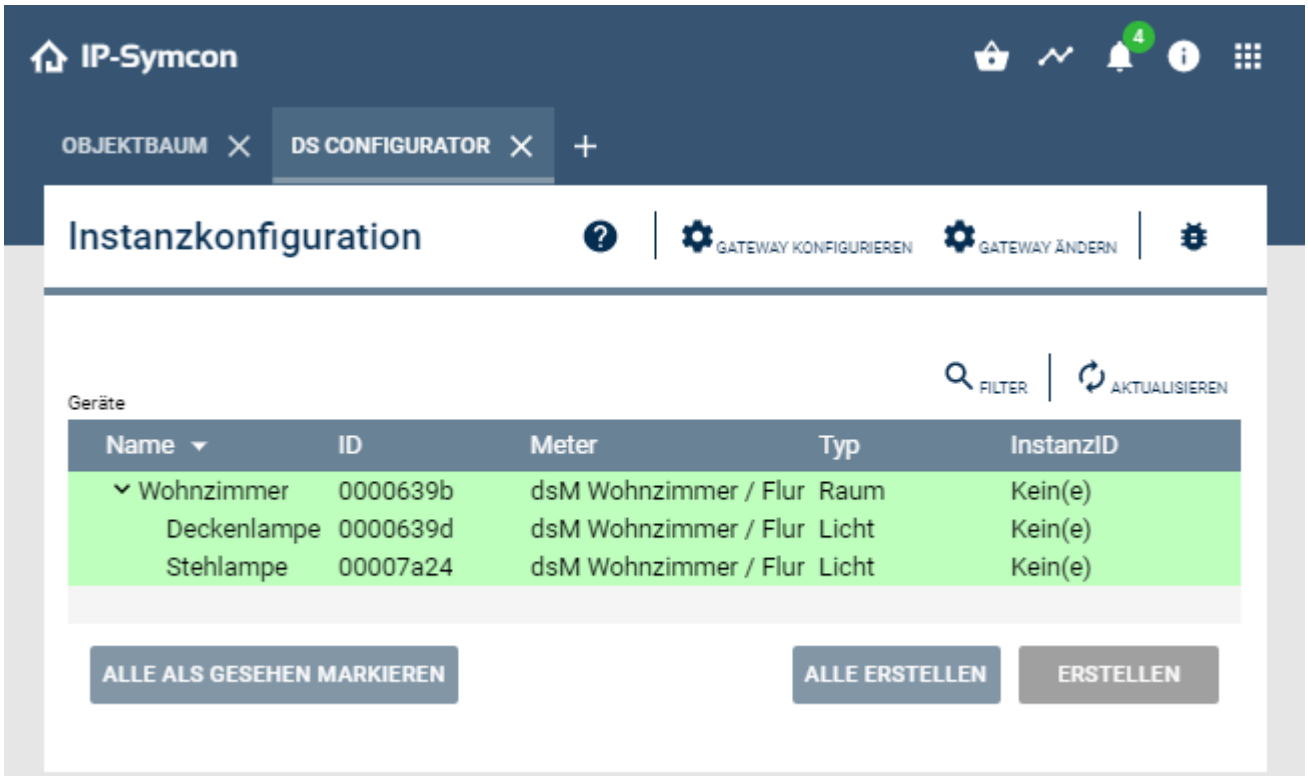
COLOR	MEANING	REMARK
Green	New device that was not configured in IP-Symcon yet	Device can be added as instance
Red	Device is not available any more, but is still configured in IP-Symcon	Instance can be removed from IP-Symcon
Gray	The existing instance is configured differently than recommended	The Create button changes to Review to show the recommended settings
White	Device available and configured in IP-Symcon	No further actions required

SCREENSHOTS

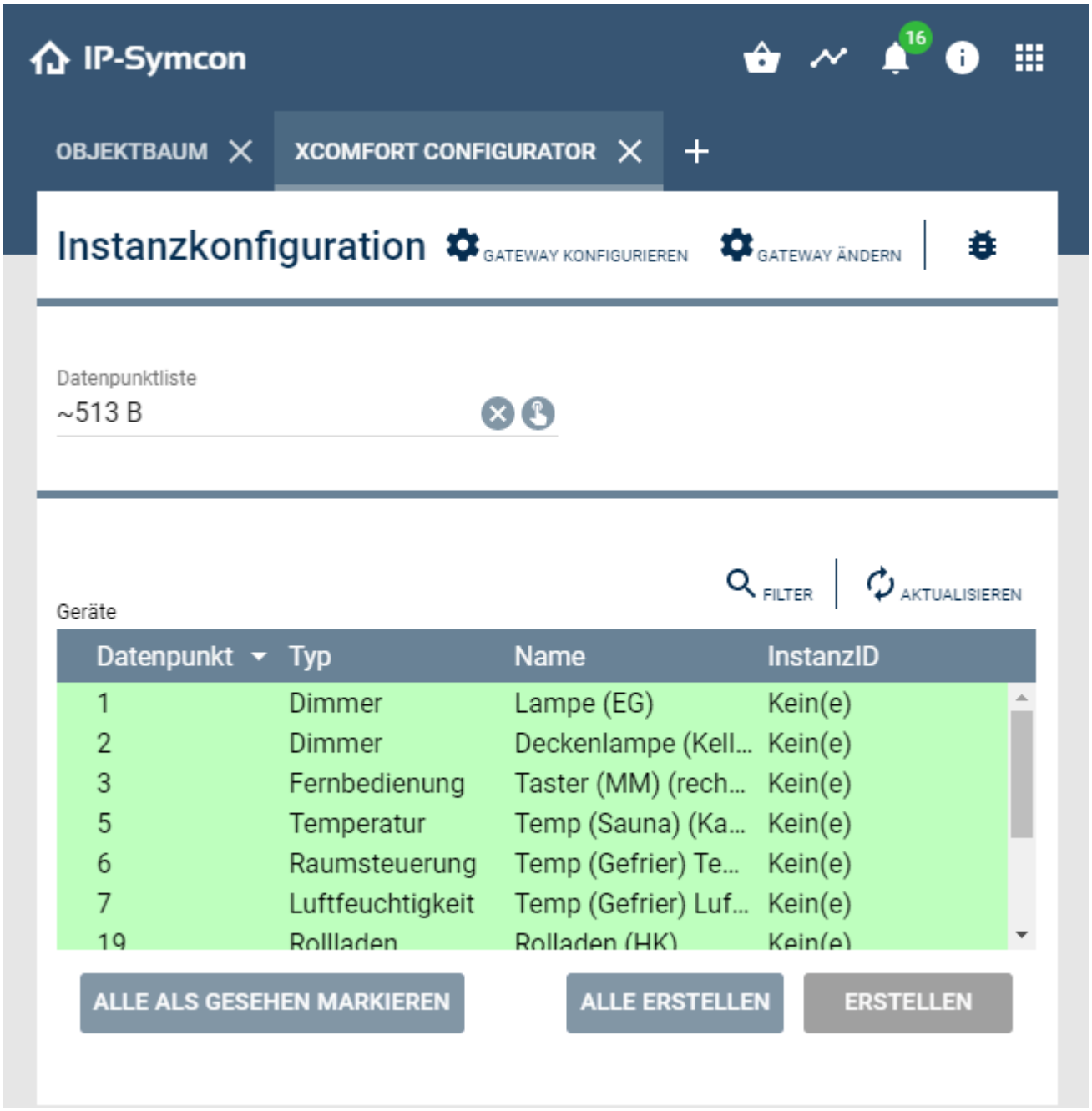
1-WIRE CONFIGURATOR



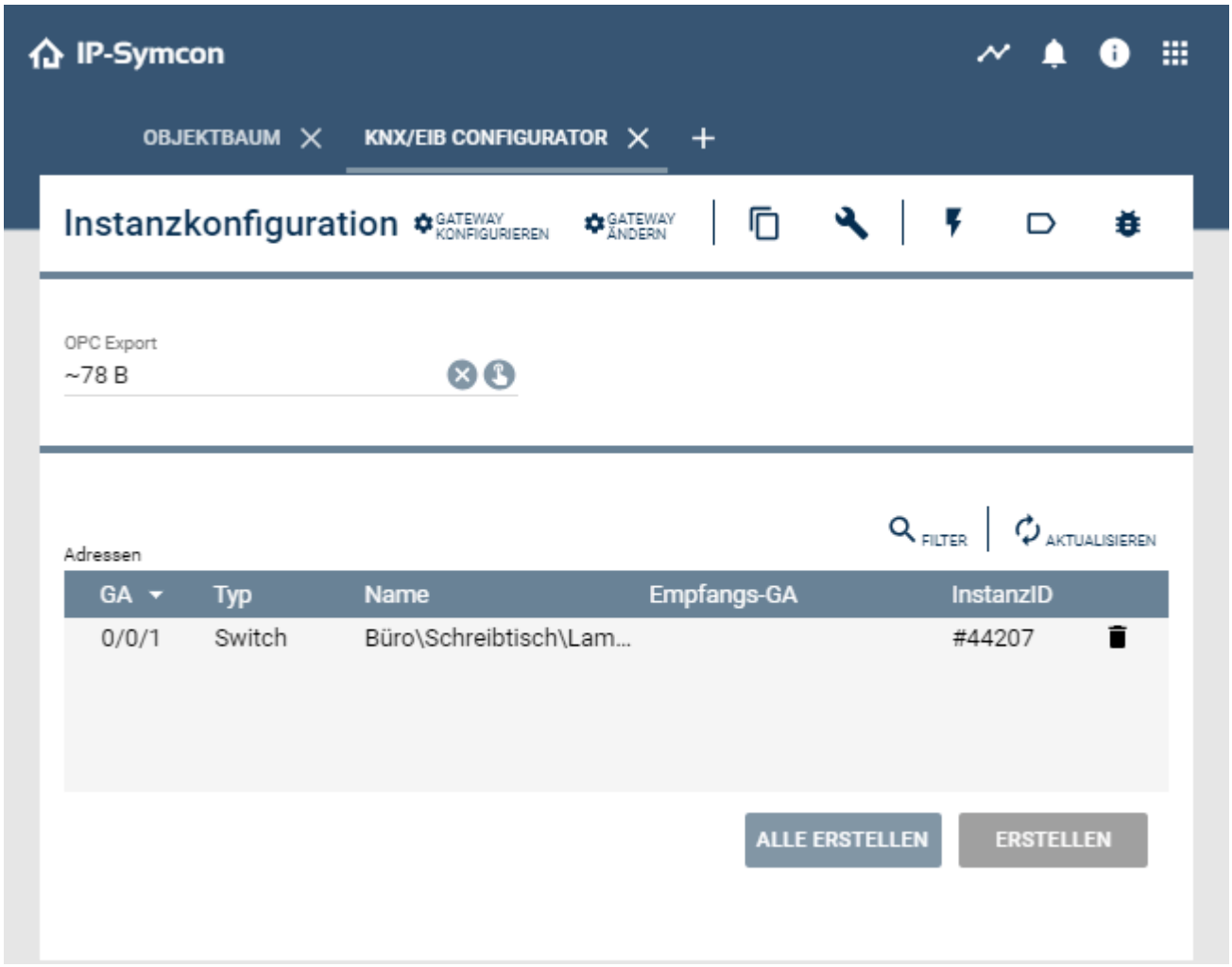
DIGITALSTROM CONFIGURATOR



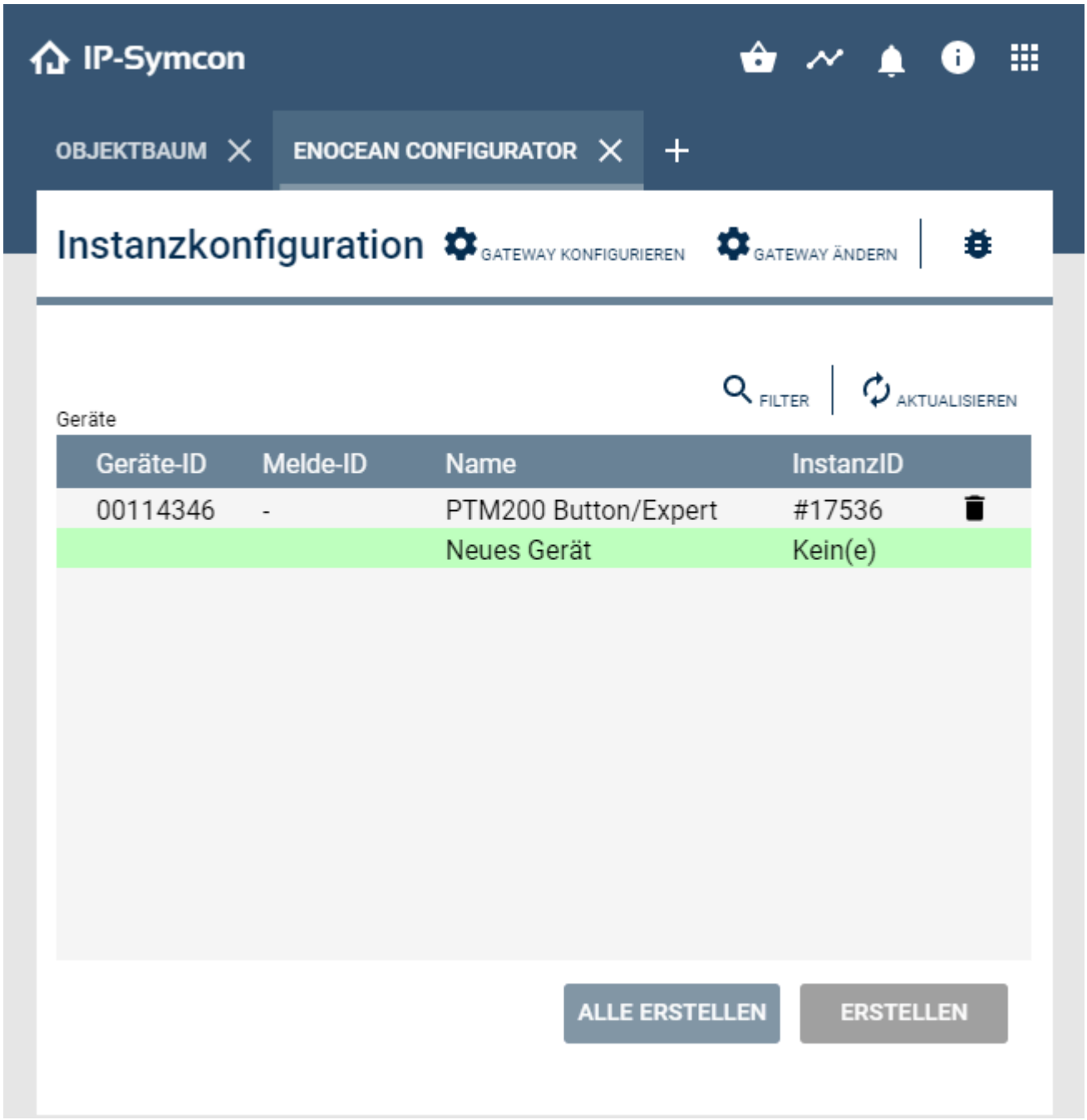
XCOMFORT CONFIGURATOR



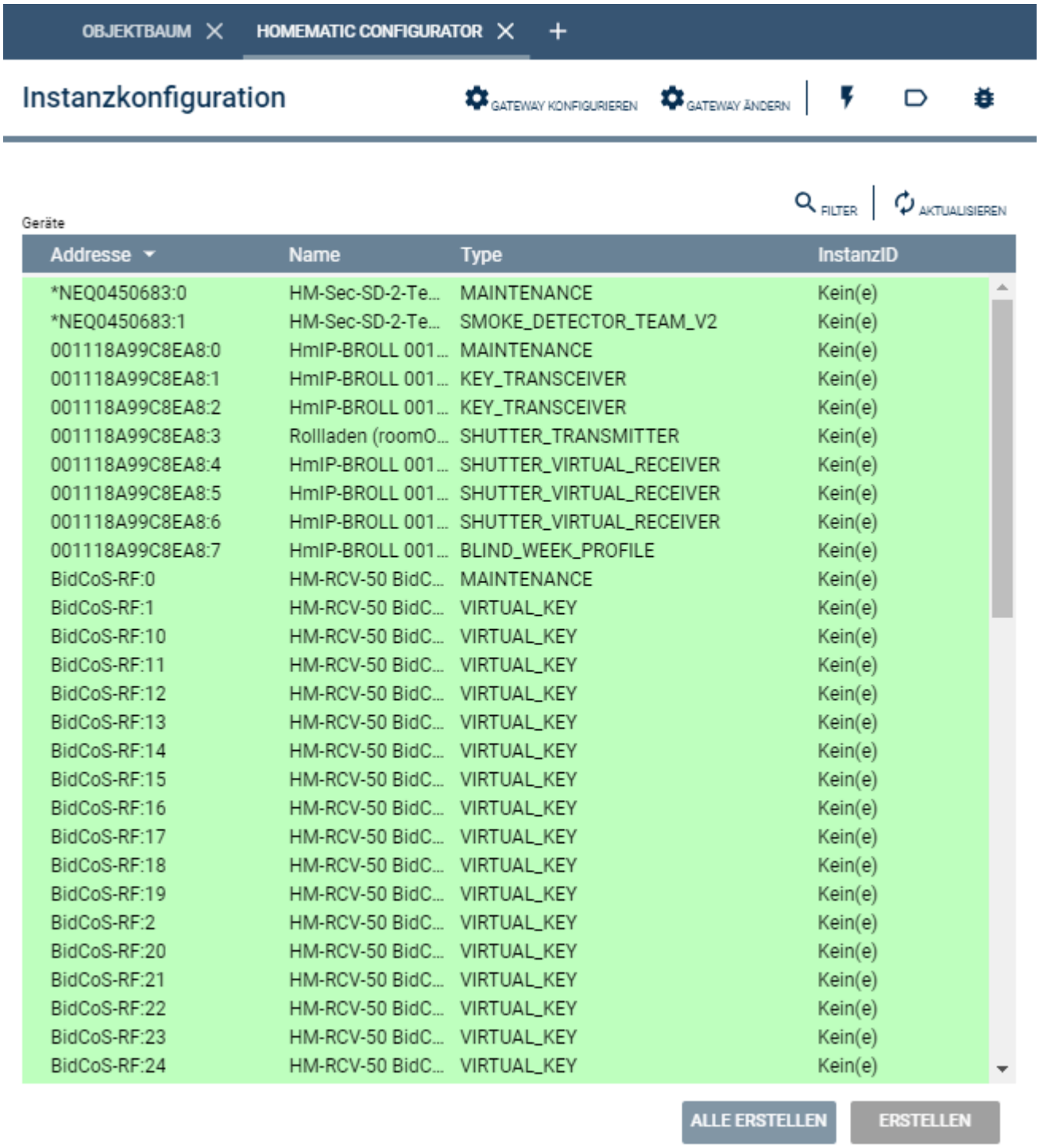
KNX CONFIGURATOR



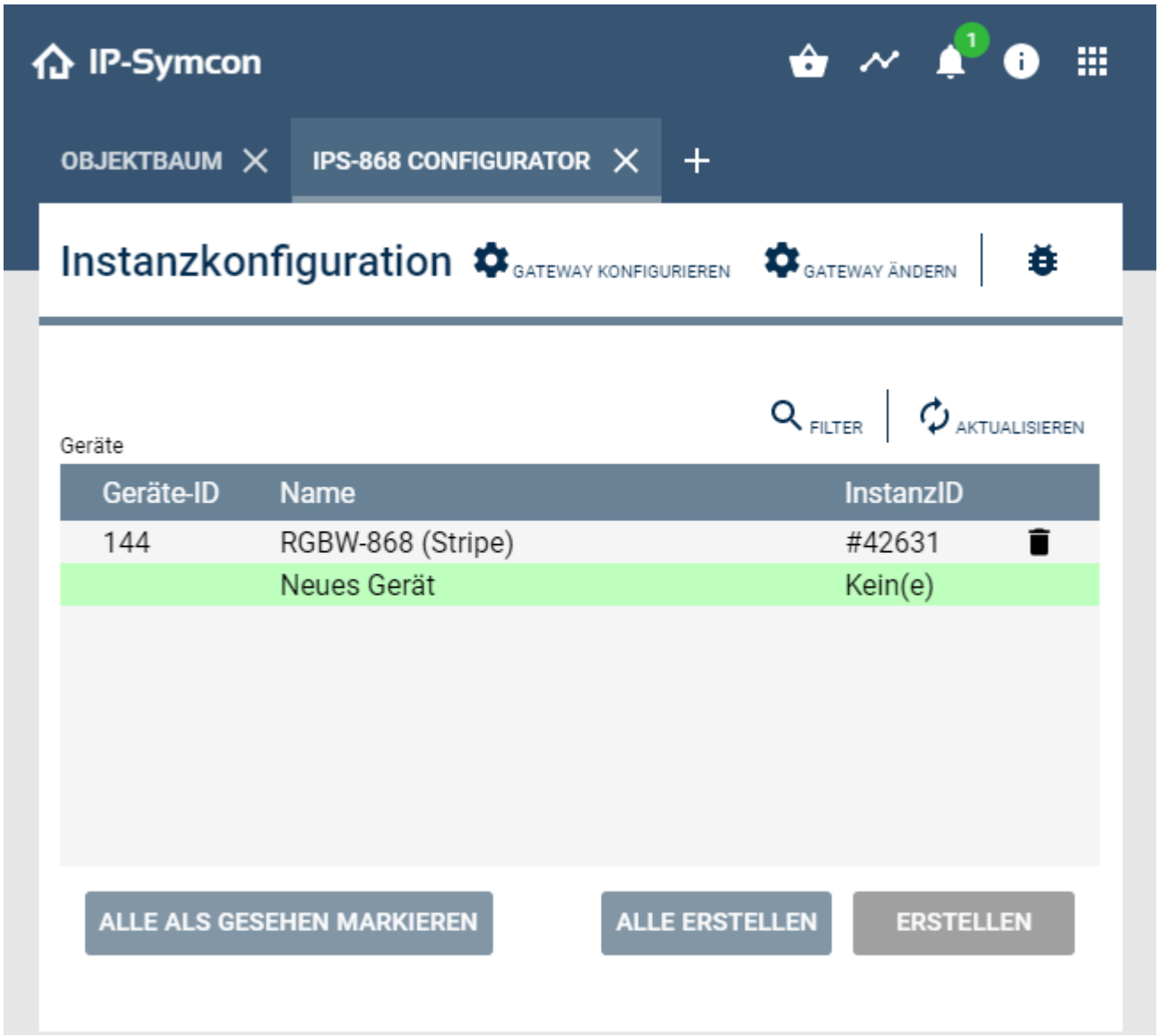
ENOCEAN CONFIGURATOR



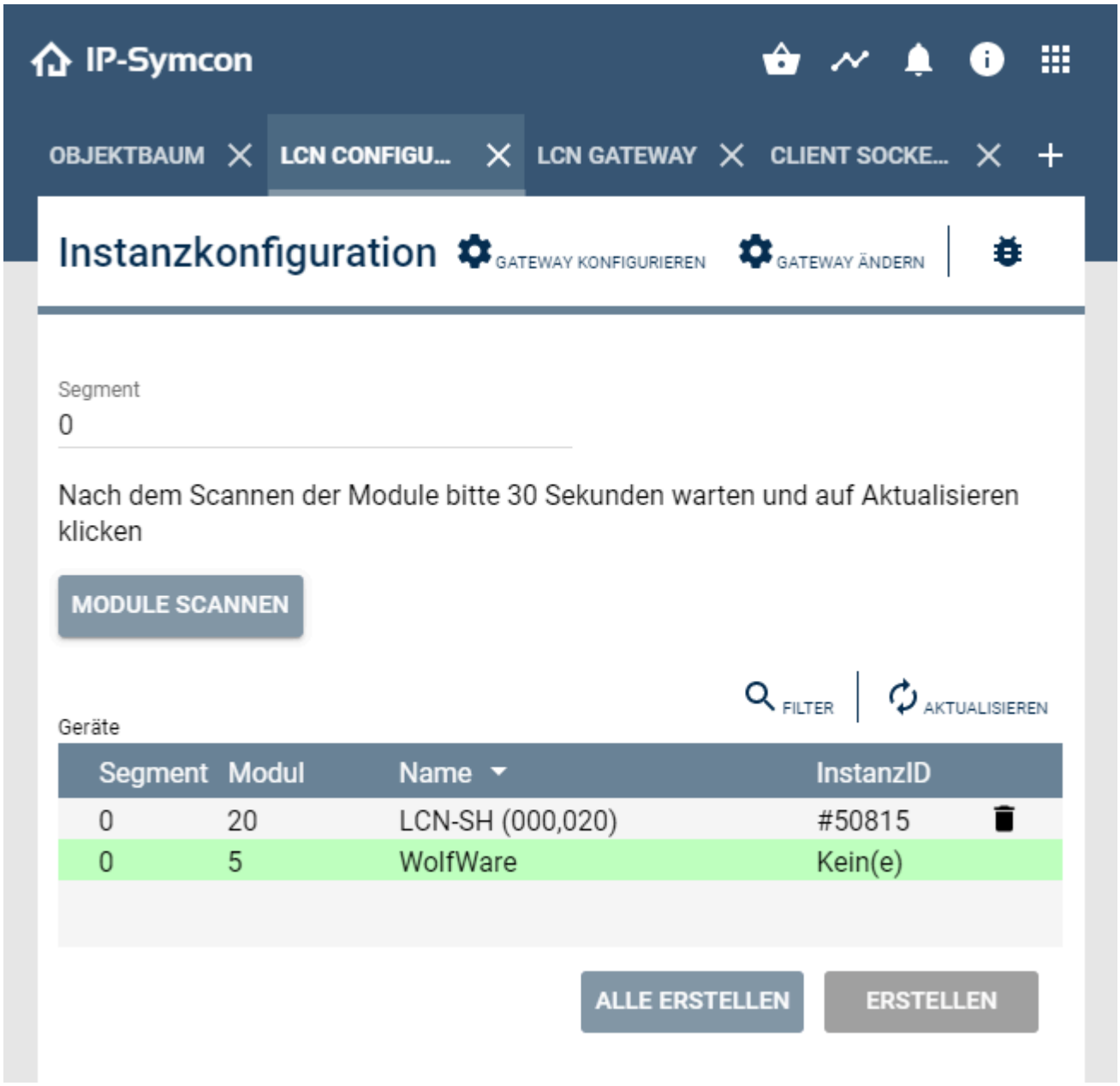
HOMEMATIC CONFIGURATOR

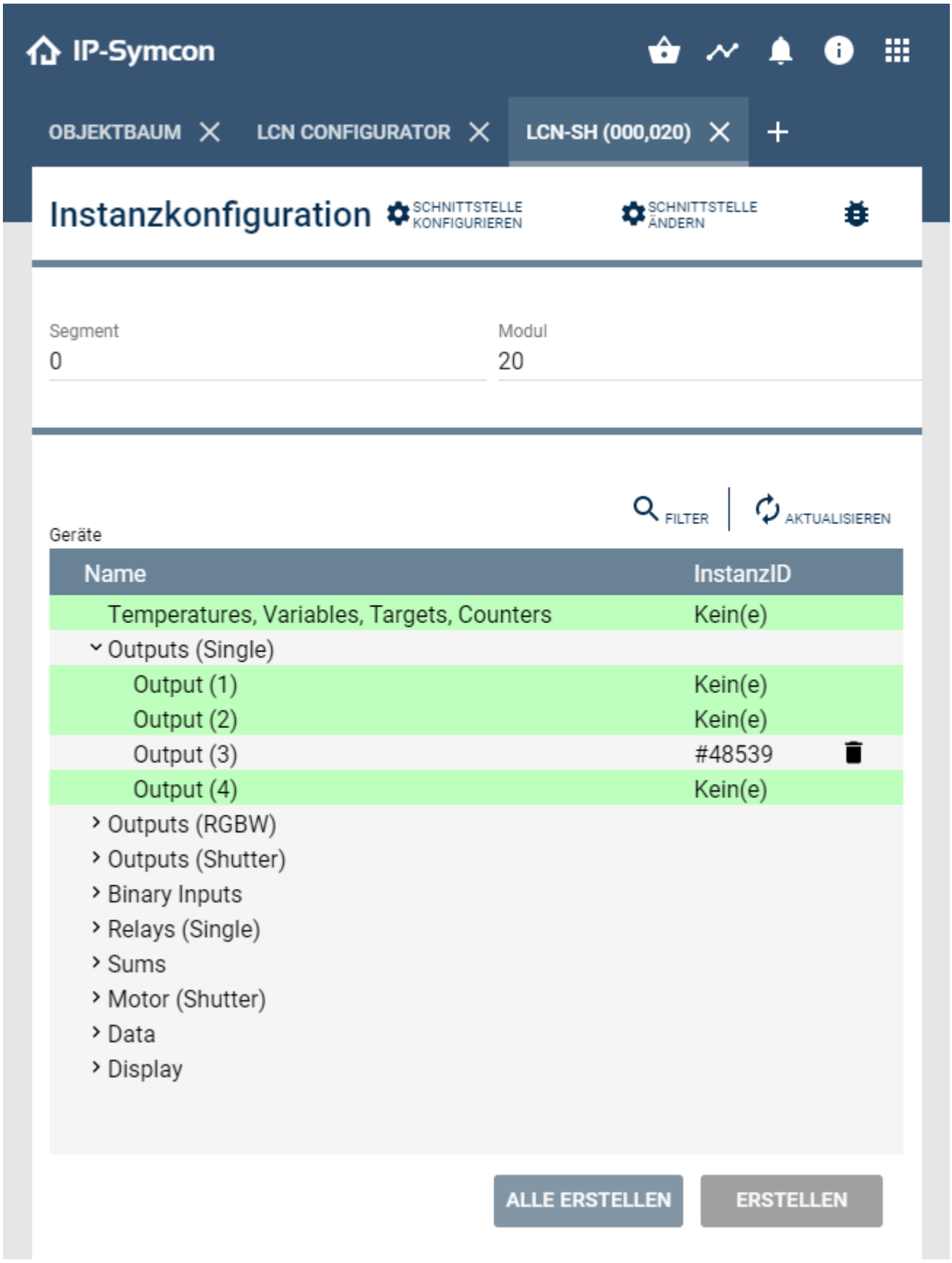


IPS-868 CONFIGURATOR

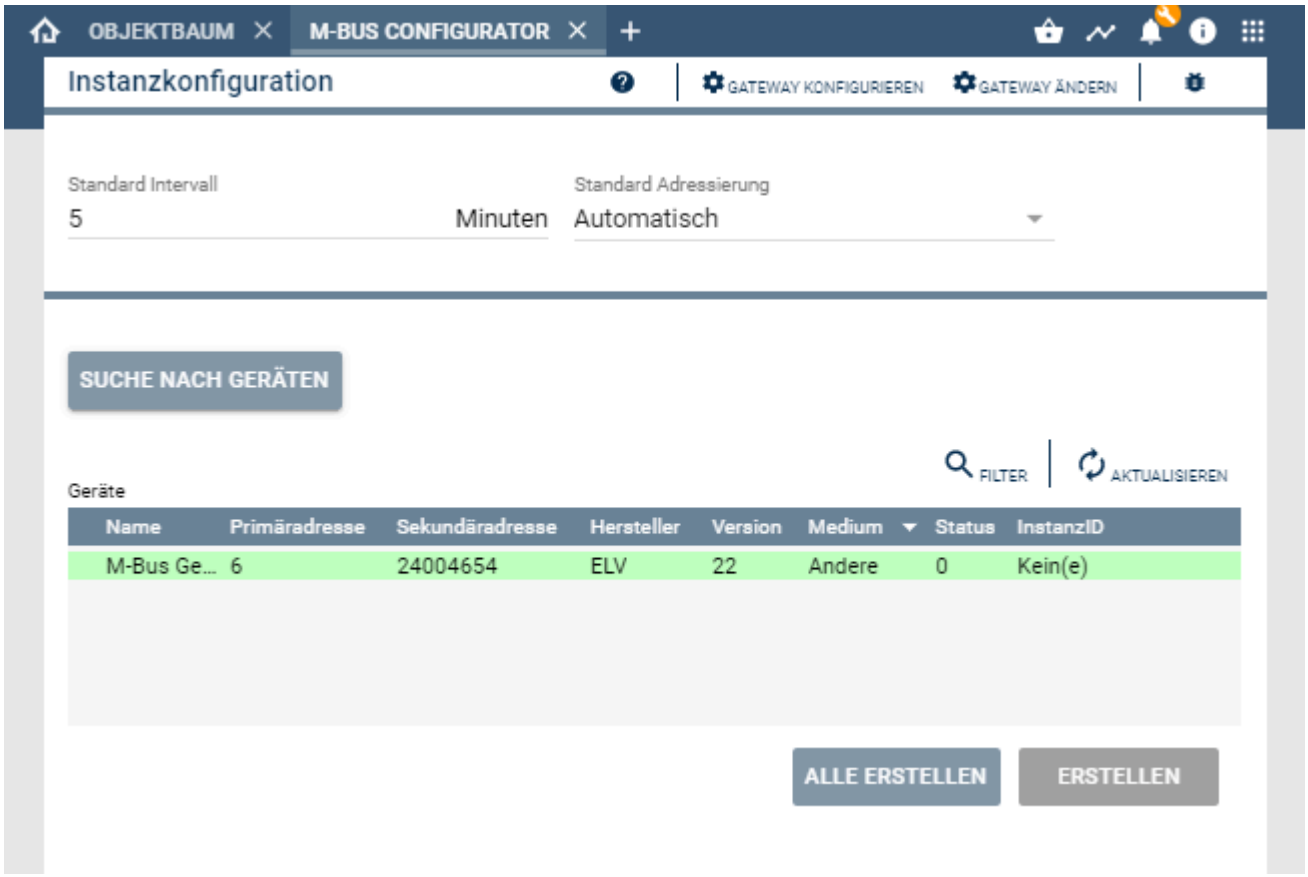


LCN CONFIGURATOR

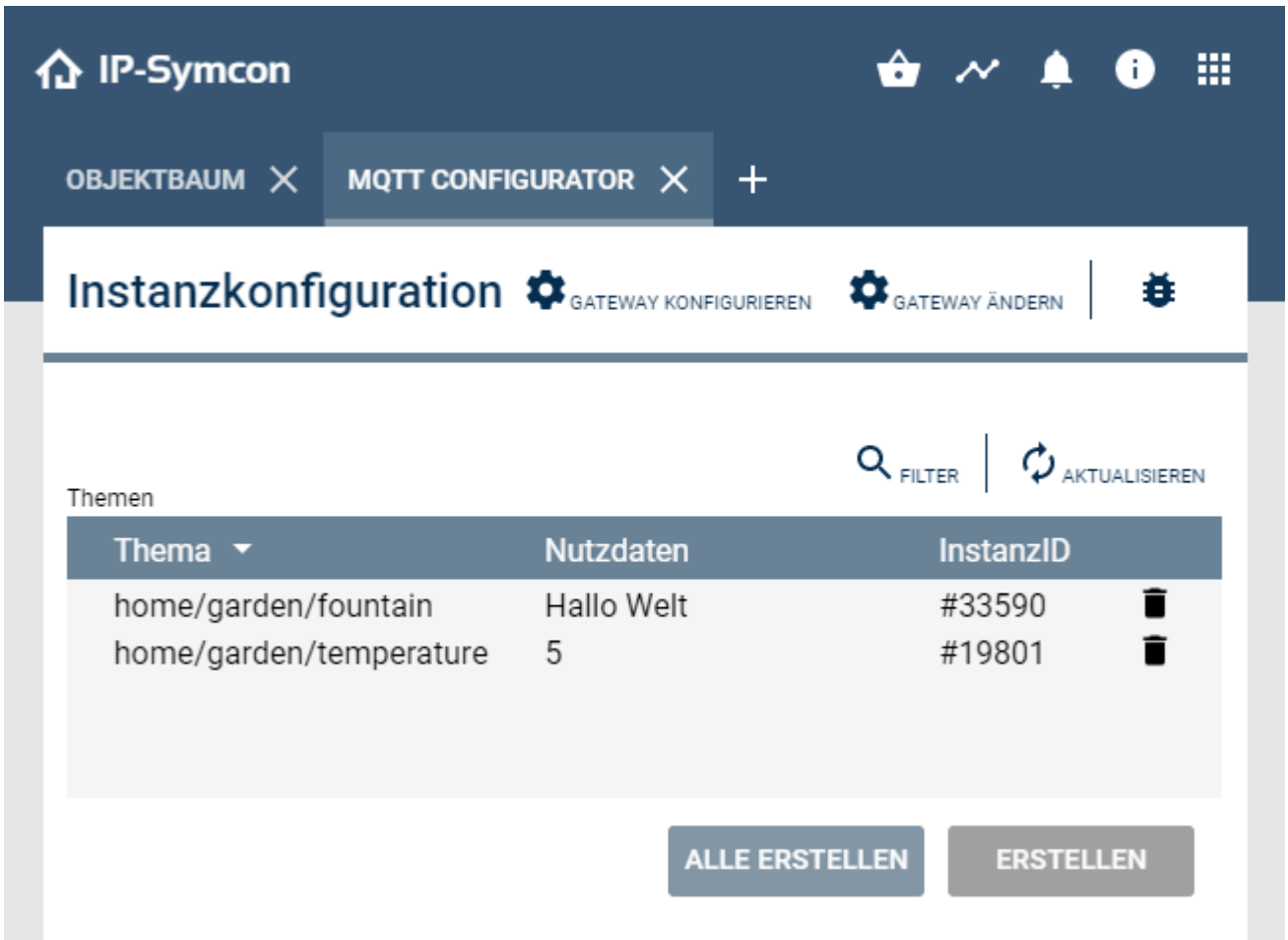

IP-Symcon
OBJEKTBAUM
LCN CONFIGU...
LCN GATEWAY
CLIENT SOCKE...
Instanzkonfiguration
GATEWAY KONFIGURIEREN
GATEWAY ÄNDERN
Segment
0
Nach dem Scannen der Module bitte 30 Sekunden warten und auf Aktualisieren klicken
MODULE SCANNEN
FILTER
AKTUALISIEREN
Geräte
Segment
Modul
Name
InstanzID
0
20
LCN-SH (000,020)
#50815
0
5
WolfWare
Kein(e)
ALLE ERSTELLEN
ERSTELLEN



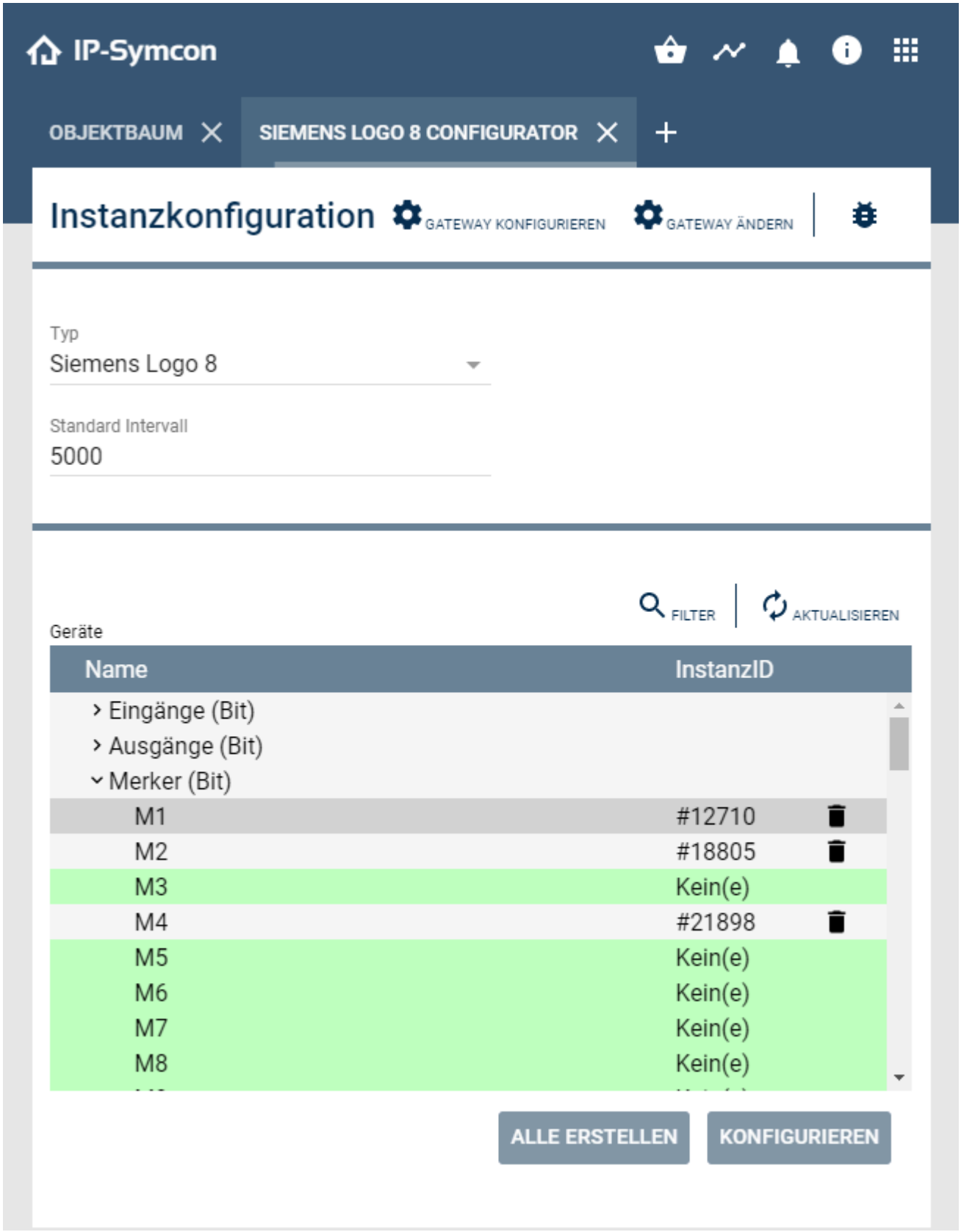
M-BUS CONFIGURATOR



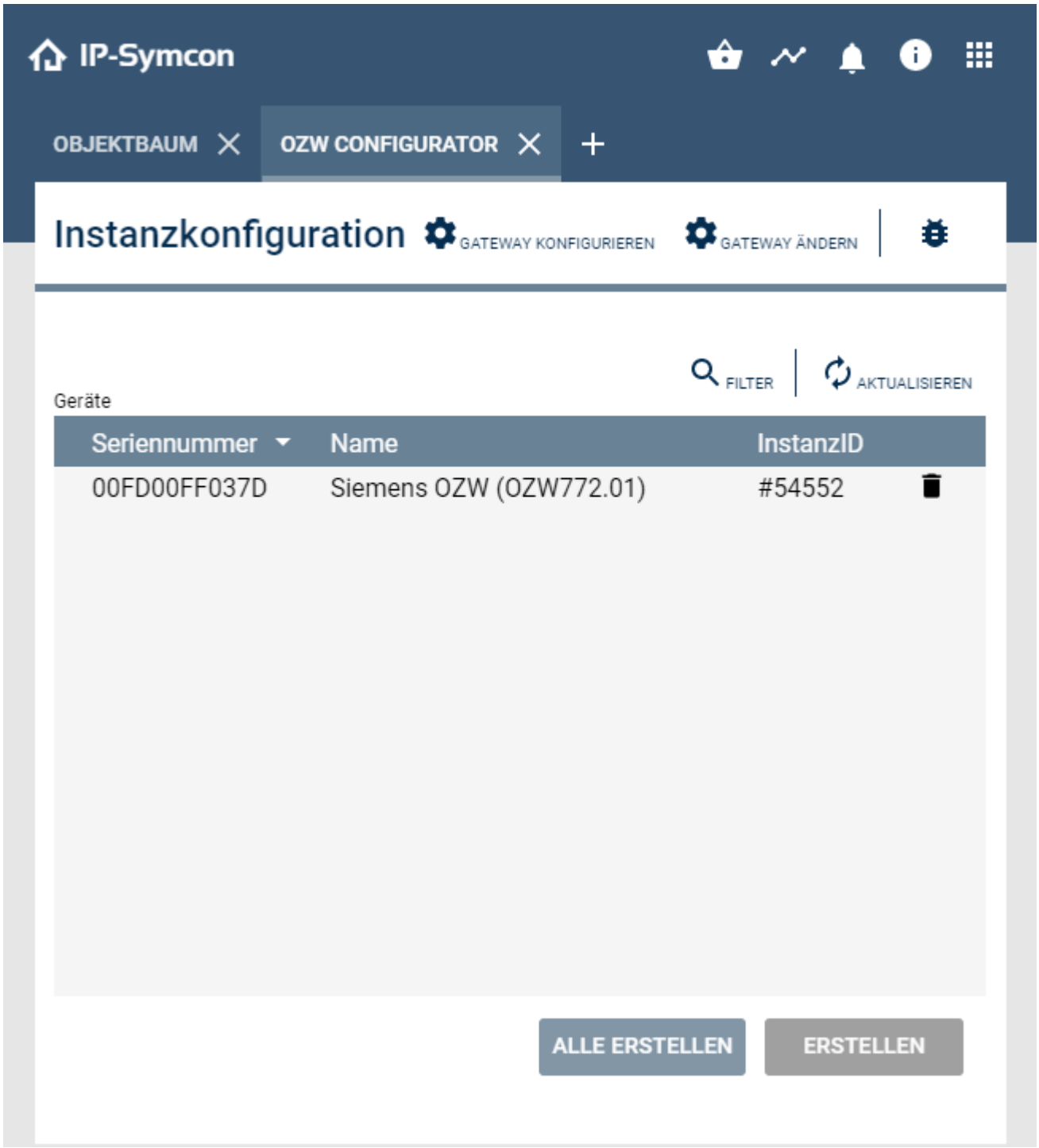
MQTT CONFIGURATOR



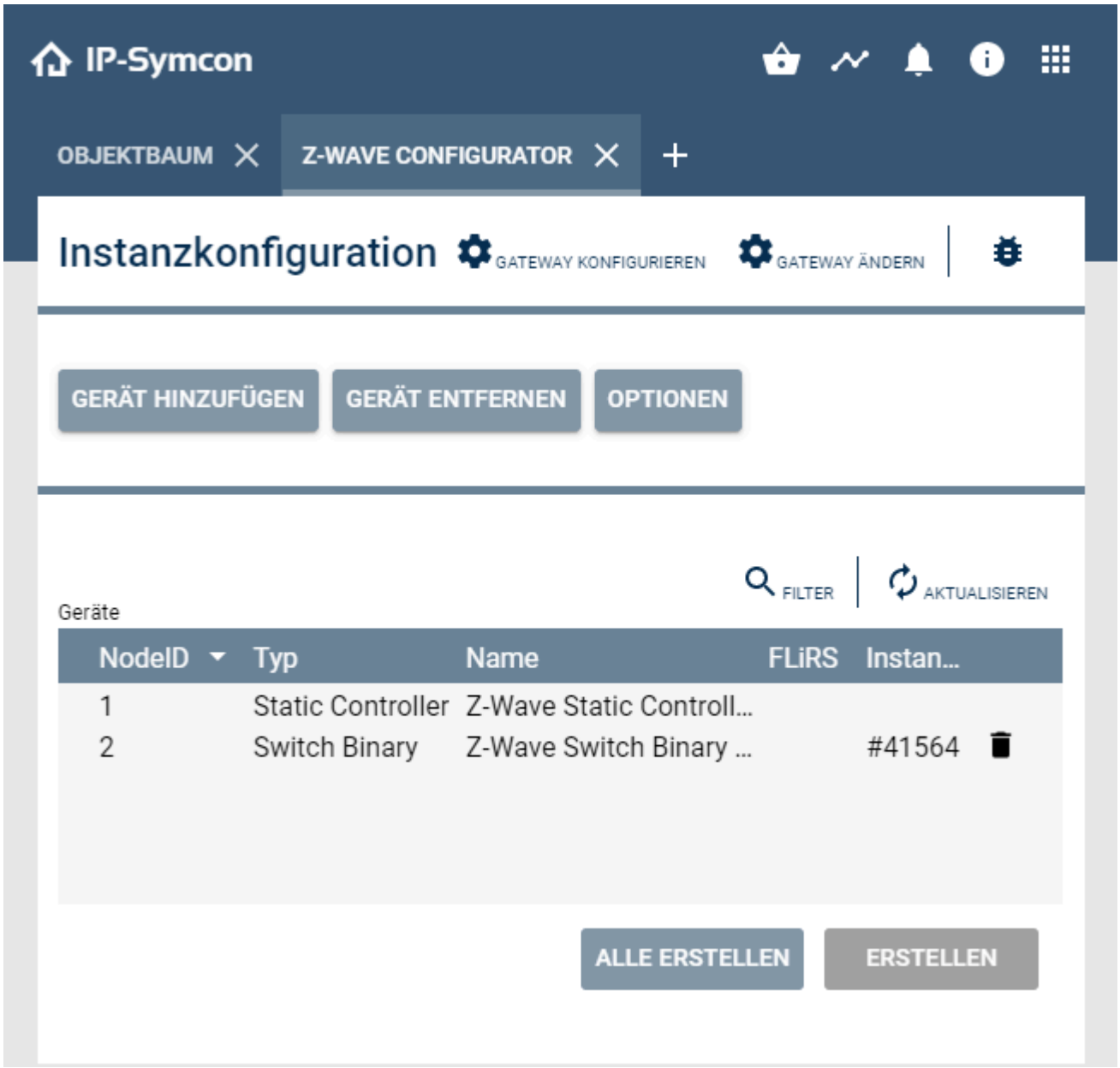
SIEMENS LOGO



SIEMENS OZW CONFIGURATOR



Z-WAVE CONFIGURATOR


IP-Symcon
OBJEKTBAUM
Z-WAVE CONFIGURATOR
Instanzkonfiguration
GATEWAY KONFIGURIEREN
GATEWAY ÄNDERN
GERÄT HINZUFÜGEN
GERÄT ENTFERNEN
OPTIONEN
FILTER
AKTUALISIEREN
Geräte
NodeID
Typ
Name
FLiRS
Instan...
1
Static Controller
Z-Wave Static Controll...
2
Switch Binary
Z-Wave Switch Binary ...
#41564
ALLE ERSTELLEN
ERSTELLEN

Variables

A variable is a container for a value. The value can be boolean, float, integer, or string. Usually, variables contain status values of devices. In addition, new user defined variables can be created to store individual values which could be used for visualization or scripts.



System Variables are no normal variables in the usual sense as they are only available within scripts.

IP-Symcon uses variables to exchange or store data. Variables that contain status values enable the use of devices, e.g., show the temperature of a thermometer in the visualization or activate and deactivate the lighting.

DESCRIPTION

There are two types of variables within IP-Symcon:

STATUS VARIABLES

Every instance of a device can have one or more status variables that are created automatically. These variables store the status of the device, e.g., on/off, temperature in °C, humidity, or battery.
The timing of updates and changes to a status or its value depend on the specific system. While a weather station periodically sends environmental data, the status of other systems like window contacts needs to be requested. The request interval is set in the configuration of the corresponding instance.
It is important to realize that status variables cannot be changed by the user. The variables merely reflect the status and datasets of the device.
[w]Do not delete status variables. Deleting can cause unpredictable behavior of an instance.
[/w]
Example 1:
When changing the wanted room temperature for a heating control system, a command is executed that sends the wanted value to the device. The status variable "Temperature" itself is not changed as it merely shows the current temperature. The required commands can be found in the corresponding Module Reference. Alternatively, the command "RequestAction " can be executed directly on the variable. IP-Symcon will check for the dedicated command by itself and execute it.
Example 2:
A more abstract example in form of a motor vehicle: To accellerate or decellerate the vehicle, the corresponding pedals need to be pushed. It is not sufficient to set the speedometer needle to the desired speed.



Conclusion: Status variables merely provide information about the device and are explicitely marked as "Read Only".

USER DEFINED VARIABLES

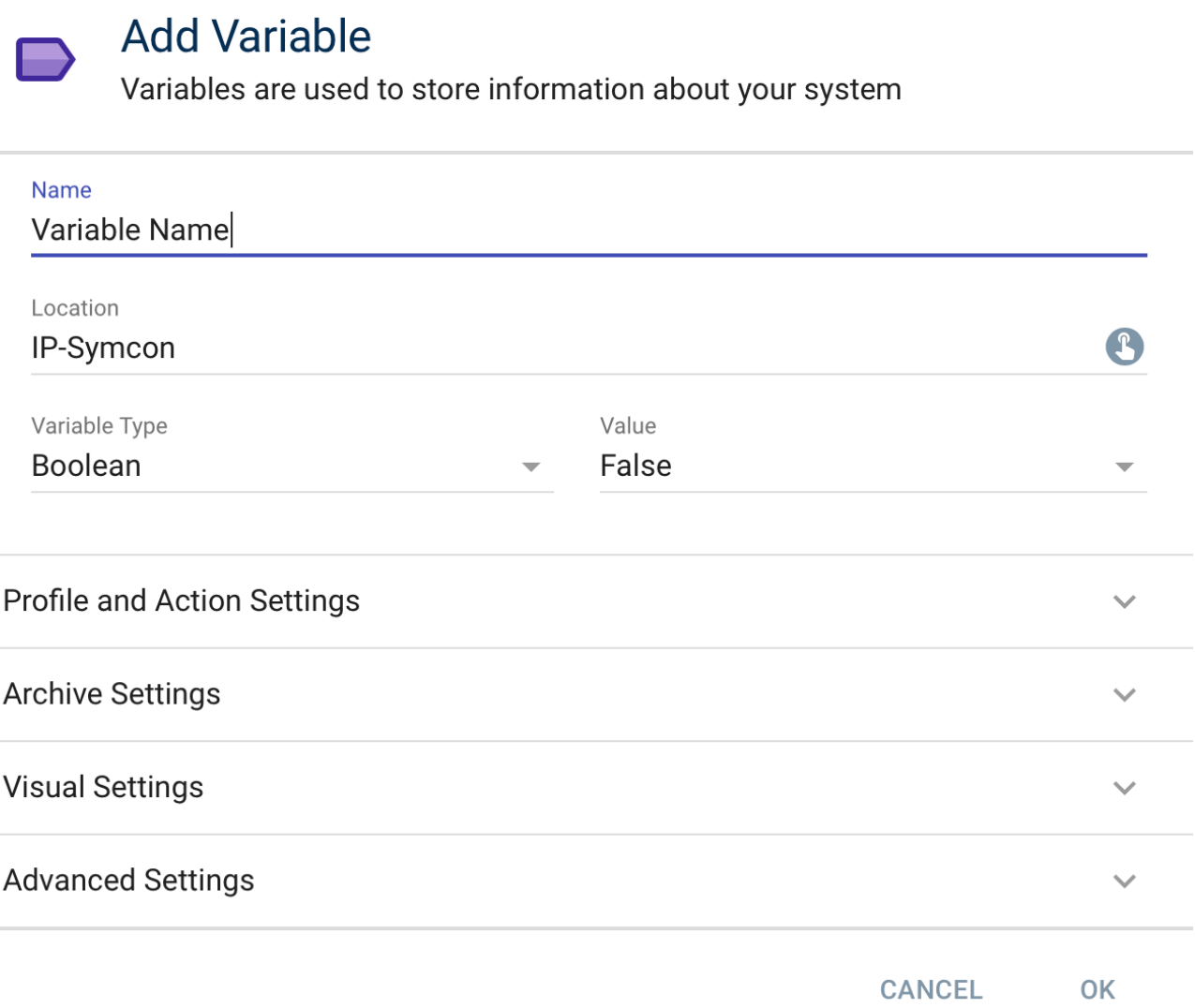
User defined variables can be created individually and contain one of four data types (see Table Variable Types). User defined variables can be manipulated by the user.
Example 1:
Integrating a previously unknown device, e.g., an AV amplifier, into IP-Symcon can be done with user defined variables.
Example 2:
A user defined variable can be used to convert data, e.g., the temperature in Celsius(°C) into Fahrenheit(°F).

VARIABLE TYPES

VARIABLE TYPE	DESCRIPTION	EXAMPLE
Boolean	True / False	On or Off
Float	Floating-point number	231.956
Integer	Whole numbers	-10, -4, 0, 32, 472
String	Text	“Hello IP-Symcon. Hello World”

CREATE NEW VARIABLES

A variable can be created in the object tree via "+" or "Right click" -> "Add object" -> "Variable". When using the second approach, the variable is created directly below the chosen object und does not need to be categorized afterwards. A variable type and a presentation are chosen in the following window. Logging can also be activated for the variable. The Aggregation decides the type of logging.
Before confirming the creation with "OK", the variable should have a meaningful name. Optionally, a description and an Icon can be chosen for the variable. The location of the variable is set with the dialog "Location" or can be changed afterwards via Drag and Drop in the object tree.

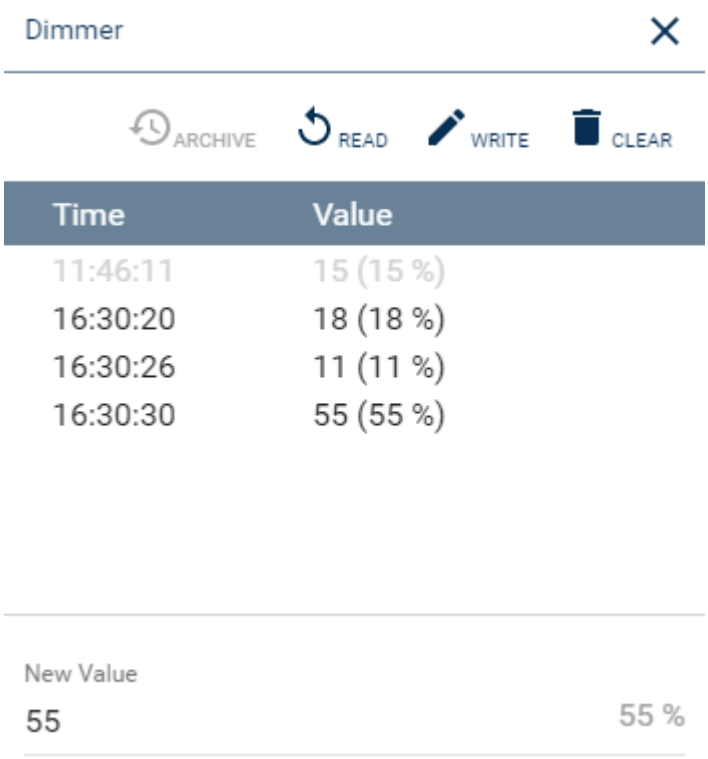


VISUALIZATION OF VARIABLES

The display of the variables in the visualization can be adjusted so that a variable is displayed as a selection or slider, for example. These options can also be changed at a later date. A more detailed description of the options can be found here: Variable visualizations

EDIT VARIABLE VALUES

The profile of a variable or its logging can be changed via double click on the name of the variable.
The "Surveillance Mode" can record the values of a variable or change it. It is accesed with a double click on the value of the variable. It is also possible to choose "Modify Variable" via context menu.
This function is meant primarily for testing. For example, it can be verified if the heating is activated on the simulated temperature or check the visualization of the variable in the WebFront.
The following picture shows the window for the "Surveillance Mode" that shows the ongoing changes to the variable and enables the modification. A modification can be repeated by clicking the button "Write".



The value of a status variable can be modified even though it is marked as "Read Only". When modifying the value all corresponding Events are executed. The modification can be used to test user defined scripts but does not change the actual values of the physical device.

VARIABLE ACTIONS

A Variable Action can be assigned to a variable via "Custom Action". It is called when the variable is clicked in the visualization, e.g., WebFront or a mobile app. In addition, the variable action is executed if the variable is switched, e.g., via the function RequestAction .

Variable Actions

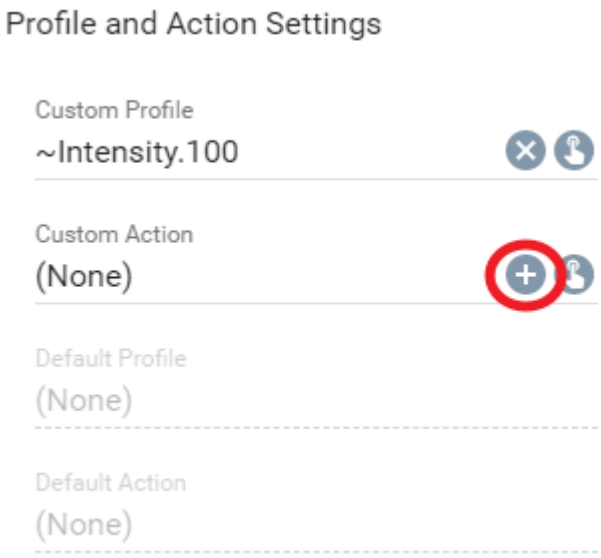
A variable action is executed when a variable is clicked via visualization.
The prepared Action Script is executed. Additional data like the new "desired" value and the ID are passed via System Variables.

CHOOSE VARIABLE ACTION

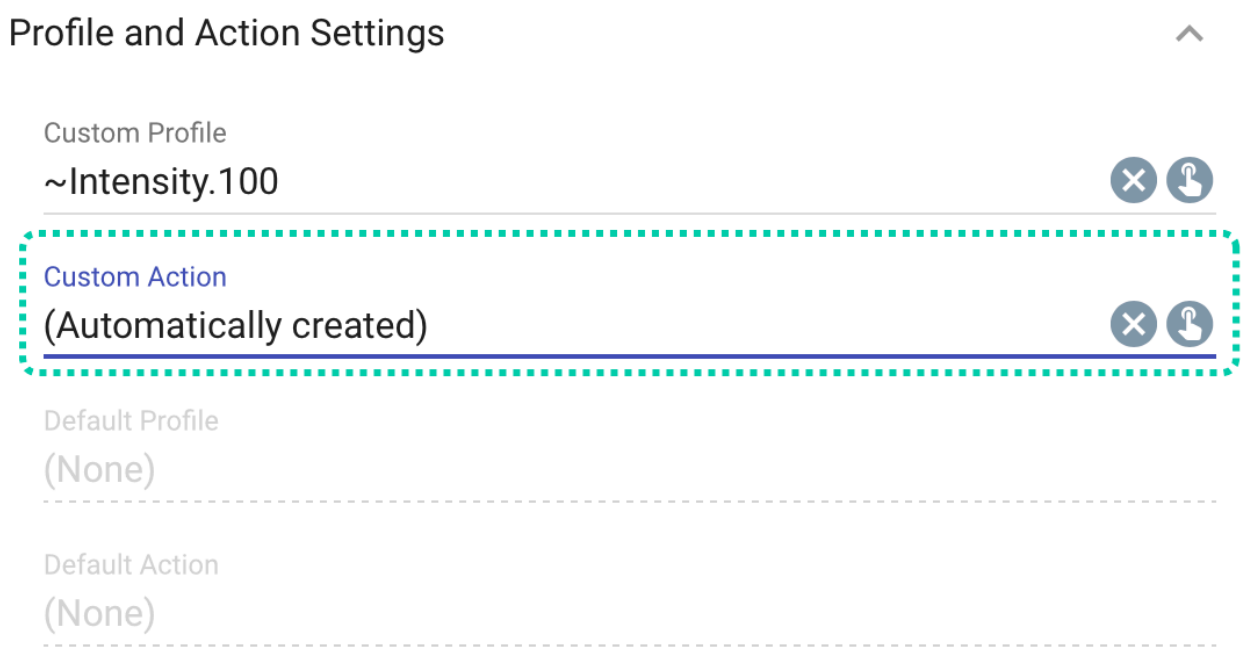
It is possible to choose an action with the "Edit variable" dialog by choosing "Custom action". The dialog "Edit Variable" is accessible when creating a variable, via double click on the variable within the object tree, or via right click on the variable and choosing "Edit object".



Some variables of added modules contain a "Default action". It can be overwritten with a "Custom action".



With the "+", a variable action can be created which only handles that the variable changes accordingly if it is clicked in the WebFront. This automatically created action script is displayed as "(Automatically created)"



CREATE VARIABLE ACTION

A chosen variable action is an action script.
For more information, refer to PHP Scripts and Action Scripts.

Variable Presentations

Require: Symcon >= 8.0
Each variable has a variable presentation. Variable presentations are used to convert variable values such as True, False or 0, 1, 2, 3 into a human-readable form and to provide them with the necessary context data (e.g. °C). The variable presentation therefore specifies how a variable is displayed in the visualization. It also determines the control element with which the variable value can be changed.

SELECT PRESENTATION

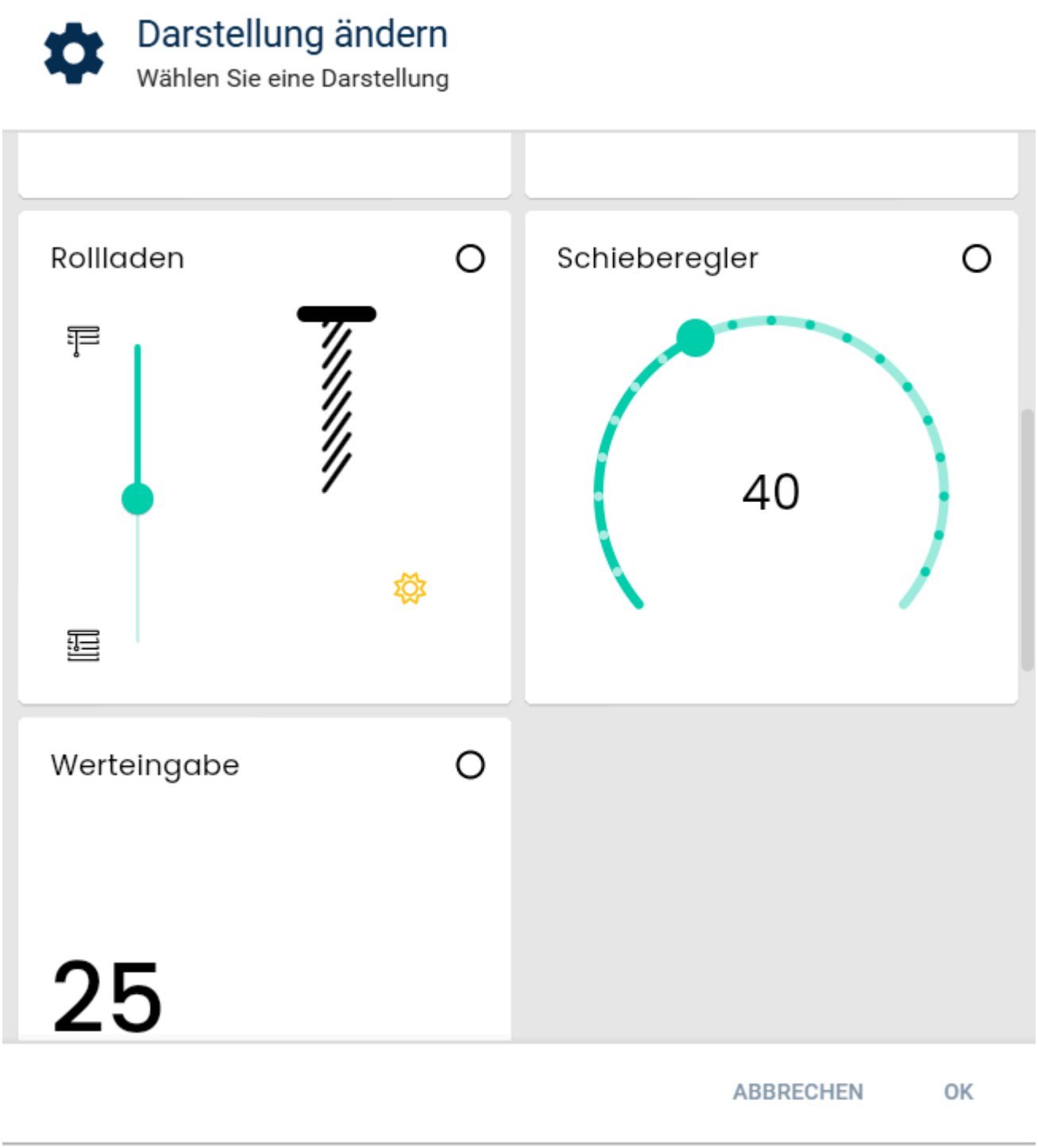
A presentation can be selected in the "Edit variable" dialog in the "Variable presentation" section. This dialog can be accessed when creating a variable or afterwards via "Object tree" -> "Double-click" or via "Object tree" -> "Right-click" -> "Edit object".



Some variables of added modules contain a "standard display". This can be overwritten by clicking on the "Overwrite default" button. The original status can then be restored using the "Reset to default" button.



The selection icon for the "Display" entry opens the display dialog in which a display can be selected. All presentations are always displayed in the presentation dialog. If the configuration of the current variable does not allow a display, it is grayed out. When you move the mouse over the grayed-out element, a text is displayed describing which requirements must be met in order for the presentation to be displayed



Each display offers many different parameters to customize it according to the requirements. The available parameters and their function are described in more detail under the respective entries under Object presentation.

PREVIEW

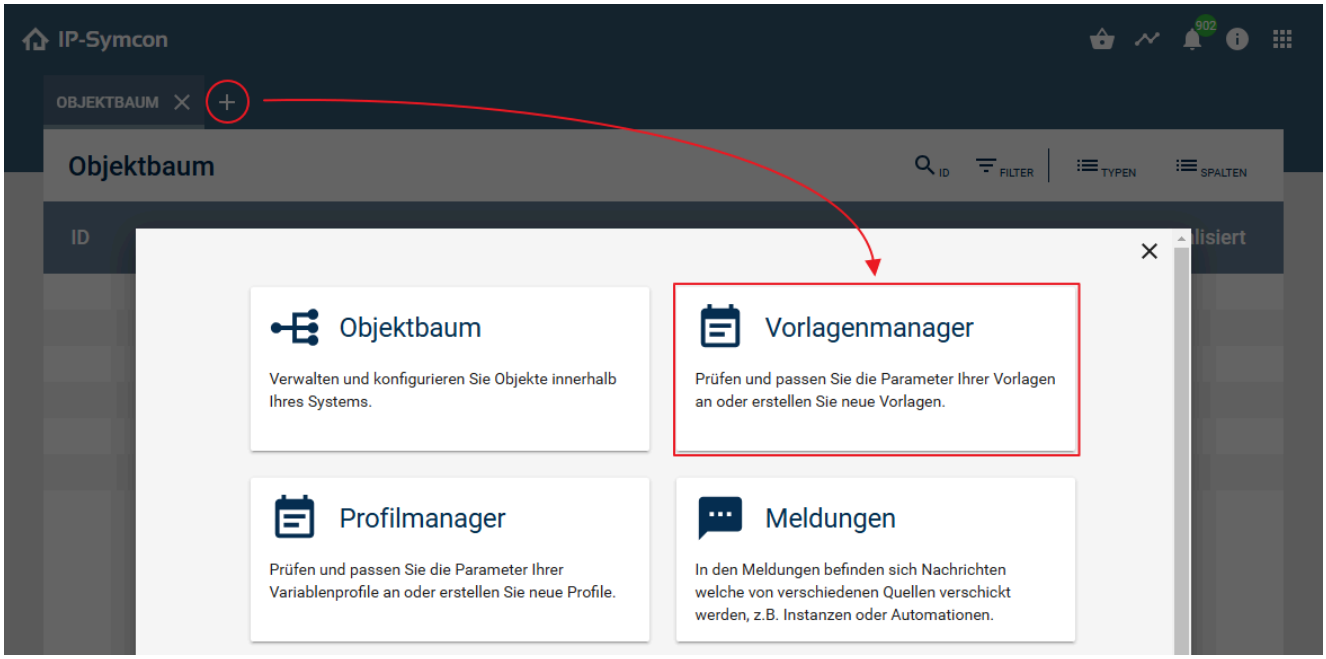
Each adjustment can be observed directly in the preview next to the display selection. The preview can be opened in fullscreen by clicking on the magnifying glass icon. Here, as in the normal visualization, the size of the tile can be adjusted using the pencil icon to see how the display behaves in different sizes.

TEMPLATES

If the parameters of a display are changed, these only apply to the currently selected variable. If several variables are to use the same parameters, templates can be used. Templates can be selected under the field for presentations. If a template has been selected, the presentation uses the parameters of the template, even if these are adjusted later. However, parameters can still be changed in the variable dialog. If the parameters no longer correspond to the values of the template or the higher-level presentation, "(Custom)" is displayed in the Template field and the set parameters apply exactly to this variable again. Custom templates can be created and edited in the Template manager.

Template Manager

Require: Symcon >= 8.0
The template manager can be opened via the "+" in the table list.

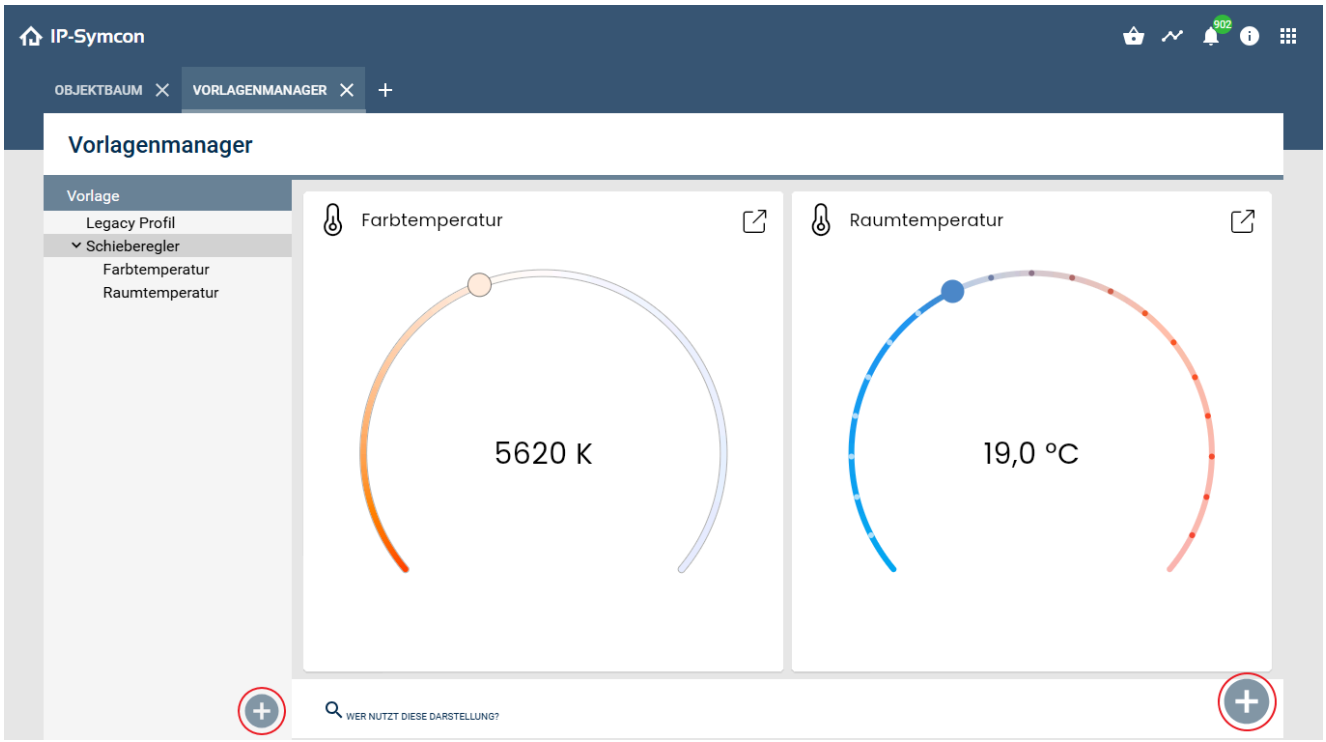


STRUCTURE

The template manager consists of a tree on the left-hand side, in which templates can be selected, grouped according to presentations, and a central detail area, which adapts to the current selection in the tree. If no entry is selected, all presentations are displayed as a preview in the detail area. As an alternative to the selection in the tree, a specific presentation can be opened here by clicking on the "Open" symbol at the top right of each tile. If a presentation is selected, the detail area shows a preview for each template of this presentation and also enables direct opening here. The "Who uses this presentation?" button at the bottom of the detail area displays all variables that use this presentation. Once a template has been selected, a corresponding configuration form appears, similar to the variable configuration form, in which the parameters of the template can be configured. The name of the template can also be customized. As each template is identified in the background by a unique ID and not the name, the name can be changed without hesitation. It only serves to make it easier for the user to understand. In the lower area, a list of variables that use this template can be displayed via "Who uses this template?". A new template can be created via "Duplicate", which is initially configured with the same parameters.

Information about the individual parameters of the presentations can be found in the Object presentations.

CREATE TEMPLATES



A template can be added via the "+" in the list on the left-hand side. The desired display is selected in a dialog and the template is then configured. If a display is selected, you can alternatively click on the "+" on the right-hand side to directly create a template for the selected display.
To build on an existing template, select it and click on "Duplicate" at the bottom.

DELETE TEMPLATES

User-defined templates can be deleted via the trash can icon next to the name in the list on the left-hand side. The templates provided by IP-Symcon cannot be deleted.

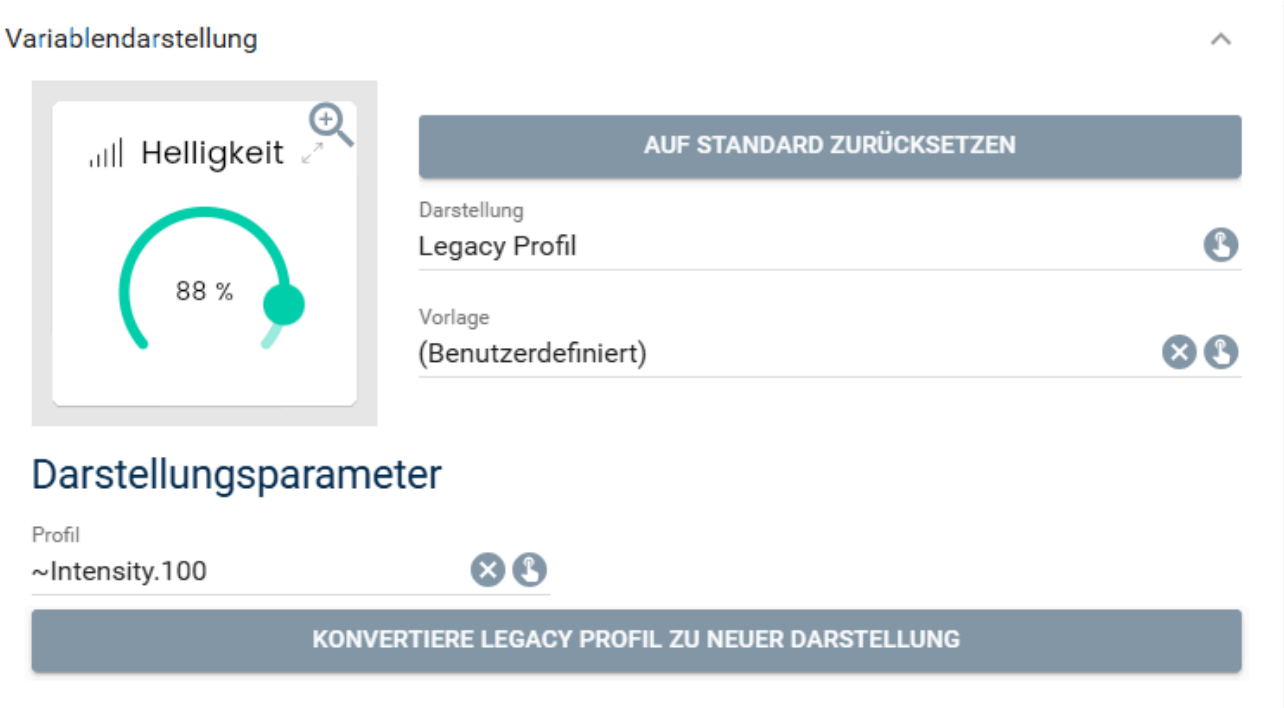
Variable Profiles

Variablen Profiles are still supported. However, version 8.0 introduced Variable Presentations as modern and flexible way to configure the presentation of variables. Variable Profiles can still be used via the presentation Legacy Profile.

Every variable can have a variable profile. Variable profiles are primaryly used to make variable values like True, False, or 0, 1, 2, 3, ... easier readable by adding context, e.g., adding °C to temperatures. Thus, the variable profile defines how a variable is visualized.

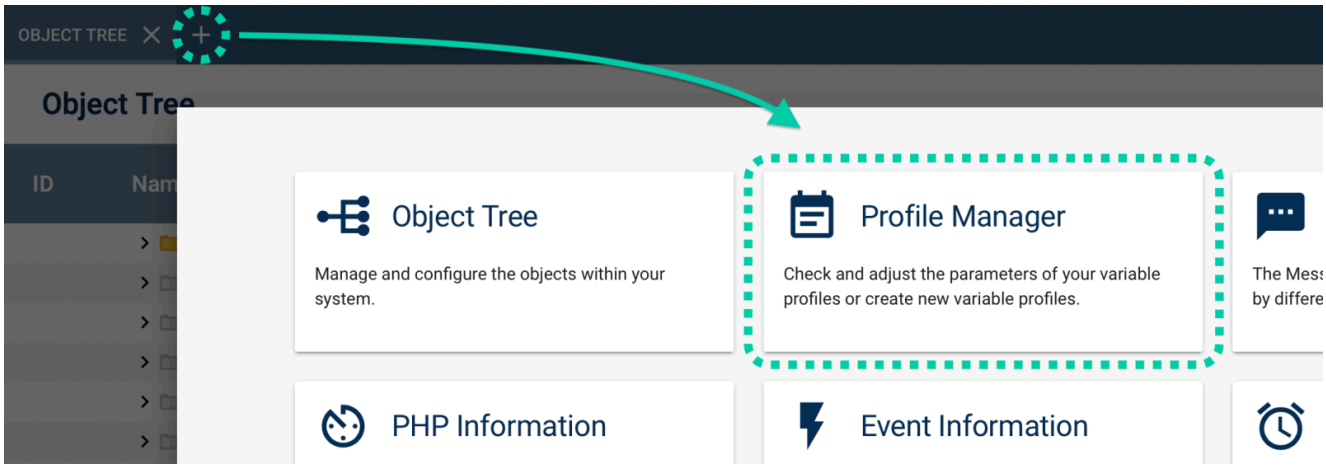
CHOOSE PROFILE

Starting from version 8.0, a variable profile is assigned via the variable presentation Legacy Profile. The dialog "Edit Variable" is accessible when creating a variable, via double click on the variable within the object tree, or via right click on the variable and choosing "Edit Object".
Within the "Variable presentation" section, first select the presentation "Legacy Profile". Afterwards, the desired profile can be chosen under "Presentation parameters".



CREATE PROFILE

If no included profile suits a variable, it is possible to create individual profiles. A created profile can be used for any number of variables.
To edit a variable profile, the profile manager needs to be opened. It can be opened via "+" -> "Profile Manager" at the tab list or via the "Edit Variable" dialog using the finger button next to the profile selector under "Presentation parameters".



It is possible to choose a pre-defined profile, to duplicate a profile, or create a new one. A new profile is created by clicking the "+" at the bottom left. Afterwards, the profile needs to be named. After saving the profile, it will appear in the list.



Even though the variable profiles are differentiated after type (Boolean, Integer, Float, String), a profile name needs to be unique

Examples for different visualizations of variable profiles in the WebFront are available at Object Depiction

Included profiles can be recognized by the tilde (~) in front of the profile name. These profiles are created from the system and cannot be edited. The tilde must not be used in individual profile names.

The following changes to a profile are possible:

PARAMETER	TYPE	DESCRIPTION
Prefix	All types	String in front of the current value, e.g., a fixed description
Suffix	All types	String after the current value, e.g., °C; Special case: When using % the value is converted into a percentage by using Min/Max. (Value = (Value – Min) * 100 / (Max – Min))
Min Value	Integer, Float	Smallest possible value of the variable for the visualization
Max Value	Integer, Float	Biggest possible value of the variable for the visualization
Step Size	Integer, Float	The step size determines the number of clickable selections within the visualization. Consider a profile with Min = 0, Max = 100. A step size of 25 would make the values 0, 25, 50, 75, 100 available. If the step size is 0 and assoziations are available, all assoziation texts are listed. In this case, the usually shown arrows for selection are not available. This field is only evaluated if an action or an action script is assigned to the visualized variable.
Digits	Float	Defines the number of decimal places.
Default Icon	All types	If no icon is available via assoziations the default icon is used. If the parameter is not set, the object Icon is used. All available icons are listed here: WebFront Icons

PARAMETER	TYPE	DESCRIPTION
Associations	All types	**Boolean:** A text and an icon can be chosen for each possible value (True, False) which is shown instead of the real value. This only affects the visualization. The value of the variable remains the same. **Integer/Float**: 1. Like Boolean, it is possible to choose a text and an icon. In the simplest case, a text and an icon needs to be defined for each value. 2. The second possibility is to omit values and only choose a text and an icon for some values. In that case, the visualized text and icon correspond to the next lowest assoziated value. For example, if the assoziated values are 0, 50, and 100, the assoziations would refer to the following values: 0(0-49), 50(50-99), and 100(100). 3. It is possible to generate an expression that contains the current value. The placeholder %d (int) or %f (float) are used as placeholder for the current value. For example would be "LightValue%d" shown as "LightValue4". **Remark:** If one of the two parameters (Text / Icon) is left empty, the normal value or the default icon are used. To show an empty text, you need to use a space character. **String (since version 6.0):** A text and an icon can be chosen for each String value. The order of associations is based on the order of creation. Sorting is not possible. Thus, putting new associations in order requires the deletion and recreation of all following associations.



Profiles are only used for visualization. If the current value of a variable is outside of the interval between Min and Max or is not described by the specified assoziations, it is possible that the output of the variable is wrong or is not shown at all.



The maximum number of simultaneous associations per profile is 128.

EXAMPLE

When Min=0 and Max=20 is defined in the profile and the suffix % is used, but the variable value is 40, the shown visualizaion is 200%. The values for Min/Max do not limit the variables!



Depending on the variable type the used placeholder is %d or %f. The number of decimal places can be influenced by using the following format: %.2f for two decimal places.



To use a percentage symbol (%) within an assoziation text, a double percentage symbol (%%) needs to be entered.



A possibility to use the variables in a formatted way in scripts is the function GetValueFormatted .

Automations

Automations are used to define individually adjusted behavior. IP-Symcon supports flow scripts and PHP scripts. Flow scripts use integrated actions that can be clicked together easily. A PHP script is the option for experts and uses the programming language PHP.
Automations in IP-Symcon:

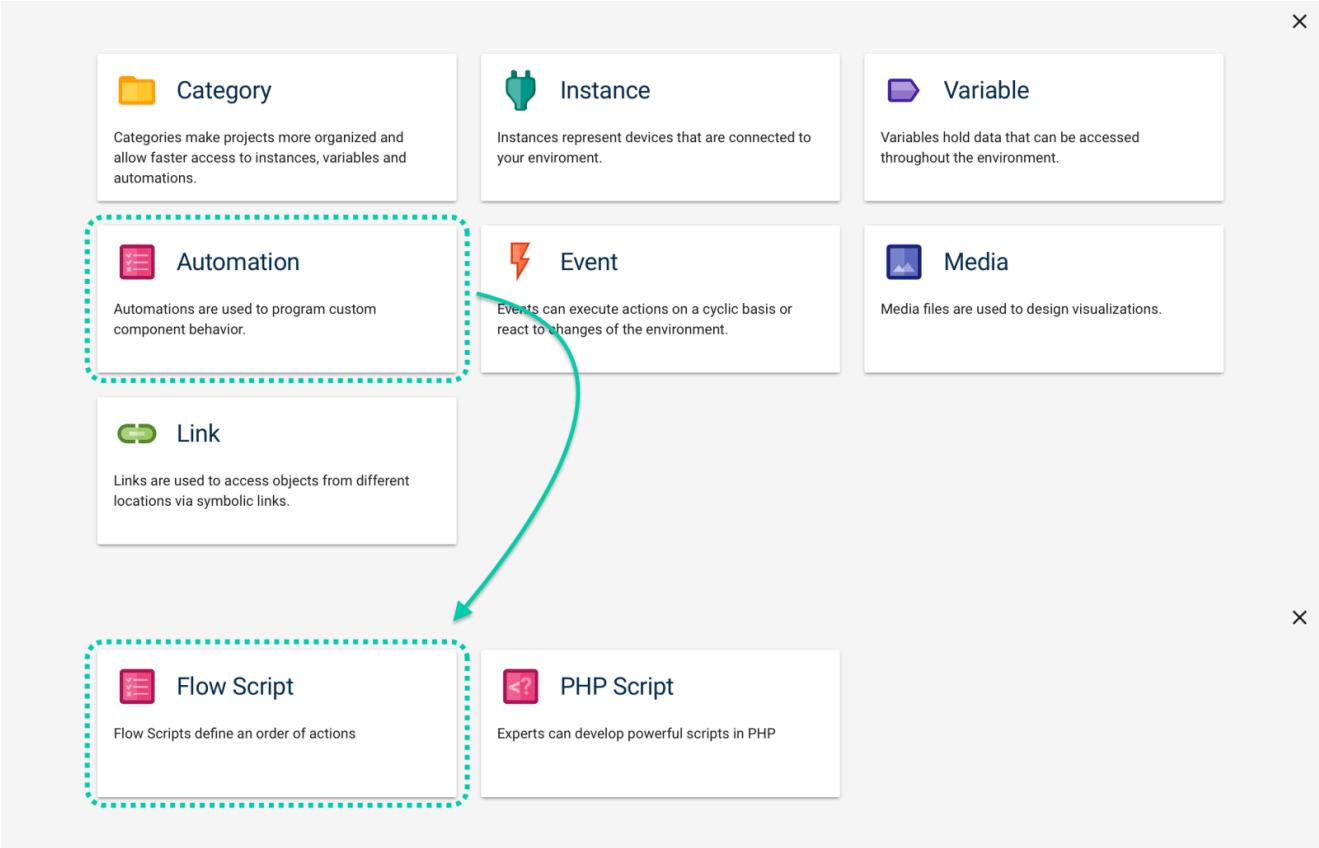
Flow Scripts
PHP Scripts (for experts)

Flow Scripts

A flow script is a sequence of Actions . These can be used without a single line of source code and thus are useable without any programming knowledge.

ADD FLOW SCRIPT

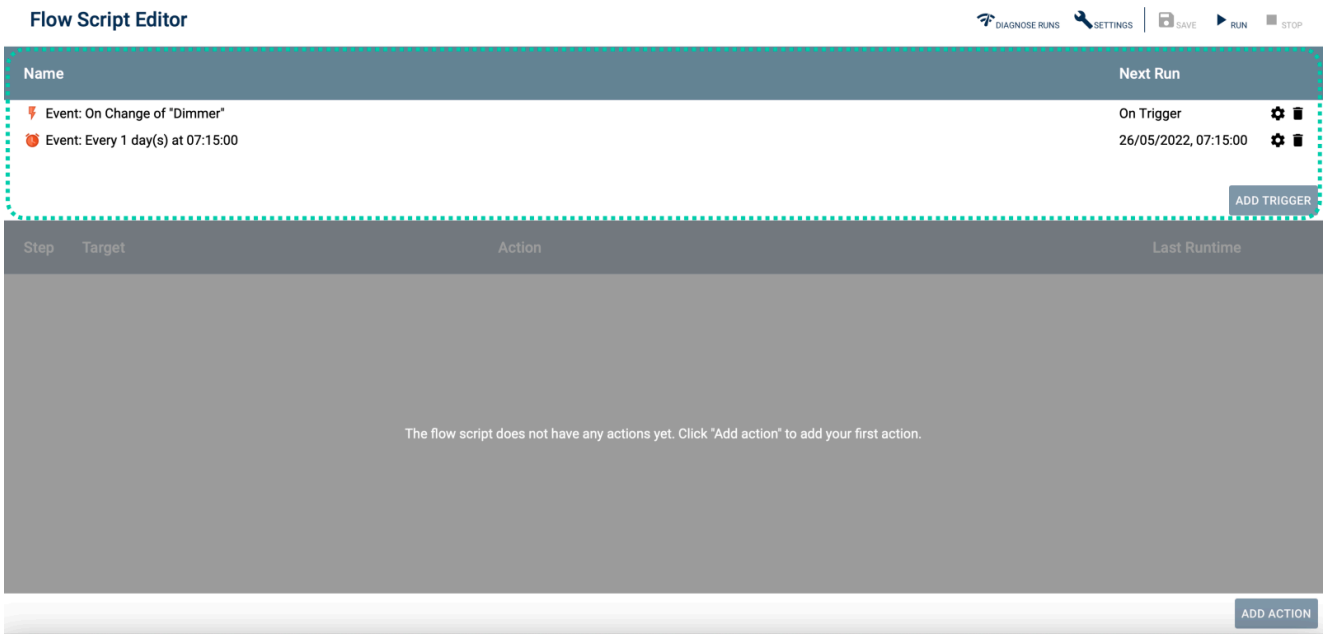
A new flow script can be added via "+" -> "Automation" -> "Flow Script". It is recommended to give the flow script a unique and meaningful name.
The name can be changed at any time and the flow script can be moved within the tree view via Drag & Drop. Alternatively, the flow script can be directly created at the wanted position via "Right click" -> "Add Object" -> "Automation" -> "Flow Script".



ADD TRIGGER

A trigger is an event and defines when a flow script is executed. The specific configuration is explained at Events.

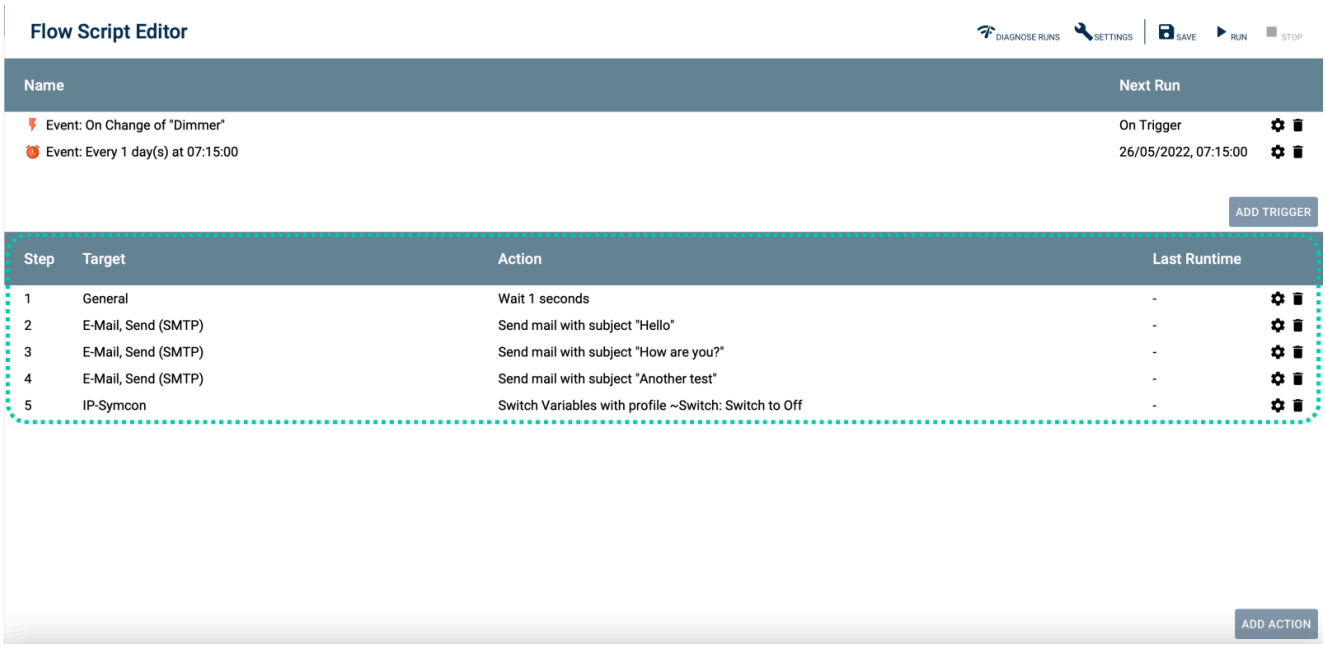
COLUMN	DESCRIPTION
Name	The name of the trigger in the object tree
Next Execution	When or how the trigger is triggered
Gear	Open the configuration for the trigger
Garbage Bin	Delete the trigger

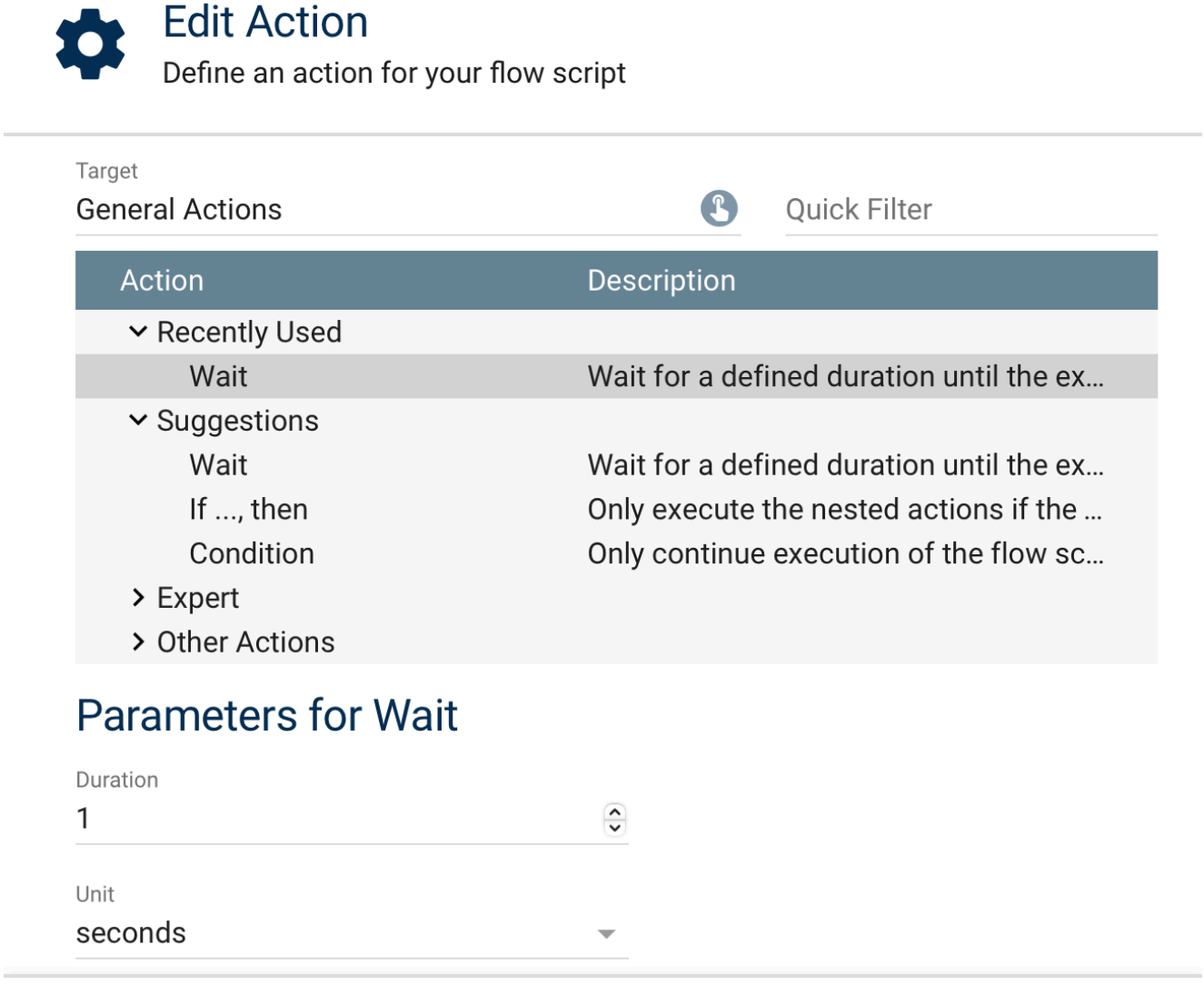


ADD ACTION

An action is specific for a selected target. An action can trigger a multitude of processes, e.g., switching a device, a condition for further subactions, or a break in the flow that is continued after some time. As such, complex processes can be constructed without a single line of source code.

COLUMN	DESCRIPTION
Step	The position of the action within the flow script. Some actions have substeps
Target	Displays the target of the action. It can be an object or a general action, e.g., waiting or a condition
Action	A description of the action to be executed
Last Runtime	The runtime of the step during the last execution of the flow script
Gear	Open the configuration of the action
Garbage Bin	Delete the action


Flow Script Editor
DIAGNOSE RUNS
SETTINGS
SAVE
RUN
STOP
Name
Next Run
Event: On Change of "Dimmer"
On Trigger
Event: Every 1 day(s) at 07:15:00
26/05/2022, 07:15:00
ADD TRIGGER
Step
Target
Action
Last Runtime
1 General Wait 1 seconds -
2 E-Mail, Send (SMTP) Send mail with subject "Hello" -
3 E-Mail, Send (SMTP) Send mail with subject "How are you?" -
4 E-Mail, Send (SMTP) Send mail with subject "Another test" -
5 IP-Symcon Switch Variables with profile ~Switch: Switch to Off -
ADD ACTION


Edit Action
Define an action for your flow script
Target
General Actions
Quick Filter
Action
Description
Recently Used
Wait
Wait for a defined duration until the ex...
Suggestions
Wait
Wait for a defined duration until the ex...
If ..., then
Only execute the nested actions if the ...
Condition
Only continue execution of the flow sc...
Expert
Other Actions
Parameters for Wait
Duration
1
Unit
seconds
CANCEL
OK

If a red exclamation mark appears in the list, the last execution was erroneous. The error message can be displayed with a click on the exclamation mark.

CONTEXT MENU

A right click on the list of actions shows additional options in a context menu.

LIST	DESCRIPTION
Add Action before/after	An action is added before or after the clicked row.
Duplicate	The clicked action is duplicated, with subactions if applicable. The duplicated action is position after the original one.

GENERAL ELEMENTS

MENU ENTRY	DESCRIPTION
Diagnose Runs	Opens a dialog that is used for an analysis of previous executions. Erroneous, aborted and succesful actions are listed here, see below
Settings	The behavior of the flow script can be defined here, see Behavior during Execution and Behavior on Error
Save	Save the current configuration of the flow script
Run	Save and run the flow script
Stop	Stop the current execution of the flow script

ANALYZE EXECUTION

The execution of a flow script goes through different status. These are described textually and coded with a color. In addition, a single execution can be expanded. This displays the actions that were executed during that run.

STATUS	DESCRIPTION
Queued	The execution is still in the queue (Color: White)
Running	The execution is still running (Color: White)
Aborted	The execution was aborted (Color: Yellow)
Done	The execution finished succesfully (Color: Green)
Done (Error)	The execution finished but some steps had errors (Color: Yellow)
Error	The execution was aborted due to an error (Color: Red)
Skipped	Is caused by "Reject further execution when running" (Color: Yellow)

BEHAVIOR ON RUNNING EXECUTIONS

This option describes the behavior if the flow script is executed again while another execution is still running.

BEHAVIOR	DESCRIPTION
Reject further execution when running	New executions are ignored while another execution is still running
Abort current execution on new execution	The current execution is aborted on a new execution
Add execution into queue	Multiple executions of the flow scripts are added to a queue and executed one after the other.

BEHAVIOR ON ERROR

BEHAVIOR	DESCRIPTION
Abort remaining steps on Error	On an error, the whole execution is aborted
Continue remaining steps on Error	On an error, the execution is continued
Retry this step on Error	On an error, the erroneous step is repeated up to the defined number of retries or until succesful. The option "Continue execution even if all retries have failed" defines whether the step is skipped on retries without success or if the whole execution is aborted

Actions

An action is a command that can be used without complex programming or a script. Depending on a selected target, multiple fitting actions are listed. Each action provides individual configuration and options.

ADD ACTION

Actions are used in Flow Scripts, Events, Instance Configuration and the "Test Commands" Dialog. How they are added and configured is explained in the corresponding section. However, all possibilities are configured in the same way:

Step 1: Select a target - This loads and displays possible actions
Step 2: Selecting an action - The actual selection of an action (Quick Filter Search is possible)
** The actions are sorted into different categories and thus enable a quick access
Step 3: Configuration of Parameters - The actual values for the action are set here

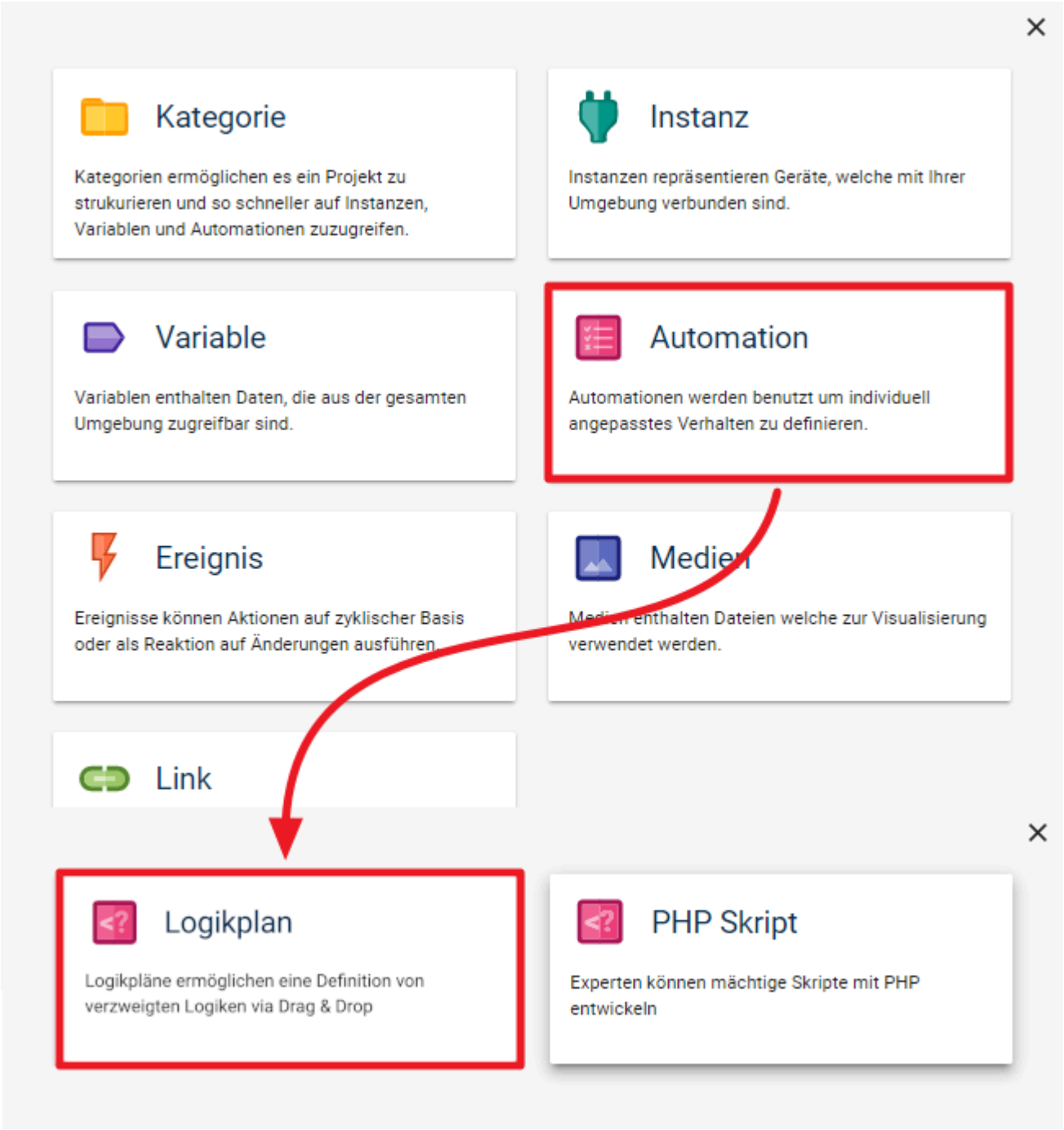
Logic Scripts

Require: Symcon >= 7.0
Logic Scripts is a tool that can be used to create individual logic scripts and automations in IP-Symcon. Typical use cases are

- Group switches for light
- Staircase automation
- Sensor evaluation

ADD LOGIC SCRIPT

A new logic script can be added via "+" -> "Automation" -> "Logic Script". It is recommended to give the logic scipt a meaningful name.
The name can be changed at any time and the logic script can be moved to a different position in the object tree using drag & drop.
Alternatively, the creation can be done directly at the desired position via "right click" -> "Add object" -> "Automation" -> "Logic Script".



COMPONENTS OF THE EDITOR

The logic script editor is divided into the following areas:

NAME	DESCRIPTION
Menu	The menu provides central functions such as saving, zooming or executing a logic script.
Panels	The Logic Script Editor provides 2 panels: - The repository with the list of all available modules - The PropertyEditor for displaying and editing properties Use drag & drop to change the position of the panels; 4 predefined positions are available. If you place several panels in one position, tabs are automatically created.

NAME	DESCRIPTION
Drawing area	The drawing area is the central component of the editor and allows you to place and edit the individual modules.

Menu

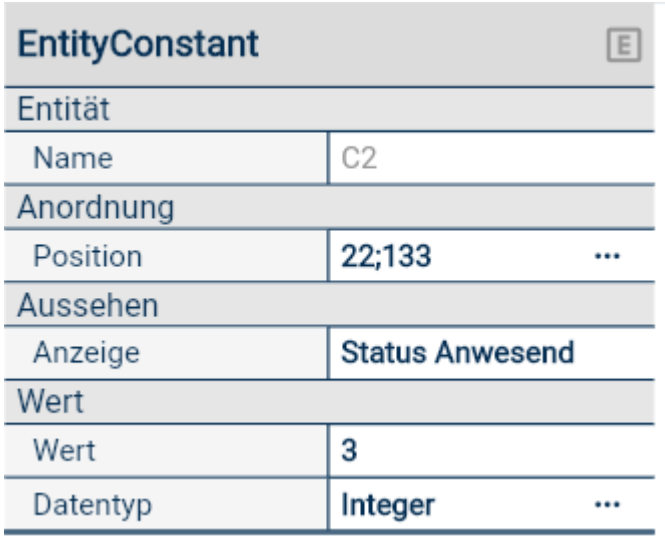


The following functions are available:

NAME	DESCRIPTION
Administration	Click the Administration button to display the available actions for the logic script. After activating the button, the administration menu opens. The following actions are available: - Export logic script - Import logic script
Zoom	The Logic Script Editor offers the option of zooming in and out of a view in 10% increments; details can be found in the Drawing area section.
Undo and Redo	Menu group for canceling or restoring changes, details can be found in the artboard area.
Navigate back and forward	- Menu group for navigating in the history, these controls are only available in History mode. - Press the Back button to navigate back in the history. After pressing the button, the next older data in the history is loaded and displayed. - Press the Forward button to navigate forwards in the history. After pressing the button, the next newer data in the history is loaded and displayed.
Edit	Press the Edit button to activate the edit mode of the Logic Script Editor. After activating the button, the logic script can be edited.
Live	Press the Live button to activate the live mode of the Logic Script Editor. After activating the button, the current values of the logic script execution are displayed on the drawing area.
History	Click the History button to activate the history mode of the Logic Script Editor. After activating the button, you can navigate through the history of the individual executions.
Execute	Press the Execute button to start a manual execution of the logic script. After activating the button, the Automation Logic Script is executed in IP-Symcon and the current values are displayed on the drawing area.
Save	Click the Save button to save your changes to the logic script. After activating the button, the logic script is saved on the IP-Symcon server. When switching to live or history mode, the logic script is saved automatically.
Info	Press the Info button to display the program information. After activating the button, the dialog with the license and program information is displayed.

Property Editor

The Property Editor panel displays all available properties of a control element.



Each property consists of a name (left column) and an associated value (for technicians: key-value pair; right column).
When you create a new object on the drawing area, its properties are assigned default values from IP-Symcon. To customize properties to your needs:

Click in the respective value fields and change the values.

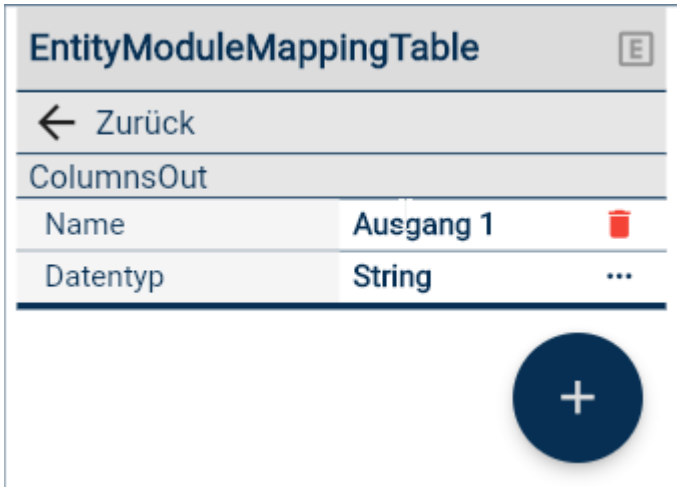
Save your changes from time to time (this may take a few seconds depending on your connection speed).

OVERVIEW

Outputs	Define outputs of a mapping module
The Properties	Meaning of the properties
Inputs	Define the inputs of a mapping module.
Mappings	Editing the assignments of a mapping module

Outputs

To define the outputs of a mapping module, press the menu button of the Outputs property of your mapping module. The definition of the inputs is loaded and displayed:



To configure an output of the mapping module:
Add a new entry to the selection using +.
Select the entries one after the other and edit their properties (see below).
Once you have defined all outputs, exit the editor by clicking the Back button.

THE FOLLOWING OPTIONS ARE AVAILABLE

NAME	DESCRIPTION
Name	Name of the output
Data type	Defines the data type of the output, the following options are available: Boolean, Integer, Float, String

The Properties

The properties described here are important for your work with IPS logic scripts.
A description of specific properties of a module can also be found in Repository .

GROUP: ENTITY

NAME	DESCRIPTION
EntityName	The name of the object in the logic script.

GROUP: VARIABLE

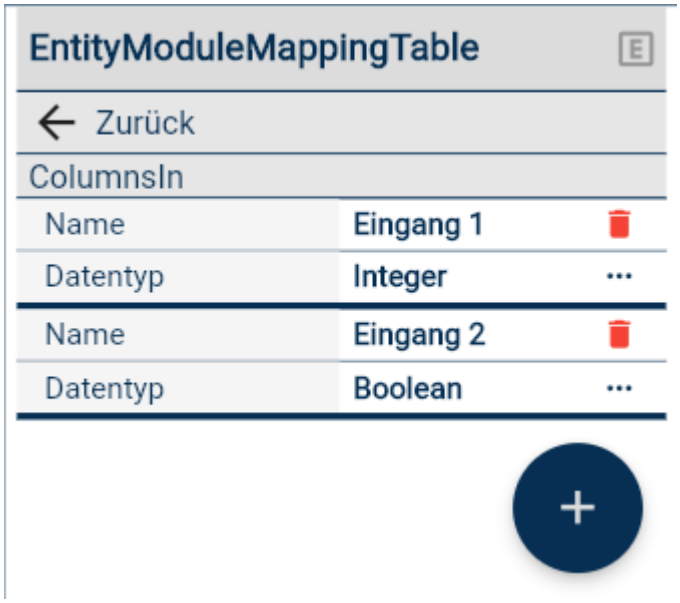
NAME	DESCRIPTION
ID	ID of the variable in IP-Symcon.
Name	Name of the variable in IP-Symcon.

GROUP: INSTANCE

NAME	DESCRIPTION
ID	ID of the instance in IP-Symcon.
Name	Name of the instance in IP-Symcon.

Inputs

To define the inputs of a mapping module, press the menu button of the Inputs property of your mapping module. The definition of the inputs is loaded and displayed:



To configure an input of the mapping module:

Add a new entry to the selection using +.

Select the entries one after the other and edit their properties (see below).

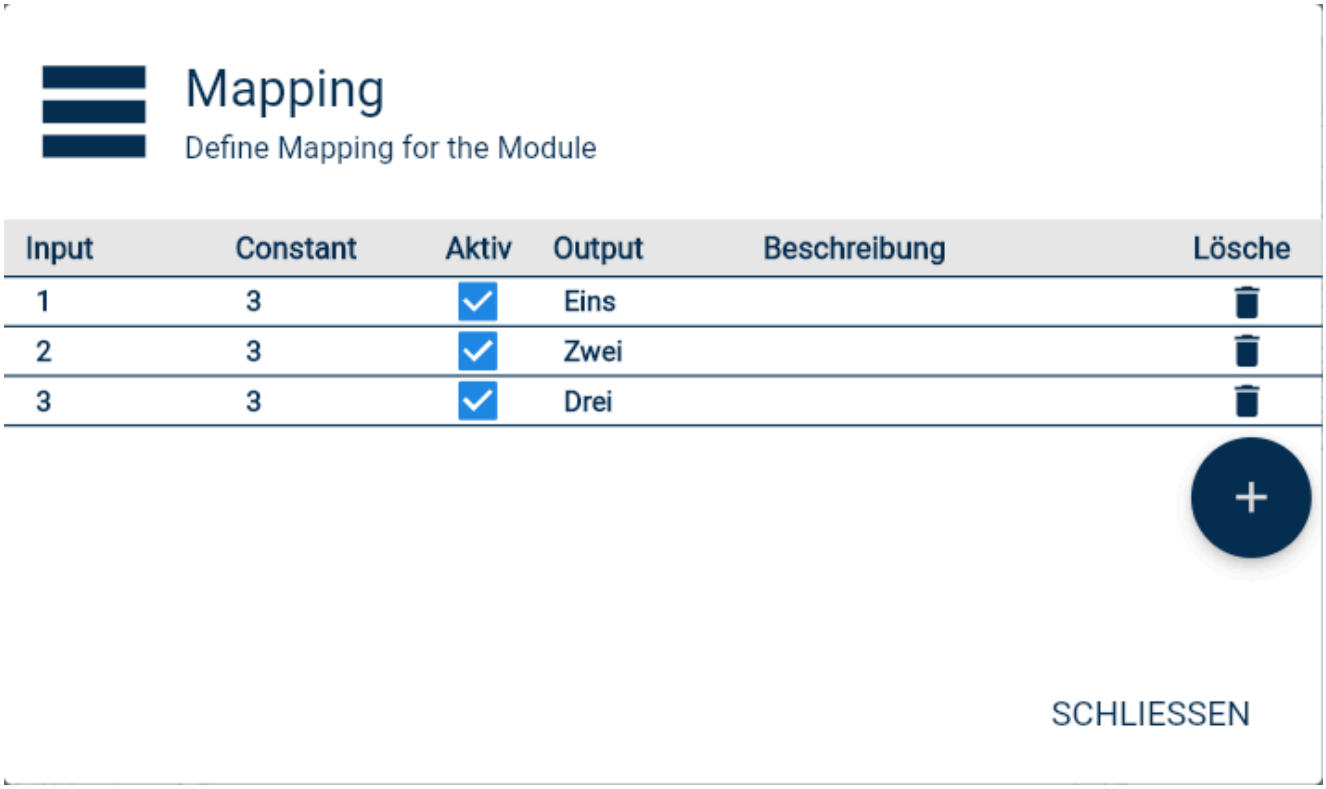
When you have defined all inputs, exit the editor by clicking the Back button.

The following options are available

NAME	DESCRIPTION
Name	Name of the input
Data type	Defines the data type of the input, the following options are available: Boolean, Integer, Float, String

Mappings

To define the individual mappings of a mapping module, press the Edit button of your mapping module. The definition of the mappings is loaded and displayed:



The editor provides a column for each input and for each output; when evaluating the module, the first entry in the list is used where all values at the input match.
To configure a mapping of the mapping module:

Add a new entry to the selection using +.
Select the value for the mapping for each input.
With "Active" you can deactivate a mapping without having to delete it immediately.
Select the desired value for each output.
Description provides a brief documentation of the current mapping line. This entry has no effect on the actual execution of the module.
Once you have defined all outputs, exit the editor by clicking the Back button.

Repository

The panel repository offers all available logic script modules, which can be placed on the drawing area simply by "drag and drop".
The repository offers the following options for finding modules:

ALPHABETICAL SORTING

Select the Alphabetical sorting option to sort all available modules in the repository by name.



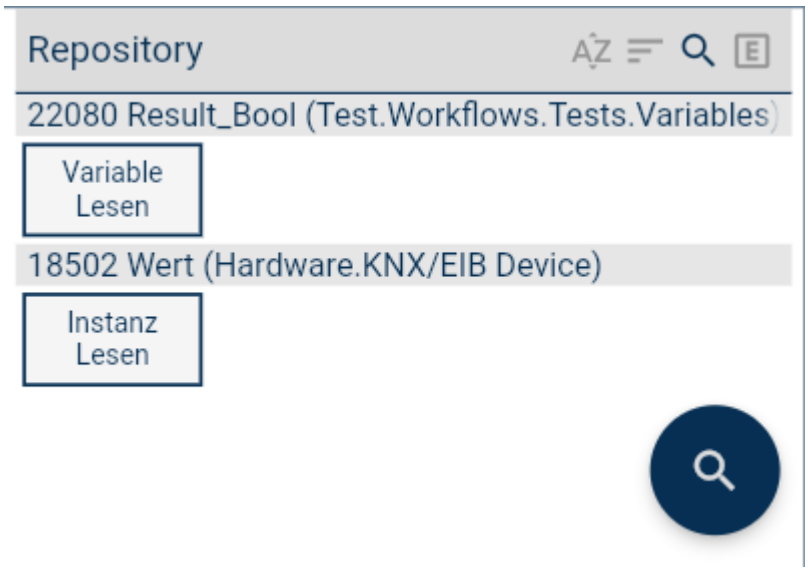
GROUPED BY USAGE

Select the option Grouped by usage () to group all available modules in the repository according to their usage:

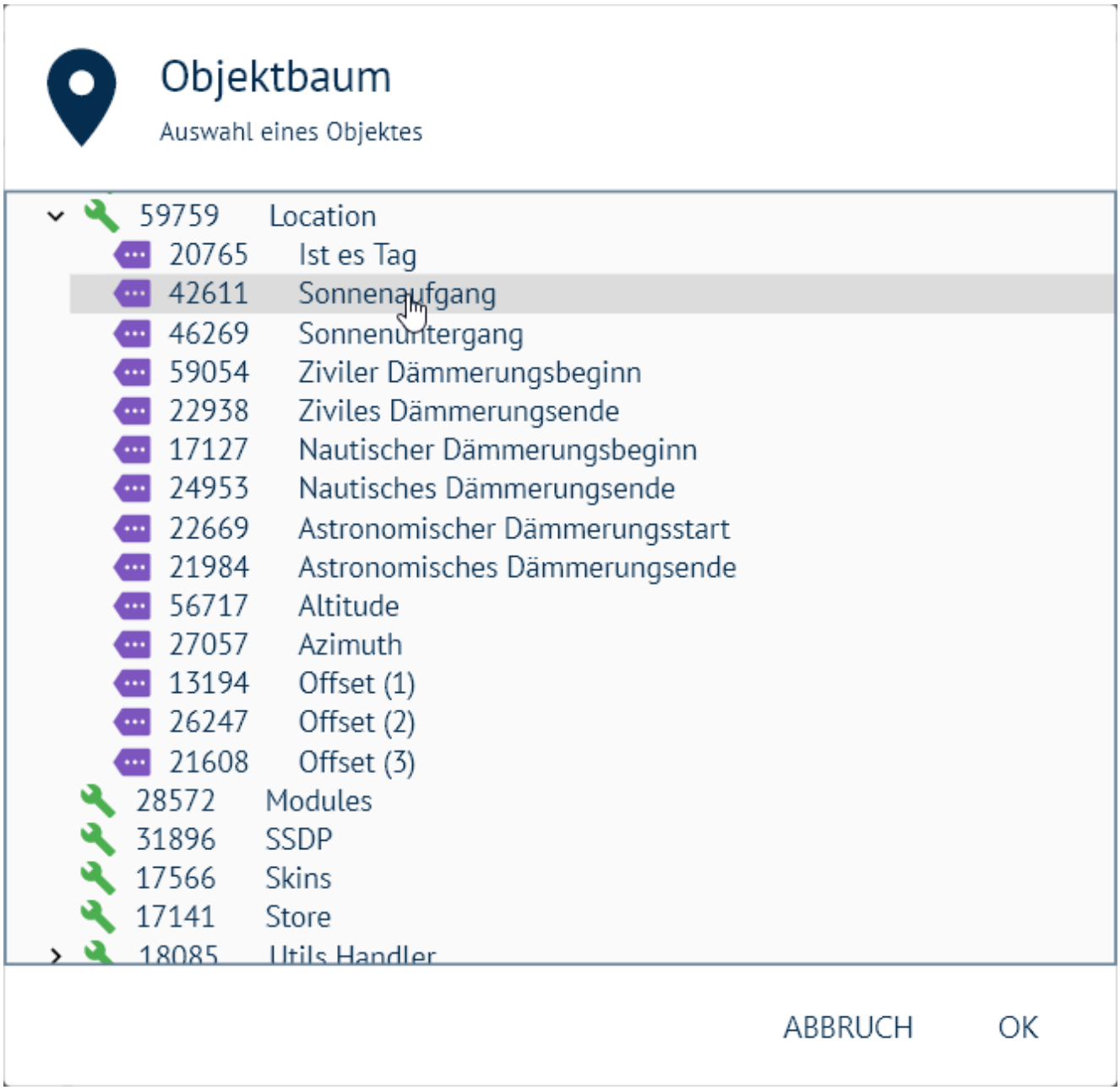


SEARCH BY ID

Select the Search by ID option to search for all available modules that can be used for a specific ID in IP-Symcon.



Click on the "Search" button to search for modules for an IP-Symcon object. After clicking on the "Search" button, the dialog for searching for an ID is opened:



After selecting an ID and confirming with OK, all available modules for the selected object are displayed.

EXPERT MODE

Select the option Expert mode () to activate the "Expert mode", after activating the option the repository also displays rarely used modules.



An overview of the relevant IP-Symcon properties, regardless of the object type, can be found in the section The properties

OVERVIEW OF THE MODULES

For a better overview, the editor modules have been grouped according to their use.

ACTIONS

Executing special actions

Custom Action	Execute any PHP code
Function call	Call an IP Symcon instance function.
IRTrans	Sending IR commands.
Log Message	Writing the log message.
Mail Notification	Sending a mail notification.
Push Notification	Sending a push notification.
Script execution	Calling scripts
Variable with ID	Modification of a variable with referencing by ID at runtime.

CONDITION

Evaluation of conditions

Conditional value	Set value if a certain condition applies
Condition	Conditional branching

DOCUMENTATION: COMMENT

Insert comments into the logic script.

EVENTS

Define events to start the automation automatically

On change	Execution of the logic script when a variable is changed.
On update	Execution of the logic script when a variable is updated.
At specific value	Execution of the logic script when a certain variable value is reached.
When exceeded	Execution of the logic script when a variable value is exceeded.
When falling below	Execution of the logic script when the value falls below a variable value.
weekly schedule (1 group)	Weekly plan with 1 group.
Weekly schedule (2 groups)	Weekly schedule with 2 groups.
Weekly schedule (7 groups)	Weekly schedule with 7 groups.

INFORMATION

Reading information

Script Information	Information on the current script execution
System Information	Date and time of the IP Symcon system
Variable Inforamtion	Information on a variable

CONSTANT

Define constant values

LOGIC

Logical modules

Not	Logical non
Or	Logical Or
And	Logical and

MAPPING TABLE

Create complex mapping logic.

MATHMATICS

Perform mathematical operations.

Addition	Addition of two or more values
Division	Division of two values
Formula	Calculation of a value based on a user-defined formula.
Maximum	Maximum of several values
Minimum	Minimum of several values
Modulo	Calculation of the modulo of two values
Multiplication	Multiplication of two values
Rounding	Rounding a value
Subtraction	Subtraction of two values

SUBLOGIC SCRIPTS

Combine several modules into a sublogic script

Sublogic script Parameter	Create sublogic script parameters.
Sublogic Script Output	Create sublogic script output.
Sublogic script Input	Create sublogic script input.

Sublogic script Module	Create sublogic script.

TIMER

Definition of timers.

Once a day	Cyclical timer with daily triggering.
One time Timer	One-time execution of a timer.
Timer Variable	Timer with variable for displaying the remaining running time.
Timer cyclic minute	Cyclical timer with triggering at one-minute intervals.
Timer cyclic second	Cyclical timer with triggering at one-second intervals.
Timer cyclic hour	Cyclical timer with triggering at hourly intervals.

CONVERSION

Conversion of values to another data type

Date to String	Converts a Unix timestamp into a string
Date to time	Converts a Unix timestamp into the individual time components.
Format message	Formats several values into one message.
Invert value	Inverts a value
Value to Boolean	Converts a value into a Boolean.
Value to float	Converts a value into a float.
Value to integer	Converts a value into an integer.
Value to string	Converts a value into a string.
Convert range	Convert a value from one value range to another.
Time to Date	Converts individual time components into a Unix timestamp.

READ/ WRITE VARIABLES

Reading and writing variables and instances.

Read Instance	Instances as input of a logic script
Write Instance	Instances as the output of a logic script
Read Variable	Variable as input of a logic script
Write Variable	Variable as the output of a logic script
Logic Script Variable	Create variable through the logic script

COMPARE

Compare values

Equal	Comparing values for equality or inequality

Larger/smaller	Compare values to Greater than, Greater equal to, Equal to, Smaller equal to or Smaller.
In value range	Checks a value for a specific value range

Actions

Modules of the "Actions" group enable the execution of IP-Symcon Automation scripts or various other IP-Symcon actions.

Custom Action	Execute any PHP code
Function call	Call an IP Symcon instance function.
IRTrans	Sending IR commands.
Log Message	Writing the log message.
Mail Notification	Sending a mail notification.
Push Notification	Sending a push notification.
Script execution	Calling scripts
Variable with ID	Modification of a variable with referencing by ID at runtime.

Custom Action

The "Custom action" module offers the option of executing any PHP code.



PROPERTIES

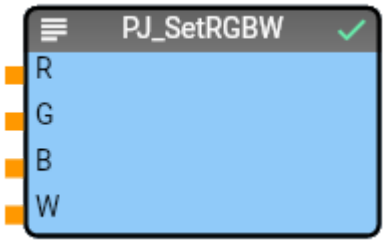
NAME	DESCRIPTION
Number of inputs	Number of inputs that the module should provide.
PHP code	PHP code to be executed. The inputs can be referenced in the PHP code via the variables $value1 to $value8.

DATA POINTS INPUT

NAME	DESCRIPTION
Value x	Depending on the "Number of inputs" property, 1 to 8 data points are available at the input.

Function call

The "Function call" module offers the option of calling an IP Symcon instance function.



PROPERTIES

NAME	DESCRIPTION
ID	ID of the instance where a function is to be called
Function name	Name of the function to be called.

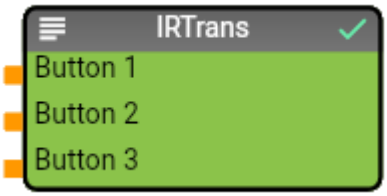
After entering a valid ID, the DropDown function already provides all possible function names.

DATA POINTS INPUT

NAME	DESCRIPTION
Parameters	The inputs of the module vary depending on the selected function

IRTrans

The "IRTrans" module offers the option of controlling an IRTrans instance. Up to 3 buttons can be controlled.



PROPERTIES

NAME	DESCRIPTION
Remote control	Remote control to be used for sending the buttons.
Instance ID	Instance to be used for sending the IR commands.

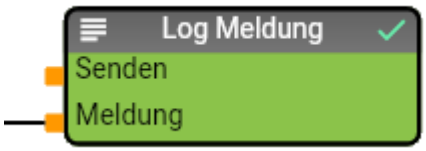
If commands are to be sent to different devices, several IRTrans modules must be used

DATA POINTS INPUT

NAME	DESCRIPTION
Button 1	1st IR command to be sent.
Button 2	2nd IR command to be sent.
Button 3	3rd IR command to be sent.

Log Message

The "Log message" module offers the option of writing a message to the IP-Symcon log.



PROPERTIES

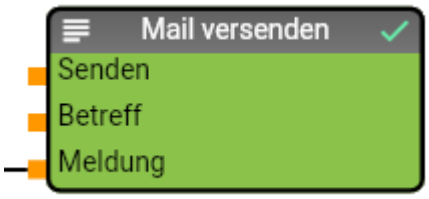
NAME	DESCRIPTION
Sender	Text to be used in the Sender field of the log message.

DATA POINTS INPUT

NAME	DESCRIPTION
Send	TRUE on this input triggers the sending of the message. If the input is not connected, a mail is always sent.
Message	Text for the log message.

Mail Notification

The "Mail" module offers the option of sending mail messages.



PROPERTIES

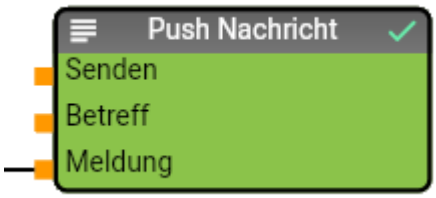
NAME	DESCRIPTION
InstanceID Mail	ID of the SMTP instance to be used for sending the mails.

DATA POINTS INPUT

NAME	DESCRIPTION
Send	TRUE on this input triggers the sending of the message. If the input is not connected, a mail is always sent.
Subject	Text to be used in the subject of the message.
Message	Text for the mail message.

Push Notification

The "Push Notification" module offers the option of sending a push notification to a mobile device.



PROPERTIES

NAME	DESCRIPTION
Notification instance	ID of the notification instance (notification control) to be used for sending the push notification.
Object for notification	ID of the object (view or WebFront configurator) to which the push notification is to be sent.

DATA POINT INPUT

NAME	DESCRIPTION
Send	TRUE on this input triggers the sending of the message. If the input is not connected, a notification is always triggered.
Subject	Text to be used in the subject of the message.
Message	Text for the message.

Script execution

The "Script" module offers the option of integrating other scripts or automations into the logic script.



PROPERTIES

NAME	DESCRIPTION
ScriptID	ID of the script.
Sender	Value for the SENDER variable when the script is called.
VariableID	Value for the variable VARIABLE when the script is called.

DATA POINTS INPUT

NAME	DESCRIPTION
Value	Value for the variable VALUE when the script is called.

Variable with ID

The "Variable with ID" module offers the option of changing a variable that is only referenced at logic script runtime via the ID of the variable.



DATA POINTS INPUT

NAME	DESCRIPTION
Variable	ID of the variable to be changed.
Value	Value to be set.

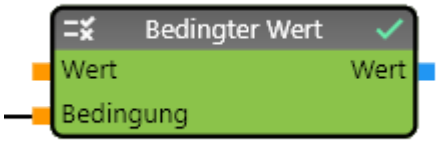
Condition

Modules of the "Condition" group enable the conditional execution of parts of the logic script.

Conditional value	Set value if a certain condition applies
Condition	Conditional branching

Conditional value

The "Conditional value" module only sets a value if a condition at the input has been evaluated to TRUE.



DATA POINTS INPUT

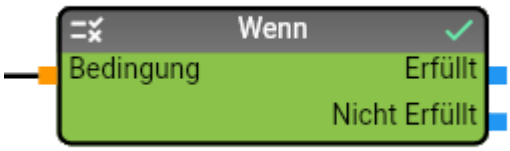
NAME	DESCRIPTION
Value	Value to be set.
Condition	Logical value that is checked as a condition.

DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Value to be set. The following branch of the logic script is only executed if the condition at the input was evaluated to TRUE.

Condition

The "Condition" (If) module offers the option of integrating a conditional branch into the logic diagram. If the condition at the input is fulfilled, the "Fulfilled" output returns the value "True", otherwise the "Not fulfilled" output returns the value "True". The other output does not return a value.



DATA POINTS INPUT

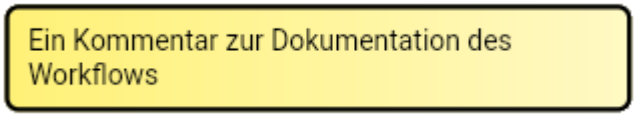
NAME	DESCRIPTION
Condition	Logical value that is checked as a condition.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Fulfilled	The following branch is only executed if the condition is evaluated to TRUE.
Not fulfilled	The following branch is only executed if the condition is evaluated to FALSE.

Documentation: Comment

The "Comment" module offers the option of inserting text to comment on the logic script.



PROPERTIES

NAME	DESCRIPTION
Text	Comment Text.

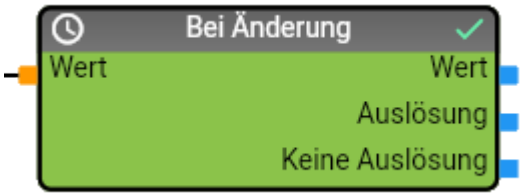
Events

Modules in the "Events" group enable the logic script to be executed automatically when the value of a variable changes.

On change	Execution of the logic script when a variable is changed.
On update	Execution of the logic script when a variable is updated.
At specific value	Execution of the logic script when a certain variable value is reached.
When exceeded	Execution of the logic script when a variable value is exceeded.
When falling below	Execution of the logic script when the value falls below a variable value.
weekly schedule (1 group)	Weekly plan with 1 group.
Weekly schedule (2 groups)	Weekly schedule with 2 groups.
Weekly schedule (7 groups)	Weekly schedule with 7 groups.

On change

The "On change" module enables the logic script to be executed when a specific variable has been changed (new value is assigned to the variable).



DATA POINTS INPUT

NAME	DESCRIPTION
Value	Data point of the variable whose change is to trigger execution of the logic script.

DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Data point of the variable. The following branch of the logic script is only executed if the associated variable has been changed.
Triggering	Data point of the module that returns TRUE if the linked variable has been changed. The following branch of the logic script is only executed if the variable has been changed.
No triggering	Data point of the module that returns TRUE if the linked variable has not been changed. The following branch of the logic script is only executed if the variable has not been changed.

APPLICATION EXAMPLE

Given are 1 Homematic motion detector and 1 Homematic actuator for switching a light:

Bewegungsmelder	HomeMatic Device	
BRIGHTNESS	Integer	160
INSTALL_TEST	Boolean	true
MOTION	Boolean	Untätig
Licht	HomeMatic Device	
INHIBIT	Boolean	false
STATE	Boolean	Aus
WORKING	Boolean	false

View of the two actuators in the WebFront



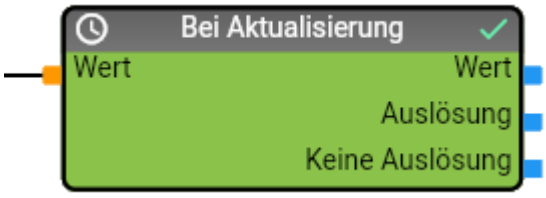
A logic script is now to be created in Logic Scripts that automatically switches a light on or off again when the motion detector is triggered.
You can implement this behavior by using a With change, predefined instance modules for setting the values can be found in the object tree of the logic script designer.



In this specific case, the logic script is called up each time the status variable of the motion detector is changed and applies its status to the light.

On update

The "On update" module enables the logic script to be executed when a specific variable is updated (same value or different value is assigned to the variable).



DATA POINTS INPUT

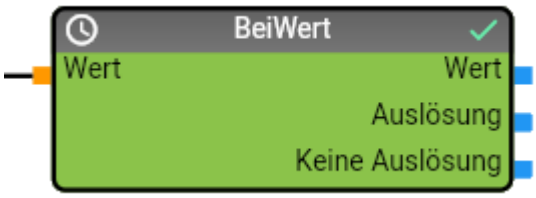
NAME	DESCRIPTION
Value	Data point of the variable whose update is to trigger execution of the logic script.

DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Data point of the variable. The following branch of the logic script is only executed if the associated variable has been updated.
Triggering	Data point of the module that returns TRUE if the linked variable has been updated. The following branch of the logic script is only executed if the variable has been updated.
No triggering	Data point of the module that returns TRUE if the linked variable has not been updated. The following branch of the logic script is only executed if the variable has not been updated.

At specific value

The "At specific value" module enables the logic script to be executed when a variable reaches a specific value.



PROPERTIES

NAME	DESCRIPTION
Value for triggering	Value at which the module should trigger.
Trigger subsequent events	If this option is activated, the module triggers each time the variable is updated if the value corresponds to the specified limit. If the option is deactivated, the module only triggers once; the module only triggers again if the value exceeds or falls below the limit and reaches it again.

DATA POINTS INPUT

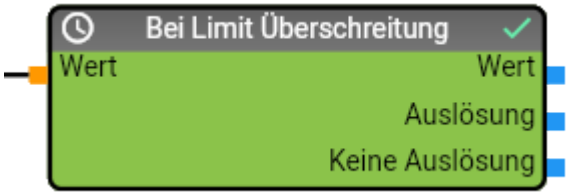
NAME	DESCRIPTION
Value	Data point of the variable whose change is to trigger execution of the logic script.

DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Data point of the variable. The following branch of the logic script is only executed if the "Update" output also returns TRUE.
Triggering	Data point of the module that returns TRUE if the value of the linked variable has reached the specified value. The subsequent branch of the logic script is only executed if the value has been reached.
No triggering	Data point of the module that returns TRUE if the value of the linked variable has not reached the specified value. The subsequent branch of the logic script is only executed if the value has not been reached.

When exceeded

The "When exceeded" module enables the logic script to be executed when a variable exceeds a certain value.



PROPERTIES

NAME	DESCRIPTION
Value for triggering	Value above which the module should trigger.
Trigger subsequent events	If this option is activated, the module triggers every time the variable changes if the value is above the specified limit. If the option is deactivated, the module only triggers once; the module only triggers again if the value falls below the limit and exceeds it again.

DATA POINTS INPUT

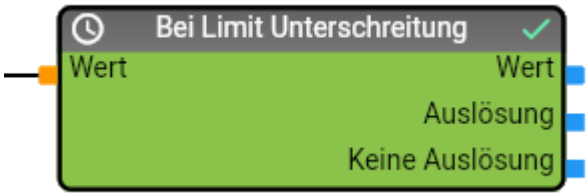
NAME	DESCRIPTION
Value	Data point of the variable whose exceeding should trigger execution of the logic script.

DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Data point of the variable. The following branch of the logic script is only executed if the "Update" data point also returns TRUE.
Triggering	Data point of the module that returns TRUE if the value of the linked variable has exceeded the specified value. The subsequent branch of the logic script is only executed if the value has been exceeded.
No triggering	Data point of the module that returns TRUE if the value of the linked variable has not exceeded the specified value. The subsequent branch of the logic script is only executed if the value has not been exceeded.

When falling below

The "When falling below" module enables the logic script to be executed when a variable falls below a certain value.



PROPERTIES

NAME	DESCRIPTION
Value for triggering	Value below which the module should trigger.
Trigger subsequent events	If this option is activated, the module triggers each time the variable changes if the value is below the specified limit. If the option is deactivated, the module only triggers once; the module only triggers again if the value exceeds the limit and falls below it again.

DATA POINTS INPUT

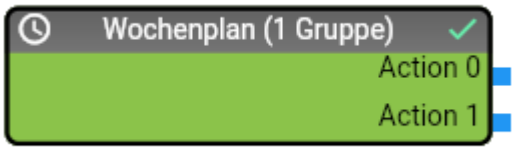
NAME	DESCRIPTION
Value	Data point of the variable whose undershoot should trigger execution of the logic script.

DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Data point of the variable. The following branch of the logic script is only executed if the "Update" output also returns TRUE.
Triggering	Data point of the module that returns TRUE if the value of the linked variable has fallen below the specified value. The subsequent branch of the logic script is only executed if the value has fallen below the specified value.
No triggering	Data point of the module that returns TRUE if the value of the linked variable has not fallen below the specified value. The subsequent branch of the logic script is only executed if the value has not fallen below the specified value.

weekly schedule (1 group)

The module "Weekly schedule (1 group)" offers the possibility to link the logic script with an IP-Symcon weekly schedule with 1 group.



PROPERTIES

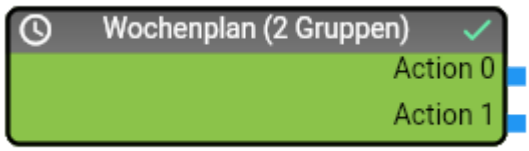
NAME	DESCRIPTION
Number of actions	Number of actions available for configuring the weekly schedule in the visualization.
Actions	List of weekly schedule actions. Each weekly schedule action can be assigned a name and a color.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Action x	Data point of the respective weekly plan action, if the corresponding action is triggered, the data point is evaluated to TRUE. The subsequent branch of the logic script is only executed if this data point has been evaluated to TRUE.

Weekly schedule (2 groups)

The module "Weekly schedule (2 groups)" offers the possibility to link the logic script with an IP-Symcon weekly schedule with 2 groups.



PROPERTIES

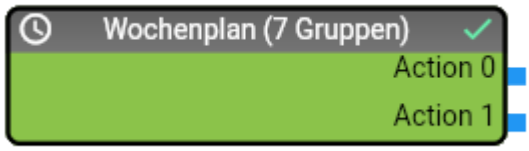
NAME	DESCRIPTION
Number of actions	Number of actions available for configuring the weekly schedule in the visualization.
Actions	List of weekly schedule actions. Each weekly schedule action can be assigned a name and a color.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Action x	Data point of the respective weekly plan action, if the corresponding action is triggered, the data point is evaluated to TRUE. The subsequent branch of the logic script is only executed if this data point has been evaluated to TRUE.

Weekly schedule (7 groups)

The module "Weekly schedule (7 groups)" offers the possibility to link the logic script with an IP-Symcon weekly schedule with 7 groups.



PROPERTIES

NAME	DESCRIPTION
Number of actions	Number of actions available for configuring the weekly schedule in the visualization.
Actions	List of weekly schedule actions. Each weekly schedule action can be assigned a name and a color.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Action x	Data point of the respective weekly plan action, if the corresponding action is triggered, the data point is evaluated to TRUE. The subsequent branch of the logic script is only executed if this data point has been evaluated to TRUE.

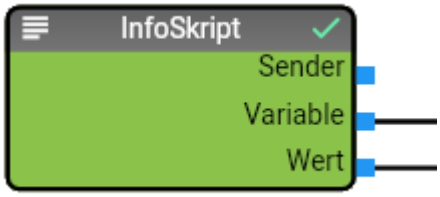
Information

Module der Gruppe "Informationen" ermöglichen die Einbindung von diversen IP-Symcon Informationen in den Logic Script.

Script Information	Information on the current script execution
System Information	Date and time of the IP Symcon system
Variable Inforamtion	Information on a variable

Script Information

The "ScriptInfo" module offers the option of integrating information on the current script execution.

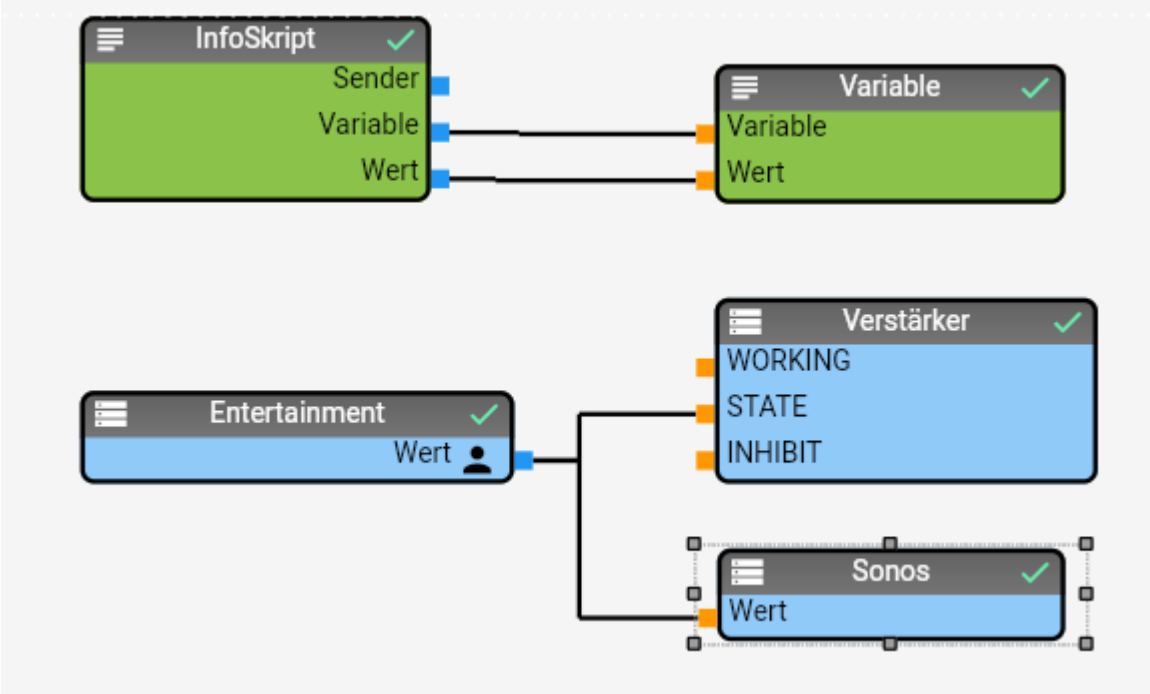


DATA POINTS OUTPUT

NAME	DESCRIPTION
Sender	The sender variable of the current script execution (value of the IP-Symcon script variable $_IPS['SENDER']).
Variable	The variable ID of the current script execution (value of the IP-Symcon script variable $_IPS['VARIABLE']).
Value	The value variable of the current script execution (value of the IP-Symcon script variable $_IPS['VALUE']).

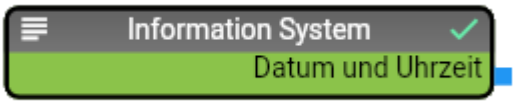
APPLICATION EXAMPLE

A typical application example for the "ScriptInfo" module is the realization of an action script for a variable.
The logic script receives the value of the visualization and assigns it to the relevant variable.
Further information on the subject of action scripts.
The logic script is assigned to the variable as an action script by integrating the variable into the logic script and selecting the "As action script" option in the property editor.
Alternatively, the logic script script can also be manually assigned to the variable in the console.
In this example, the value of the "Entertainment" variable is set first (operation via the WebFront or IPSView client) and then the value is used to control an amplifier and a Sonos player using an automation script.


InfoSkript
Sender
Variable
Wert
Variable
Variable
Wert
Verstärker
WORKING
STATE
INHIBIT
Entertainment
Wert
Sonos
Wert

System Information

The "SystemInfo" module provides information about the IP-Symcon system.

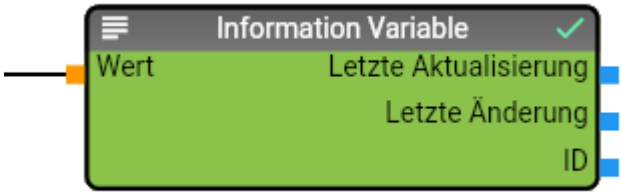


DATA POINTS OUTPUT

NAME	DESCRIPTION
Timestamp	Unix Timestamp of the current system time.

Variable Inforamtion

The "VariableInfo" module provides information on a variable.



DATA POINTS INPUT

NAME	DESCRIPTION
Value	Connection to the data point value of a variable from which the information is to be displayed.

DATA POINT OUTPUT

NAME	DESCRIPTION
Last update	Unix Timestamp of the last update of the variable.
Last change	Unix timestamp of the last change to the variable.
ID	ID of the variable.

Constant

The "Constant" module offers the option of integrating fixed values into the logic script; a data type, a value and a name for the display can be assigned to the module in the properties.



PROPERTIES

NAME	DESCRIPTION
Display	Name displayed in the diagram for the constant.
Value	Value of the constant as a string.
Data type	Data type of the constant.

DATA POINTS OUTPUT

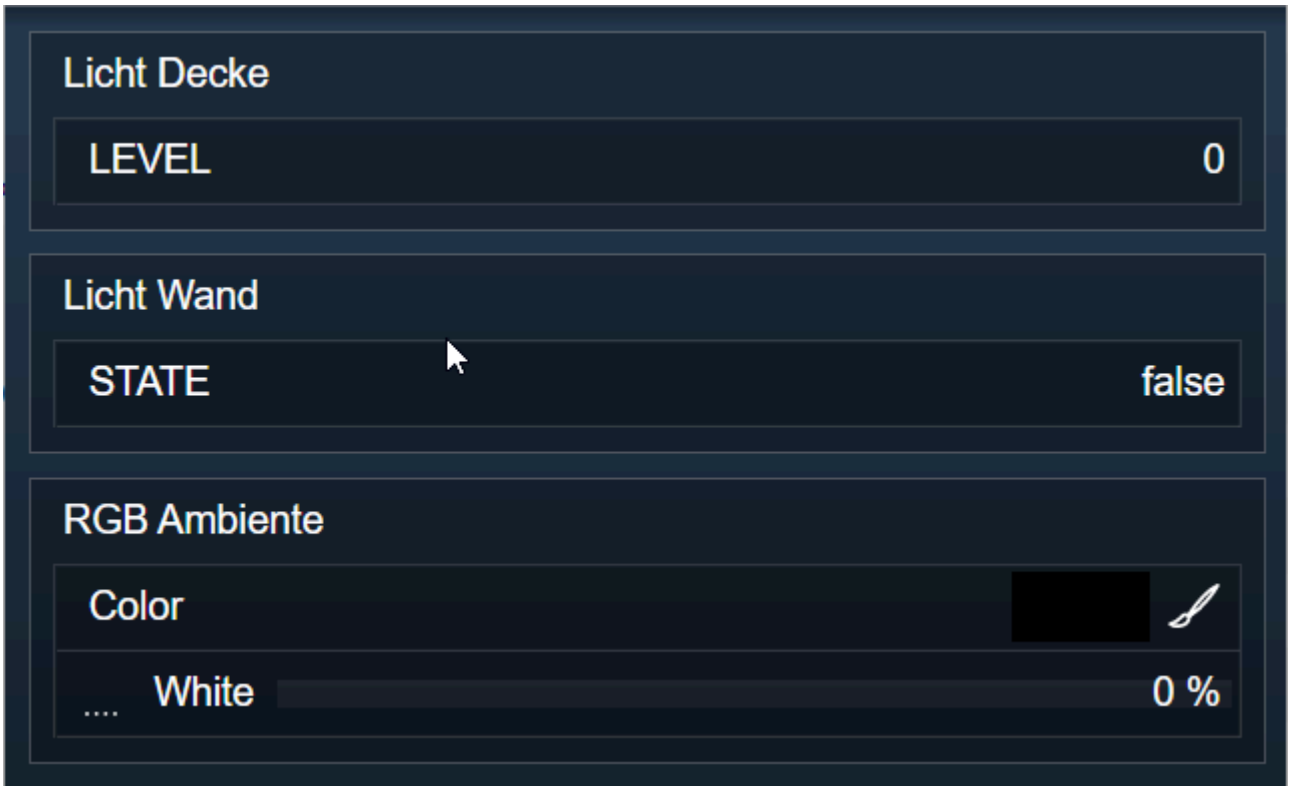
NAME	DESCRIPTION
Value	The value of the constant

APPLICATION EXAMPLE

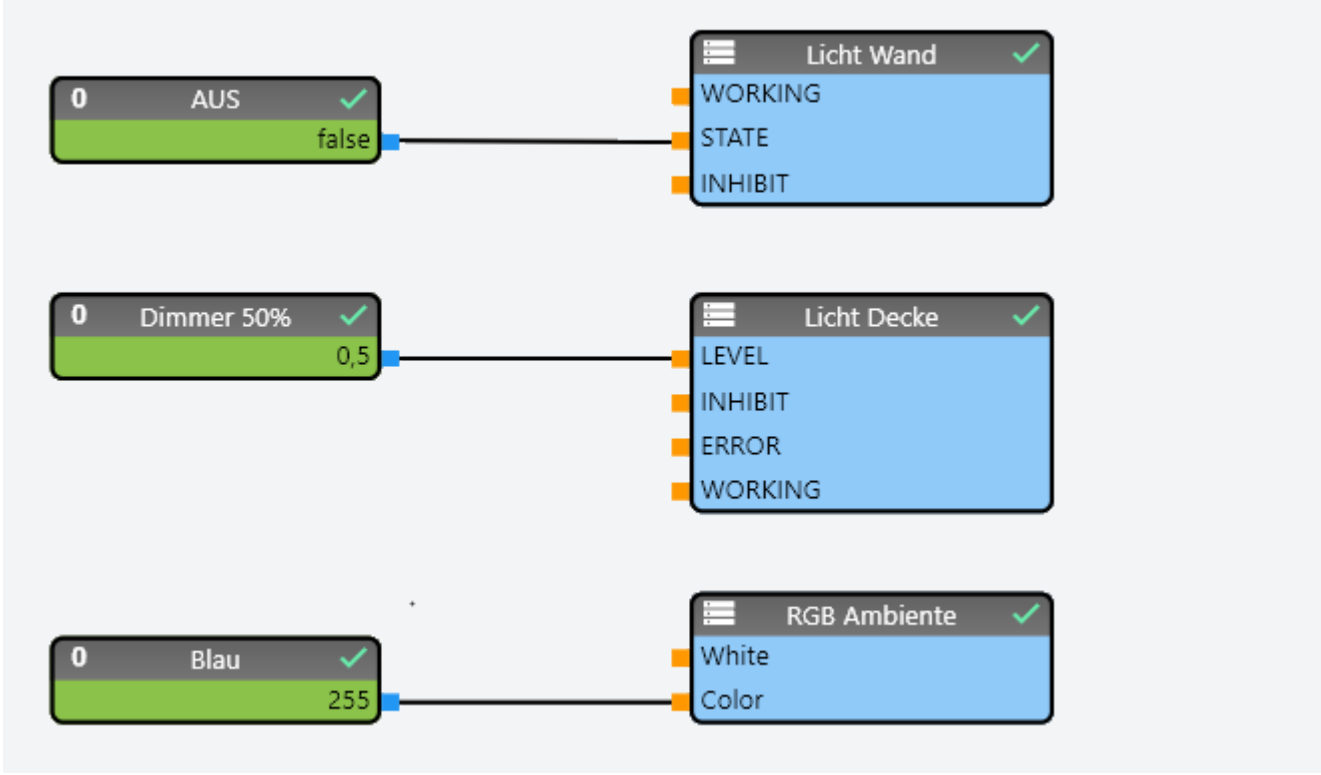
Given are 1 Homematic actuator for switching a light, 1 Homematic actuator for dimming a light and an IP-Symcon 868 actuator for operating an RGB strip:

Licht Decke	HomeMatic Device	
ERROR	Integer	0
INHIBIT	Boolean	false
LEVEL	Float	0
WORKING	Boolean	false
Licht Wand	HomeMatic Device	
INHIBIT	Boolean	false
STATE	Boolean	false
WORKING	Boolean	false
RGB Ambiente	IPS-868 Stripe	
Color	Integer	
White	Integer	0 %

The actuators can be switched separately via the WebFront:



A logic script is now to be created in Logic Scripts that sets a specific lighting scenario and assigns predefined values to the individual lighting fixtures.
You can realize these values by using constants, predefined instance modules for setting the values can be found in the Object tree of the Logic Script Designer.



In this specific case, the RGB luminaire is assigned the color blue, the wall light is switched off and the dimmer is set to 50%.

Logic

Modules in the "Logic" group enable the integration of logic modules.

Not	Logical non
Or	Logical Or
And	Logical and

Not

The "Not" module offers the option of implementing a logical NOT in the logic script.



DATA POINTS INPUT

NAME	DESCRIPTION
Value	Data point for input value.

DATA POINT OUTPUT

NAME	DESCRIPTION
Inverted	Returns TRUE if the value at input Value is FALSE and returns FALSE if the value at input Value is TRUE.

Or

The "Or" module offers the option of implementing a logical OR link in the logic script.



PROPERTIES

NAME	DESCRIPTION
Number of inputs	Provides 2 to 8 module inputs.

DATA POINTS INPUT

NAME	DESCRIPTION
Value 1	Data point for comparison value 1
Value 2	Data point for comparison value 2

DATA POINT OUTPUT

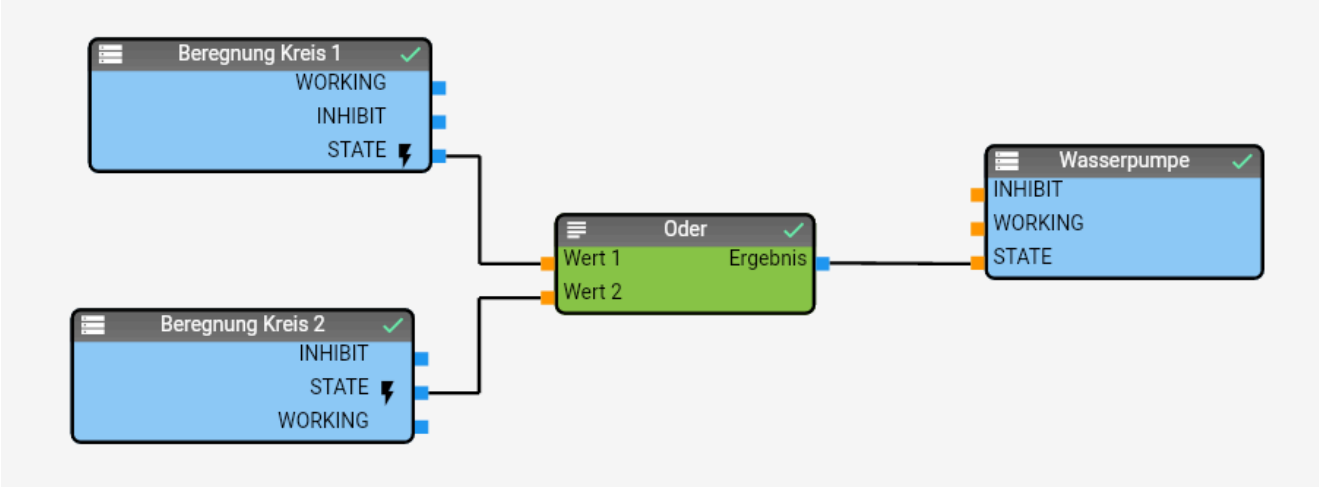
NAME	DESCRIPTION
Result	Returns TRUE if the value of one input is TRUE.

APPLICATION EXAMPLE

Given are 2 Homematic actuators for controlling a garden irrigation system and 1 Homematic actuator that acts as the main switch for the irrigation system:

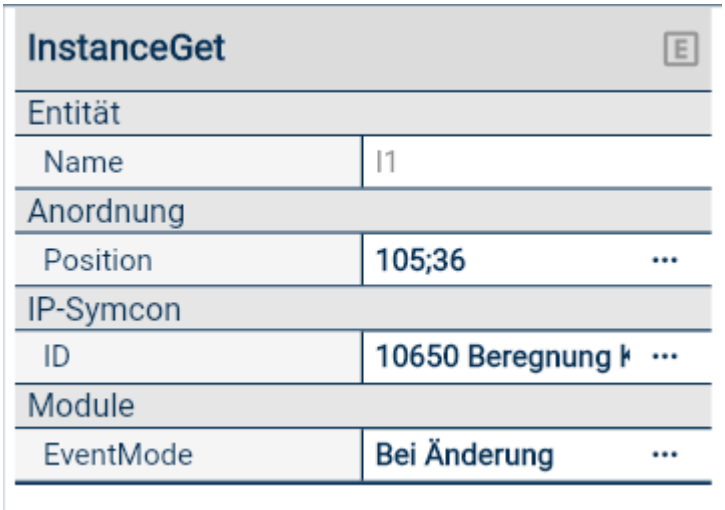
Beregnung Kreis 1	HomeMatic Device	
INHIBIT	Boolean	false
STATE	Boolean	true
WORKING	Boolean	false
Beregnung Kreis 2	HomeMatic Device	
INHIBIT	Boolean	false
STATE	Boolean	false
WORKING	Boolean	false
Wasserpumpe	HomeMatic Device	
INHIBIT	Boolean	false
STATE	Boolean	true
WORKING	Boolean	false

A logic script is now to be created in Logic Scripts that automatically controls the garden pump. If one of the two irrigation actuators is activated, the main switch should also be activated. If both actuators are off, the main switch should also be deactivated.
You can implement this behavior using an Or module; predefined modules for the instances can be found in the object tree of the logic script editor.



How it works:
A change to the "STATE" outputs of the irrigation circuit automatically calls up the logic script as soon as one of the two irrigation actuators is switched (regardless of whether via a visualization, a weekly schedule or another script).
If one of the two actuators has been activated, the or returns TRUE and activates the main switch .
If both actuators are deactivated, the OR returns FALSE and deactivates the main switch again.

For automatic execution of the logic script when the variable is changed, you must set the "Logic Script execution" property of the "Read instance" module to the value "On change"



And

The "And" module offers the option of implementing a logical AND link in the logic script.



PROPERTIES

NAME	DESCRIPTION
Number of inputs	Provides 2 to 8 module inputs.

DATA POINTS INPUT

NAME	DESCRIPTION
Value 1	Data point for comparison value 1
Value 2	Data point for comparison value 2

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Returns *TRUE* if the value at all inputs are *TRUE*.

Mapping table

The "Mapping table" module offers the option of mapping up to 8 inputs with a complex mapping logic to up to 8 outputs. You can define the values for the respective outputs for various combinations of values at the inputs.



PROPERTIES

NAME	DESCRIPTION
Inputs	Definition of the inputs. Up to 8 inputs can be defined, each input must be assigned a name and a data type.
Outputs	Definition of outputs. Up to 8 outputs can be defined, each output must be assigned a name and a data type.
Mappings	Definition of the mappings.

DATA POINTS INPUT

NAME	DESCRIPTION
Input x	Data points defined in the Inputs property

DATA POINTS OUTPUT

NAME	DESCRIPTION
Output x	Data points defined in the Outputs property

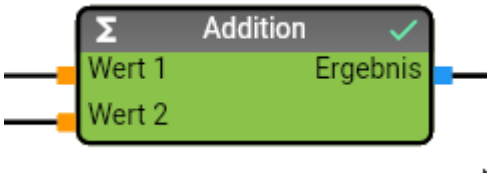
Mathmatics

Modules in the "Mathematics" group enable the calculation of mathematical expressions in the logic script.

Addition	Addition of two or more values
Division	Division of two values
Formula	Calculation of a value based on a user-defined formula.
Maximum	Maximum of several values
Minimum	Minimum of several values
Modulo	Calculation of the modulo of two values
Multiplication	Multiplication of two values
Rounding	Rounding a value
Subtraction	Subtraction of two values

Addition

The "Addition" module offers the option of adding 2 values.



PROPERTIES

NAME	DESCRIPTION
Number of inputs	Provides 2 to 8 inputs of the module.

DATA POINTS INPUT

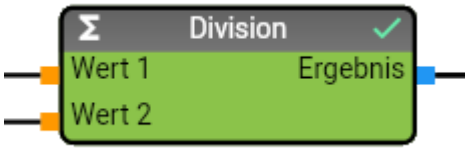
NAME	DESCRIPTION
Value 1	Data point for value 1.
Value 2	Data point for value 2.

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Sum of the inputs.

Division

The "Division" module offers the option of dividing 2 values.



DATA POINTS INPUT

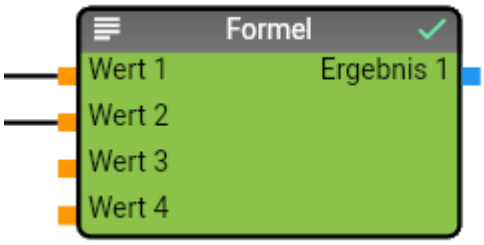
NAME	DESCRIPTION
Value 1	Data point for value 1
Value 2	Data point for value 2

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Result from the calculation of value 1 by value 2

Formula

The "Formula" module offers the option of calculating up to 8 values based on a user-defined formula with up to 8 different input values.



PROPERTIES

NAME	DESCRIPTION
Formula	PHP code to be executed when the module is called. The values of the input are available via the variables $value1, $value2, $value3, $value4, $value5, $value6, $value7 and $value8. The result can be returned to the logic script via the variables $result1, $result2, $result3, $result4, $result5, $result6, $result7 and $result8.
Number of outputs	Number of data points at the output (value range 1-8)
Number of inputs	Number of data points at the input (value range 1-8)

DATA POINTS INPUT

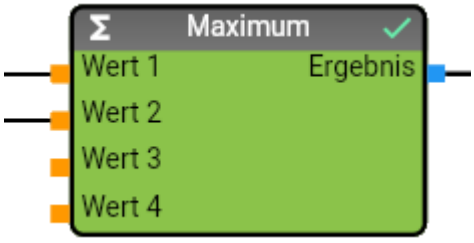
NAME	DESCRIPTION
Value 1	Data point for value 1
Value 2	Data point for value 2
Value 3	Data point for value 3
Value 4	Data point for value 4
Value 5	Data point for value 5
Value 6	Data point for value 6
Value 7	Data point for value 7
Value 8	Data point for value 8

DATA POINT OUTPUT

NAME	DESCRIPTION
Result 1	Result 1 from the calculation with the user-defined formula.
Result 2	Result 2 from the calculation with the user-defined formula.
Result 3	Result 3 from the calculation with the user-defined formula.
Result 4	Result 4 from the calculation with the user-defined formula.
Result 5	Result 5 from the calculation with the user-defined formula.
Result 6	Result 6 from the calculation with the user-defined formula.
Result 7	Result 7 from the calculation with the user-defined formula.
Result 8	Result 8 from the calculation with the user-defined formula.

Maximum

The "Maximum" module offers the option of calculating the maximum from up to 4 different values.



DATA POINTS INPUT

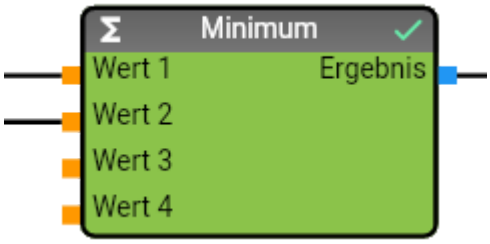
NAME	DESCRIPTION
Value 1	Data point for value 1
Value 2	Data point for value 2
Value 3	Data point for value 3
Value 4	Data point for value 4

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Maximum of input values Value 1 to Value 4

Minimum

The "Minimum" module offers the option of calculating the minimum from up to 4 different values.



DATA POINTS INPUT

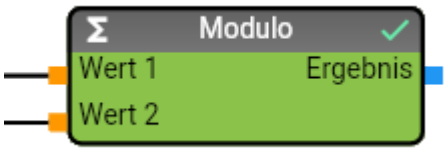
NAME	DESCRIPTION
Value 1	Data point for value 1
Value 2	Data point for value 2
Value 3	Data point for value 3
Value 4	Data point for value 4

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Minimum of input values value 1 to value 4

Modulo

The "Modulo" module offers the option of calculating the modulo of 2 values.



DATA POINTS INPUT

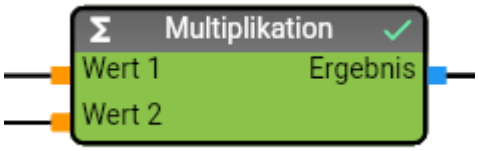
NAME	DESCRIPTION
Value 1	Data point for value 1
Value 2	Data point for value 2

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Result from the calculation Value 1 modulo Value 2

Multiplication

The "Multiplication" module offers the option of multiplying 2 values.



DATA POINTS INPUT

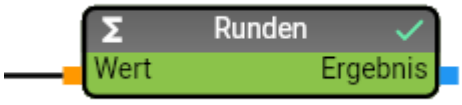
NAME	DESCRIPTION
Value 1	Data point for value 1
Value 2	Data point for value 2

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Result from the calculation value 1 times value 2

Rounding

The "Rounding" module offers the option of rounding a value.



PROPERTIES

NAME	DESCRIPTION
Digits	Number of digits to round to.

DATA POINTS INPUT

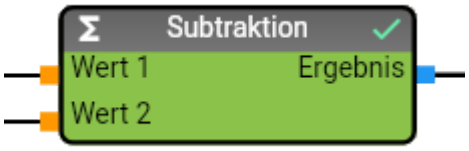
NAME	DESCRIPTION
Value 1	Data point for input value.

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Result from the calculation Value rounded.

Subtraction

The "Subtraction" module offers the option of subtracting 2 values.



DATA POINTS INPUT

NAME	DESCRIPTION
Value 1	Data point for value 1
Value 2	Data point for value 2

DATA POINT OUTPUT

NAME	DESCRIPTION
Result	Result from the calculation Value 1 minus Value 2

Sublogic scripts

"Sublogic scripts" make it possible to group several modules into a new unit and assign parameters and data points to them. This group of modules can then be addressed in the "main logic script" as an independent module, and it is also possible to export or import these sublogic scripts into a file.

Sublogic script Parameter	Create sublogic script parameters.
Sublogic Script Output	Create sublogic script output.
Sublogic script Input	Create sublogic script input.
Sublogic script Module	Create sublogic script.

Sublogic script Parameter

The "Parameter" module offers the option of creating a parameter for a sublogic script. Parameters can subsequently be set in the properties of the sublogic script module.



PROPERTIES

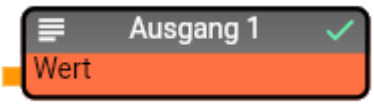
NAME	DESCRIPTION
Display	Name of the parameter, specifies the name of the parameter as it should be displayed in the property list.
Data type	Data type of the parameter
Default value	Default value of the parameter to be assigned after the initial creation of the module.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	Current value of the parameter that can subsequently be used in the Sublogic Script.

Sublogic Script Output

The "Output" module offers the option of creating an output data point for a sublogic script. Outputs are made available as data points of the sublogic script module and can be connected to other modules there.



PROPERTIES

NAME	DESCRIPTION
Display	Name of the output, specifies the name of the data point as it should be displayed in the main logic script module.
Data type	Data type of the parameter

DATA POINT INPUT

NAME	DESCRIPTION
Value	Value to be made available to the main logic script via the output.

Sublogic script Input

The "Input" module offers the option of creating a data point input for a sublogic script. Inputs are made available as data points of the sublogic script module and can be connected to other modules there.



PROPERTIES

NAME	DESCRIPTION
Display	Name of the input, specifies the name of the data point as it should be displayed in the main logic script module.
Data type	Data type of the parameter
Mandatory	Specifies whether the input must be connected.

DATA POINTS INPUT

NAME	DESCRIPTION
Default value	Default value if the data point was not connected.

DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Current value that was transferred to the data point of the sublogic script module.

Sublogic script Module

The "Sublogic script" module offers the option of combining several modules into one unit and inserting them into your main logic script.



The following functions are available:

NAME	DESCRIPTION
Open / Edit	Click on Open / Edit to open a sublogic script for editing. After clicking the button, the sublogic script is opened and can be edited.
Back	Click on Back to exit the sublogic script editor. After clicking the button, the current changes to the sublogic script are saved and the main logic script is reloaded.

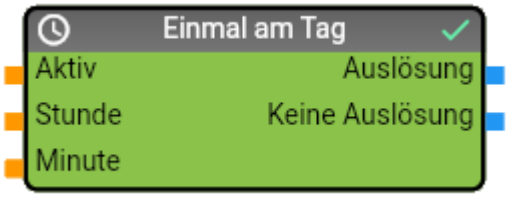
Timer

Modules of the "Timer" group to automatically execute the logic script via a timer.

Once a day	Cyclical timer with daily triggering.
One time Timer	One-time execution of a timer.
Timer Variable	Timer with variable for displaying the remaining running time.
Timer cyclic minute	Cyclical timer with triggering at one-minute intervals.
Timer cyclic second	Cyclical timer with triggering at one-second intervals.
Timer cyclic hour	Cyclical timer with triggering at hourly intervals.

Once a day

The "Timer cyclic second" module allows you to implement a timer that calls up the logic script every day at a specific time.



PROPERTIES

NAME	DESCRIPTION
Time	Time for triggering the timer.

DATA POINTS INPUT

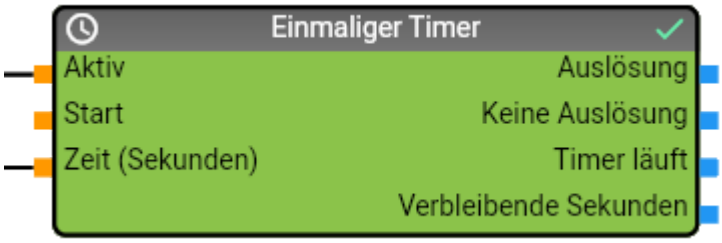
NAME	DESCRIPTION
Active	Activation of the timer; if TRUE, the timer is activated and triggers the logic script at the set time every day.
Hour	Data point of the module that controls the hour of triggering. If the input is not connected, the value of the "Time" property is used for triggering.
Minute	Data point of the module that controls the minute of triggering. If the input is not connected, the value of the "Time" property is used for triggering.

DATA POINT OUTPUT

NAME	DESCRIPTION
Triggering	Data point of the module that returns TRUE when the module has triggered. This occurs every second when the timer is active. The subsequent branch of the logic script is only executed if the timer has been triggered.
No triggering	Data point of the module that returns TRUE if the module has not triggered. The subsequent branch of the logic script is only executed if the timer has not been triggered.

One time Timer

The "One-time timer" module enables the time-delayed execution of certain parts of the logic diagram.



DATA POINTS INPUT

NAME	DESCRIPTION
Restart/Active	Activation of the timer, if TRUE the timer is activated and triggers the logic diagram again after a certain time. If the timer is already running, the remaining time is reset to the original value.
Start	Start the timer, if TRUE the timer is activated if it is not yet active and triggers the logic plan again after a certain time.
Time (seconds)	Time in seconds after which the timer should be triggered.

DATA POINTS OUTPUT

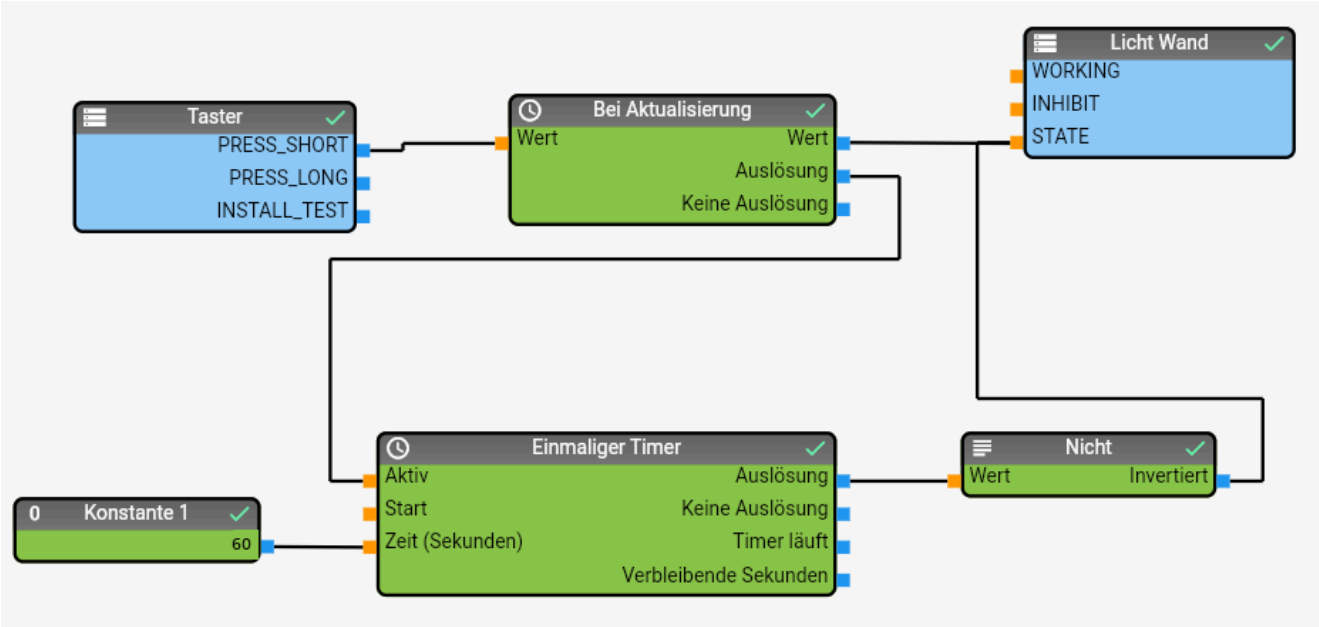
NAME	DESCRIPTION
Triggering	Data point of the module that returns TRUE when the module has triggered. The following branch of the logic script is only executed if the timer has been triggered.
No triggering	Data point of the module that returns TRUE if the module has not triggered. The following branch of the logic script is only executed if the timer has not been triggered.
Timer running	Data point of the module that returns TRUE if the timer has been started but has not yet been triggered.
Remaining seconds	Returns the number of seconds that must elapse before the timer is triggered.

APPLICATION EXAMPLE

Given are 1 Homematic push-button and 1 Homematic actuator for switching a light:

Licht Wand	HomeMatic Device	
INHIBIT	Boolean	false
STATE	Boolean	false
WORKING	Boolean	false
Taster	HomeMatic Device	
INSTALL_TEST	Boolean	false
PRESS_LONG	Boolean	true
PRESS_SHORT	Boolean	true

A logic script is now to be created in the logic script that switches on a light when the push-button is triggered and switches it off again after one minute.
You can implement this requirement by using a one-time timer module; predefined instance modules for the push-button or activating the light can be found in the object tree of the logic script designer.



Functionality:

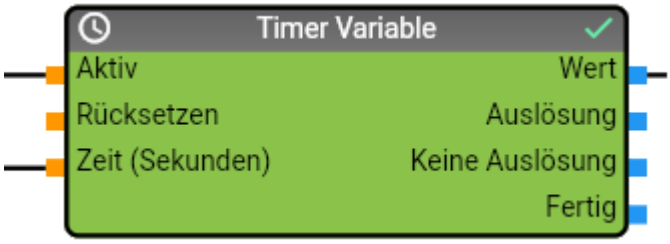
The PRESS_SHORT variable of a Homematic push-button always has the value TRUE; when the push-button is pressed, the value of the variable is updated and only the date/time of the variable is updated. This update can be evaluated with the On update module and used to control other modules.

In the current example, the light is switched on when the PRESS_SHORT variable is updated and at the same time a timer with 60 seconds is triggered by setting the "Active" input in the One-time timer module.

The logic plan script is executed a second time by this timer after 60 seconds and switches the light off again via the Non module.

Timer Variable

The "Timer Variable" module enables the implementation of a timer based on a variable. After activating the timer, the remaining time until the timer is triggered is counted down.



DATA POINTS INPUT

NAME	DESCRIPTION
Active	Activation of the timer, if TRUE the timer is activated and triggers the logic script again after a certain time.
Reset	Deactivation of the timer; if TRUE, an active timer can be canceled.
Time (seconds)	Time in seconds after which the timer should be triggered.

DATA POINTS OUTPUT

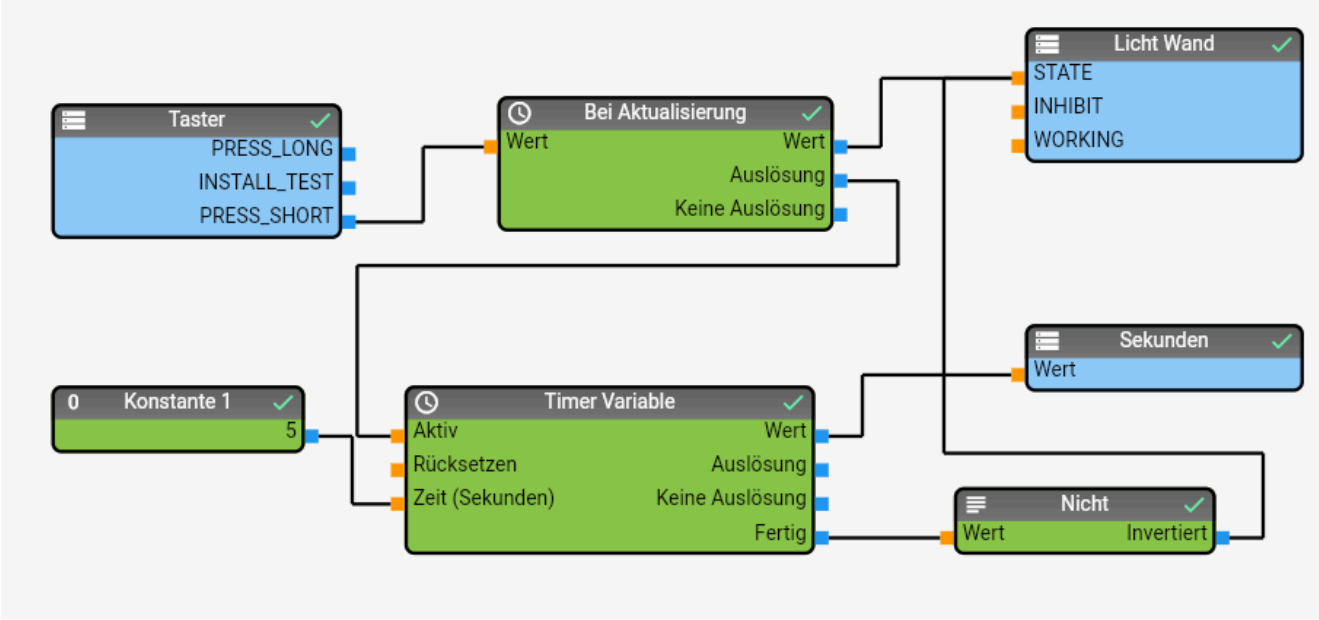
NAME	DESCRIPTION
Value	This data point of the module must be connected to a variable of type integer. The variable realizes the display of the remaining timer time.
Triggering	Data point of the module that returns TRUE when the module has triggered. This occurs every second when the timer is active. The subsequent branch of the logic script is only executed when the timer has been triggered.
No triggering	Data point of the module that returns TRUE if the module has not triggered. The subsequent branch of the logic script is only executed if the timer has not been triggered.
Done	Data point of the module that returns TRUE if the timer has ended. The following branch of the logic script is only executed if the timer has ended.

APPLICATION EXAMPLE

Given are 1 Homematic button, 1 Homematic actuator for switching a light and a variable for displaying the remaining time of a staircase timer (normal variable of type integer, which has a profile with suffix "Sec" assigned).

Licht Wand	HomeMatic Device	
INHIBIT	Boolean	false
STATE	Boolean	false
WORKING	Boolean	false
Taster	HomeMatic Device	
INSTALL_TEST	Boolean	false
PRESS_LONG	Boolean	true
PRESS_SHORT	Boolean	true
Sekunden	Integer	0

A logic script is now to be created in Logic Scripts that switches on a light when the push-button is triggered and switches it off again after one minute. In addition, the remaining time in seconds should also be visualized
You can implement this requirement by using a Timer Variable module; predefined instance modules for the push-button or activating the light can be found in the object tree of the Logic Script Designer.



How it works:

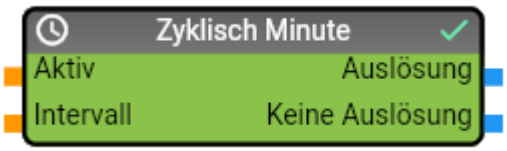
The PRESS_SHORT variable of a Homematic push-button always has the value TRUE; when the push-button is pressed, the value of the variable is updated and only the date/time of the variable is updated. This update can be evaluated with the On update module and used to control other modules.

In the current example, when the PRESS_SHORT variable is updated, the light is switched on and at the same time a timer with 60 seconds is triggered by setting the "Active" input in the Timer Variable module, which uses the "Remaining time" variable to count down the seconds. The logic script script is executed by this timer every second and updates the value in the "Remaining time" variable.

Once the timer has reached the value 0, the lamp is switched off again via the "Done" output and a non-module.

Timer cyclic minute

The "Timer cyclic minute" module allows you to implement a timer that calls up the logic script cyclically at the set interval.



PROPERTIES

NAME	DESCRIPTION
Interval	Interval in minutes for triggering the timer.

DATA POINTS INPUT

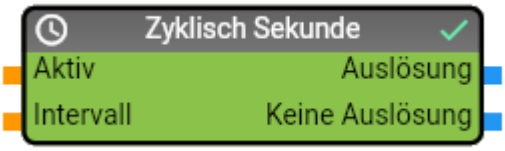
NAME	DESCRIPTION
Active	Activation of the timer; if TRUE, the timer is activated and triggers the logic script every minute at the specified interval.
Interval	Data point of the module for controlling the interval. If the input is not connected, the value of the "Interval" property is used for triggering.

DATA POINT OUTPUT

NAME	DESCRIPTION
Triggering	Data point of the module that returns TRUE when the module has triggered. This occurs every second when the timer is active. The subsequent branch of the logic script is only executed if the timer has been triggered.
No triggering	Data point of the module that returns TRUE if the module has not triggered. The following branch of the logic script is only executed if the timer has not been triggered.

Timer cyclic second

The "Timer cyclic second" module enables the implementation of a timer that calls up the logic script cyclically with the set interval.



PROPERTIES

NAME	DESCRIPTION
Interval	Interval in seconds for triggering the timer.

DATA POINTS INPUT

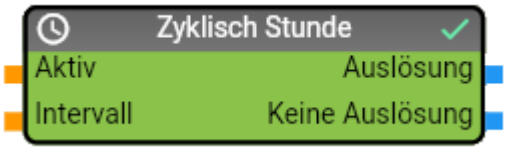
NAME	DESCRIPTION
Active	Activation of the timer; if TRUE, the timer is activated and triggers the logic script at the specified interval every second.
Interval	Data point of the module for controlling the interval. If the input is not connected, the value of the "Interval" property is used for triggering.

DATA POINT OUTPUT

NAME	DESCRIPTION
Triggering	Data point of the module that returns TRUE when the module has triggered. This occurs every second when the timer is active. The subsequent branch of the logic script is only executed if the timer has been triggered.
No triggering	Data point of the module that returns TRUE if the module has not triggered. The following branch of the logic script is only executed if the timer has not been triggered.

Timer cyclic hour

The "Timer cyclic hour" module enables the implementation of a timer that calls up the logic script cyclically with the set interval.



PROPERTIES

NAME	DESCRIPTION
Interval	Interval in hours for triggering the timer.

DATA POINTS INPUT

NAME	DESCRIPTION
Active	Activation of the timer; if TRUE, the timer is activated and triggers the logic script at the specified interval every hour.
Interval	Data point of the module for controlling the interval. If the input is not connected, the value of the "Interval" property is used for triggering.

DATA POINT OUTPUT

NAME	DESCRIPTION
Triggering	Data point of the module that returns TRUE when the module has triggered. This occurs every second when the timer is active. The subsequent branch of the logic script is only executed if the timer has been triggered.
No triggering	Data point of the module that returns TRUE if the module has not triggered. The subsequent branch of the logic script is only executed if the timer has not been triggered.

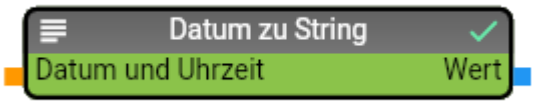
Conversion

Modules in the "Conversion" group enable the integration of modules for converting data into another format.

Date to String	Converts a Unix timestamp into a string
Date to time	Converts a Unix timestamp into the individual time components.
Format message	Formats several values into one message.
Invert value	Inverts a value
Value to Boolean	Converts a value into a Boolean.
Value to float	Converts a value into a float.
Value to integer	Converts a value into an integer.
Value to string	Converts a value into a string.
Convert range	Convert a value from one value range to another.
Time to Date	Converts individual time components into a Unix timestamp.

Date to String

The module "Date to String" offers the possibility to convert a Unix timestamp into a string.



PROPERTIES

NAME	DESCRIPTION
Date format	Format to be used for the conversion to a string.

Any PHP format can be entered for converting a date. Details on the possible formats can be found in the PHP documentation. Some general formats are already predefined and can be selected via the dropdown

DATA POINTS INPUT

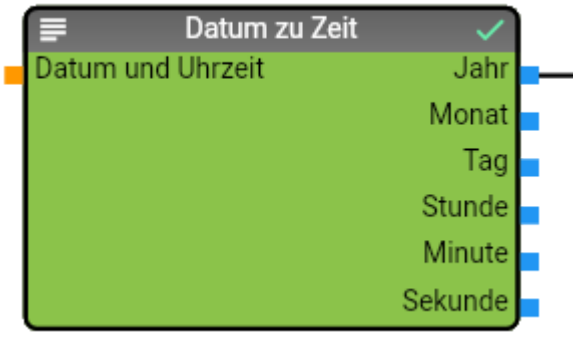
NAME	DESCRIPTION
Timestamp	Unix timestamp to be converted.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	Converted timestamp as string.

Date to time

The module "Date to Time" offers the possibility to split a Unix timestamp into the individual time components.



DATA POINTS INPUT

NAME	DESCRIPTION
Timestamp	Unix timestamp to be converted.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Year	The year from the Unix timestamp.
Month	The month from the Unix timestamp.
Day	The day from the Unix timestamp.
Hour	The hours from the Unix timestamp.
Minute	The minutes from the Unix timestamp.
Second	The seconds from the Unix timestamp.

Format message

The "Format message" module offers the option of assembling a string with one or more variable values. The module supports up to 8 different variables.



PROPERTIES

NAME	DESCRIPTION
Number of inputs	Number of inputs that the module should make available.
Format String	PHP Format String for formatting the string

DATA POINTS INPUT

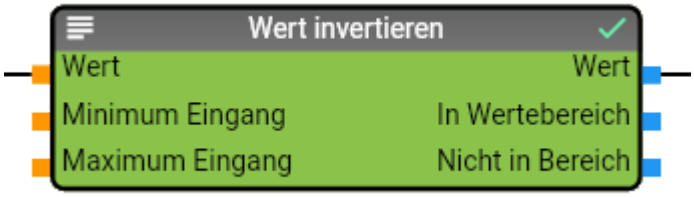
NAME	DESCRIPTION
Value 1	Value 1 for forming the message
Value 2	Value 2 for forming the message
Value 3	Value 3 for forming the message
Value 4	Value 4 for forming the message
Value 5	Value 5 for forming the message
Value 6	Value 6 for forming the message
Value 7	Value 7 for forming the message
Value 8	Value 8 for forming the message

DATA POINTS OUTPUT

NAME	DESCRIPTION
Result	Result message from the format string and the input variables.

Invert value

The "Invent value" module offers the option of inverting a value. The respective minimum or maximum can either be configured via the properties of the module or, alternatively, the module also offers inputs for the values.



PROPERTIES

NAME	DESCRIPTION
Outputs Range	Activates the data points at the "In range" and "Not in range" output. These data points are not activated by default.
Inputs Limit	Activates the data points at the input for the limits ("Minimum input" and "Maximum input"). These data points are not activated by default.
Maximum Input	Specifies the maximum for the inversion of the value, is only used if the data points for the limits are not activated.
Minimum input	Specifies the minimum for the inversion of the value, is only used if the data points for the limits are not activated.

DATA POINTS INPUT

NAME	DESCRIPTION
Value	Value to be inverted
Minimum input	Minimum value for the inversion. This data point is only available if the "Inputs limit" property has been activated.
Maximum input	Maximum value for the inversion. This data point is only available if the "Inputs Limit" property has been activated.

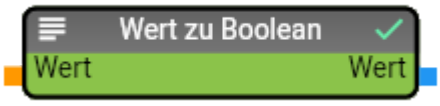
DATA POINT OUTPUT

NAME	DESCRIPTION
Value	Converted value
In range	Returns **TRUE** if the value to be inverted was within the minimum and maximum. This data point is only available if the "Outputs range" property has been activated.

NAME	DESCRIPTION
Not in range	Returns **TRUE** if the value to be inverted was outside the minimum and maximum. This data point is only available if the "Outputs range" property has been activated.

Value to Boolean

The "Value to Boolean" module offers the option of converting a value into a Boolean.



DATA POINTS INPUT

NAME	DESCRIPTION
Value	Value to be converted.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	Converted value as Boolean.

Value to float

The "Value to float" module offers the option of converting a value into a float.



DATA POINTS INPUT

NAME	DESCRIPTION
Value	Value to be converted.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	Converted value as float.

Value to integer

The "Value to integer" module offers the option of converting a value to an integer.



DATA POINTS INPUT

NAME	DESCRIPTION
Value	Value to be converted.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	Converted value as an integer.

Value to string

The "Value to string" module offers the option of converting a value into a string.



DATA POINTS INPUT

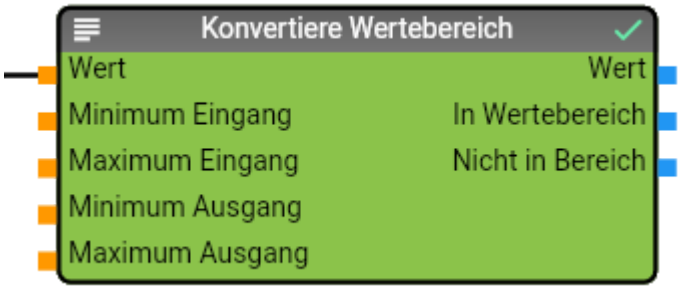
NAME	DESCRIPTION
Value	Value to be converted.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	Converted value as a string.

Convert range

The "Convert range" module offers the option of converting a value from one value range to another value range. For example, it is possible to use the module to convert a value in the range 0 to 1 to the value range 0 to 100.



PROPERTIES

NAME	DESCRIPTION
Outputs range	Activates the data points at the "In range" and "Not in range" output. These data points are not activated by default.
Inputs Limit	Activates the data points at the input for the limits ("Minimum input" and "Maximum input"). These data points are not activated by default.
Maximum input	Specifies the maximum for the input range of the value, is only used if the data points for the limits are not activated.
Minimum input	Specifies the minimum for the input range of the value, is only used if the data points for the limits are not activated.
Maximum output	Specifies the maximum for the output range of the value, is only used if the data points for the limits are not activated.
Minimum output	Specifies the minimum for the output range of the value, is only used if the data points for the limits are not activated.
Rounding to digits	Specifies the number of digits to be used for rounding the converted value.

DATA POINTS INPUT

NAME	DESCRIPTION
Value	Value to be inverted
Minimum input	Minimum value for the input range. This data point is only available if the "Connector for limits" property has been activated.
Maximum input	Maximum value for the input range. This data point is only available if the "Connector for limits" property has been activated.
Minimum output	Minimum value for the output range. This data point is only available if the "Connector for limits" property has been activated.

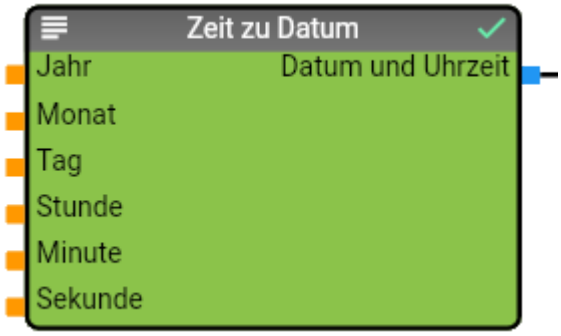
NAME	DESCRIPTION
Maximum output	Maximum value for the output range. This data point is only available if the "Connector for limits" property has been activated.

OUTPUT DATA POINTS

NAME	DESCRIPTION
Value	Converted value
In range	Returns "true" if the value to be converted was within the minimum and maximum. This data point is only available if the "Connector for range check" property has been activated.
Not in range	Returns "true" if the value to be converted was outside the minimum and maximum. This data point is only available if the "Connector for range check" property has been activated.

Time to Date

The module "Time to Date" offers the possibility to convert single time components into a Unix timestamp.



DATA POINTS INPUT

NAME	DESCRIPTION
Year	The year for the Unix timestamp (if no value is set, the current year is used).
Month	The month for the Unix timestamp (if no value is set, the current month is used).
Day	The day for the Unix timestamp (if no value is set, the current day is used).
Hour	The hours for the Unix timestamp (if no value is set, the current hour is used).
Minute	The minutes for the Unix timestamp (if no value is set, the current minute is used).
Seconds	The seconds for the Unix timestamp (if no value is set, the current second is used).

DATA POINTS OUTPUT

NAME	DESCRIPTION
Timestamp	Unix timestamp formed from the individual time components.

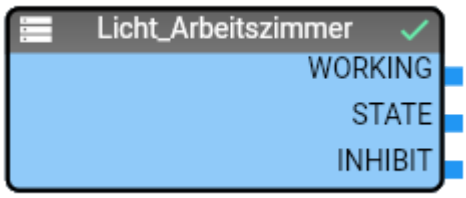
Read/ write variables

Modules in the "Read/write variables" group enable variables or instances to be read and set in IP-Symcon.

Read Instance	Instances as input of a logic script
Write Instance	Instances as the output of a logic script
Read Variable	Variable as input of a logic script
Write Variable	Variable as the output of a logic script
Logic Script Variable	Create variable through the logic script

Read Instance

The "Read instance" module offers the option of integrating instances as logic script input; variables of the corresponding instance can be used to control other modules.



PROPERTIES

NAME	DESCRIPTION
ID	The ID of the instance in IP-Symcon.
Logic Script execution	Enables the automatic execution of the logic script when a variable is changed.

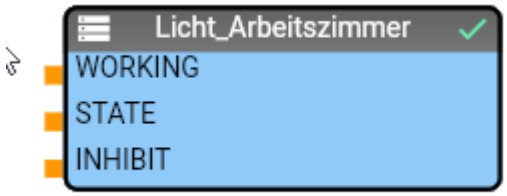
The following values are available for selection during logic script execution:
None - no automatic execution of the logic script
On change - automatic execution of the logic script when the value of a linked variable changes.
On update - automatic execution of the logic script when the value of a linked variable is updated.
As action script - the logic script script is stored as an action script for each variable linked in the instance. When operated in the GUI (WebFront or IPSView), the logic script is called up with the new value.
Additional information on the topic of action scripts.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value instance variable	The values of all instance variables are available as data points.

Write Instance

The "Write Instance" module offers the option of integrating instances as an output of the logic script.



PROPERTIES

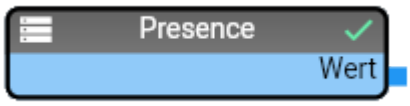
NAME	DESCRIPTION
ID	The ID of the instance in IP-Symcon.

DATA POINTS INPUT

NAME	DESCRIPTION
Value instance variable	Instance variables that have an action script stored are available as data points.

Read Variable

The "Read Variable" module offers the option of integrating normal variables as logic script input.



PROPERTIES

NAME	DESCRIPTION
ID	The ID of the variable in IP-Symcon.
Logic Script execution	Enables the automatic execution of the logic script when the variable is changed.

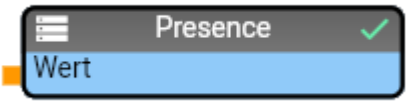
The following values are available for selection for logic script execution:
None - no automatic execution of the logic script
On change - automatic execution of the logic script when the value of the variable changes.
On update - automatic execution of the logic script when the value of the variable is updated.
As action script - the logic script script is stored as an action script for the variable. When operated in the GUI (WebFront or IPSView), the logic script is called up with the new value.
Additional information on the topic of action scripts.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	The value of the variable.

Write Variable

The "Write Variable" module offers the option of integrating normal variables as an output of the logic script.



PROPERTIES

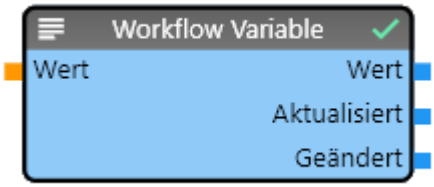
NAME	DESCRIPTION
ID	The ID of the variable in IP-Symcon.
Use action script	If TRUE, the action script is used to set the variable value. If no action script is available, the value of the variable is set directly.
Sender	Value for the SENDER variable when the action script is called.

DATA POINTS INPUT

NAME	DESCRIPTION
Value	The value of the variable.

Logic Script Variable

The "Logic Script Variable" module offers the option of having a variable created by the logic script itself.



PROPERTIES

NAME	DESCRIPTION
Data type	Data type of the variable
Default value	Value of the variable to be assigned after the initial creation of the variable.
Profile name	Name of the profile that is to be assigned to the variable after the initial creation of the variable.
Variable name	Name of the variable
Logic Script as action script	If this option is activated, the logic script itself is called up to set a value via a visualization. In this case, you must take care of assigning the new value in the logic script, otherwise the value will not be set.

DATA POINTS INPUT

NAME	DESCRIPTION
Value	Value of the variable.

DATA POINTS OUTPUT

NAME	DESCRIPTION
Value	Current value of the variable
Update	Signals an update of the variable via the input
Change	Signals a change of the variable via the input

Compare

Modules in the "Compare" group allow you to compare values in the logic script.

Equal	Comparing values for equality or inequality
Larger/smaller	Compare values to Greater than, Greater equal to, Equal to, Smaller equal to or Smaller.
In value range	Checks a value for a specific value range

Equal

The "Equal" module offers the opportunity to compare different values for equality or inequality.



DATA POINTS INPUT

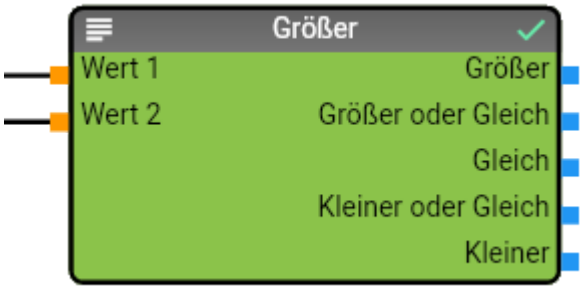
NAME	DESCRIPTION
Value 1	Data point for comparison value 1
Value 2	Data point for comparison value 2

DATA POINT OUTPUT

NAME	DESCRIPTION
Equals	Returns TRUE if the value at input value 1 is equal to the value at input value 2.
Not equal	Returns TRUE if the value at input value 1 is not equal to the value at input value 2

Larger/smaller

The "Larger/smaller" module offers the option of comparing different values for Larger, Larger equal, Equal, Smaller equal or Smaller.



DATA POINTS INPUT

NAME	DESCRIPTION
Value 1	Data point for comparison value 1
Value 2	Data point for comparison value 2

DATA POINT OUTPUT

NAME	DESCRIPTION
Greater	Returns TRUE if the value at input value 1 is greater than the value at input value 2.
Greater than or equal to	Returns TRUE if the value at input Value 1 is greater than or equal to the value at input Value 2.
Equal	Returns TRUE if the value at input Value 1 is equal to the value at input Value 2.
Less than or equal to	Returns TRUE if the value at input Value 1 is less than or equal to the value at input Value 2.
Smaller	Returns TRUE if the value at input Value 1 is smaller than the value at input Value 2.

In value range

The "In value range" module offers the option of checking whether a value is within a certain value range. The limits for the value range can be defined via properties or via separate module inputs.



PROPERTIES

NAME	DESCRIPTION
Inputs Limits	Activates the data points at the input for the limits ("Lower Limit" and "Upper Limit"). These data points are not activated by default.
Upper Limit	Specifies the maximum for the input range of the value, is only used if the data points for the limits are not activated.
Lower Limit	Specifies the minimum for the input range of the value, is only used if the data points for the limits are not activated.

DATA POINTS INPUT

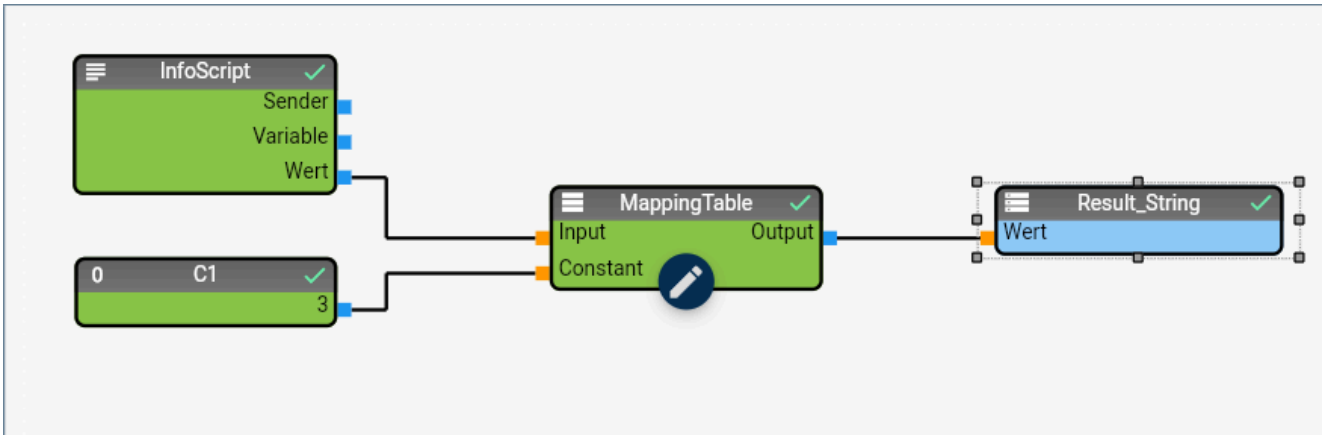
NAME	DESCRIPTION
Value	Value to be evaluated
Lower Limit	Minimum value for the input range. This data point is only available if the "Input limits" property has been activated.
Upper Limit	Maximum value for the input range. This data point is only available if the "Inputs Limits" property has been activated.

DATA POINT OUTPUT

NAME	DESCRIPTION
In range	Returns TRUE if the value to be converted was within the minimum and maximum range.
Not in range	Returns TRUE if the value to be converted was outside the minimum and maximum.
Limited value	If the value is outside the upper or lower limit, the value is corrected to the respective limit.

Drawing Area

The drawing area is located in the middle of the Logic Script Editor. Here you can combine the individual modules into a logic script.



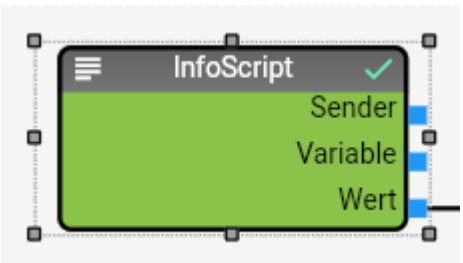
The drawing area is intuitive to use; the functionality is very similar to other programs with graphical components. In this section you will find an overview of the available functions and actions.

PLACE OBJECT

To place a new object on the drawing area:
  Select the object in the object tree or from the Repository menu.
  Drag the selected control element onto the drawing area (drag & drop).
The object is drawn with the default values from IP-Symcon (size, color ...). You must then customize it for your purposes (see below).

SELECT / DESELECT OBJECTS

To select a single object:
  Click on the object.
To select several objects at the same time:
  Hold down <Shift> while clicking on different objects.
To select all objects on the drawing area at the same time:
  Press <Ctrl>+A.
You can recognize selected objects by the eight handles (boxes) on the frame:



To cancel the selection again:

Click on an empty area of the drawing area.
The entire selection made is canceled.

CHANGE PROPERTIES

The stored values and properties of the selected object are saved in the Property-Editor. You can change them there as required.
If you have selected several objects at the same time, you can change the properties that appear in all selected objects (properties that only appear in individual objects are not displayed). Your change will then be applied to all currently selected objects at the same time.

CUT, COPY, PASTE, DELETE

Logic Scripts supports the standard commands for manipulating objects and content:

NAME	DESCRIPTION
Cut	Cut context menu or the key combination **<Ctrl>+X**.
Copy	Copy context menu or the key combination **<Ctrl>+C**.
Paste	Paste context menu or the key combination **<Ctrl>+V**.
Delete	Delete context menu or the **<Delete>** key.

UNDO / REDO

Logic Scripts supports "Undo / Redo", i.e. canceling or restoring changes:

NAME	DESCRIPTION
Undo	Menu+Undo or the key combination **<Ctrl>+Z**.
Restore	Menu+Restore or the key combination **<Ctrl>+<Shift>+Z**.

ZOOMING A LOGIC SCRIPT

The Logic Scripts Designer offers the option to zoom in and out of a view in steps of 10%. This only changes the display in the Designer; the display on the client is not changed.

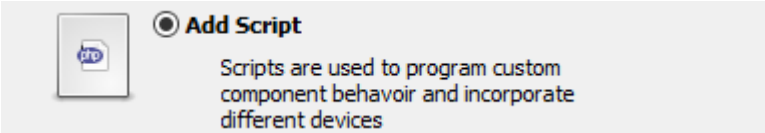
NAME	DESCRIPTION
Zoom In	Select Menu+Zoom In or press the <Ctrl>++ key combination to enlarge the display by one level.
Zoom Out	Select Menu+Zoom Out or press the <Ctrl>+- key combination to reduce the display by one level.
Zoom	Select Menu+Zoom to set the zoom functionality to a specific value. After pressing the button, a drop-down window is displayed that allows you to select the zoom level.

PHP Scripts

A PHP script is a concatenation of more or less complex commands. IP-Symcon uses and supports every functionality of the programming language PHP. Numerous examples and further documentation are found in our community forum and throughout the internet.

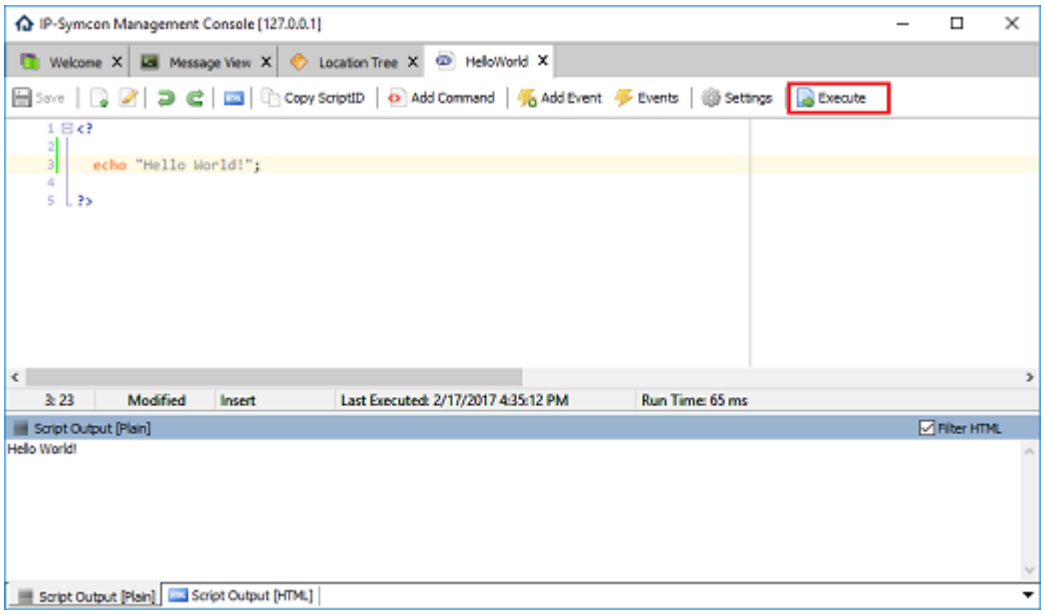
ADD PHP SCRIPT

A new PHP script can be added to a project via "+" -> "Automation" -> "PHP Script". It is recommended to give the PHP script a unique and meaningful name.
The name can be changed at any time and the PHP script can be moved within the tree view via Drag & Drop. Alternatively, the PHP script can be directly created at the wanted position via "Right Click" -> "Add object" -> "Automation" -> "PHP Script.



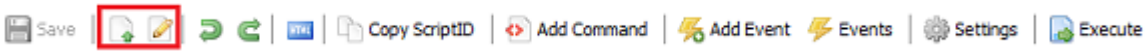
EDIT AND EXECUTE PHP SCRIPTS

A PHP script is required to begin with a PHP Tag (<?php). Every command needs to end with a semicolon.



The PHP script can be executed manually via "Execute".
In that case, the execution is saved automatically. The time stamp of the last execution as well as the runtime are updated. The view can be changed between [Text] and [HTML].
Faulty inputs in the PHP script can be corrected via "Undo" or "Redo."
"Export to HTML" shows the complete PHP script in a browser.

IMPORT AND RENAME PHP SCRIPTS



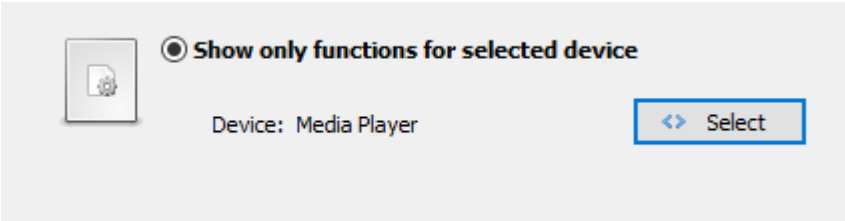
When an existing PHP script (e.g., from the forum) was downloaded, it can be inserted quickly and uncomplicated with the button "Import". After pressing the button "Import" the launched menu can be used to select the wanted file. By default, all PHP scripts within IP-Symcon are stored as files with names according to their ID. For better readability, the file name can be changed at any time with the button "Rename".

ADD COMMAND

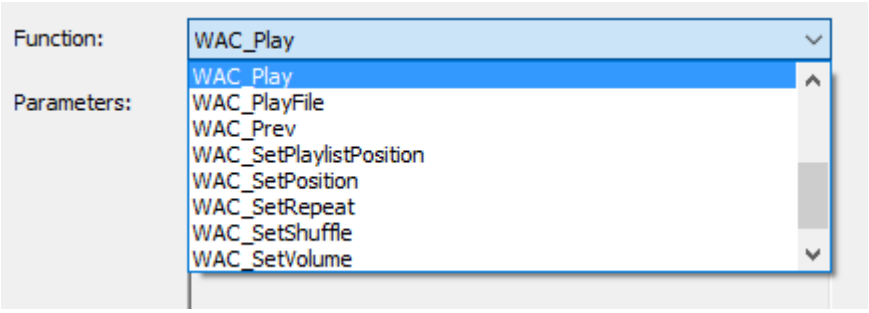
The button "Add Command" creates a new command at the position of the cursor.



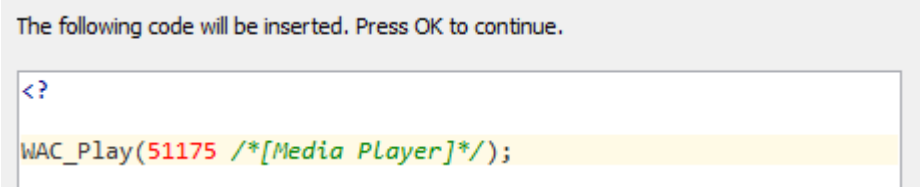
To handle the multitude of commands, it is recommended to select the affected device. Here is an example for the sound output "Media Player":



A selection shows all device specific functions:



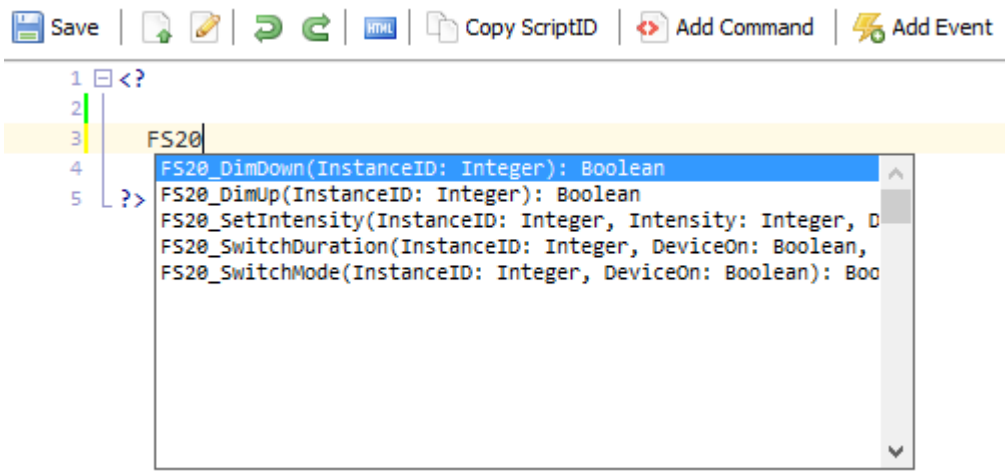
The result is the command with the according ID of the selected device.



The documentation for all selectable commands is shown in the Command Reference and Module Reference. Another possibility is the search function in the upper right area of the IP-Symcon website. This function offers a quick way to find the documentation and examples for a command.

SHOW AND USE COMMAND LIST

Autocompletion can be used when entering commands by pressing "Ctrl + Space" within a PHP script. The shown list contains all IP-Symcon functions. The selection can be narrowed by entering the first letters of a command. For example, typing "'f' + 's' + '2' + '0'" shows all FS20 commands. The wanted command can be chosen by double click or via arrows + Enter.



SYSTEM VARIABLES IN PHP SCRIPTS

A complete list is found at System Variables.

HINT

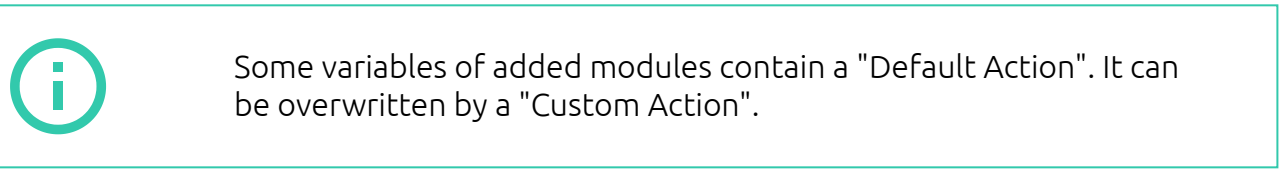
A list of all keyboard-shortcuts.

Action Scripts

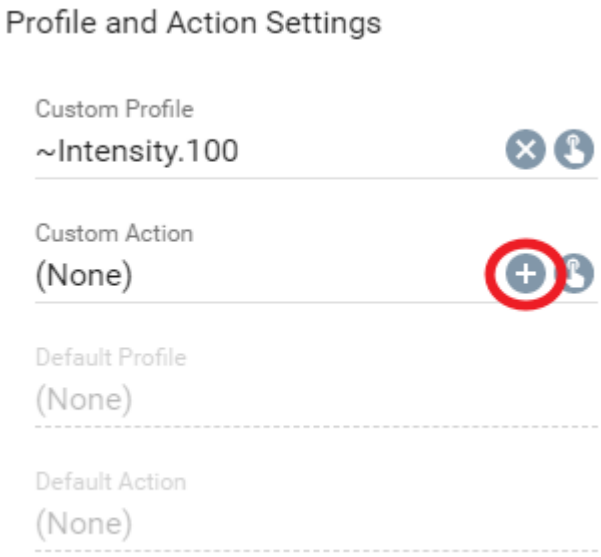
Action scripts are scripts that are called when a Variable is clicked in a Visualization (e.g., WebFront, Mobile).
The action script needs to be linked to the corresponding variable as "Custom Action".
Special variables for called scripts are shown in System Variables.
An action script can be used by multiple variables. The most common variant is shown below: "Set Variable only".

SELECT ACTION SCRIPT

The selection of an action script is the same as the selection of a Variable Action .



Some variables of added modules contain a "Default Action". It can be overwritten by a "Custom Action".



CREATE ACTION SCRIPT

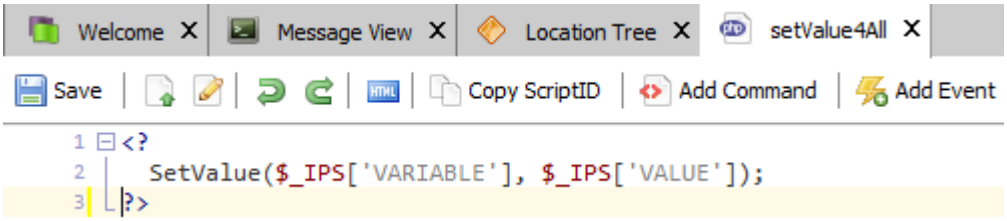
An action script is a regular script file. It becomes an action script by linking it to a variable.
An action script should describe the switching process or at least a command that sets the variable to the requested state:

EXAMPLE 1

```
//Set Variable only
SetValue($_IPS['VARIABLE'], $_IPS['VALUE']);
```

EXAMPLE 2

```
//Set variable accordingly when switching was sucessful
if (FS20_SwitchMode(12345, (boolean)$_IPS['VALUE'])) {
SetValue($_IPS['VARIABLE'], $_IPS['VALUE']);
}
```



BAD ACTION SCRIPTS

Some criterias should be regarded when creating an action script. Otherwise, unexpected behavior could appear.

SETTING A VARIABLE WITHOUT CHECKING FOR SUCESSFUL SWITCHING

Most functions return a "True" when they succeed. This should be checked.
If this is not done, the state of IP-Symcon and the device are not consistent any more. Thus, the configured automatic control could fail.

```
//Variable is set to new value even if the switching was not sucessful.
//This can lead to consistency faults.
FS20_SwitchMode(12345, (boolean)$_IPS['VALUE']);
SetValue($_IPS['VARIABLE'], $_IPS['VALUE']);
```

SWITCHING A DEVICE VIA EVENT WHEN THE VARIABLE CHANGES

There should be no additional event to switch a device after the corresponding variable was changed. This approach cannot detect switching problems of the device. Thus, error messages are hidden and the variable value could be different from the device value.
Consider a dimmer that is set from 0% to 50% via WebFront.
The variable is instantly set to 50% und the switch command is executed asynchonously by the additional event. The action script cannot detect if the switch command failed and the variable within IP-Symcon stays at 50% even though the device value is 0%.



System Variables

System variables are variables that are available in every PHP script. The user can utilize them to create efficient PHP scripts that can solve generic tasks. The according system variables are set automatically when the PHP script is called, depending on the trigger. The following tables providing an overview over the variables which are available in which PHP scripts.



Regard upper and lower case when writing these variables (Case Sensitive)!

GENERAL SYSTEM VARIABLES

The following system variables are always available.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['SELF']	ScriptID of the current PHP script
$_IPS['THREAD']	ThreadID of the current PHP script
$_IPS['SENDER']	Trigger of the current PHP script. Further global variables are possible, depending on the value. Possible values are further specified at Additional System Variables

ADDITIONAL SYSTEM VARIABLES

The following system variables are available, depending on the value of $_IPS['SENDER'].

Execute
HeatingControl
RegisterVariable
RunScript
Shutdown
ShutterControl
Startup
StatusEvent
TimerEvent
Variable
VoIP (since version 5.2)
Watchdog
WebFront
WebHook (since version 4.0)
WebInterface
WebOAuth (since version 4.0)

ACTION

When the PHP script was executed by a RequestAction .

SYSTEM VARIABLE	DESCRIPTION
$_IPS['SENDER']	By default, this "Action" can be set individually with RequestActionEx
$_IPS['VALUE']	Value with which the variable was called
$_IPS['VARIABLE']	ID of the variable that is to be switched

EXECUTE

When the PHP script was run from the administration console.
No additional system variables.

HEATINGCONTROL

When the PHP script was called from an event of HeatingControl.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['INSTANCES']	IDs of the sending instances
$_IPS['INVERTS']	States if a device is inverted (true/false)
$_IPS['VALUE']	States if a device should heat (true) or not (false)

REGISTERVARIABLE

When the PHP script was called by a RegisterVariable instance.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['INSTANCE']	ID of the triggering RegisterVariable instance
$_IPS['VALUE']	Value of the puffer that was received from the splitter or I/O module
$_IPS['CLIENTIP'] (only for I/O ServerSocket)	Received IP adress of the Client
$_IPS['CLIENTPORT'] (only for I/O ServerSocket)	Receiving port of the Client
$_IPS['TYPE']	Status of the receiving port (0 = Data; 1 = Connected; 2 = Disconnected)

RUNSCRIPT

When the PHP script was called by an IPS_RunScript function. In addition, IPS_RunScriptEx can be called to pass additional parameters.
No additional system variables.

SHUTDOWN

During IP-Symcon Shutdown (see EventControl).
No additional variables.

SHUTTERCONTROL

When the PHP script was called by a ShutterControl Module.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['DIRECTION']	Moving direction: **0** = Stop **1** = Up **2** = Down
$_IPS['DURATION']	Moving duration in milliseconds
$_IPS['INSTANCE']	InstanceID that was set in ShutterControl
$_IPS['INSTANCE2']	InstanceID #2 that was set in ShutterControl

STARTUP

During IP-Symcon Startup (see EventControl).
No additional system variables.

STATUSEVENT

When the PHP script is called by a state change of an instance.
Further informations are found at EventControl and IPS_GetInstance.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['INSTANCE']	InstanceID for state change
$_IPS['STATUS']	State of the instance. A list of possible values is found here: IPS_GetInstance
$_IPS['STATUSTEXT']	A short text according to the state

TIMEREVENT

When the PHP script was called by a cyclic or a schedule event.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['ACTION']	ID of the calling action (only schedule event)
$_IPS['EVENT']	ID of the triggered event
$_IPS['TARGET']	ID of the superior object

VARIABLE

When the PHP script was called be a trigger event.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['EVENT']	ID of the calling event
$_IPS['OLDCHANGED'] (since Version 5.1)	Timestamp of the last change of the old value
$_IPS['OLDVALUE']	Value of the affected variable before switching
$_IPS['OLDUPDATED'] (since Version 5.1)	Timestamp of the last update of the old value
$_IPS['TARGET']	ID of the superior object
$_IPS['TRIGGER']	Type of the calling event
$_IPS['VALUE']	Value of the affected variable when switching
$_IPS['VARIABLE']	ID of the affected variable

VOIP (SINCE VERSION 5.2)

When the PHP script is executed by an event of the VoIP Module

SYSTEM VARIABLE	DESCRIPTION
$_IPS['CONNECTION']	ChannelID of the connection
$_IPS['DATA']	The variable can contain the following values depending on $_IPS['EVENT']: **Incoming:** Phone number from which the call is made **PlayFinish:** File name of the played file **DTMF:** Value of the pressed key
$_IPS['EVENT']	The variable can contain the following values: **Incoming:** Incoming call **Connect:** Established connection **Disconnect:** Terminating connection **PlayFinish:** Audio file has finished playing **DTMF:** Received a DTMF sound
$_IPS['INSTANCE'] (ab Version 5.4)	InstanceID of the VoIP Instance

WATCHDOG

When the PHP script is called by the Watchdog.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['STATUSTEXT']	The passed state text
$_IPS['VALUE']	Value of the variable that is outside of its interval
$_IPS['VARIABLE']	ID of the variable that is outside of its interval

WEBFRONT

When the PHP script was called by the WebFront, Tile Visualization, or IPSView.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['CONFIGURATOR']	ID of the currently used visualization
$_IPS['VALUE'] (only for action script)	New value of the variable
$_IPS['VARIABLE'] (only for action script)	ID of the changing variable

WEBHOOK (SINCE VERSION 4.0)

When the PHP script was called by a WebHook control.

SYSTEM VARIABLE	DESCRIPTION
$_SERVER['HOOK']	Complete URL of the called WebHook

Furthermore, the WebHook type provides the same variables as WebInterface.

WEBOAUTH (SINCE VERSION 4.0)

When the PHP script was called by a OAuth Control.

SYSTEM VARIABLE	DESCRIPTION
$_SERVER['OAUTH']	Complete URL of the called WebOAuth

Furthermore, the WebOAuth type provides the same variables as WebInterface.

WEBINTERFACE

When the PHP script was called by the Webserver, e.g., in the 'user' folder.

SYSTEM VARIABLE	DESCRIPTION
$_IPS['INSTANCE']	ID of the calling Webserver instance
$_GET	Received data via GET
$_POST	Received data via POST
$_SERVER['DOCUMENT_ROOT']	Contains the Document Root folder that contains the currently executed PHP script as it is set within the configuration of the server.
$_SERVER['PHP_AUTH_PW']	If HTTP authentication is used, this variable contains the password the user provided
$_SERVER['PHP_AUTH_USER']	If HTTP authentication is used, this variables contains the user name the user provided
$_SERVER['PHP_SELF']	File/Path of the started PHP script
$_SERVER['QUERY_STRING']	If available, the query string that was used to access the website
$_SERVER['REMOTE_ADDR']	The IP adress that the user used to access the website
$_SERVER['REMOTE_PORT']	The port that the user used to access the website

SYSTEM VARIABLE	DESCRIPTION
$_SERVER['REQUEST_METHOD']	Contains the request method that was used for the access, e.g., "GET", "HEAD", "POST", or "PUT"
$_SERVER['REQUEST_URI']	The URI that was used to access the current website, e.g., "/index.html"
$_SERVER['SCRIPT_NAME']	Contains the path of the current PHP script. This can be useful for websites that link to themselves

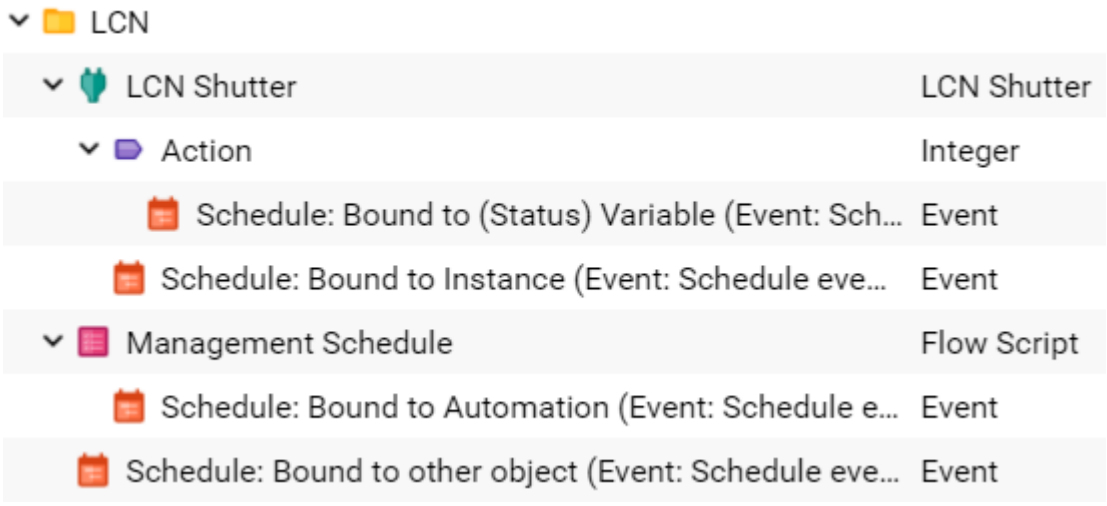
In addition, all request headers are added in capital letters with a HTTP*_ prefix. Minus symbols (-) are changed to underscore symbols (_*). For example, the header 'User-Agent is provided as $_SERVER['HTTP_USER_AGENT'].

Events

An event is an automatic action that is connected to a condition.
Executed events modify binded objects. The binding is done by attaching the event to the corresponding object in the IP-Symcon management console. (see: (link: page://qks1ZxXFt67qdb3I#Binding_to_Action_Target text: Binding Types))
Events are executed depending on the event type and the configured condition. (see: (link: #Event_Types text: Event Types))

BINDING TO ACTION TARGET

Die bound object is used as target for an Action . As is common with actions, an action can be selected depending on the target and configured.
The action of an event determines what happens when the event is triggered. For more information, see Actions



EVENT TYPES

There are three event types.

EVENT TYPE	DESCRIPTION
Trigger	An event that is connected to a specific variable.
Cyclic	The event is run at a defined time point once or repeatedly.
Schedule	A schedule that is configured in the WebFront runs the event.

Access to system variables:
IP-Symcon automatically offers a number of variables that can be accessed within a script.
However, these are only available if the script was called by an event.
See:
System Variables

CONDITIONS

Conditions expand events by further options that need to be fulfilled for the event to trigger. This can be used to realized multiple conditions.
Additional conditions can be added in the expansion panel "Further Conditions". In this panel, new conditions can be added by clicking "Add".
The types Variable, Date, Day of the week, and Time can be selected. For the type "Variable", a variable needs to be selected. Afterwards, the remaining options will appear.
Finally, the comparison operator (see table Comparison Operators) and the compared value need to be defined before confirming the condition with "OK".
In the table, all defined conditions are listed. An X symbol shows that a condition is currently not fulfilled while a dash icon symbolizes that it is fulfilled. The gear icon can be clicked to modify the condition, the trash icon will remove it.
In the dropdown "Evaluation" it can be defined whether all conditions need to be fulfilled for the event to trigger or if a single conditions suffices.
The Live State shows if the event would currently trigger based on the conditions or not.

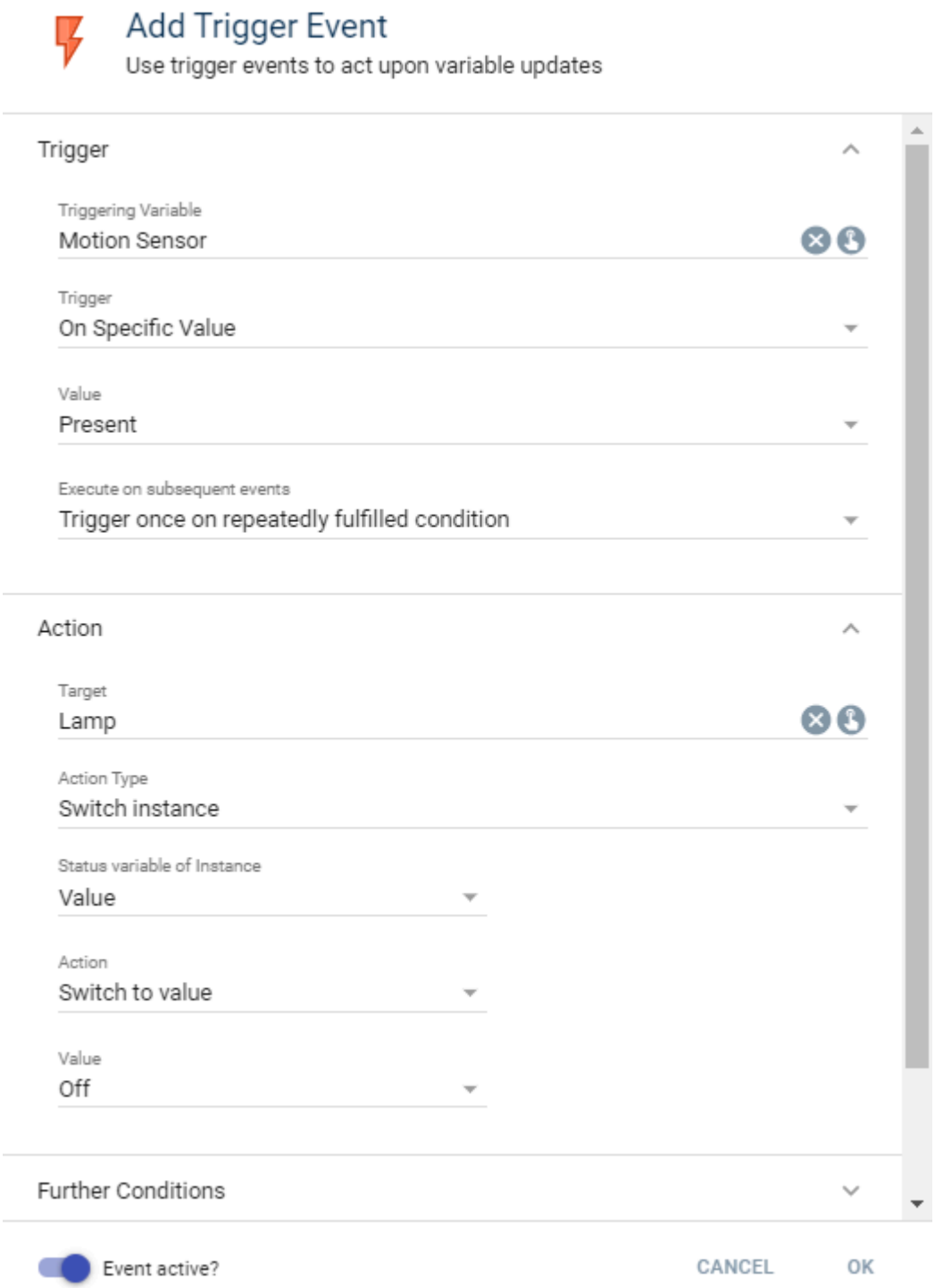
COMPARISON OPERATORS

Comparison Operators describe the behavior when a comparison is fulfilled and when it is not fulfilled.

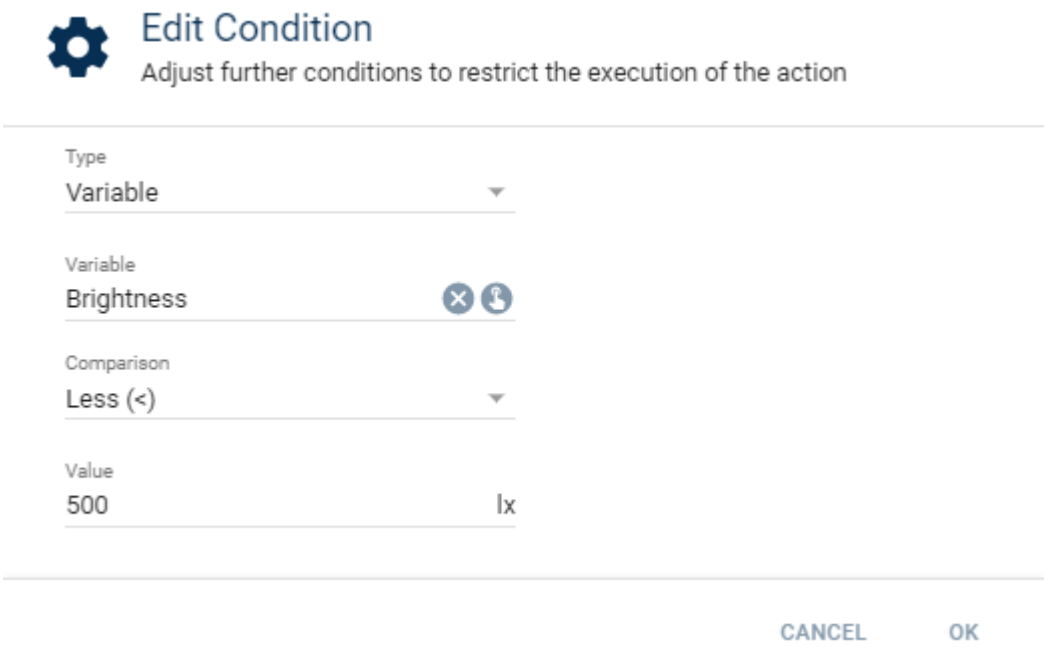
OPERATOR	EXPLANATION
Equal (=)	Fulfilled if the current value is equal to the comparison value. (e.g., true = true, 123 = 123)
Not Equal (≠)	Fulfilled if the current value is not equal to the comparison value. (e.g., false ≠ true, 123 ≠ 122 or 124)
Greater (>)	Fulfilled if the current value is truly greater than the comparison value. (e.g., 123 > 122, 09:00 > 08:59)
Greater or Equal (≥)	Fulfilled if the current value is greater or equal than the comparison value. (e.g., 123 ≥ 122 or 123, 09:00 ≥ 08:59 or 09:00)
Less (<)	Fulfilled if the current value is truly less than the comparison value. (e.g., 123 < 124, 09:00 < 09:01)
Less or Equal (≤)	Fulfilled if the current value is less or equal than the comparison value. (e.g., 123 ≤ 124 or 123, 09:00 ≤ 09:01 or 09:00)

EXAMPLE

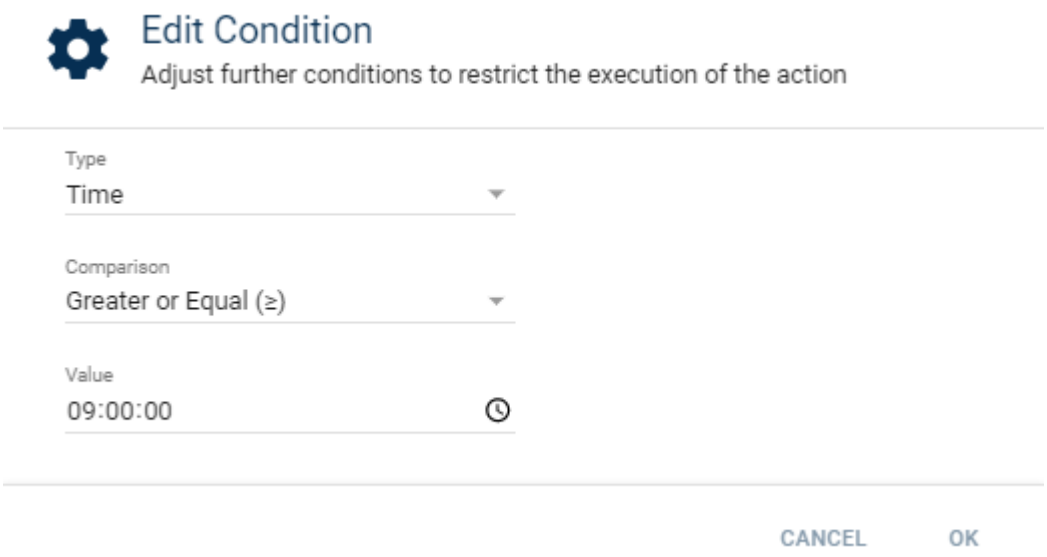
A lamp should only be switched on when a motion sensor is activated.
In addition, this should only trigger if it is too dark (Brightness Value below 500) and it is between 09:00 and 17:00.
First, a triggered event is created. The motion sensor is defined as trigger on the value "Present" and the lamp is selected as target instance, switching its value to "On".



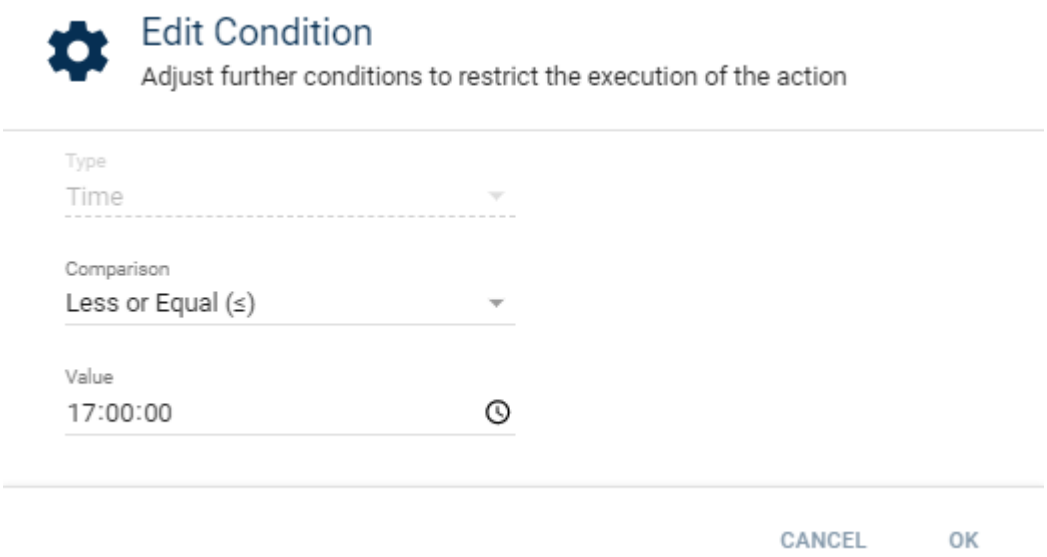
In the tab "Further Conditions", the additional rules are created via "Add".
The light value "Brightness" is configured as "less" than 500.



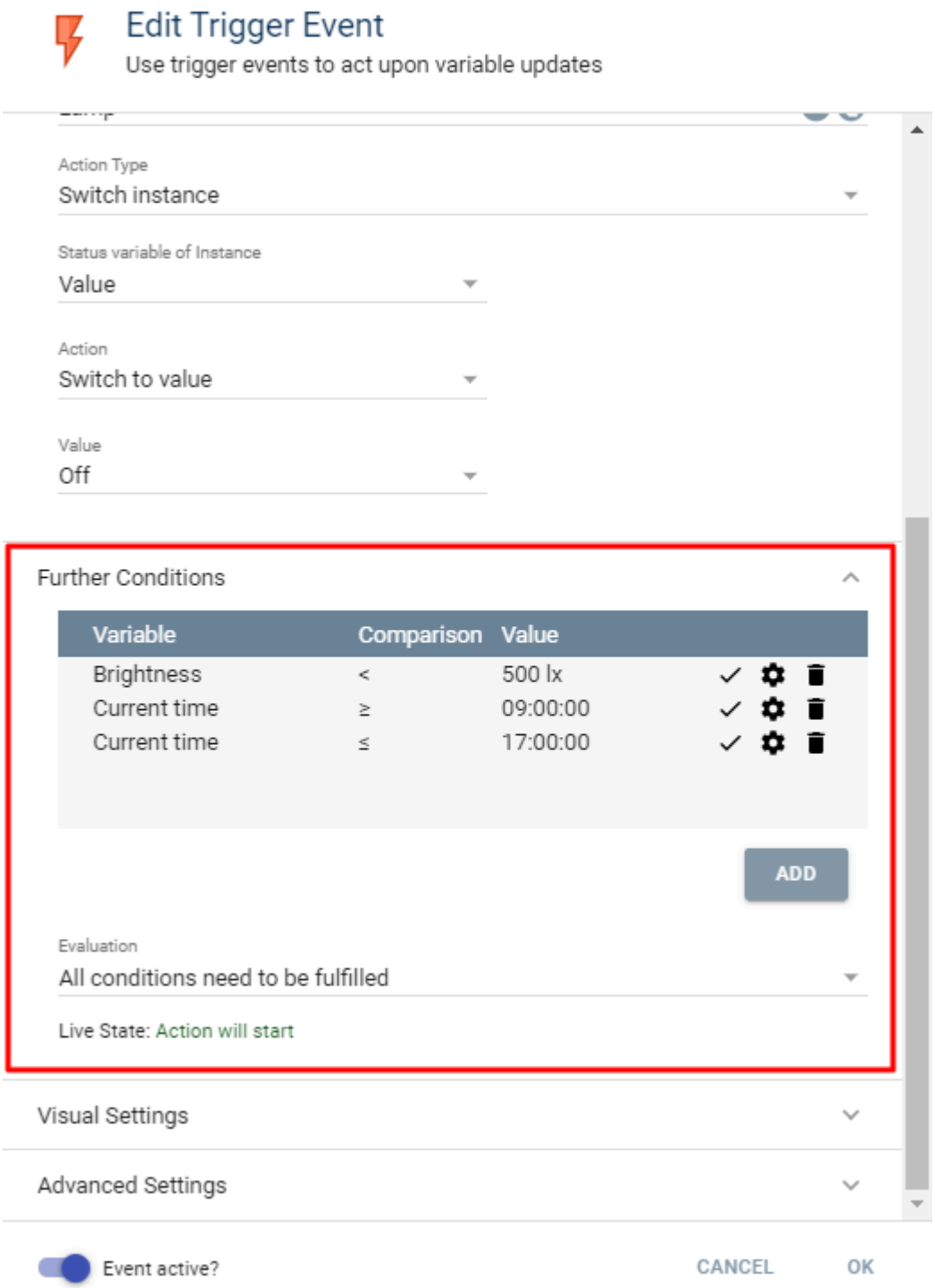
The time is configured as "greater or equal" to 09:00.



The time is configured as "less or equal" to 17:00.



In the overview, all three rules are listed and the live state shows if the light would be switched on if the motion sensor would trigger at this moment.

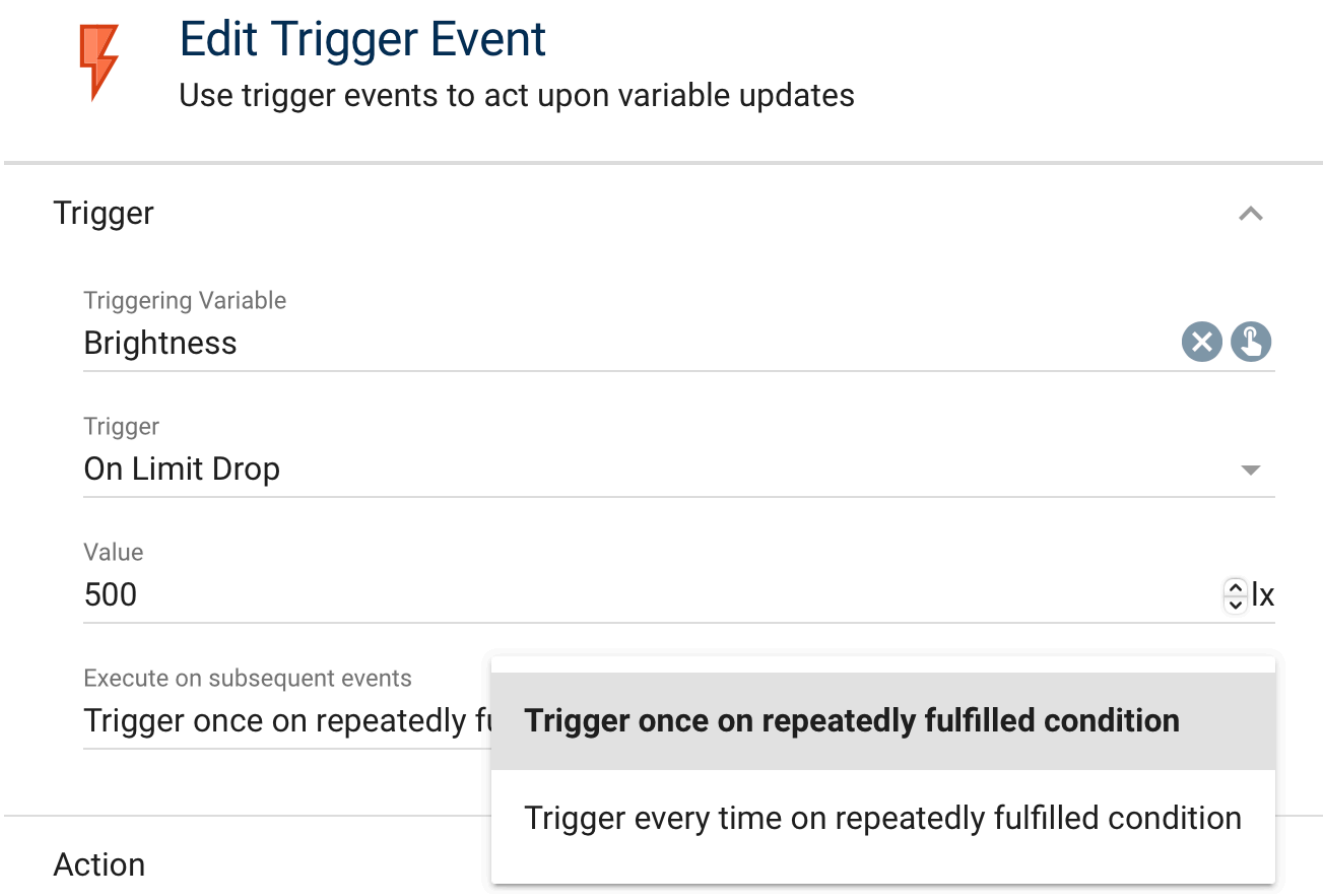

Edit Trigger Event
Use trigger events to act upon variable updates
Action Type
Switch instance
Status variable of Instance
Value
Action
Switch to value
Value
Off
Further Conditions
Variable
Comparison
Value
Brightness
<
500 lx
Current time
≥
09:00:00
Current time
≤
17:00:00
ADD
Evaluation
All conditions need to be fulfilled
Live State: Action will start
Visual Settings
Advanced Settings
Event active?
CANCEL
OK

Trigger

This event type is executed when the connected variable is updated, independently of the time point of execution.
There are five usable triggers.

Scripts that are run by a trigger event can use these System Variables.

TRIGGER



TYPE OF TRIGGER	DESCRIPTION
On Change	The event is executed whenever the variable changes.
On Update	The event is executed for every received variable value. It is also run when the received and the current value are identical.
On Limit Drop/On Limit Exceed (Option "Execute subsequent events")	The event is executed when the value of the variable drops below or rises above a threshold. The threshold is set in the field "Value".

TYPE OF TRIGGER	DESCRIPTION
On specific Value (Option "Execute subsequent events")	The event is executed when the variable reaches a specific value that is specified in "Value".

OPTIONS

EXECUTE SUBSEQUENT EVENTS

Trigger once on repeatedly fulfilled condition: The event is executed once. The event can only be executed again after the condition was not fulfilled at least once. After it was not fulfilled, dropping below, rising above, or reaching the specified value executes the event again.
Trigger every time on repeatedly fulfilled condition: The event is executed on every variable update as long as the condition is fulfilled.

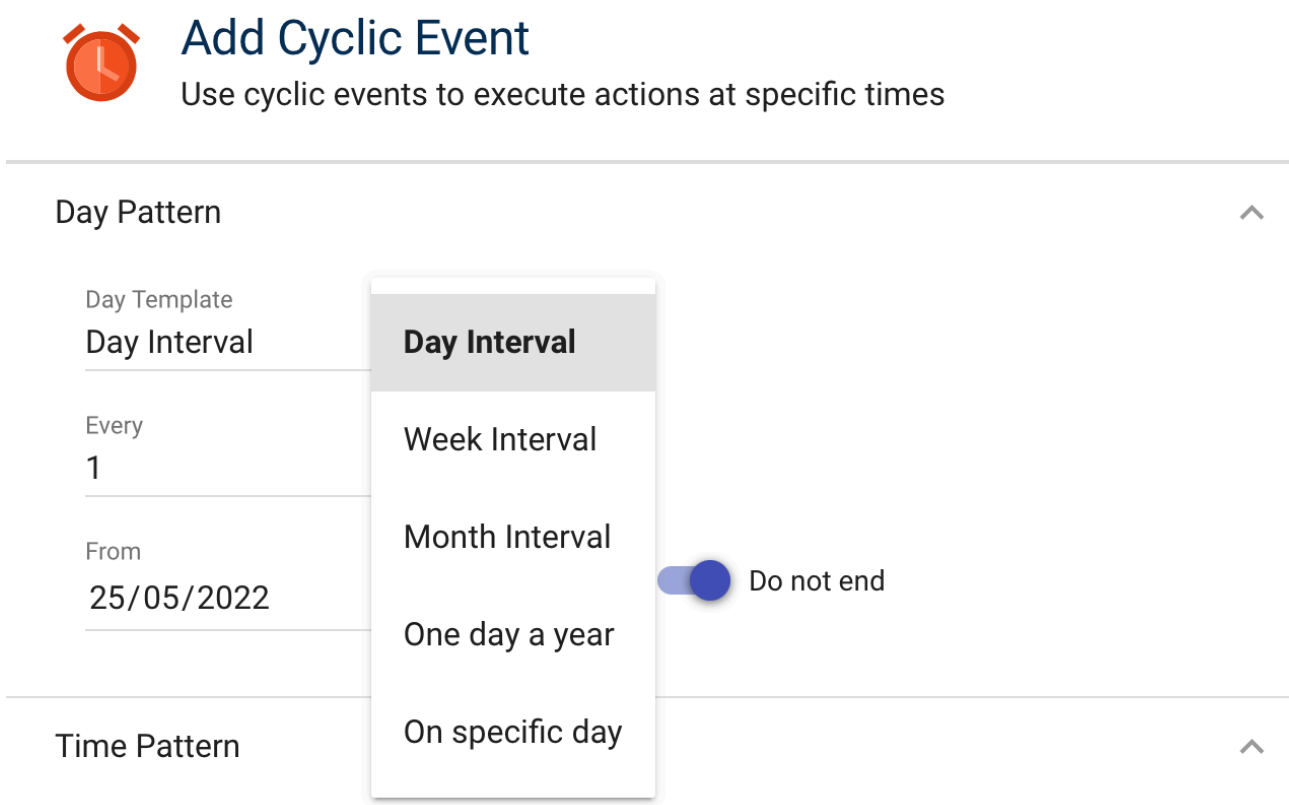
Cyclic

This type of event is executed at a specific time and is repeated cyclically. The point of time is defined by a combination of "Day Pattern" and "Time Pattern".

Scripts that are run by a cyclic event offer these System Variables

DAY PATTERN:

There are 5 different options

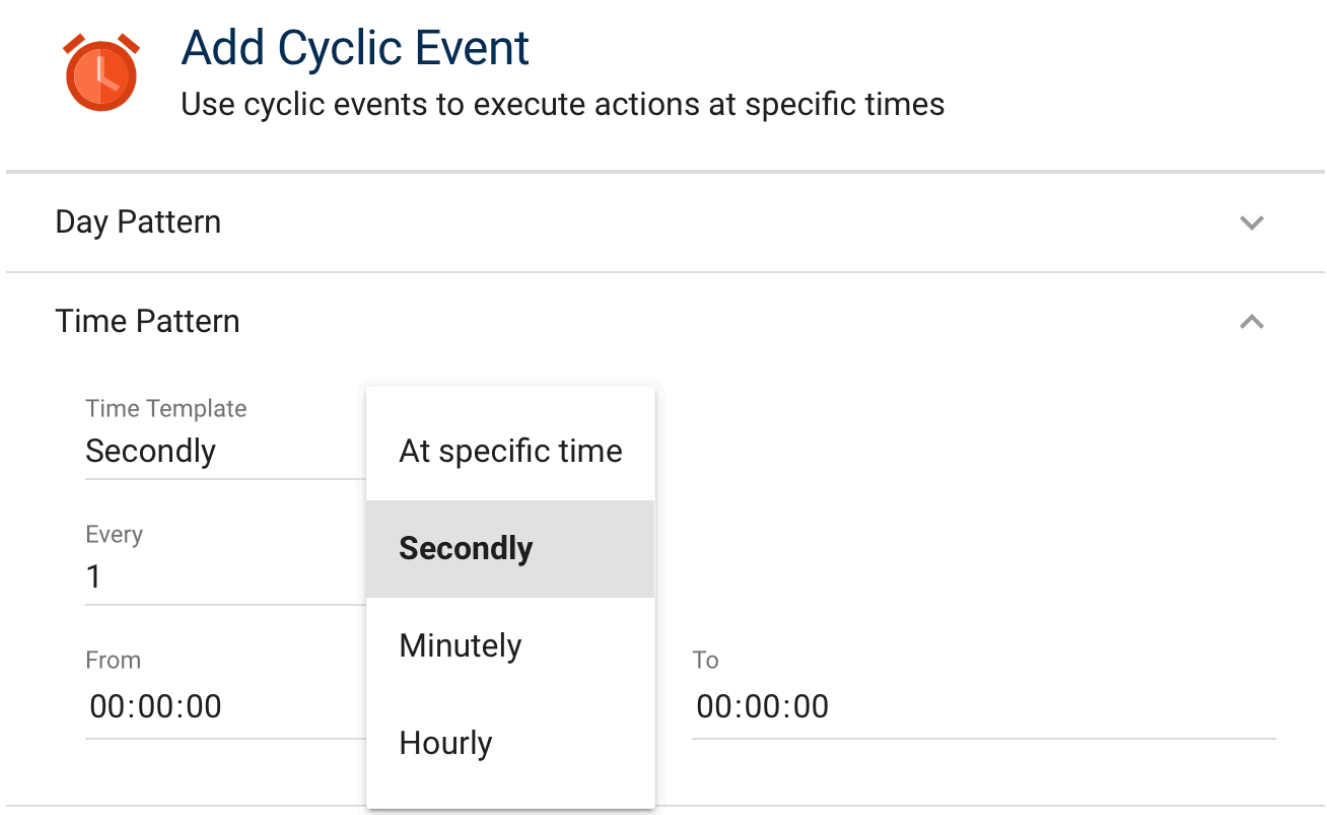


DAY PATTERN	DESCRIPTION
Day Interval	Defines a day interval (1 = every day; 2 = every second day; ...) in which the event is executed.
Week Interval	Defines a week interval (1 = every week; 2 = every second week; ...) and the days in which the event is executed.
Month Interval	Defines a month interval (1 = every month; 2 = every second month; ...) and the days in which the event is executed.
One day a year	Defines a specific day in the year on which the event is executed.

DAY PATTERN	DESCRIPTION
On specific day	Defines a specific date on which the event is executed.

TIME PATTERN:

There are 4 different options



TIME PATTERN	DESCRIPTION
At specific time	Defines a specific time on which the event is executed.
Secondly	Defines a second interval on which the event is executed.
Minutely	Defines a minute interval on which the event is executed.
Hourly	Defines an hour interval on which the event is executed.

SPECIAL OPTIONS:

From/To:
It is possible to define start and end time.

SETTING	DESCRIPTION
On Day Pattern	Defines the date interval in which the event is active.
On Time Pattern	Defines the daily interval in which the event is active.
From	Start Time = Defines the beginning of the interval.

SETTING	DESCRIPTION
To	End Time = Defines the end of the interval.
"Do not end"	No end time. If deactivated, the "To" option is displayed.

EXAMPLES:

EVERY FRIDAY AT 09:00:

Day Pattern = Weekly with value one and Friday checked
Time Pattern = At specific time with 09:00:00

EVERY 5 MINUTES FROM 12:02:10 UNTIL 16:00:00:

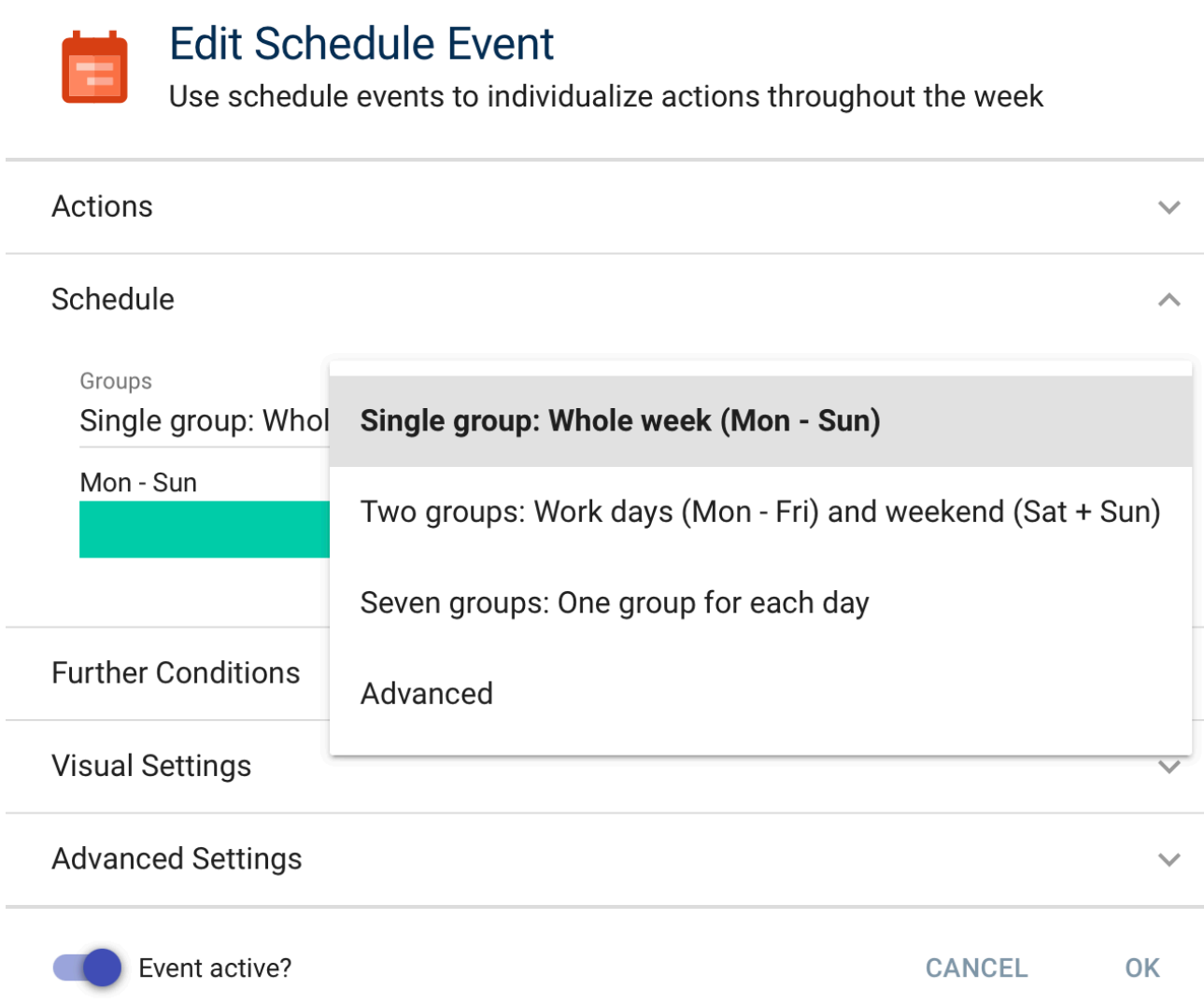
Day Pattern = Daily with value 1
Time Pattern = Minutely with value 5
From: 12:02:10 To: 16:00:00
No hook at "Do not end"
First execution at 12:02:10
Second execution at 12:07:10
...
--continuing until-->
...
Last execution at 15:57:10

Schedule

The schedule is a graphical tool for the configuration of weekly procedures.
The plan for the week, the group type, the actions, and the connection type are chosen in the management console.
The schedule actions and its changes are defined in the visualization "WebFront" or the configuration of the schedule.

GROUP TYPES

There are four options



TYPE	DESCRIPTION
Single group: Whole week (Mon - Sun)	All week days are summarised in a single group.
Two groups: Work days (Mon-Fri) and weekend (Sat + Sun)	Splits the week into two groups and allows the days from Mon-Fri and Sat+Sun to be configured independently.

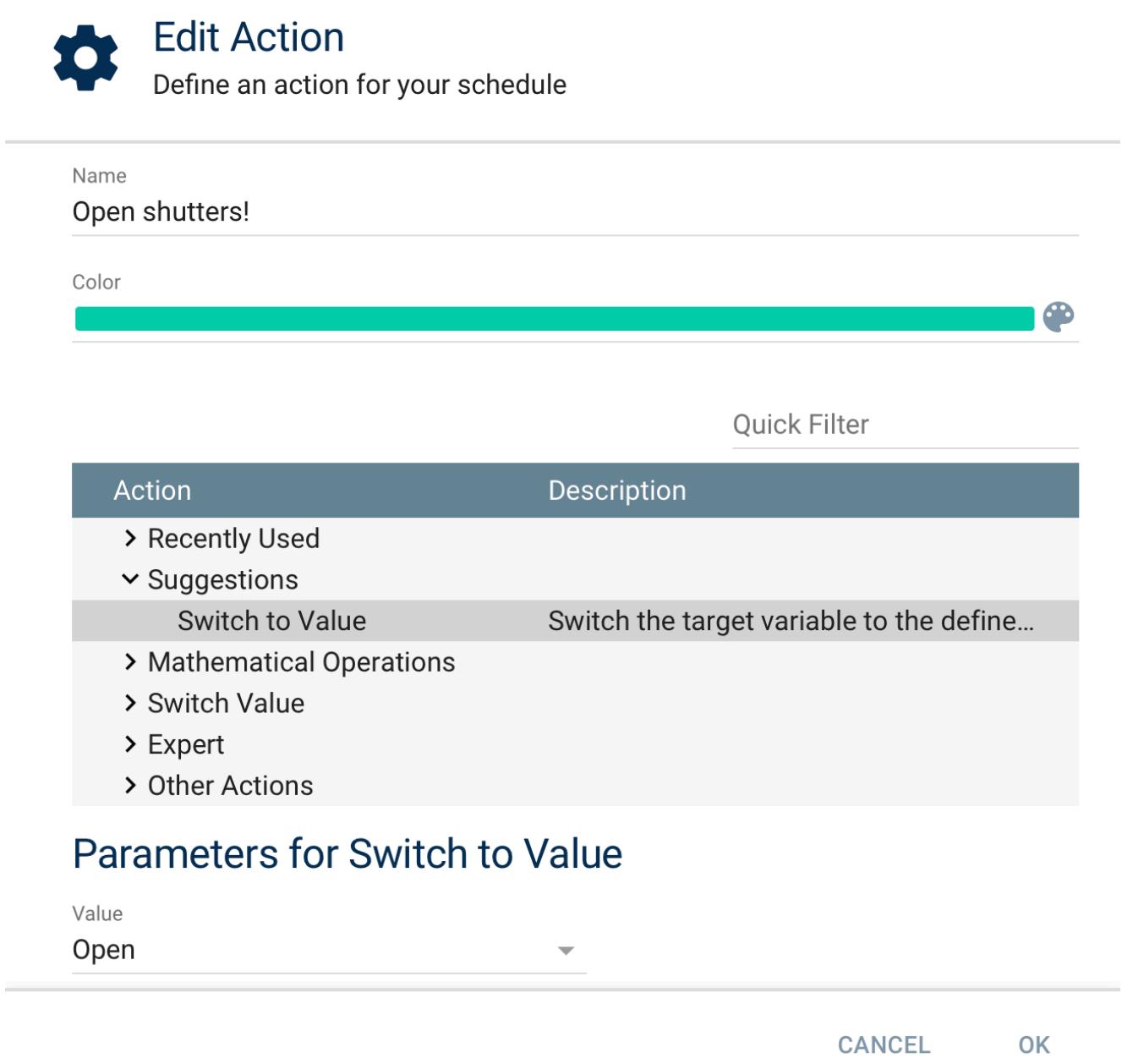
TYPE	DESCRIPTION
Seven groups: One group for each day	Every week day can be configured individually.
Advanced	The week days can be combined as required. Each combination is summarized into one group and can be configured.

ACTIONS

The schedule offers actions depending on the selected target. At least two schedule actions need to be configured. As part of the schedule action, an Action is defined for the target.

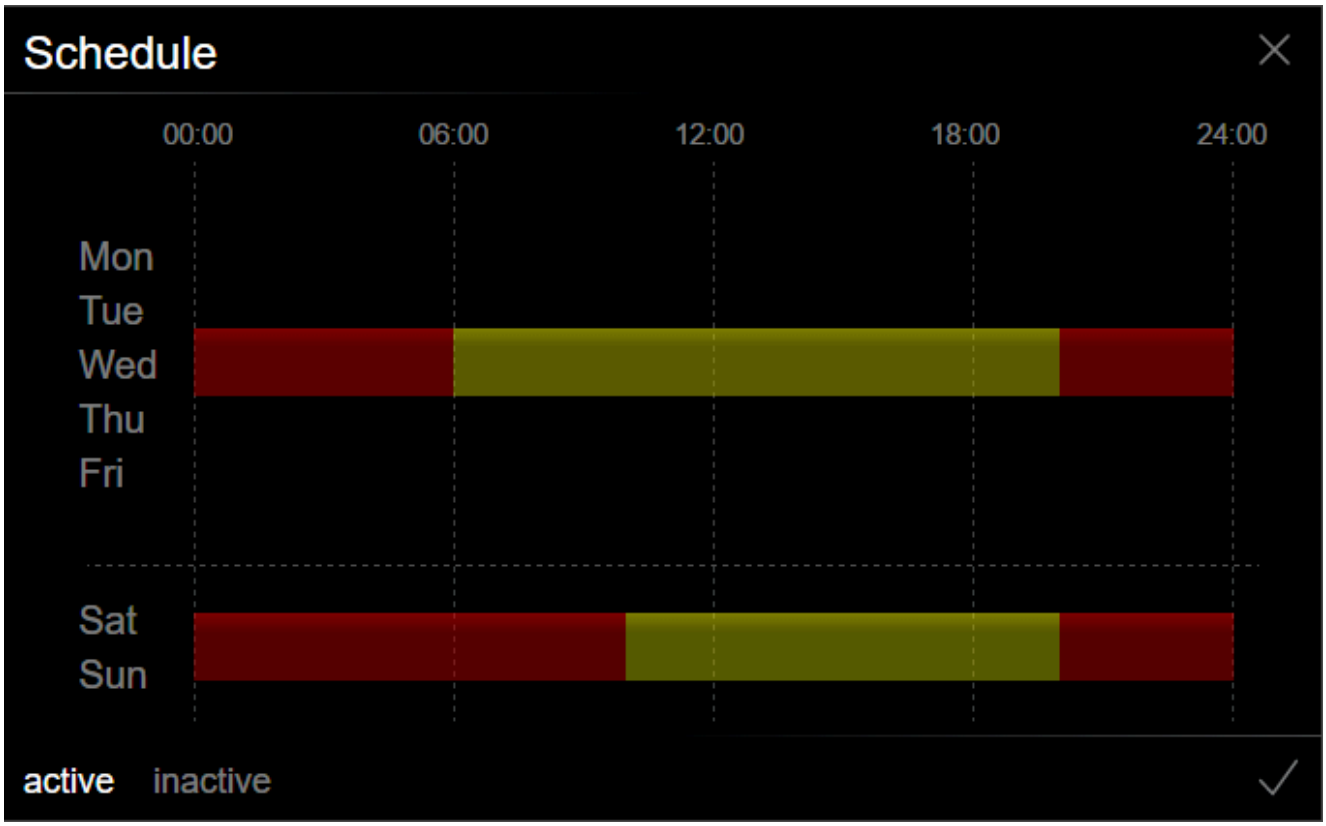
ADD SCHEDULE ACTION

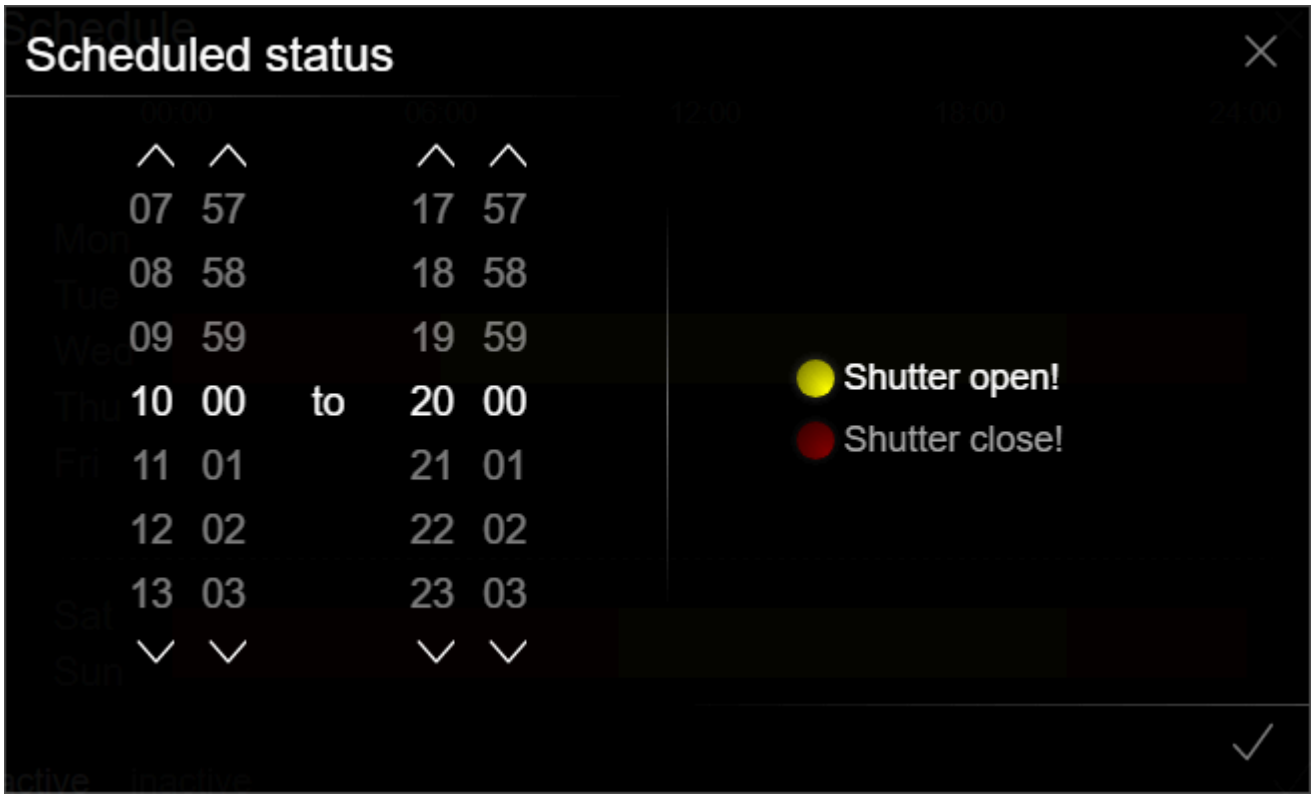
A schedule action can be added by clicking on "Add". For the schedule action, a name, a color, and an Action are selected.



CONFIGURATION OF THE SCHEDULE ACTIONS

The durations of the actions can be configured within the Management Console or the WebFront.
Within the grafical interface, the following applies:
Single Click configures an existing state.
Click & Pull creates a new state.



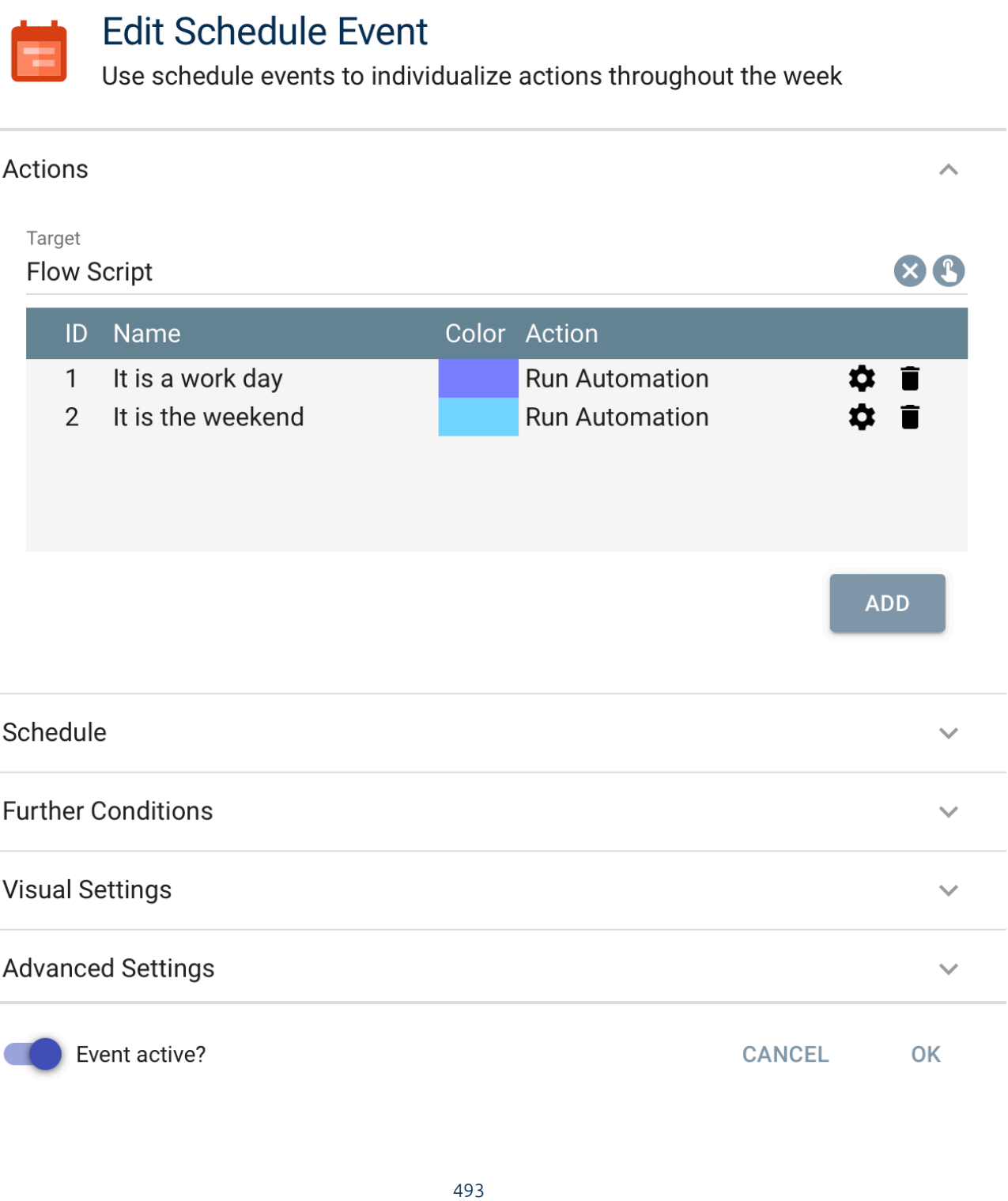


The selected action is only executed when the action changes.

Important remark:
Previously set actions are not undone by an action change.
This means, when an action activates a light, it is not automatically deactivated when the action changes unless the deactivation is described explicitely.

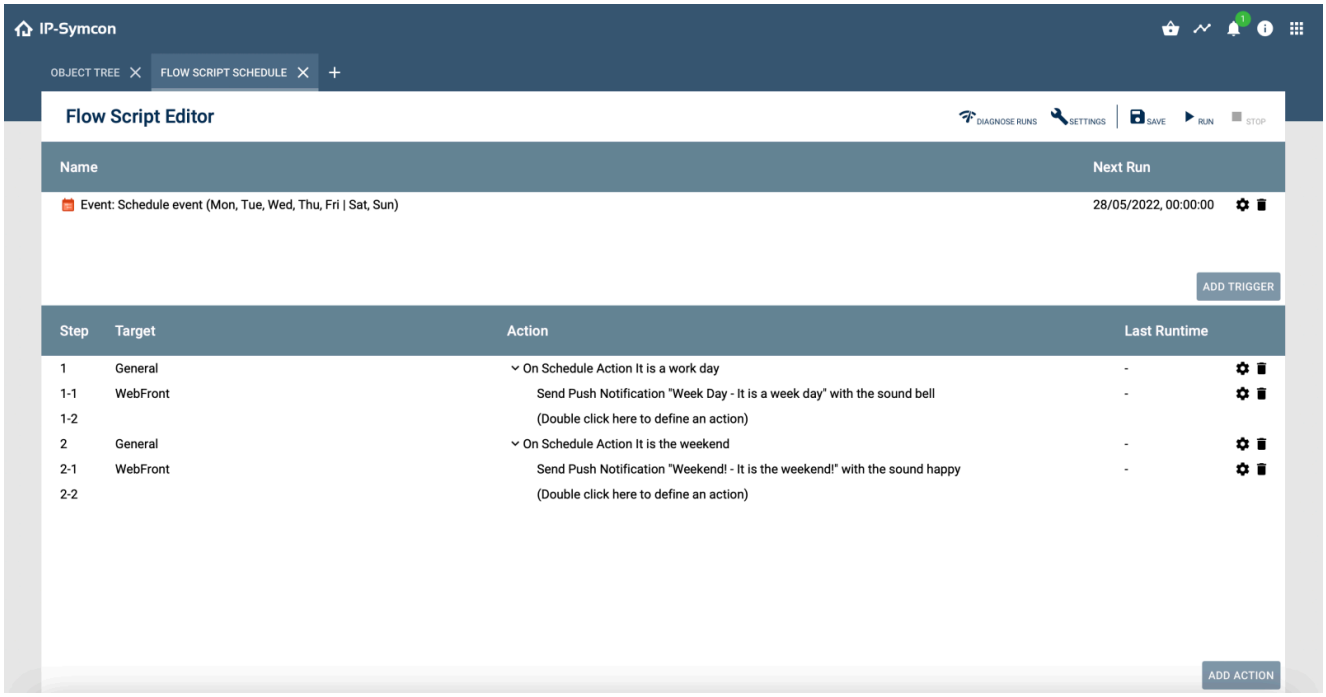
BIND TO FLOW SCRIPT

If the action "Run Automation" is selected for a Flow Script, the action "On Schedule Action" can be used to execute a number of actions on a specific schedule action. When the first schedule is added as trigger, the "On Schedule Action"-actions are added automatically.



EXAMPLE WITH FLOW SCRIPT

This is an example flow script which is executed via schedule.



BIND TO PHP SCRIPT

If the action "Run Automation" is selected for a System Variable $_IPS['ACTION'].

EXAMPLE WITH SCRIPT

This is an example PHP script which is executed via schedule.

```
//switch over the state IDs
switch ($_IPS['ACTION']) {
case 1: //ID 1
SetValueBoolean(39540 /*[Test Environment\WorkDay]*/, true);
echo "Hello World, on a work day";
break;
case 2: //ID 2
SetValueBoolean(39540 /*[Test Environment\WorkDay]*/, false);
echo "Hello World, Yay week end!!";
break;
}
```

HINT FOR EXPERTS

It is possible to check the currently active schedule action via script.

```
$e = IPS_GetEvent($id);
$actionID = false;
// Loop through all groups
foreach($e['ScheduleGroups'] as $g) {
// Check if the group is responsible for today
if($g['Days'] & date("N") > 0) {
// Search for current trigger point. We exploit the property that the
trigger points are sorted ascendingly.
foreach($g['Points'] as $p) {
if(date("H") * 3600 + date("i") * 60 + date("s") >= $p['Start']['Hour']
* 3600 + $p['Start']['Minute'] * 60 + $p['Start']['Second']);
} else {
break; // Once passed, we can abort.
}
}
break; // The loop can be canceled once we have found our day. Every
day must be in exactly one group.
}
}
var_dump($actionID);
```

Media

IP-Symcon supports sound and image files, charts, documents, streams, and views from IPSStudio as media.

Object Tree

ID	Name	Type	Value
	IP-Symcon		
44707	Energy Consumption	Media	media/44707.chart
22877	Football	Media	media/football.png
30038	Grant ch 5 & 7 Required Analyses	Media	media/Grant ch 5 & 7 Requir...
46742	iPhone Original SMS Sound	Media	media/iPhone Original SMS ...
22798	My Stream	Media	rtsp://192.168.0.25

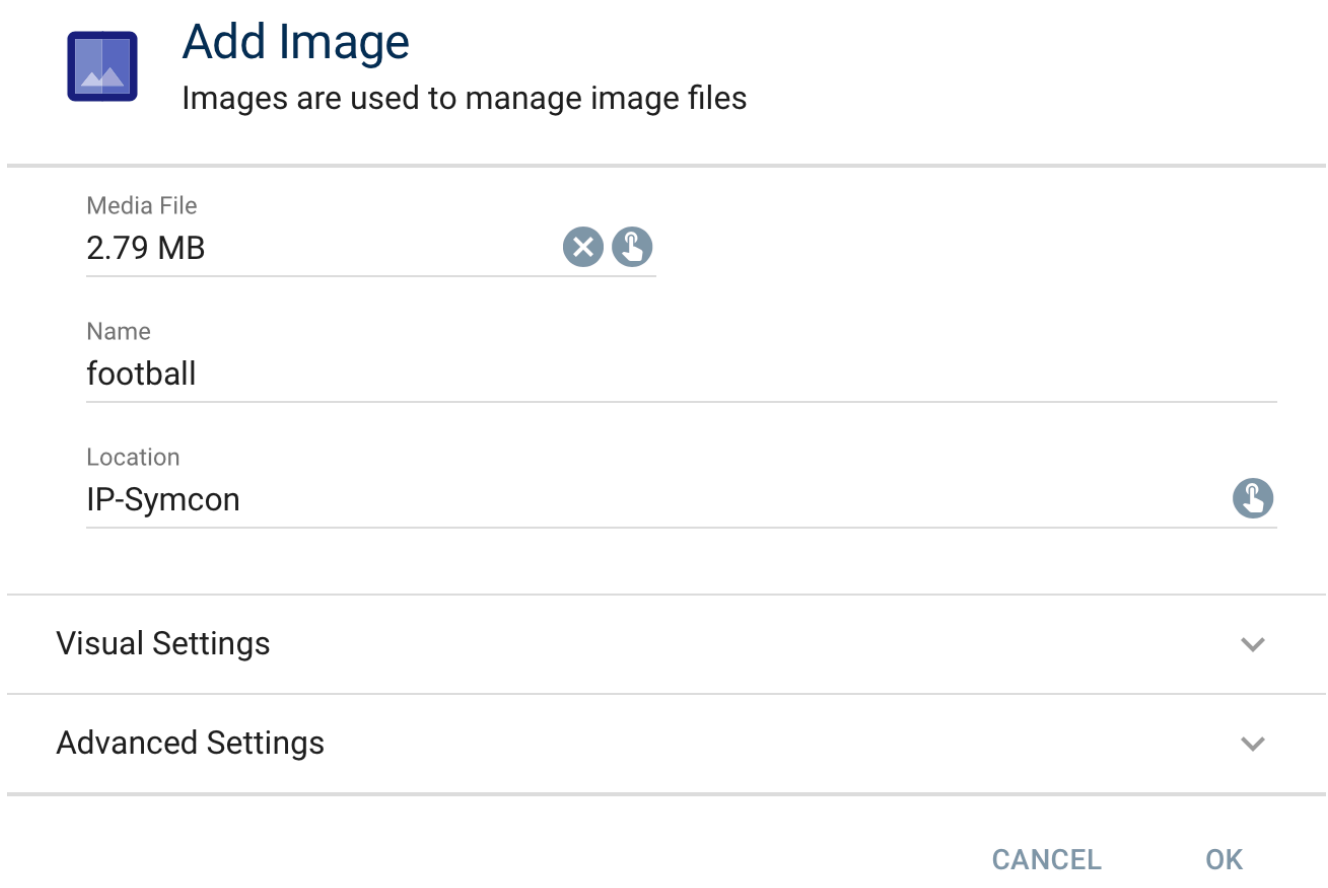
ADD MEDIA

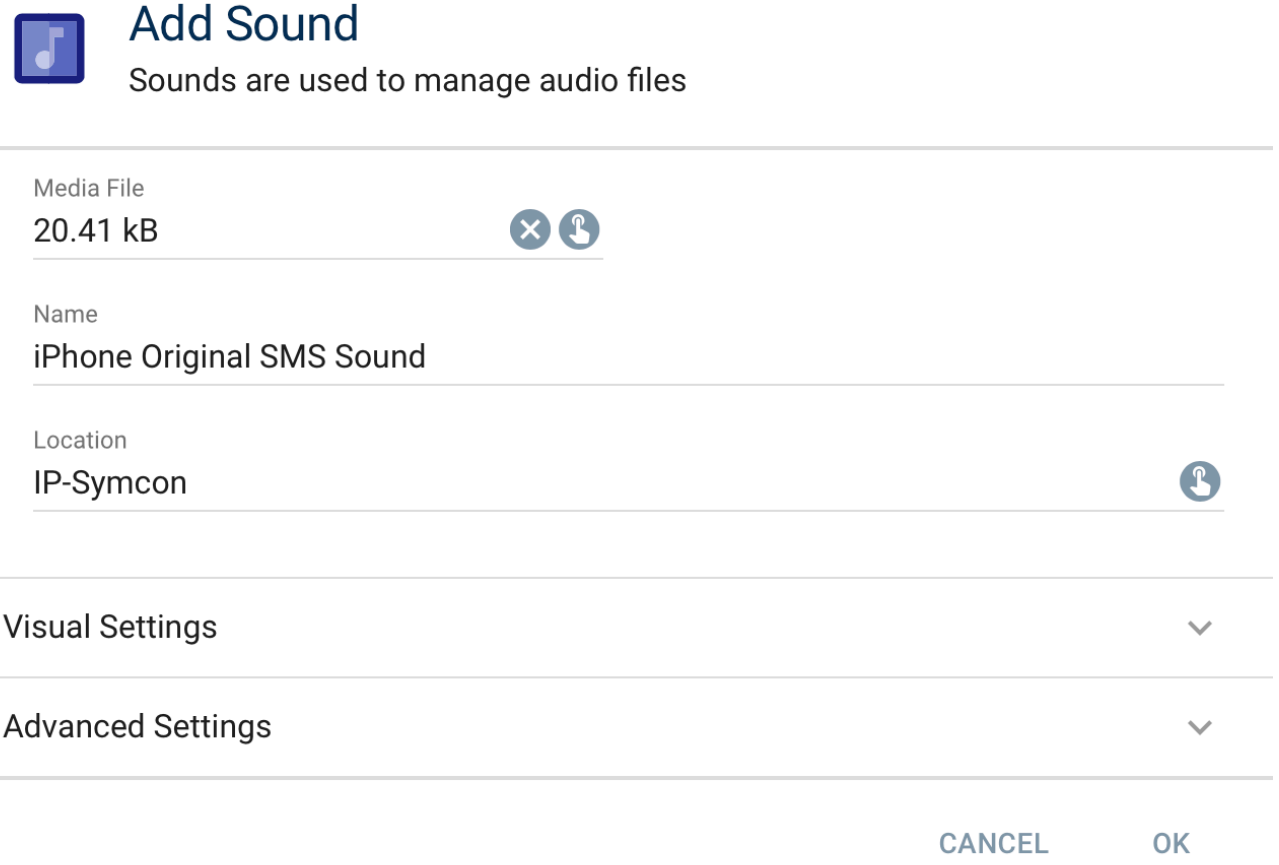
The tutorial for the configuration of each type of media is contained within the corresponding subcategory.
It is possible to add the following types of media:
Image/Sound
Charts
Documents
Streams
Views from IPSStudio

Image/Sound

ADD IMAGE OR SOUND FILES

The dialog "Add object -> Media -> Image or Sound" is used to add an image or sound file to the media pool.
Any image or sound file from the hard drive, network, internet, etc. can be chosen for upload and will be copied into the media pool of IP-Symcon.





Added Image/Sound objects are available in the object tree.
A preview can be shown with a double click on the media object in the object tree.

Image

Preview of the image



CLOSE

Sound

Preview of the sound



CLOSE

ALLOWED FILE EXTENSIONS

ENDING	SINCE VERSION
.jpg	3.4
.jpeg	3.4
.gif	3.4
.png	3.4
.ico	3.4
.webp	8.1

All files can be added - however, the correct display in the tile visualization is only guaranteed for these types

Charts

Charts can present one or more variables as graphs. Using variables in a chart requires that the variables are logged. Configuring the logging of variables is explained in the Archive Control.

CONFIGURATION IN IP-SYMCON

A new chart can be added via Right click -> "Add object" -> "Media" -> "Chart" at the selected location. The selected (logged) variables are used as data source for the individual graphs.

EDIT CHART

General properties of the chart.

OPTION	DESCRIPTION
Add	Add a single variable -> Opens the "Configure graph" dialog. If a graph already exits, this dialog can be opened by clicking on the gear in the list
Name	Name of the Chart
Location	Selection of the location for the chart
Type	Selection the type of presentation as line, bar, or bool chart
Visual Settings	Here, the icon, visibility, and activation can be configured
Advanced Setting	A description can be provided here

CONFIGURE GRAPH

A graph represents a single dataset for the chart.

FIELD	DESCRIPTION
Axis Profile	Selection of the Variable Profile . The defined Min and Max values are used for scaling the Y-axis (not available for Bool Chart)
Axis Side	Selection if the labels for this graph should be at the left or at the right (not available for Bool Chart)
Fill Color	Selection of the desired fill color - Transparency possible
Line Color	Selection of the desired line color
Offset	An offset by the respective time unit (see below for examples)
Title	Meaningful title which is shown in the legend. If the title is empty, the name of the variable is displayed
Variable	Selection of the displayed variable

DISPLAY

Display as Popup Graph
To display a popup graph, the chart needs to be moved to the desired category in the object tree.
Display as Full Screen Graph
A full screen graph is displayed by adding the chart as graph element with the WebFront editor.

ADDITIONAL OPTIONS FOR PRESENTATION

Selectable options that can be activated within the graph to change the presentation.

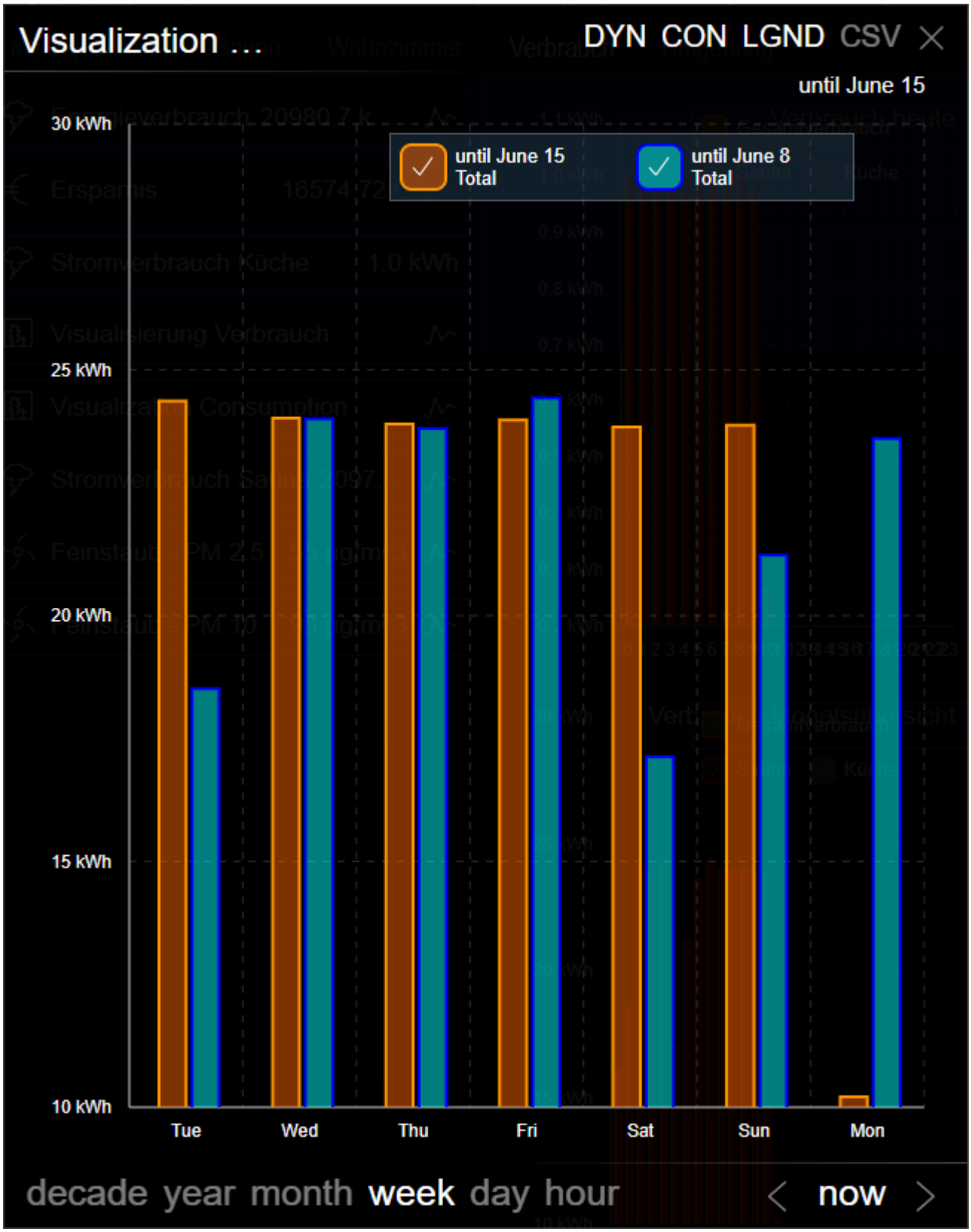
ABBREVIATION	MEANING	DESCRIPTION
CON	Continuous	The X axis is adjusted such that the right border is set to the most current value. This shows the biggest possible amount of current data
DYN	Dynamic	If activated: The Y axis is scaled according to the minimum and maximum value of the chart If deactivated: The borders of the Y axis are taken from the selected variable profile
HD	High Density	If activated: An increased precision of average values is used for the presentation of the graph. See table **Average Intervals**
Legend	Legend	(Charts only) (De-)Activates the legend
MIN/MAX	Min/Max	(Automatically generated graphs for variables only) Two additional graphs with the minimum and maximum values are shown
RAW	Raw data	If activated: The raw data is displayed. If deactivated: The aggregated values are shown

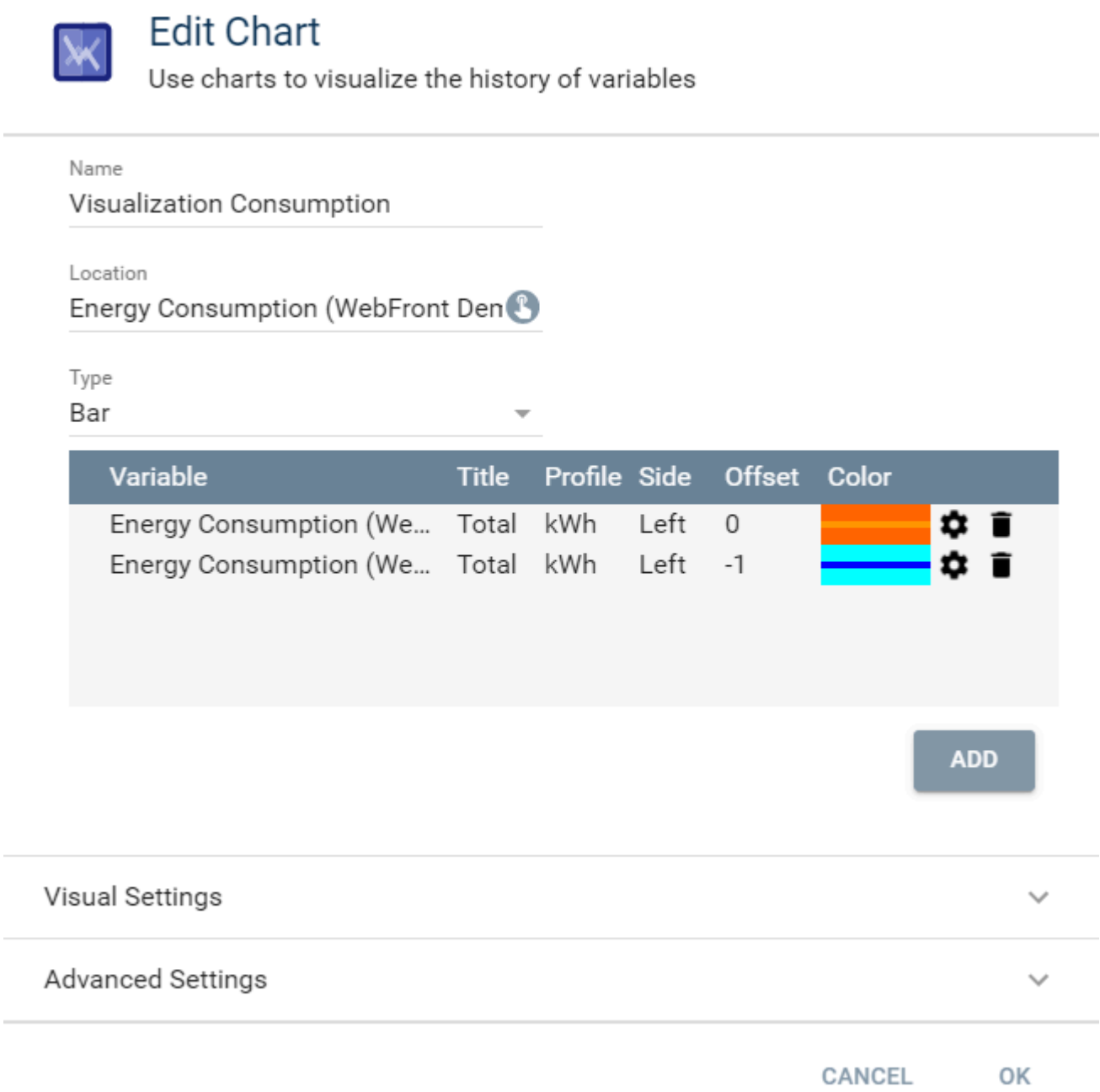
Average Intervals

TIMESPAN	REGULAR	HIGH DENSITY (HD)
Decade	Years	Months
Year	Months	Days
Month	Days	Hours
Week	Days	Hours
Day	Hours	5 minutes
Hour	5 minutes	1 minute

EXAMPLE 1 (TYPE: BAR)

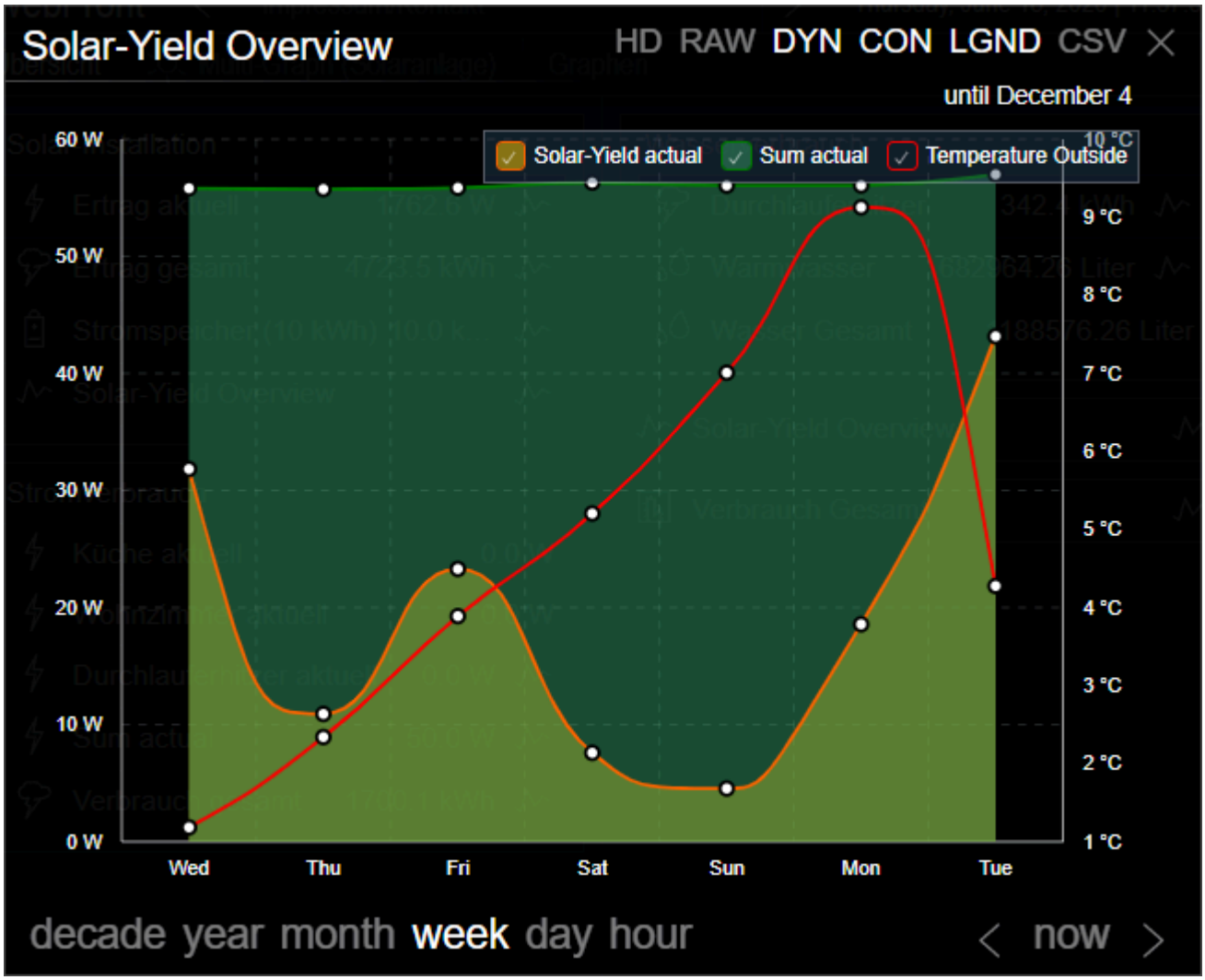
This example shows the comparison of energy costs of a heating system.
The same variable was chosen twice, but one (blue) uses a time offset of -1.
In the week presentation, a comparison between the current and the previous week is shown, in the year presentation, a comparison to the previous year, ...

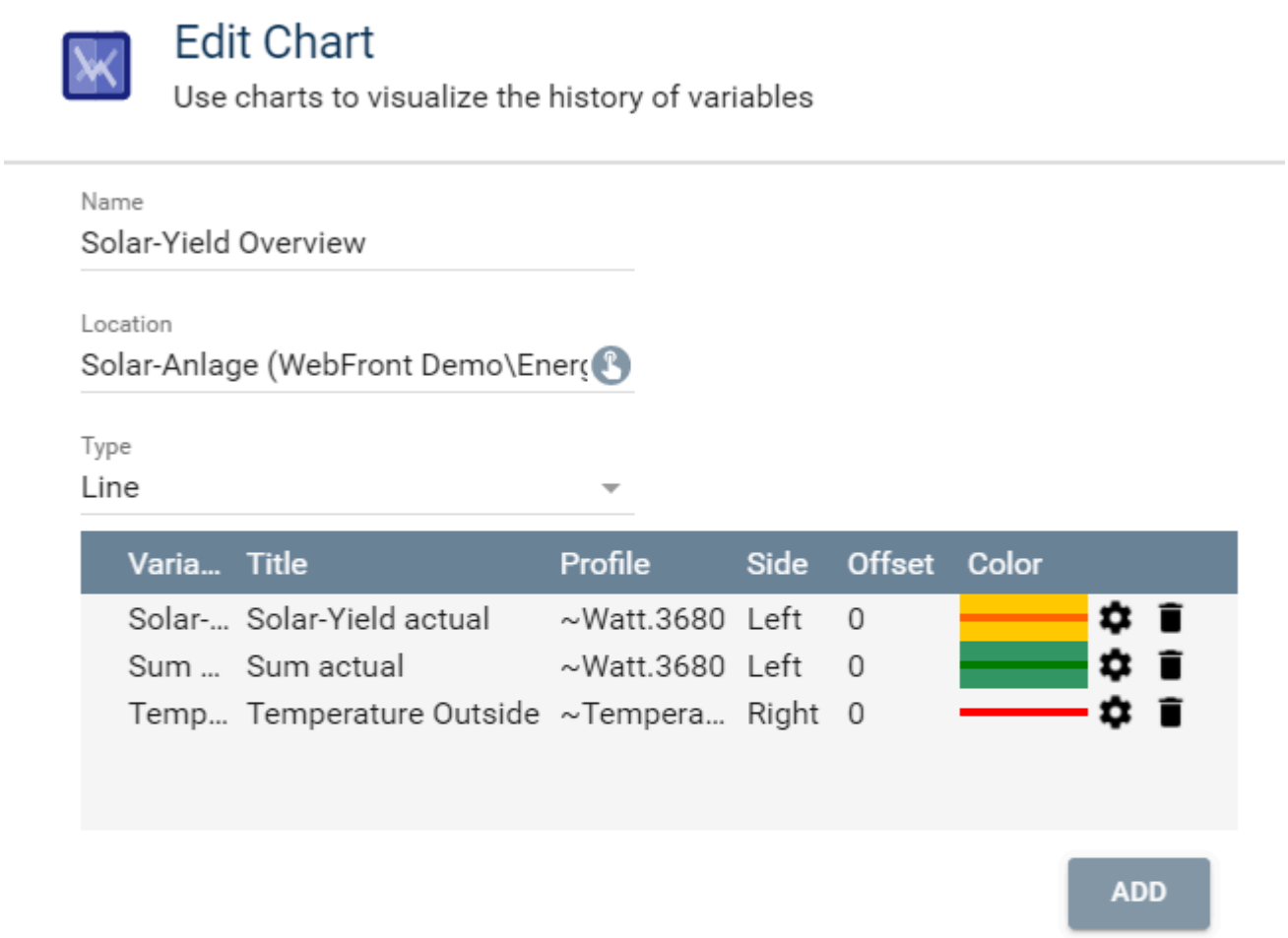

Visualization ...
DYN CON LGND CSV
until June 15
30 kWh
25 kWh
20 kWh
15 kWh
10 kWh
until June 15
Total
until June 8
Total
Tue
Wed
Thu
Fri
Sat
Sun
Mon
decade year month week day hour
now



EXAMPLE 2 (TYPE: LINE)

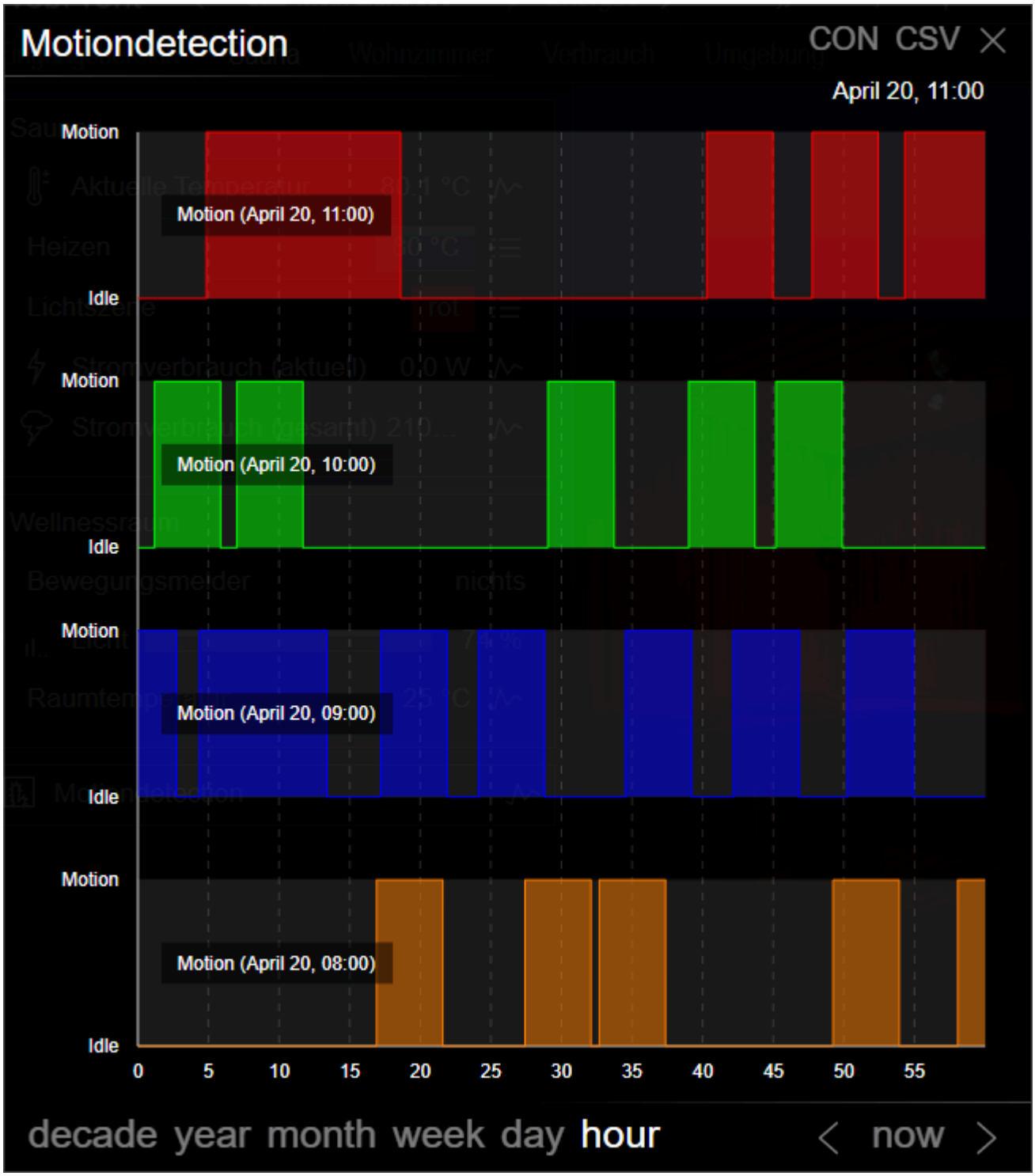
This example displays the gain of a solar collector as well as a sum of the current consumption. The values for these two graphs can be read at the left axis. In addition the exterior temperature is integrated, which uses the right axis.


Solar-Yield Overview
HD RAW DYN CON LGND CSV
until December 4
Solar-Yield actual
Sum actual
Temperature Outside
60 W
50 W
40 W
30 W
20 W
10 W
0 W
10 °C
9 °C
8 °C
7 °C
6 °C
5 °C
4 °C
3 °C
2 °C
1 °C
Wed
Thu
Fri
Sat
Sun
Mon
Tue
decade year month week day hour
now



EXAMPLE 3 (TYPE: BOOL)

This example shows a bool chart that shows the state of a motion sensor over the course of four consecutive hours.


Motiondetection
CON CSV
April 20, 11:00
Motion
Motion (April 20, 11:00)
Idle
Motion
Motion (April 20, 10:00)
Idle
Motion
Motion (April 20, 09:00)
Idle
Motion
Motion (April 20, 08:00)
Idle
0 5 10 15 20 25 30 35 40 45 50 55
decade year month week day hour
now

Edit Chart

Use charts to visualize the history of variables

Name

Motiondetection

Location

Sensors (Hardware)

Type

Bool

Variable	Title	Profile	Side	Offset	Color
Motion ...	Motion	(Automatic)	Left	0	
Motion ...	Motion	(Automatic)	Left	-1	
Motion ...	Motion	(Automatic)	Left	-2	
Motion ...	Motion	(Automatic)	Left	-3	

ADD

Visual Settings

Advanced Settings

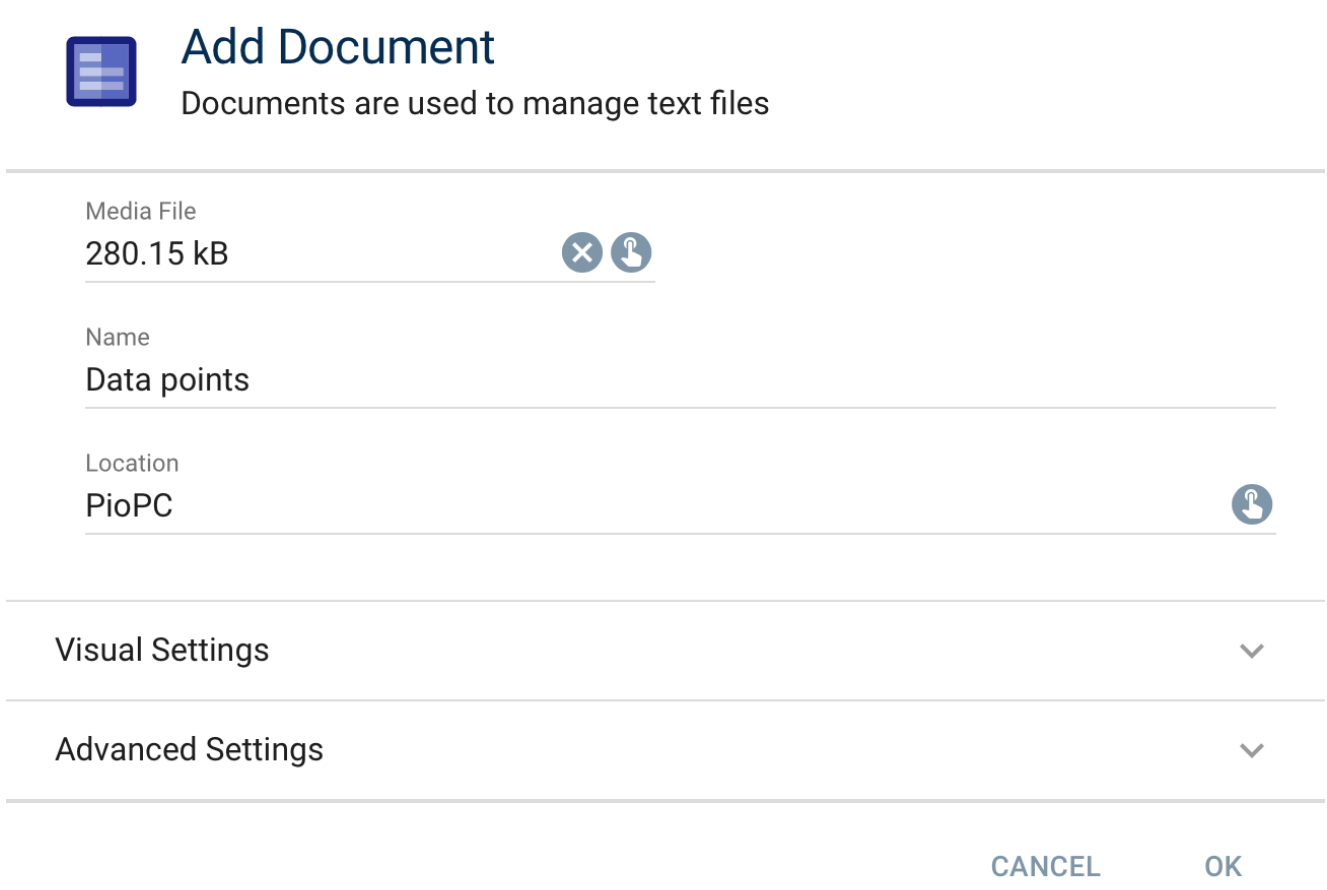
CANCEL OK

Documents

Require: Symcon >= 4.3

ADD MEDIA DOCUMENT

A new media file of the type document can be added via the "Add Object -> Media -> Document" dialog.
A new file can be created or an existing one from the hard disk, network, etc. can be selected, which IP-Symcon then copies into the media pool.
Added documents can be viewed and opened in the object tree under "Media Files".



ALLOWED FILE EXTENSIONS

ENDING	SINCE VERSION
.doc	4.3
.docx	5.2
.pdf	4.3
.txt	4.3
.xls	4.3

ENDING	SINCE VERSION
.xlsx	5.2



All files can be added, but later support via WebFront will only be available for the specified file types

Streams

ADD MEDIA STREAM

It is possible to add MJPEG and RTSP streams to the WebFront by using the special media type Stream. Further information and the address can be found in the documentation of the utilized web camera. The size of the stream can be parameterized by the configuration of the web camera. The complete address, including the username and password, needs to be entered within the corresponding dialog of IP-Symcon.



RTSP streams can currently only be displayed in Chrome, Firefox, Opera, Edge, Android (incl. apps since version 5.1), Safari (since version 5.5), and iOS (incl. apps since version 5.5).



RTSP streams need to be h264 encoded. As IP-Symcon acts as distributor for RTSP streams, limits are applied depending on the used edition.



For **Axis** cameras: To use streams in the tile visualization, you must open “Plain Config” in “Advanced Options” and activate the setting **Image.I0.MPEG.H264.PSEnabled**. Then save with **Save** and reload the tile visualization to load the stream correctly.



Add Stream

Use streams to manage RTSP/MJPEG video streams

Name
My Stream

Location
IP-Symcon

Protocol
RTSP

Stream
rtsp://192.168.0.25

Username
user

Password
••••••••

Visual Settings

Advanced Settings

CANCEL OK

Links

DESCRIPTION

Links are connected to other existing objects and represent those objects in a different location. This is done without changing the existing structures and sortings.
All properties of the original object are shown.
Only the following properties of a link can be changed:

- Name
- Icons for the WebFront
- Visibility Settings for the WebFront

All objects within the object tree can be linked:

- Categories
- Instances
- Variables
- Scripts
- Events
- Media



Links of links are not possible. However, multiple links of one object are possible.

APPLICATION

Links are primarily used to position objects in the WebFront in desired locations. Thus, instead of positioning the object itself in the locations, a link of an object is placed there instead.
Another possibility is the creation of an overview tab that centrally shows the properties of multiple distributed objects in a single location.
A link is created by right clicking the object to be linked -> "Link Object" -> Choose a location for the link.



Only the top level categories of the object tree can be accessed within the iOS or Android app. Links can be used to bring objects within subcategories back to the "surface".

EXAMPLE

A complex example is found in Use Links

Components

IP-Symcon is composed of several software components that are described in more detail in the following subsections.

Remote Access

By default, access is possible in the local network without username/password. External access (for example through Symcon Connect) is completely disabled. To allow external access or password protection in the local network, remote access can be activated. A secure password (at least eight characters with special characters) should be assigned. Once remote access has been enabled, the system can be accessed through the Management Console by entering the username (license username) and password.



The license username used here is not identical to the account username on the homepage. The username requested here is always the license email address.



If too many password errors occur, password validation will be slowed down initially and completely disabled later on.

To access IP-Symcon remotely from another machine you will need the Management Console. As soon as the Management Console is launched, the Connection Wizard will be displayed, which automatically finds all IP-Symcon servers in their local networks. After selecting the server, the user name and password must be entered in order to establish a connection.



For access from outside the local network the Connect Control can be used with an active subscription. Alternatively, if sufficient network knowledge is available, this can be achieved by configuring a port forwarding in the router.

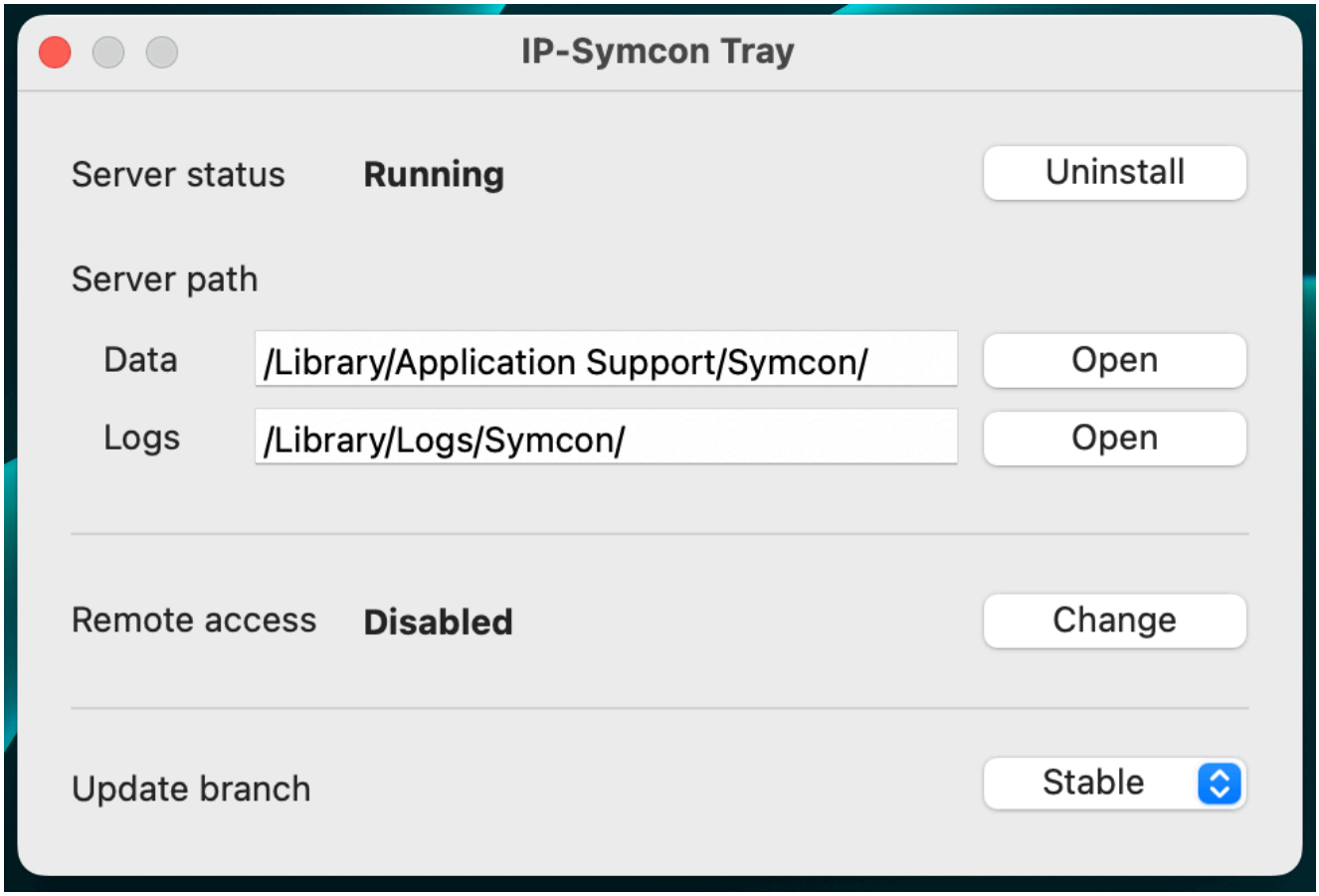
ACTIVATING / DEACTIVATING REMOTE ACCESS

In order to activate remote access, a password must be assigned. If remote access is deactivated again, the remote access password must simply be left "empty" and saved.
SymOS (SymBox)
Windows (desktop / server)
MacOS
Linux (Ubuntu)
Raspberry Pi
Docker
QNAP
Synology

Since version 4.4, remote access can also be activated within a demo license. When accessing the demo version via IPSStudio or Mediola, the user name must be left blank and only the password entered.

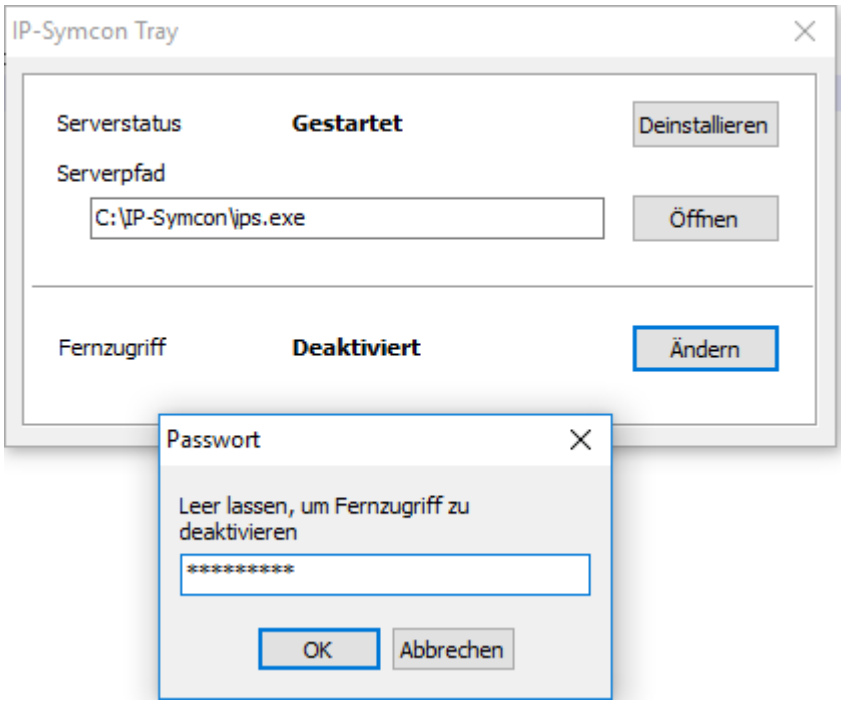
SYMOS

Via the web interface, a password can be set via "Settings -> Remote Access".



WINDOWS

Through the tray application a dialog can be opened via "Right-Click -> Information". At the bottom of the dialog, an item "Remote Access" can be found. There, the password can be set.



MACOS

Through the tray application a dialog can be opened via "Click -> Information". At the bottom of the dialog, an item "Remote Access" can be found. There, the password can be set.

LINUX

The license must be set up correctly in the Management Console. The following command must then be executed:

```
sudo nano /root/.symcon
```

There should be several entries. The following must be inserted in a new line. (Do not leave empty lines!)

```
Password=c3ltY29u
```

Then save and exit. The Service must be restarted.
The password c3ltY29u is "symcon" encoded in Base64. The password must be Base64 encoded. This tool can be used for this purpose: base64encode.org

RASPBERRY PI

The license must be set up correctly in the Management Console. The following command must then be executed:

```
sudo nano /root/.symcon
```

There should be several entries. The following must be inserted in a new line. (Do not leave empty lines!)

```
Password=c3ltY29u
```

Then save and exit. The service must be restarted.
The password c3ltY29u is "symcon" encoded in Base64. The password must be Base64 encoded. This tool can be used for this purpose: base64encode.org

DOCKER

The license must be configured in the Management Console. Afterwards, the following command must be executed:

```
sudo nano /opt/symcon/.symcon
```



The path is based on the one defined during the Installation of IP-Symcon. It can vary.

There should be several entries. The following must be inserted in a new line. (Do not leave empty lines!)

```
Password=c3ltY29u
```

Then save and exit. The service must be restarted.
The password c3ltY29u is "symcon" encoded in Base64. The password must be Base64 encoded. This tool can be used for this purpose: base64encode.org

QNAP

The license must be configured in the Management Console. Afterwards, an editor must be used to update the .symcon file. It can be found in the Root volume selecting during installation.

```
/Symcon/MeinIPSymcon/.symcon
```

The path is based on the one defined during the Installation of IP-Symcon. It can vary.

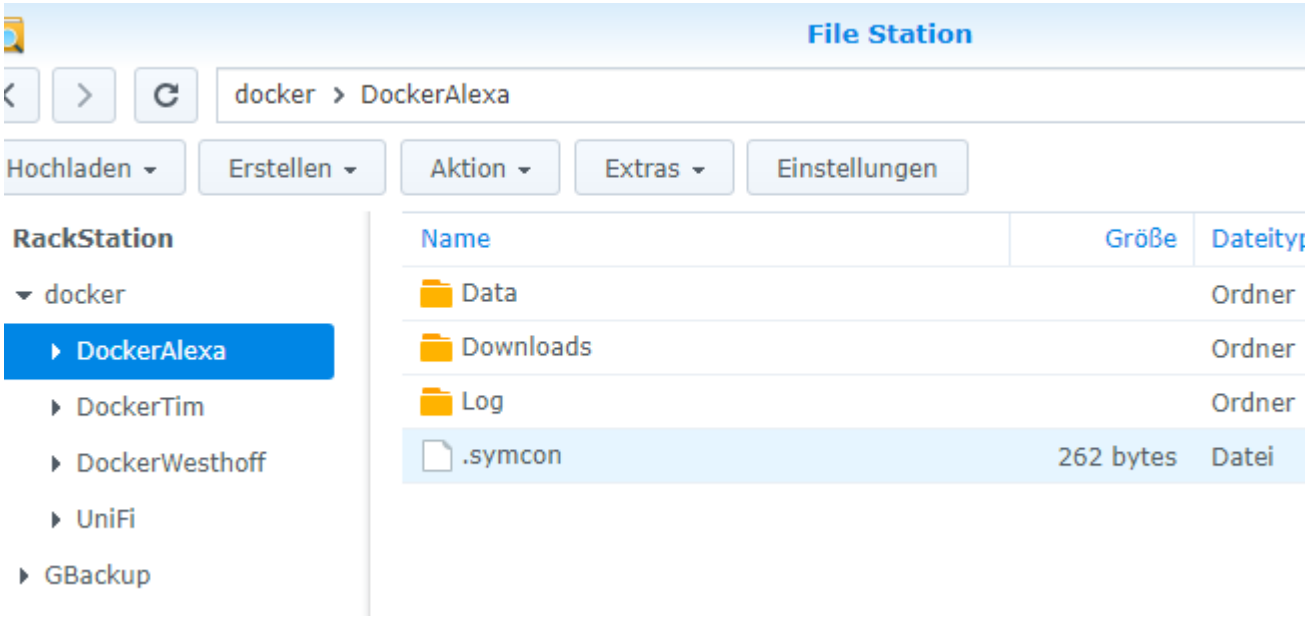
There should be several entries. The following must be inserted in a new line. (Do not leave empty lines!)

```
Password=c3ltY29u
```

Then save and exit. The service must be restarted.
The password c3ltY29u is "symcon" encoded in Base64. The password must be Base64 encoded. This tool can be used for this purpose: base64encode.org

SYNOLOGY

The license must be configured in the Management Console. Afterwards, an editor must be used to update the .symcon file. It can be found in the Root volume selecting during installation.



The path is based on the one defined during the Installation of IP-Symcon. It can vary.

There should be several entries. The following must be inserted in a new line. (Do not leave empty lines!)

```
Password=c3ltY29u
```

Then save and exit. The service must be restarted.
The password c3ltY29u is "symcon" encoded in Base64. The password must be Base64 encoded. This tool can be used for this purpose: base64encode.org

Tray

The IP-Symcon Tray Application is the interface to the IP-Symcon Service, which runs in the background. Specific settings can be administrated via the context menu. This includes, for example, starting and stopping the service and opening the Management Console.



The Tray is only available for MacOS and Windows operating systems.

The following other settings can be made via the Tray Application:

Start Live Update
Uninstall Service
Change License
Change Remote Access

Service

The Service is the centerpiece of IP-Symcon. It always runs in the background and is usually automatically launched when starting the operating system. This means that no user needs to be logged into a computer/server, to guarantee perfect operation.

CONFIGURATION FILE

The configuration (settings.json) of IP-Symcon is saved every 10 minutes or when properly shutting down. Should the power supply fail, the configuration file could become corrupt or broken.
In this case, IP-Symcon will try to use automatically saved copies of the settings.json until a working copy is found. If no functional configuration file is found, the service does not start and outputs an error message.

PHP

IP-Symcon uses PHP as its script language. As PHP is fully integrated, all of its advantages can be made use of. Among others, this means that multiple scripts can run at the same time.

VERSION HISTORY

IP-SYMCON VERSION	INTEGRATED PHP VERSION
from 1.0	5.1.x (x86 thread safe)
from 2.2	5.3.x (x86 thread safe)
from 2.5	5.4.x (x86 thread safe)
from 4.0	5.6.x (x86 thread safe)
from 5.0	7.2.x (x64 thread safe)
from 5.1	7.3.x (x64 thread safe)
from 5.5	7.4.x (x64 Thread Safe)
from 7.0	8.2.x (x64 Thread Safe)

In addition to the standard PHP functions, special IP-Symcon functions are available, through which IP-Symcon-specific settings (see Command Reference) or devices configured in IP-Symcon, can be accessed (see Module Reference).
The "php.ini" known from PHP (php.net/manual/de/ini.php) is equally available in IP-Symcon. IP-Symcon configures "extension" entries automatically, based on the available extensions that were placed in the "IP-Symcon/ext" folder.

GLOBAL INCLUDE

In order to make functions, constants, etc. available globally across all scripts, these must be defined in the file "__autoload.php". These must be present in the "IP-Symcon/scripts" folder. Multiple files can be read in within the "__autoload.php".

Inputting commands or files via "__autoinclude.inc.php" is possible, but the file is overwritten during each new update or new installation.

The PHP function "auto_prepend_file" cannot be used, as it is already called and used by IP-Symcon. This should/can only be used once throughout the system.

EXAMPLE

```
__autoload.php
require_once(IPS_GetKernelDir() . "/scripts/globalfunction.ips.php");
```

CONFIGURE

All possible configuration parameters can be taken from the PHP handbook. A useful setting for the case that some scripts (e.g. ShutterControl) eventually require it, is the extension of the maximum script runtime. When adjusting this setting, one must make sure that only a limited number of PHP scripts are run in IP-Symcon at the same time. Should all slots be occupied, due to a script having a long runtime, other scripts will be placed into a queue and only run with a delay. "Sleep" commands should therefore always be avoided and exchanged for "timer" commands.



IP-Symcon must be restarted after any changes are made to "php.ini"!

EXAMPLE

Increase maximum script runtime to 5 minutes.

```
[PHP]
max_execution_time=300
```

Interfaces

HARDWARE MODULES

sorted according to connections and alphabetical order

COMPONENTS	MANUFACTURER	CONNECTIONS	SUPPORTED DEVICES
1-Wire	Maxim Dallas	Wired	Device List
ALLNET	ALLNET	Wired	Device List
digitalSTROM	AIZO	Wired	Device List
DMX/Artnet	among others DMX4ALL	Wired	Device List
KNX	among others Siemens AG, EIBMarkt, Gira, Jung, Merten, Weinzerl	wired	Device List
eKey	eKey	Wired	Device List
IR remote control	among others IRTrans, WinLIRC	Wired	Device List
LCN	Issendorff	Wired	Device List
ModBus TCP, ModBus RTU, ModBus RTU over TCP	e.g. Wago/Beckhoff SPS	Wired	Device List
M-Bus	e.g. ALLMESS	Wired	Device List
SPS Siemens Vipa Logo	Siemens AG	Wired	Device List
Siemens OZW	Siemens AG	Wired	Device List
UVR1611	Technical Alternative	Wired	Device List
Velleman Board	Velleman	Wired	----------
WuT	W&T (Wiesemann & Theis)	Wired	----------
EnOcean	EnOcean Alliance	Wireless (868Mhz)	page://szUQAzDnAQT63wLx
FHZ1X00PC (FS20, HMS, FHT, KS300)	ELV	Wireless (868Mhz)	Device List
HomeMatic	ELV	Wireless (868Mhz)	Device List
IPS-868	IP-Symcon/ProJet	Wireless (868Mhz)	IPS-868

COMPONENTS	MANUFACTURER	CONNECTIONS	SUPPORTED DEVICES
XBee	ZigBee	Wireless (868Mhz)	----------
xComfort	Eaton	Wireless (868Mhz)	Device List
Z-Wave	among others ACT, Popp, Düwi, Merten, Innovus, Fibaro	Wireless (868Mhz)	Device List
FS10-weather	ELV	Wireless (433Mhz)	----------

VIRTUAL MODULES

in alphabetical order

DESIGNATION	FUNCTIONS	SPECIAL FEATURES
Archive Control	Saves, manages and provides logged variables.	
Calendar Control	Manages planned presence/absence and provides available variables.	
Connect Control	Opens an encrypted tunnel to the Connect Service, which permits a secured remote access.	Requires an active subscription
Cutter	Processes binary data and cuts/synchronizes them. Can be used with the interface modules.	
Event Control	Reaction to particular events, such as system start/stop, in order to save/input particular system conditions	
Heating Control	2-point regulator that switches heating actuators based on actual and set temperatures, in order to regulate using preset temperatures.	
IMAP	Serves an IMAP-based retrieval of e-mails.	
Location Control	Calculates various twilight times, based on longitude and latitude.	
MediaPlayer	WAV, MP3, WMA, Shoutcast, IceCast, WMALive	Several sound cards simultaneously
Module Control	Provides various modules, integrated over a git repository.	
Notification Control	Manages all devices registered for push notifications.	Push Notifications require an active subscription
POP3	Serves the POP3-retrieval of e-mails.	
Presence Control	Offers the possibility of assessing a presence in a particular area	

DESIGNATION	FUNCTIONS	SPECIAL FEATURES
RegisterVariable	Provides a data transfer and processing interface	
SMTP	Serves to send e-mails via SMTP.	
SMS	Serves to send SMSs with help of an external SMS provider.	
System Information	CPU Load, hard drive capacity, battery power, printer queue	
Shutter Control	Positions your shutters to an exact percentage value, even if your module can only move up or down.	Must be taught once
TextParser	Extracts data from a text that, e.g., was delivered by the WWW Reader.	
Text To Speech	Synthesizes language for direct output from a loudspeaker or into a desired WAV file with various formats/qualities.	requires SAPI 5.1 or higher (XP, W2k3, Vista)
Util Control	This module is an interface between the IP-Symcon service and the management console.	
WebHook Control	Offers the possibility of retrieving scripts via the browser.	Can be used externally when combined with Connect Control
WebServer	Provides a further port for the relevant WebFront.	SSL-encryption activated

LANGUAGE ASSISTANCE SYSTEMS

in alphabetical order

SYSTEM	MANUFACTURER	SPECIAL FEATURES
Amazon Alexa	Amazon	Requires Symcon Connect
Google Assistant	Google	Requires Symcon Connect

VISUAL MODULES

in alphabetical order

DESIGNATION	FUNCTIONS
Dummy Module	Optically creates an instance in WebFront.
Image Grabber	Automatically creates webcam images as media objects and thereby makes them directly available to WebFront.
Popup Module	Creates a pop-up in WebFront and displays linked objects.
Skin Control	This module manages available skins and allocates them to various WebFronts.

DESIGNATION	FUNCTIONS
WebFront Visualization	Manages a WebFront and offers various possible settings.

INTERFACE MODULES

in alphabetical order

DESIGNATION	FUNCTIONS	SPECIAL FEATURES
Client Socket	----------	TCP-Client
HID	----------	----------
HTTPClient	Retrieves/reads websites	Proxy, Basic Authentication Support
Multicast Socket	----------	Multicast-Client
Serial Port	----------	RS232/RS485
Server Socket	----------	TCP-Server
UDP Socket	----------	UDP-Client

Management Console

The Management Console is the tool for setting up and configuring the complete logics of the IP-Symcom Server. There are two versions of the Management Console: The web based Management Console and the Pro Console.

WEB-BASED MANAGEMENT CONSOLE

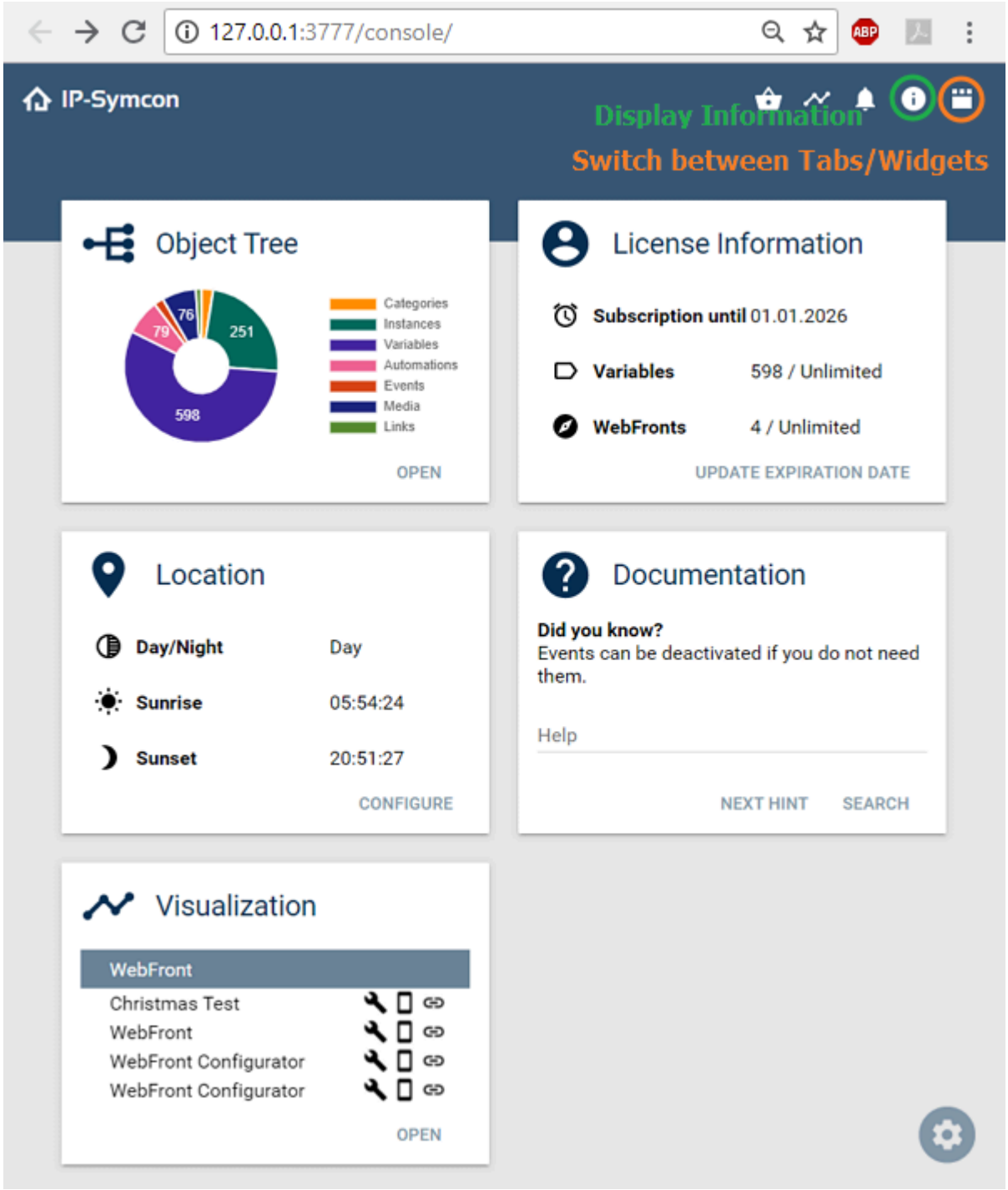
since IP-Symcon 5.0
The web-based Management Console is platform-independent and can be accessed via browser. Officially recommended browsers are: Google Chrome, Apple Safari and Opera

CONNECT TO THE IP-SYMCON SERVER

The web-based Management Console can be opened in a browser via "[IP]:3777/console/" .

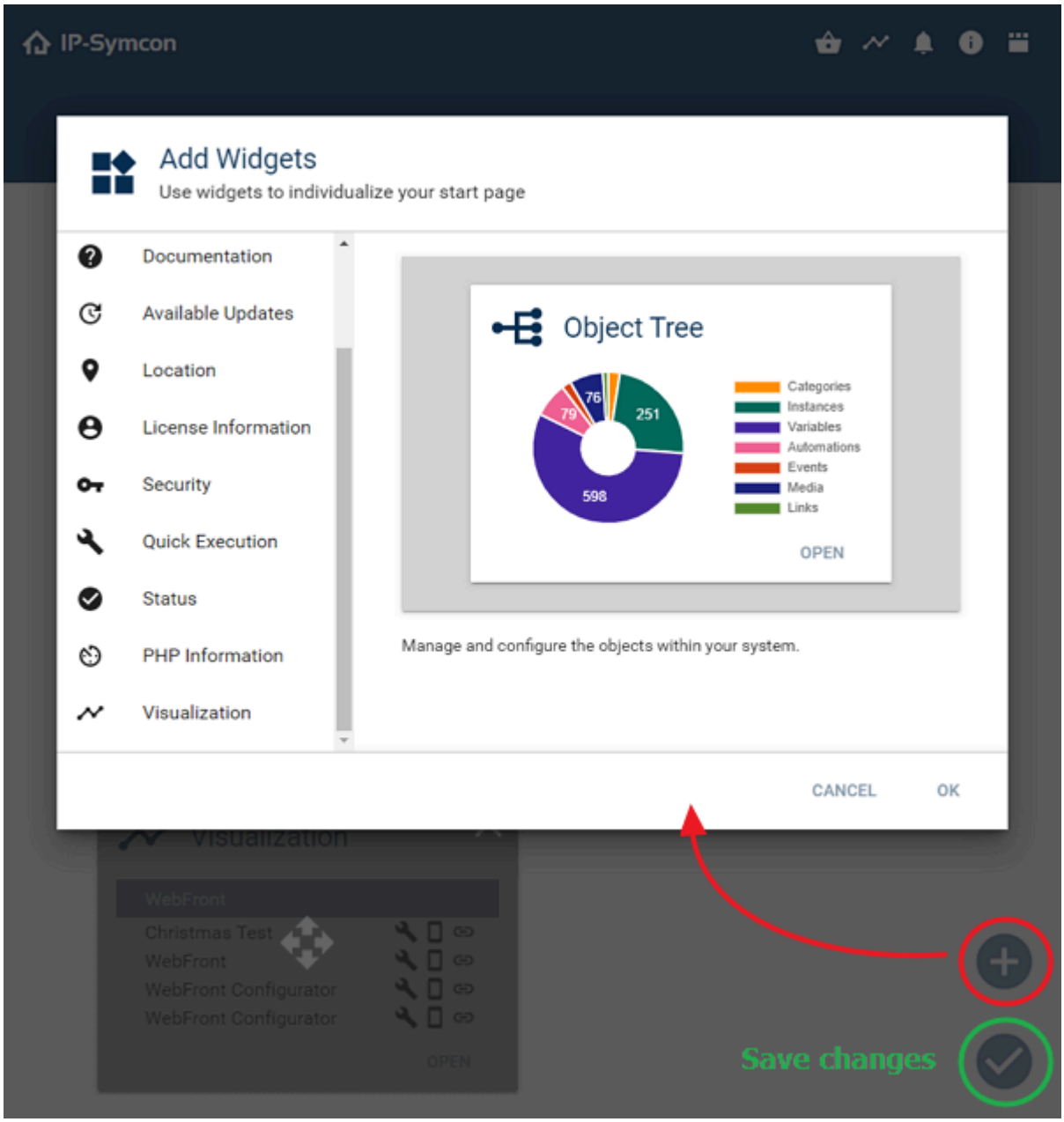
THE START PAGE

Widgets are shown on the start page of the Management Console. These are individually configurable and offer an initial view of the many activities of the system.



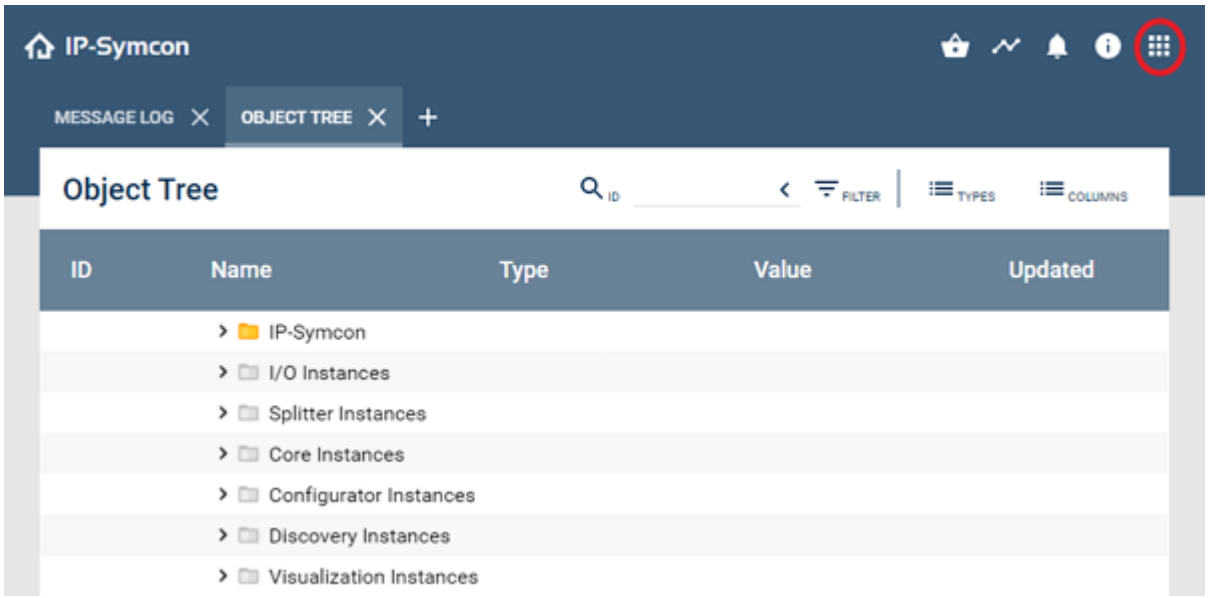
CONFIGURING WIDGETS

At the start page, one can display and remove widgets using the cogwheel button.



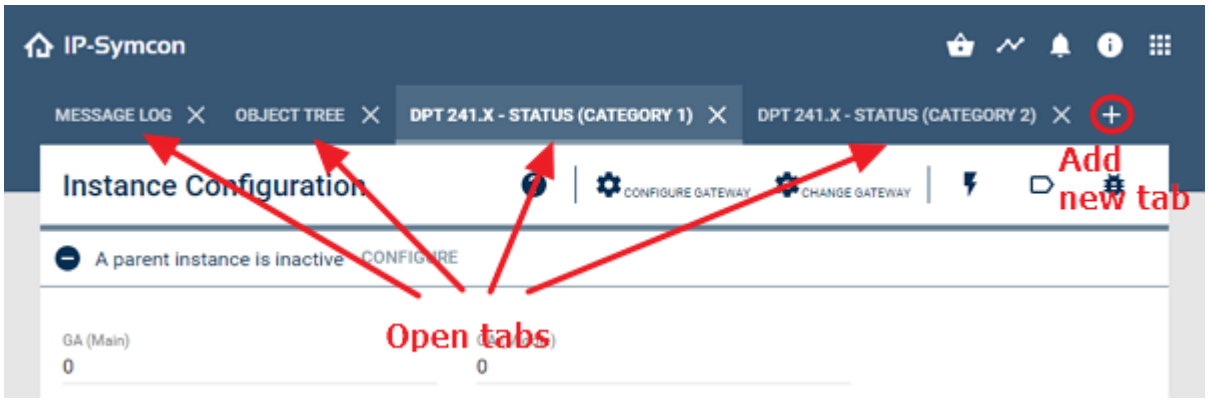
VIEW CHANGER

Using the button on the top right, one can quickly switch between the start page and the Tab List.



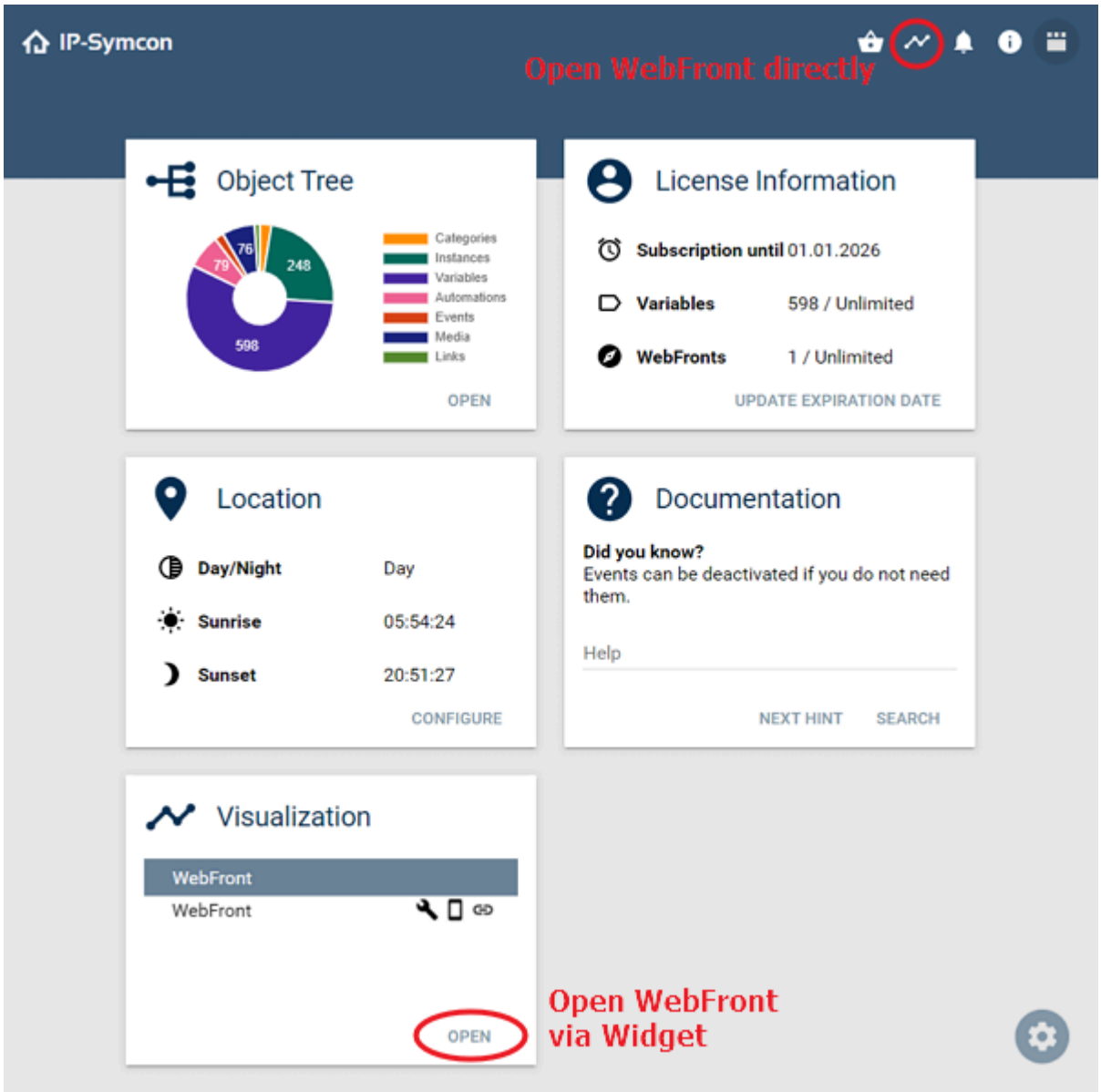
TAB LIST

Diverse tabs can be opened in the tab list. This includes the Object Tree, the Messages , or the configuration of Instances or Automations. The Plus Icon at the right of the tab list can open new tabs. If multiple tabs of objects with the same name are open, the path to those objects is displayed as well, so the corresponding object can be properly identified.



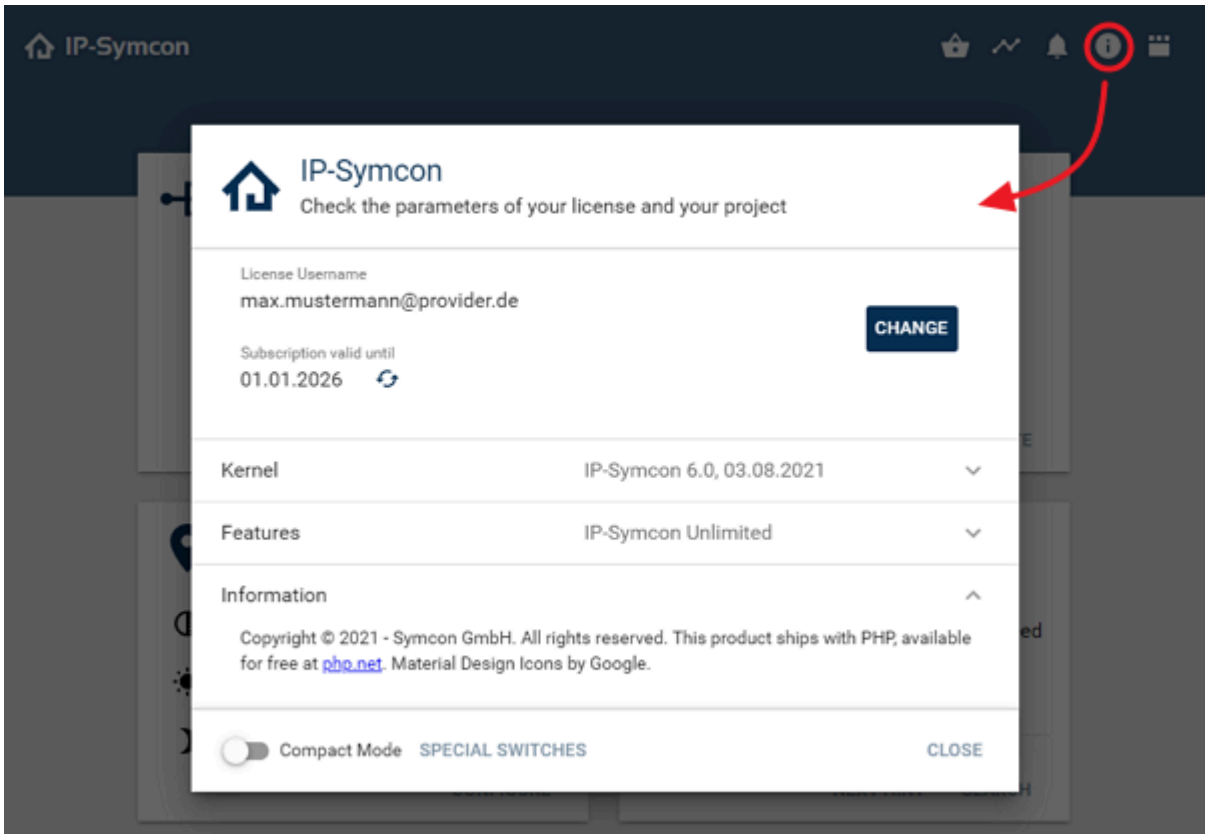
VISUALIZATION

The WebFront can be opened directly via the button on the top right.



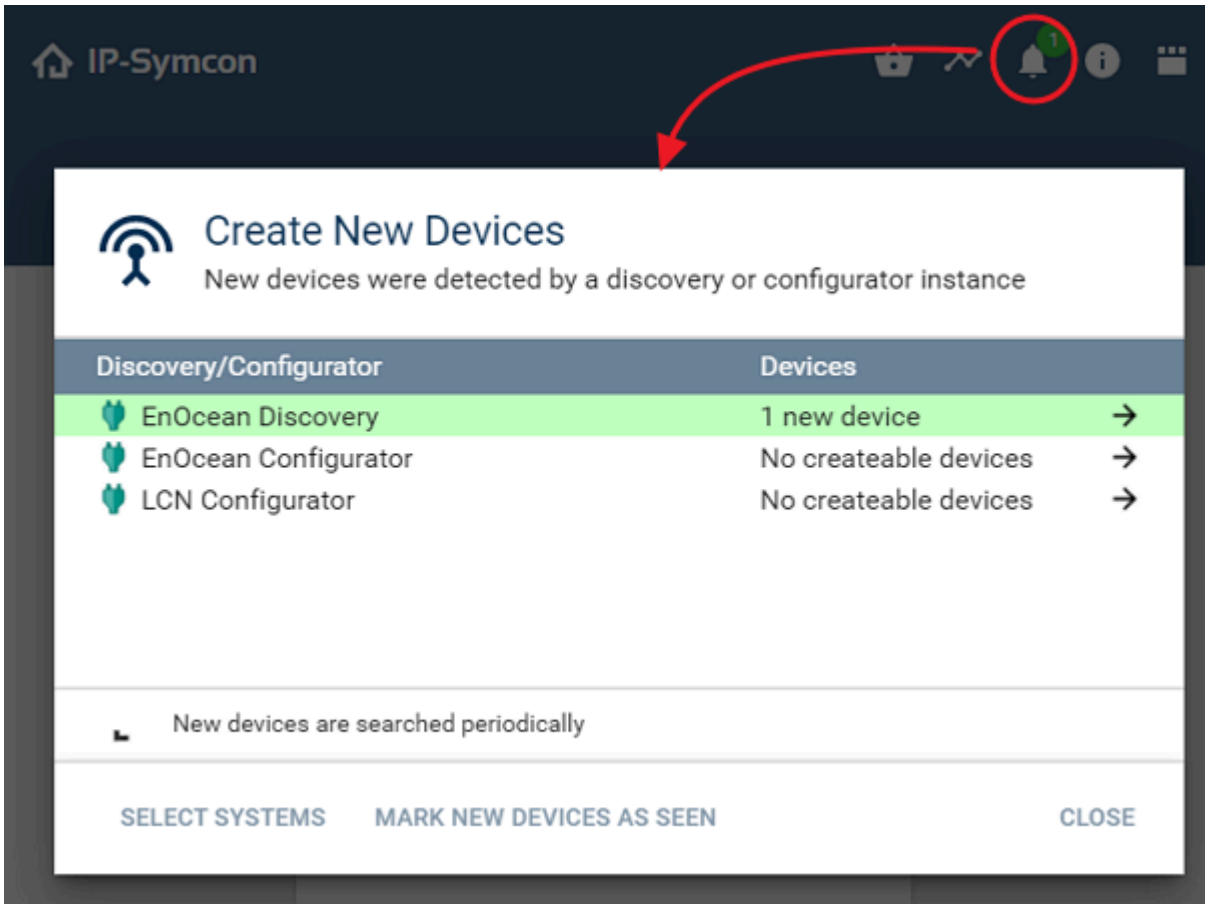
INFORMATION

A variety of information can be retrieved using the information button.
These include kernel and feature information, as well as a button to the Special Switches.



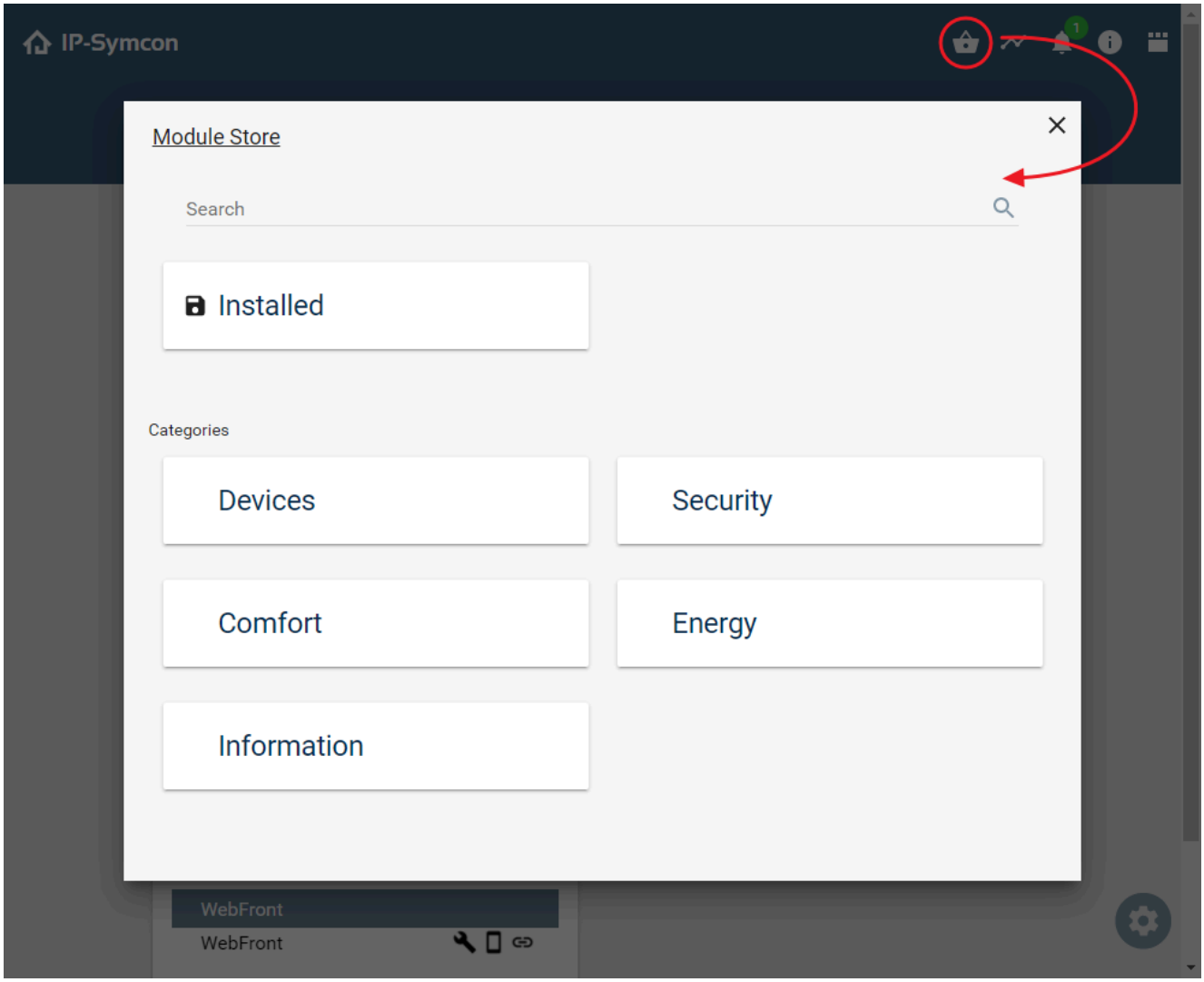
DEVICE SEARCH

Discovery instances can be configured via the Device Search button.
These inform about newly connected devices and simplify the configuration of new devices.



MODULE STORE

New modules can be installed via the Module Store. These modules expand the functionality and enable the usage of new functions or additional devices.



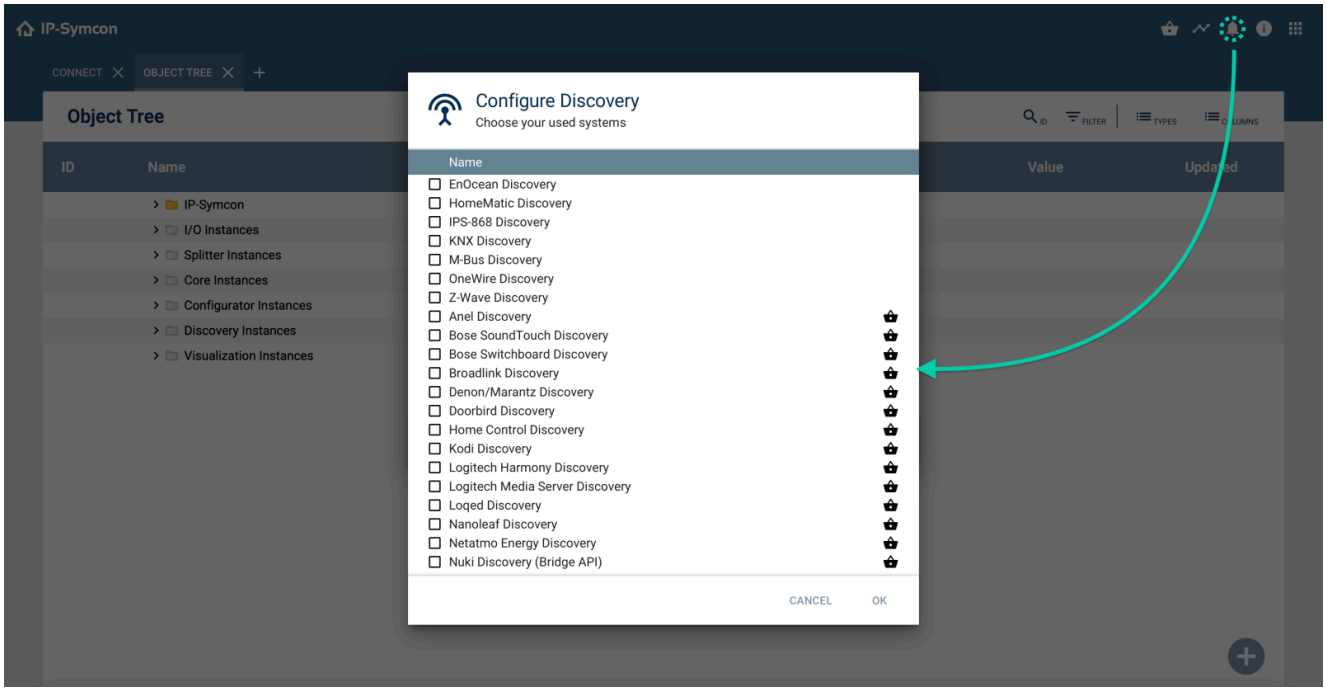
Device Search

The Device Search button can be used to set up Discoveries and view Configurators. These provide information about newly connected devices and simplify the setup of new devices.

SET UP DISCOVERY

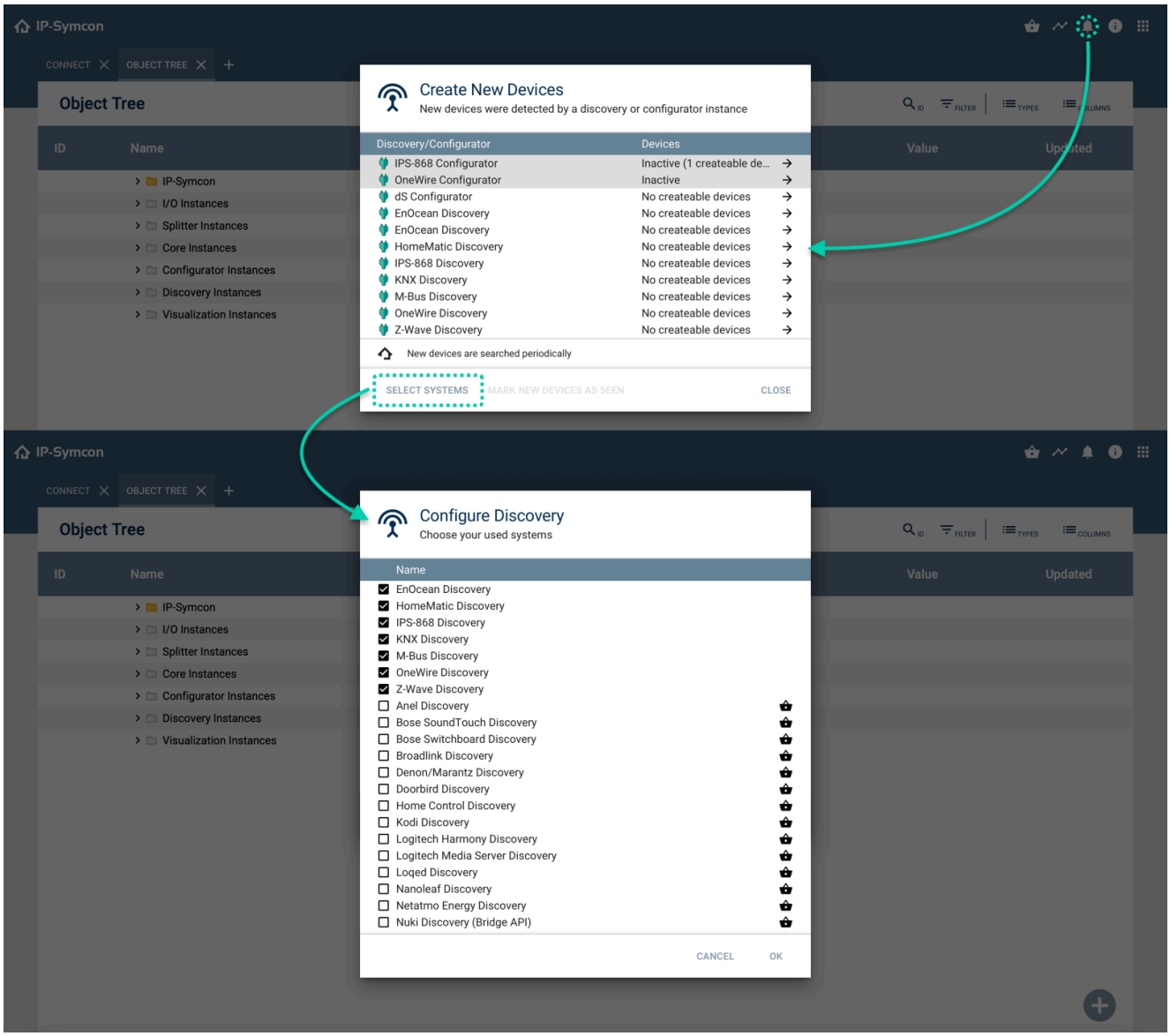
The Device Search can be accessed via the bell icon at the top right of the console.

INITIAL SETUP

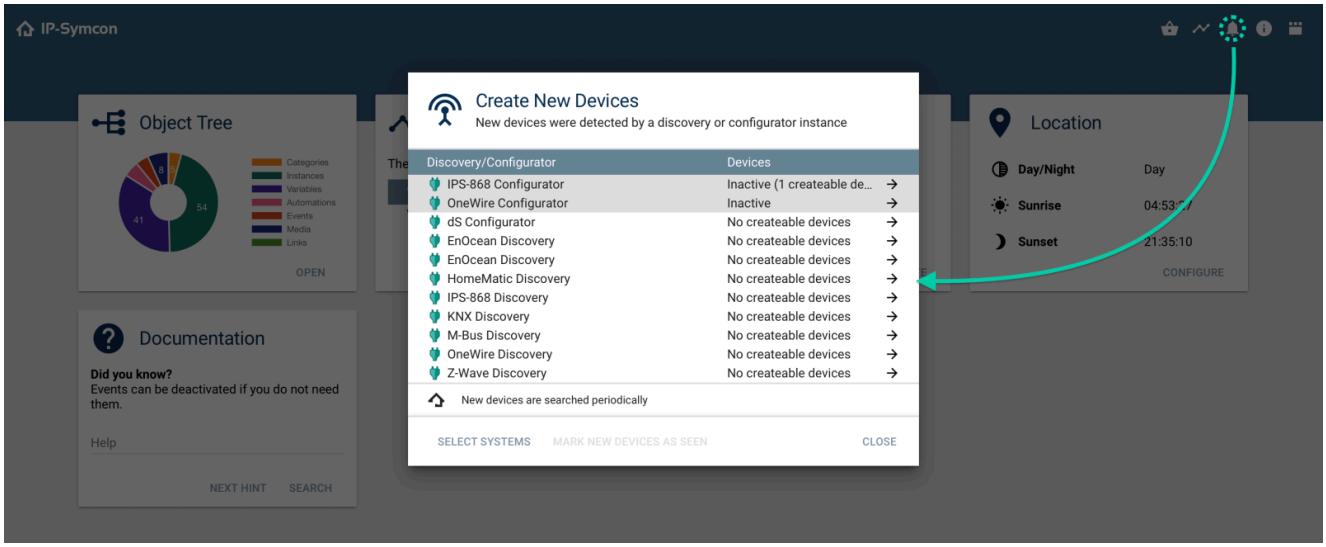


New Discoveries can be set up by ticking the box on the left side.
If there is a basket on the right edge, it is a Discovery from the Module Store. These can be opened by clicking on the basket.
When setting up a Discovery, the required IO/splitter instances are automatically created.

ADDITIONAL SETUP



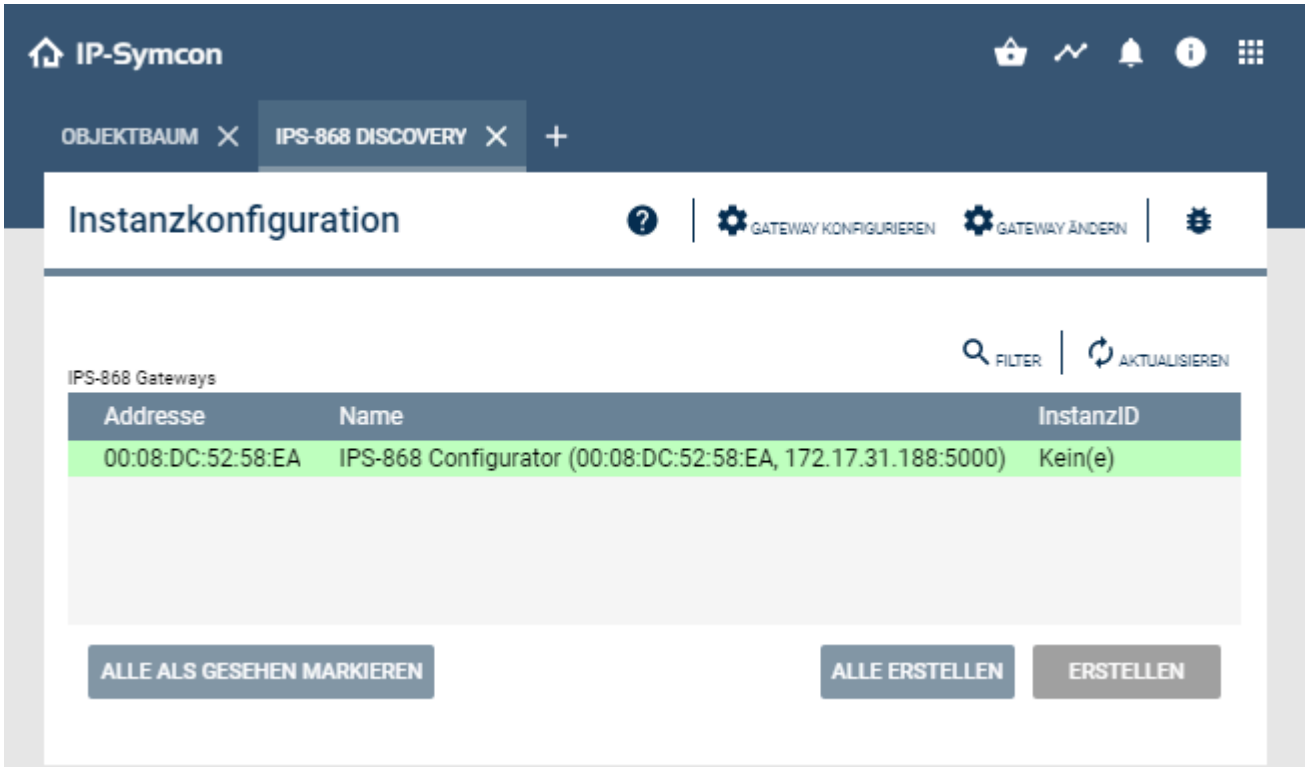
Additional Discovery Systems can be added via "Select Systems".



Configurators can be created and configured automatically using previously established Discoveries. The created configurators are displayed in the Device Search.
The configuration for the established Discoveries/Configurators can be opened using the arrows on the right side.

CREATE A CONFIGURATOR

System-specific Configurators can be created in the respective Discovery. To do this, the appropriate entry must first be selected from the list. Then "Create" has to be clicked.
The Discovery automatically mirrors the complete configuration of the Configurator making it immediately functional.



VARIOUS STATUSES

The Device Search icon shows directly whether, for example, a new device can be created. There are three statuses for this, which are indicated by the icon on the console.

STATUS UNCHANGED

If no new devices have been detected, all devices that can be created have been created or marked as "seen", only the standard symbol is displayed.



NO DISCOVERY IS SET UP

No Discovery instance has been set up yet.



NEW DEVICES CAN BE CREATED

When new devices are recognized, their number is shown in a green circle.
If no setup is desired, this message can be hidden in the Device Search window with "Mark new devices as seen"



Messages

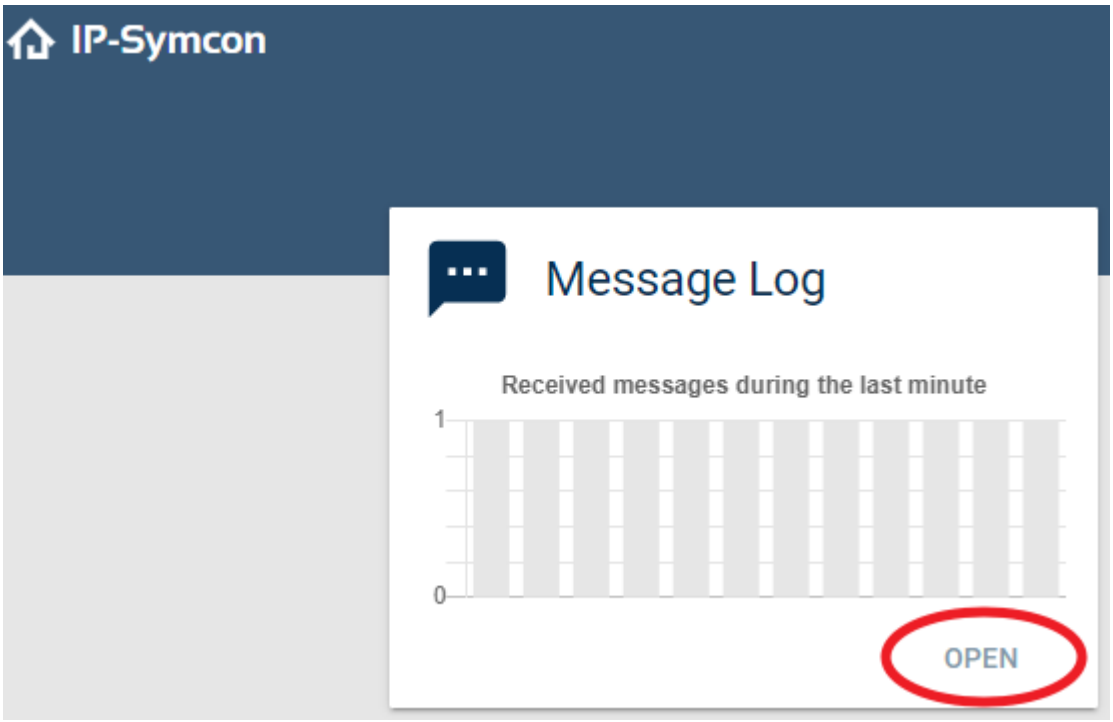
The Messages tab displays information about operations that are currently happening on the IP-Symcon server. Here, changes, activities, and also error messages are collected and can be filtered.

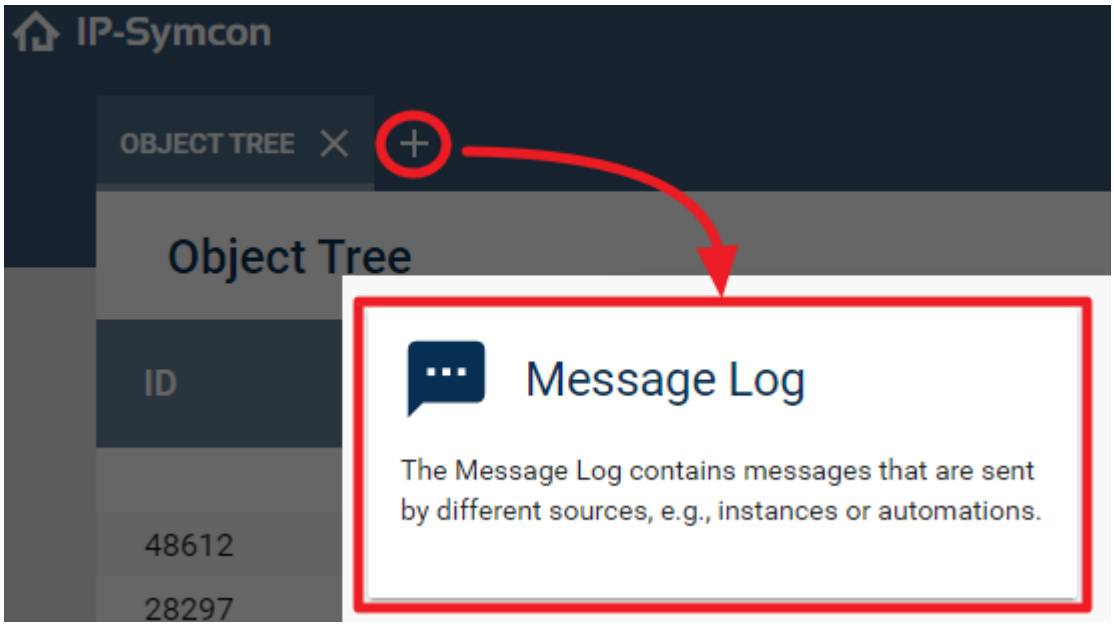


By double-clicking or "right-click -> open" on a variable within the message windows, it can be changed.

CONFIGURATION

The messages can be accessed via Message Log Widget or the plus in the tab list.
This is done by simply clicking "Open" in the widget or openening the Message Log with the plus in the tab list.





COLOR CODES

The different messages are displayed in color in the message window.
Transparent background:
A default message. (MESSAGE)
Gray background:
A custom message. (CUSTOM)
Green background:
A success message. (SUCCESS)
Purple background:
A notification message. (NOTIFY)
Yellow background:
A warning message. (WARNING)
Red background:
An Error Message. (ERROR)

MENU ITEMS

START/STOP

Starts and stops the recording of messages. Thus, the previous messages can be analyzed calmly.

CLEAR

Clears all previous messages.

LIMIT MESSAGES

The maximum number of displayed messages. When the limit is reached, the oldest messages are removed before new ones are added. This can cause the list to scroll automatically, even if the AutoScroll feature has been disabled.

AUTOSCROLL

When enabled, the window automatically scrolls to the most recent messages.

FILTER

Here you can select which type of messages should be displayed.

QUICK FILTER

The "Sender" and "Message" columns are searched for the content of the input field. The search is activated when entering the search field itself, but can be activated/deactivated by clicking on "Quick filter".
Searching for multiple terms is not possible.

Error Messages

An error message is displayed either directly or in the message log. Error messages are shown in red in the message window.
All error messages that appear in the message log are also saved in the log file in the "logs" folder.



The following applies: If an error message is not displayed directly, it will be displayed in the message log.

EXAMPLES FOR A DIRECT OUTPUT OF THE ERROR MESSAGES:

Execute in the Script Editor
Execute in WebFront and the app
Execute one's own action (Script / PHP Module) via WebFront and app
Executing Web Hooks and OAuth Hooks

EXAMPLES OF ERROR MESSAGES IN THE LOG:

Events (cyclic, trigger)
Script timer
Timers of instances (including PHP modules)
Error in the start and stop script in the Event Control
Error when evaluating a Register Variable

Module Store

New modules can be installed via the Module Store. These modules expand the range of functions and thus enable the use of new functions or additional devices.

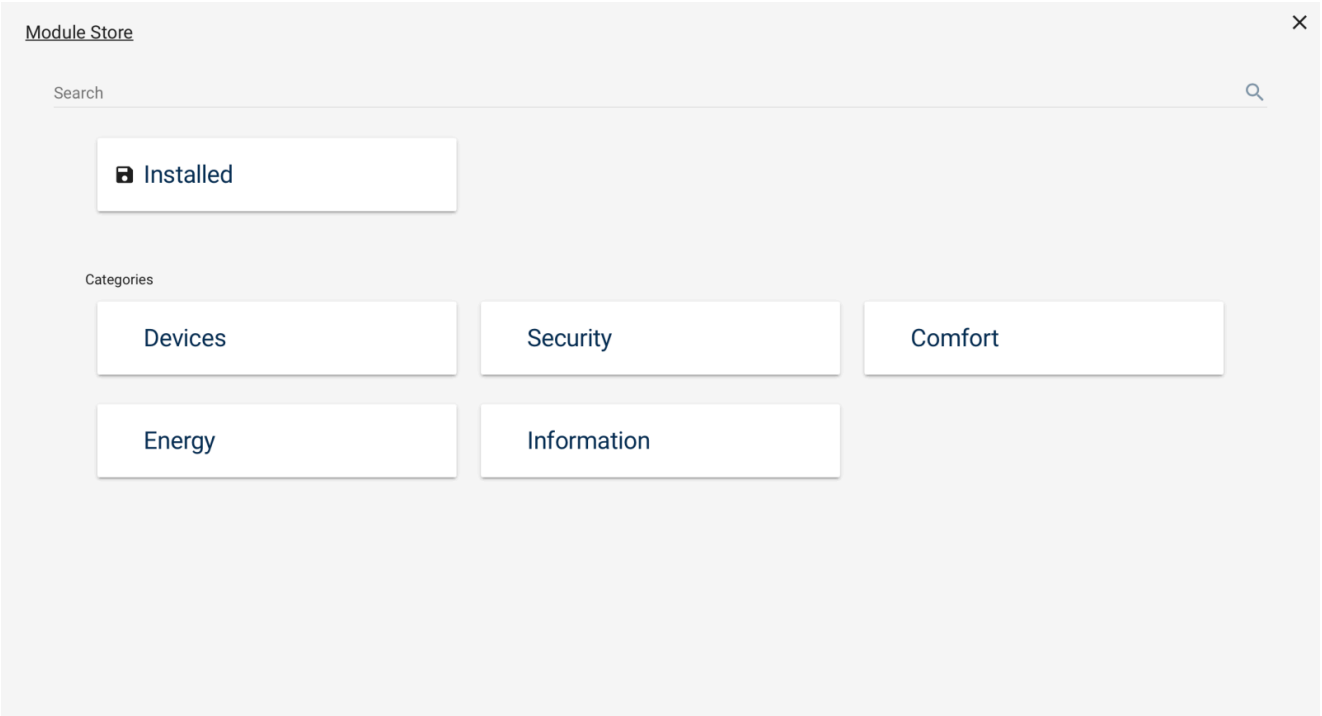
Search all available modules in the Module Store in the browser.

OPEN MODULE STORE

The module store can be reached via the basket in the upper right corner of the console.

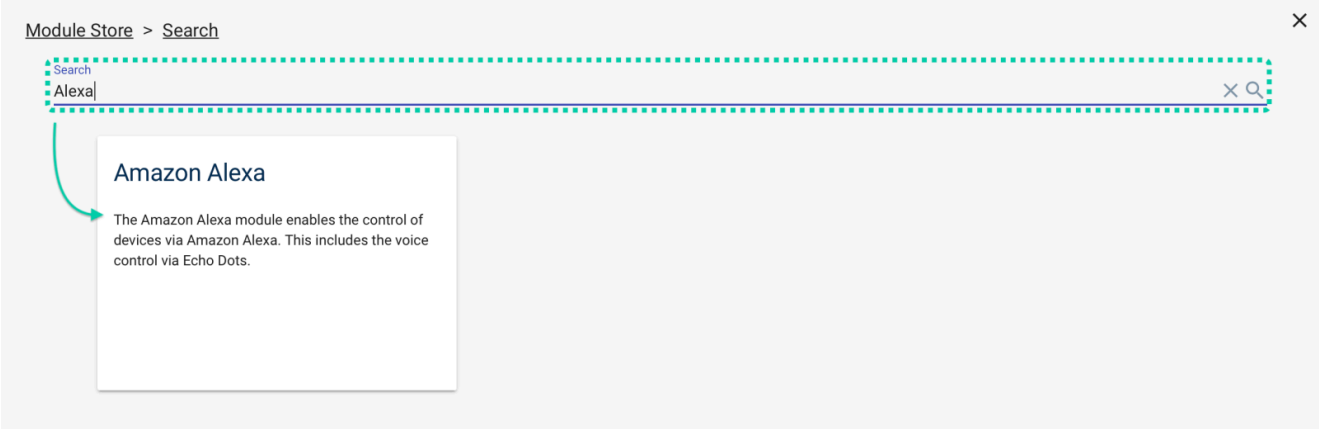


Initially, the Module Store shows different categories in which modules are offered and provides a search function. If modules have already been installed via the Module Store, they can be accessed via a corresponding special category.

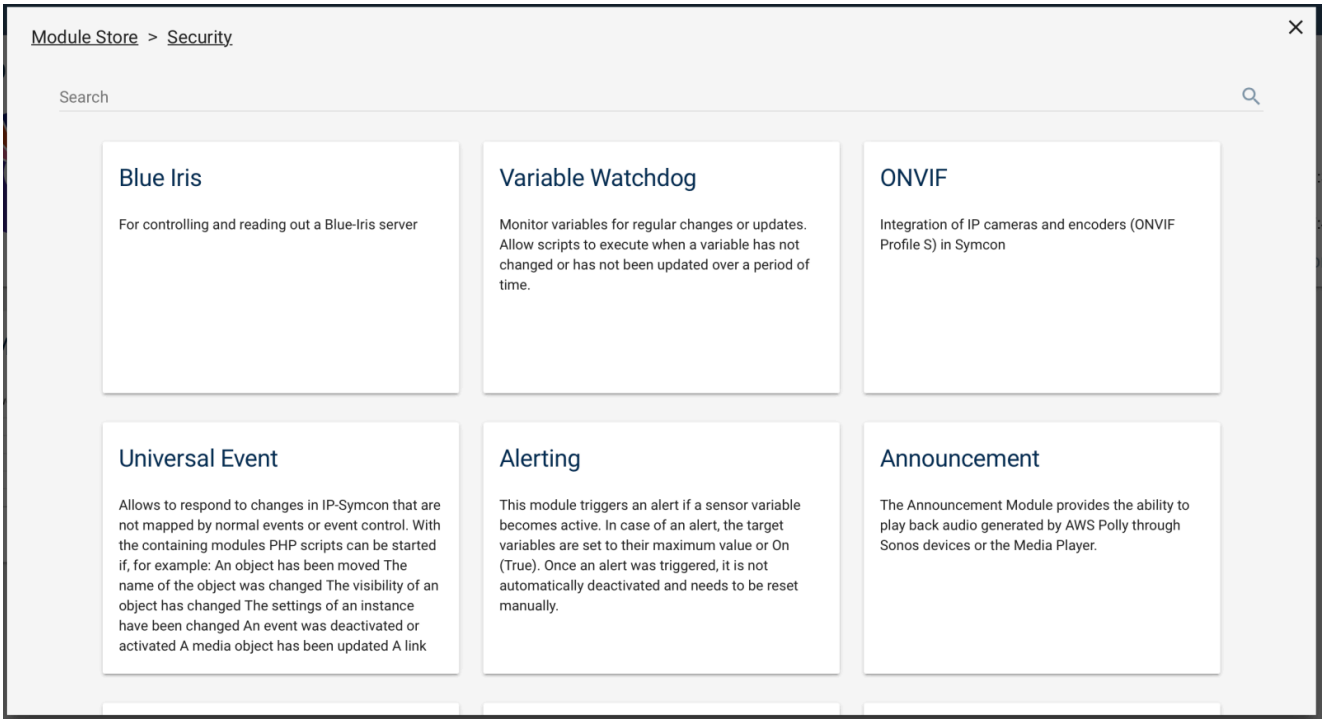


FIND MODULES

There are various options for finding a suitable module. On the one hand, the search function can be used. A search term can be entered here and confirmed by pressing the Enter key or clicking on the magnifying glass. Matching modules are then displayed.



Alternatively, a category can be selected by clicking on it to display the modules it contains. Categories can contain sub-categories, which can then also be selected. One can return to a previous category by clicking on it in the top bar. A click on "Module Store" returns one to the start screen.



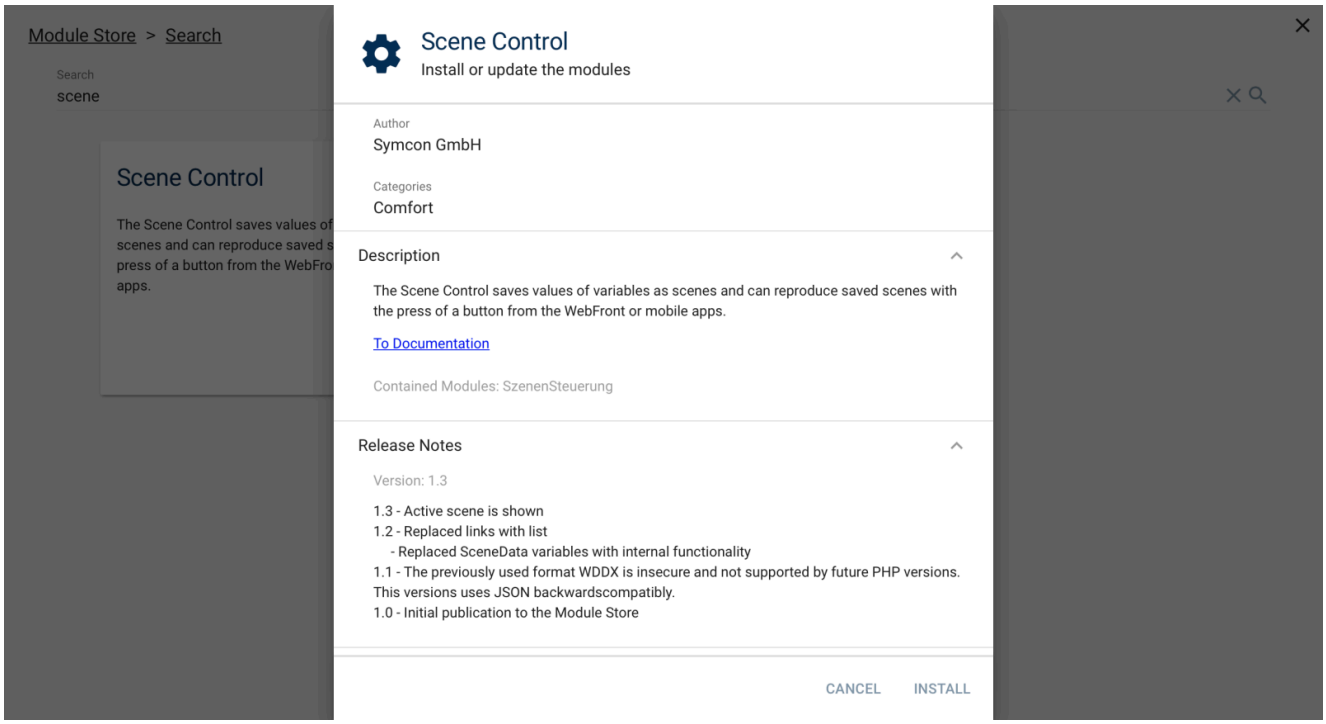
These two methods can also be combined. One can search for a term in a certain category or limit a search result afterwards by using categories.



If a search term is used, then sub-categories are also searched. Without a search term, only modules from the current category are displayed, but not their subcategories.

INSTALL MODULES

If modules are to be installed, the associated card must first be clicked. Following, a dialog opens which explains the currently selected modules.



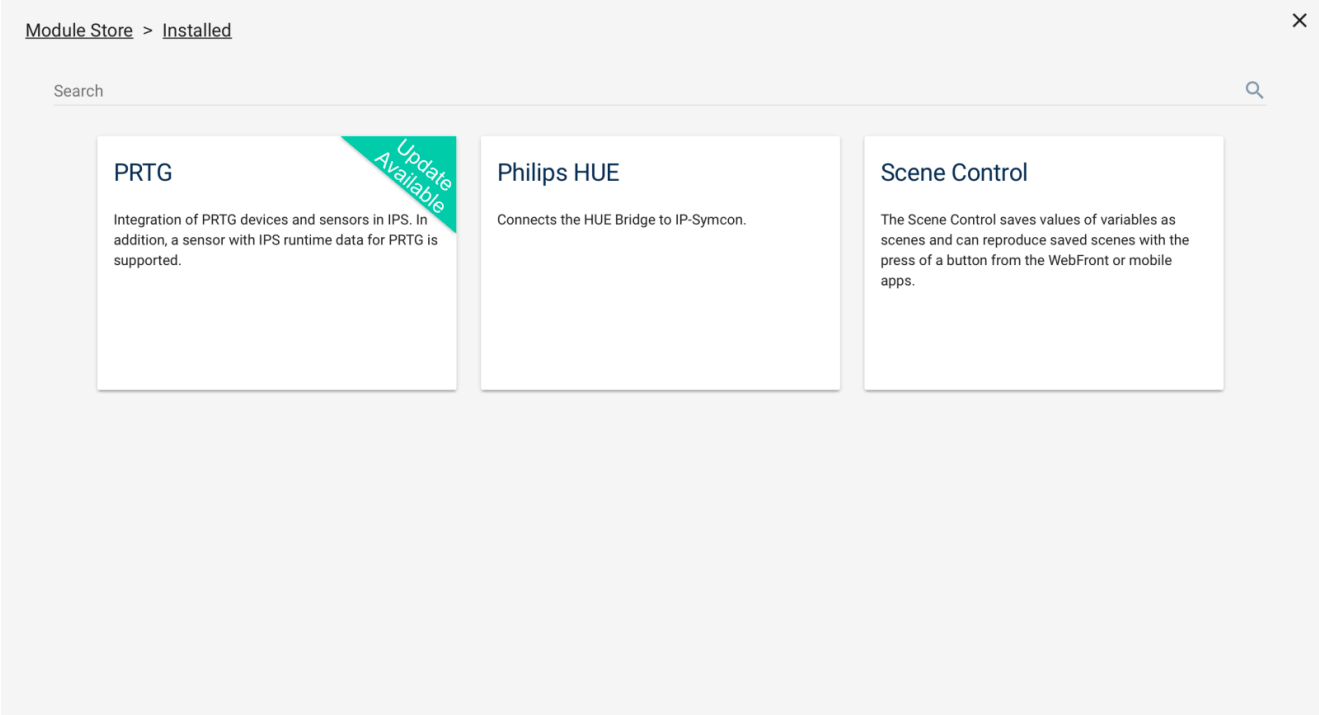
ELEMENT	DESCRIPTION
Author	The author of the modules
Categories	The categories in which the modules are contained
Description	The description of the modules. In addition, there is a link to the documentation and a list of the included modules
Version information	The current version number and a description of what is new about that version
Beta settings	If the modules have a beta version, this can be used to switch to the beta version

At the bottom of the dialog there is the "Install" button to install the modules and "Cancel" to close the dialog again.
If a beta version is to be installed, the beta settings can be expanded and "Switch to beta" may be clicked on. The module information will then be adapted to the beta version and this can be installed by clicking on "Install".

Beta versions are not checked by the Symcon-Team.

UPDATE AND REMOVE MODULES

If modules have been installed, a special category "Installed" appears on the start screen of the module store. All installed modules are listed here. If there are new versions for modules this is highlighted.



If an installed module is clicked on, the installation dialog appears again. One now has the additional option of deleting the modules by clicking on "Remove".
If there is a new version, one can install it using "Update". If one desires to change the channel afterwards, i.e. from the regular stable version to beta or the other way around, this can be done via the beta settings and is confirmed with "change channel". If the current version is installed, one can download it again via "Reinstall".

Object Tree

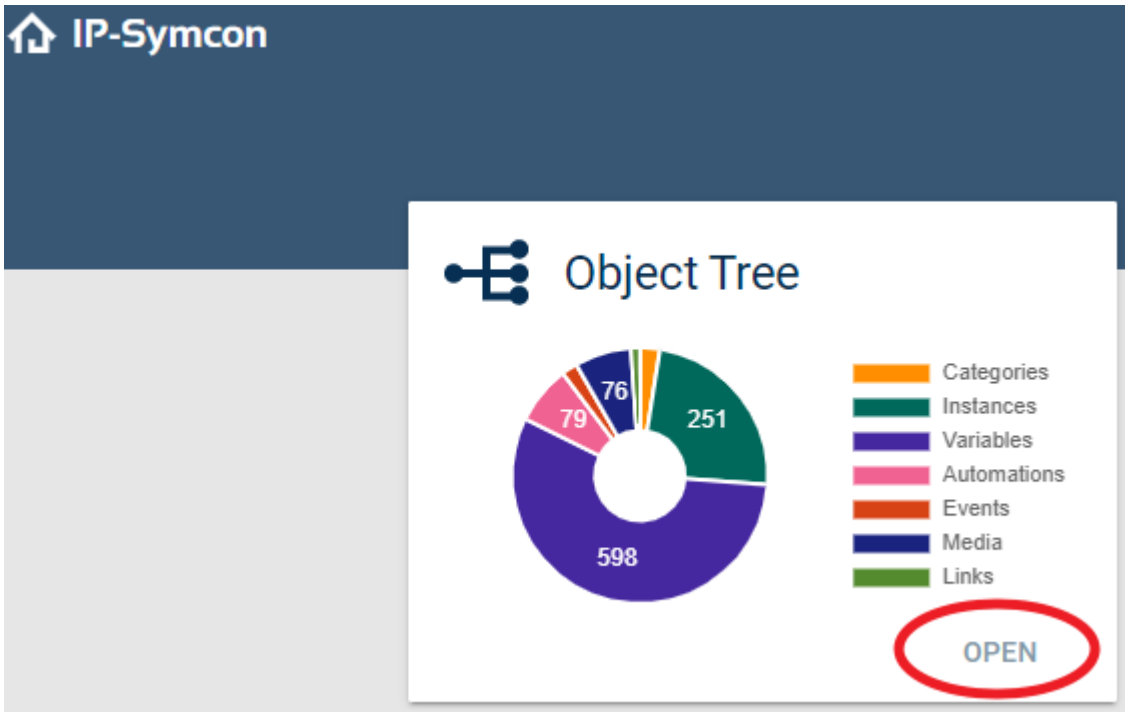
The Object Tree is used to overview and manage all objects and instances. The structure used here is also used as foundation for the WebFront. Objects can be used Drag & Drop.

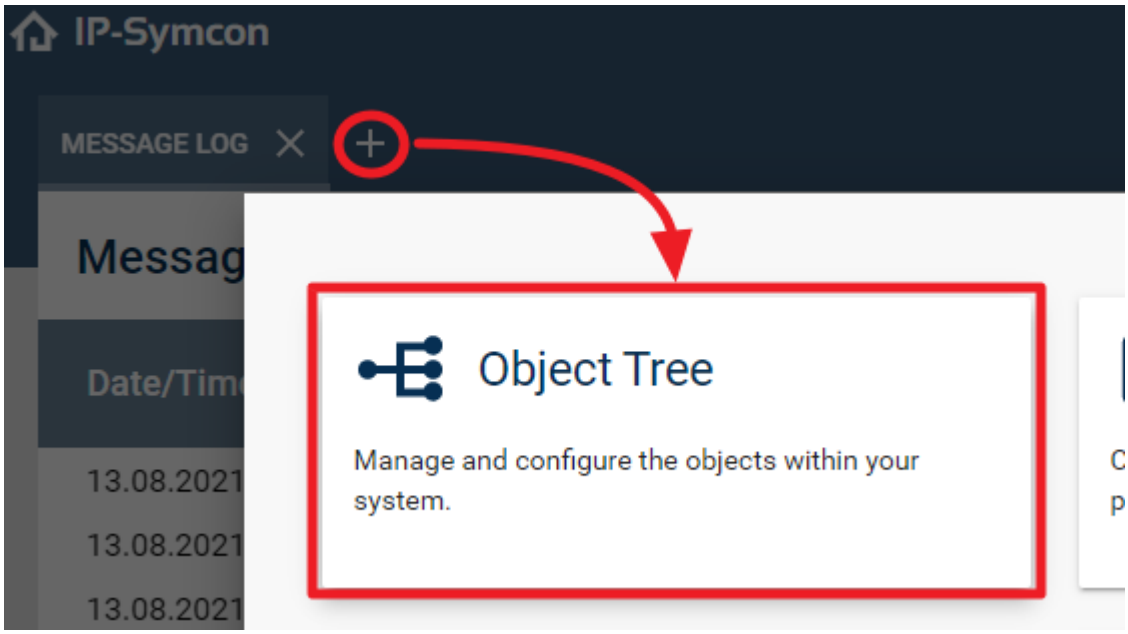


The explanation of the context menu of the different instances can be found in the Basics.

CONFIGURATION

The Object Tree can be accessed via Object Tree Widget or the plus in the tab list.
This is done by simply clicking "Open" in the widget or openening the Object Tree with the plus in the tab list.





COLOR CODES

In the Object Tree the status of the objects is displayed with color codes.
Black font:
Everything is normal and the object is displayed in the WebFront.
Light gray font:
Everything is normal and the object does not appear in the WebFront.
Gray exclamation mark:
The object is inactive.
Light gray exclamation mark:
A parent instance of the object is inactive.
Red exclamation mark:
The object has an error condition.
Light red exclamation mark:
A parent instance has an error condition.

ADDITIONAL FUNCTIONS

When hovering over time or a status icon, a tool tip with the current date or the type of complication is displayed.

MENU ITEMS

PLUS BUTTON

Via the dialog "+" at the bottom right, new objects can be created. Alternatively, by right-clicking on the desired position in the object tree, an object can be added to this position. The respective object types are described in more detail in the Basics.

ID

You can search for an ObjectID using the search input. If found, the object tree automatically jumps to the ID you are looking for and selects it.

FILTER

When entering text into the search field, the quick filter is automatically activated. It can be switched on and off again by clicking on the quick filter. It is searched in the column "Name". If the column "Description" is visible, it is searched as well.

TYPES

The visibility of specific object types can be activated or deactivated to focus the overview on specific types. If something is hidden, this button is highlighted in gray.

COLUMNS

Overview functionality to enable and disable the displayed columns.

QUICK FILTER

When entering the search field, the quick filter is automatically activated. This can be switched on and off again by clicking on the quick filter. It is only searched in the column "Location/Object".

Object Tree (physical)

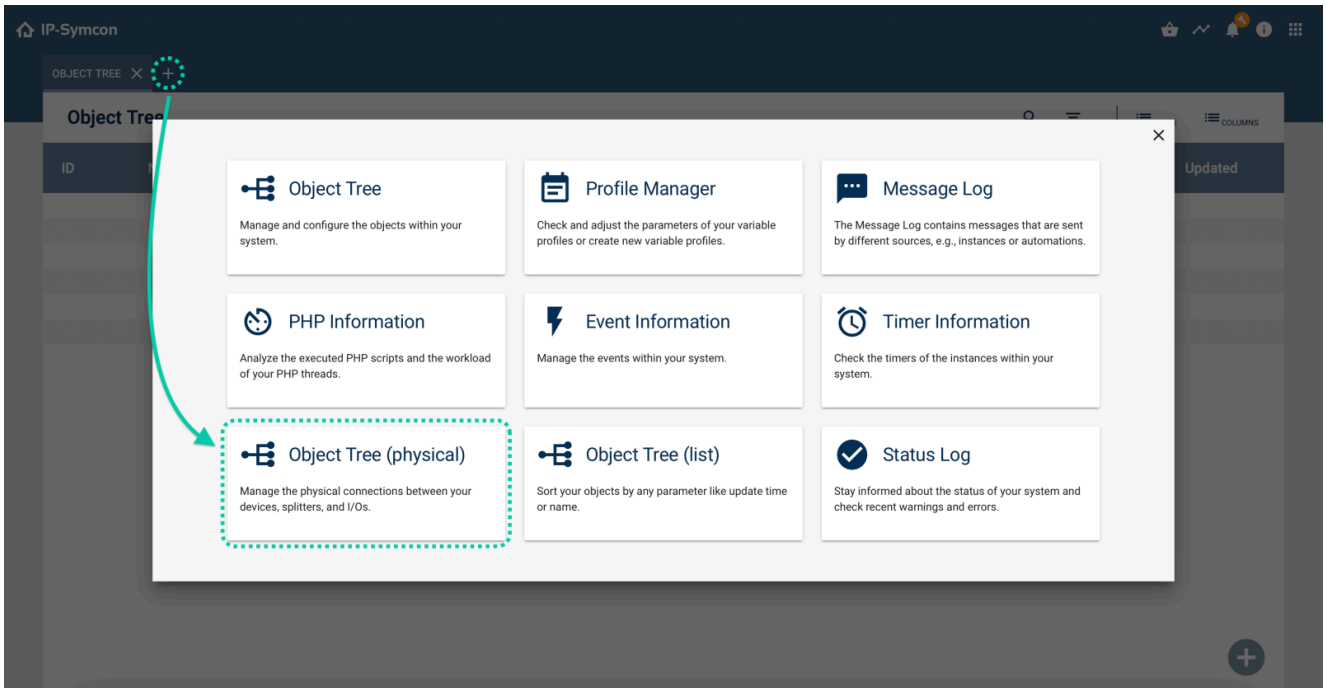
The physical view of the object tree only shows unconnected and I/O instances at the top level. In the I/O area, the devices are shown in the structure of the actual physical connection. The structure is illustrated under Parent Instances. (I/O -> Gateway -> Device). Only instances are displayed.
Unconnected instances include those that are not physically present.



The explanation of the context menu of the various instances can be looked up in the Basics.

SETUP

The object tree can be opened via the plus in the tab bar.



COLOR CODES

In the object tree the status of the objects is shown with color codes
Black font:
Everything is normal and the object is displayed in the WebFront.
Light grey font:
Everything is normal and the object is not displayed in the WebFront.
Grey exclamation mark:
The object is inactive.
Light grey exclamation mark:
A higher-level instance of the object is inactive.
Red exclamation mark:
The object has an error state.

Light red exclamation mark:
A higher-level instance has an error state.

ADJUSTING CONNECTIONS

Using drag & drop, an instance can be moved to another compatible connection and will then be assigned to it. If the target instance is not compatible, the move will be prevented.

FURTHER FUNCTIONALITIES

If a time or an exclamation mark is hovered over in the object tree, a tooltip appears which shows, for example, the entire date or the type of fault.

MENU ITEMS

ID

An ObjectID can be searched for using the search dialog. If it is found, the object tree automatically jumps to the ID one is looking for and selects it.

FILTER

With entry to the search field the filter is automatically activated. This can be activated/ deactivated by clicking on "Filter". The search is only carried out in the “Name” column.

TYPES

Here the visibility of certain objects in the object tree can be switched on and off in order to set the overview or focus on certain types. If something is hidden, this button is highlighted in grey.

COLUMNS

Overview functionality to activate and deactivate the displayed columns.

Pro Console

The Pro Console is an expert tool to setup and configure the complete logic of the IP-Symcon server. It includes all features of the web based Management Console and additional expert options.

INSTALLATION

The Pro Console can be installed for Windows, MacOS, and Linux Ubuntu.



The Pro Console was introduced with IP-Symcon 5.4. However, it is function since IP-Symcon 5.0. For older versions, not all functions are available.

WINDOWS

The current Pro Console for Windows can be downloaded here .

MACOS

The current Pro Console for MacOS can be downloaded here .

UBUNTU

If not already done, add the Symcon Repo:

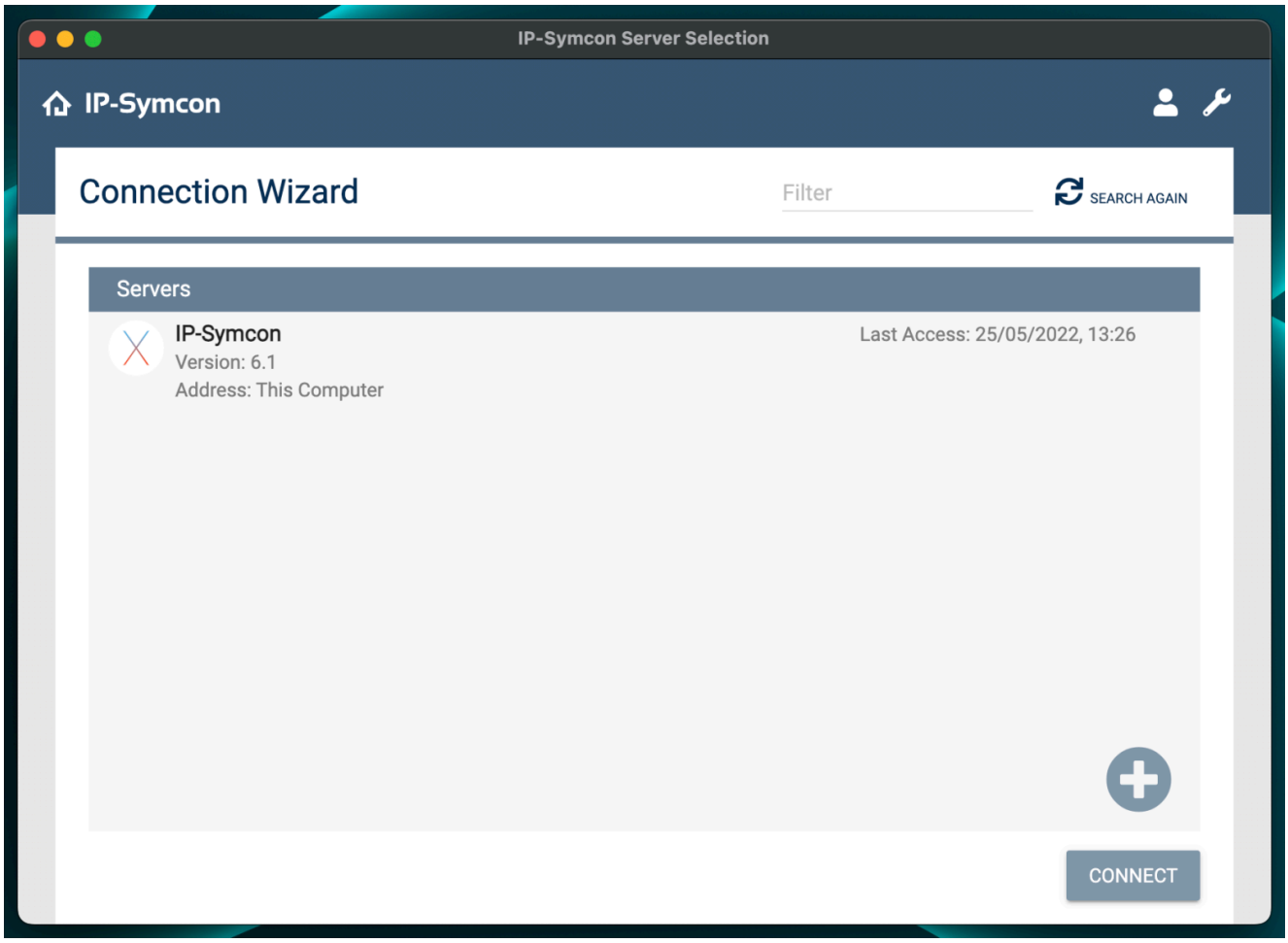
```
wget -qO- https://apt.symcon.de/install.sh | bash /dev/stdin
```

Execute the following commands:

```
sudo apt-get update
sudo apt-get install symcon-console
```

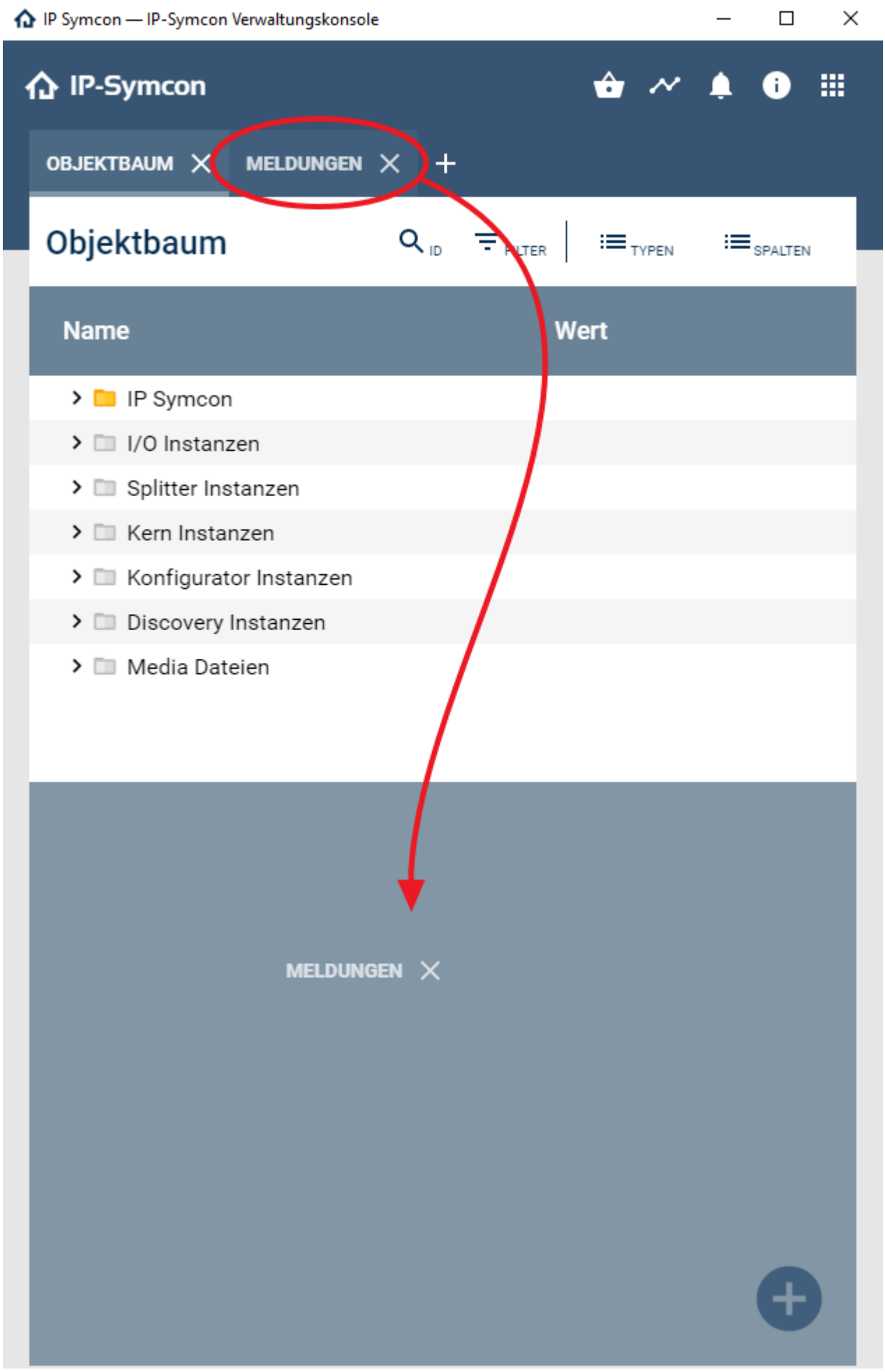
CONNECT TO THE IP-SYMCON SERVER

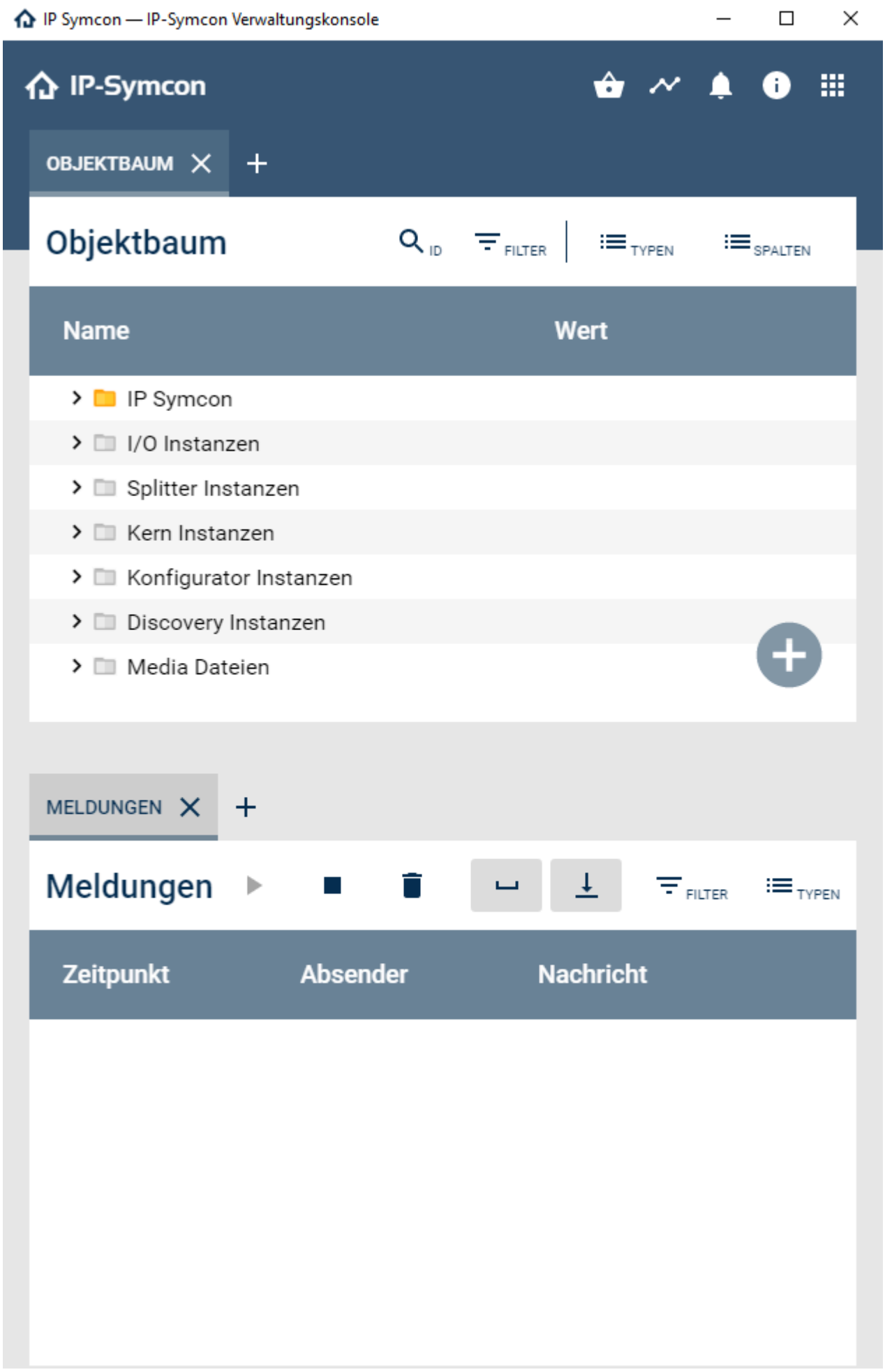
The Pro Console uses the Connection Wizard. Here, a server can be selected and connected to via "Connect" or double click.



UN- AND REDOCK

In the Pro Console, individual tabs can be un- and redocked at will. Thus, an individual work space be be designed to keep everything in sight at once. Additionally, single tabs can be shown in a seperate window by rightclicking on it and selecting "Open in New Window".


IP Symcon — IP-Symcon Verwaltungskonsole
IP-Symcon
OBJEKTBAUM
MELDUNGEN
Objektbaum
ID
FILTER
TYPEN
SPALTEN
Name
Wert
IP Symcon
I/O Instanzen
Splitter Instanzen
Kern Instanzen
Konfigurator Instanzen
Discovery Instanzen
Media Dateien
MELDUNGEN


IP Symcon — IP-Symcon Verwaltungskonsole
IP-Symcon
OBJEKTBAUM
Objektbaum
ID
FILTER
TYPEN
SPALTEN
Name
Wert
IP Symcon
I/O Instanzen
Splitter Instanzen
Kern Instanzen
Konfigurator Instanzen
Discovery Instanzen
Media Dateien
MELDUNGEN
Meldungen
FILTER
TYPEN
Zeitpunkt
Absender
Nachricht

START WITH PARAMETERS

Usually, the Pro Console directly launches the Connection Wizard. This behaviour can be adjusted via parameters.

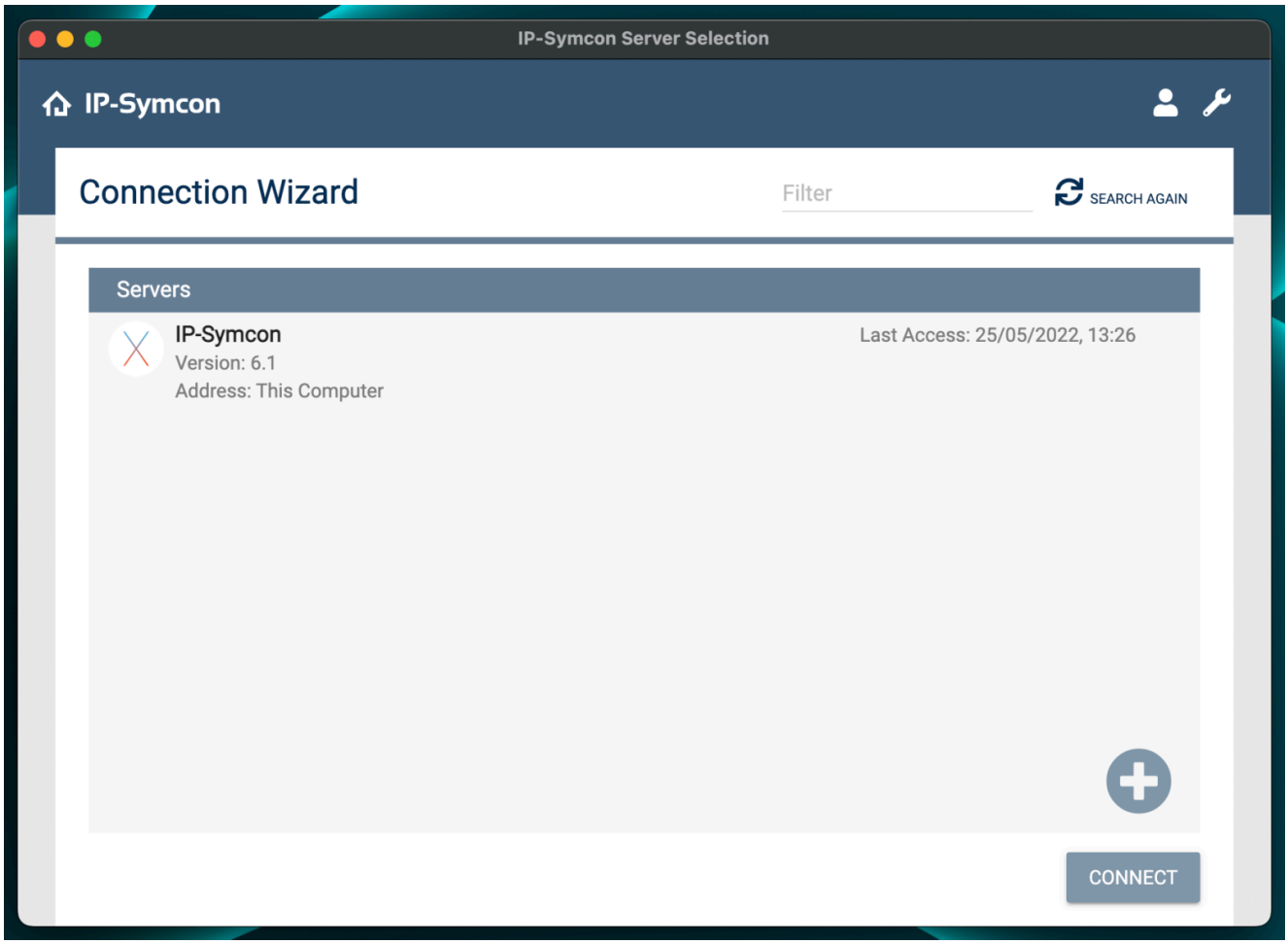
PARAMETER	DESCRIPTION	EXAMPLE FOR WINDOWS
--server=[Console]	Directly connects to a server. [Console] needs to be replaced with the address of the web based Management Console of the Server.	"IP-Symcon Management Console.exe" --server=http://127.0.0.1:3777/console/

Connection Wizard

The Connection Wizard helps to connect the Pro Console to an IP-Symcon server. The available servers are displayed in a list.

SERVERS

All servers found in the local network or entered into the License Management are listed here consecutively. To display the servers from the License Management, the corresponding Symcon account needs to be logged in. The login is done via click on the person icon top right. There, username and password from the personal area of the Symcon homepage need to be entered. Regard, that this login is seperate from the community forum and the remote access. With "Search again", it is possible to scan for new servers in the local network.



NO SERVERS FOUND

If no servers are shown in the list, there may be several reasons.
IP-Symcon Server from the License Management do not appear
It should be verified that the correct user is logged in. If a user is logged in, it says "Logged in as ..." next to the person icon. If that label is not visible, no user is logged in and thus no licenses from the License Management can be shown.
IP-Symcon Server is not powered on/started
Please start IP-Symcon on the server. Verify that the server is connected and started

correctly.

IP-Symcon server is not properly connected to the network

It should be checked if a network connection between both computers is available.

Firewall settings incorrect

Both client and server side access must be allowed in the firewall for Port 1900 / UDP.
For the connection with the management console additionally, under normal circumstances, port 82/TCP or 3777/TCP must be allowed.

Tile Visualization

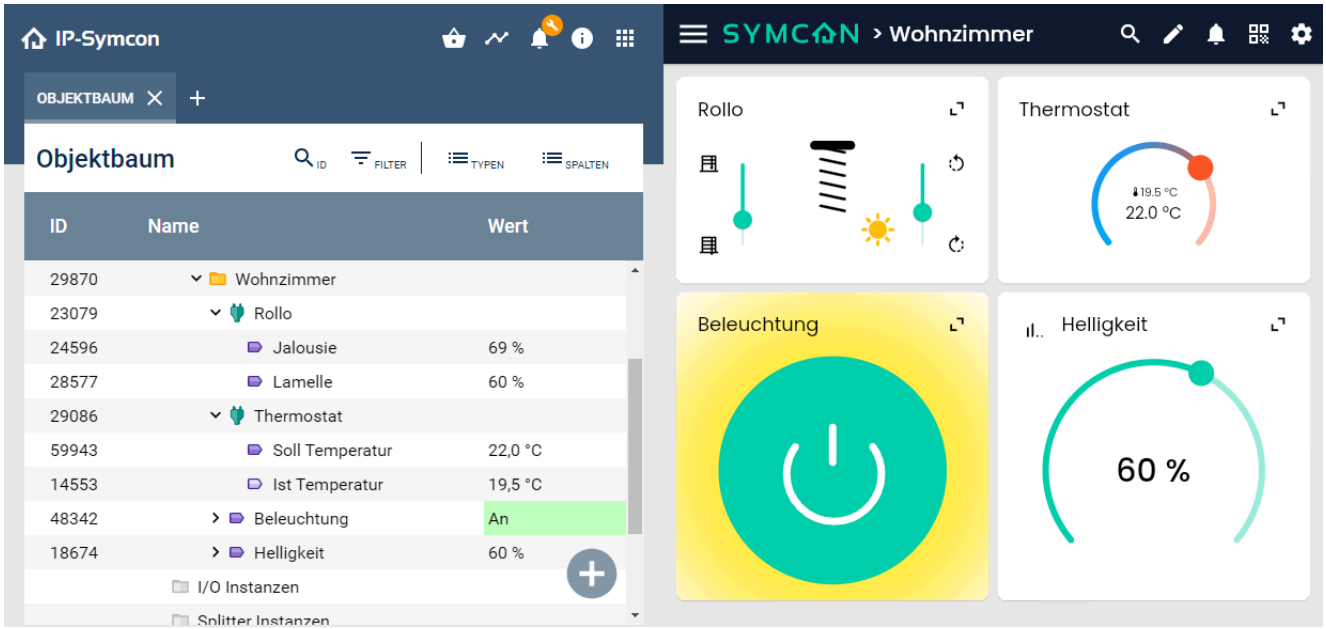
Require: Symcon >= 7.0
The Tile Visualization provides a representation of your system that visualizes the entire Object Tree or just a part of it. The visualization can be used via browser or via apps.

CONNECTING WITH THE VISUALIZATION

Connecting is one of the few points where apps and browsers differ:
Connecting with Apps
Connecting with Browser

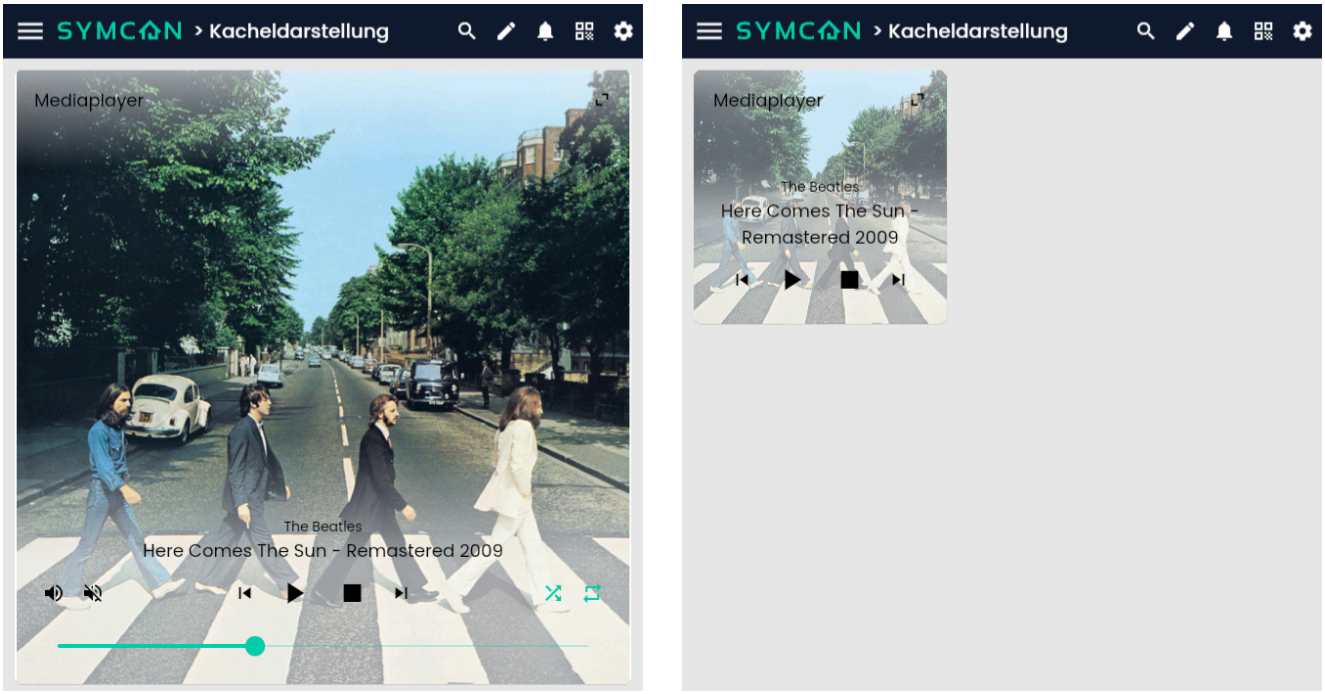
MAIN VIEW

The visualization represents objects from IP-Symcon as tiles and is oriented to the Object Tree. One page of the visualization thus corresponds to the content of one category. How the individual object types are displayed is described in the object-presentation



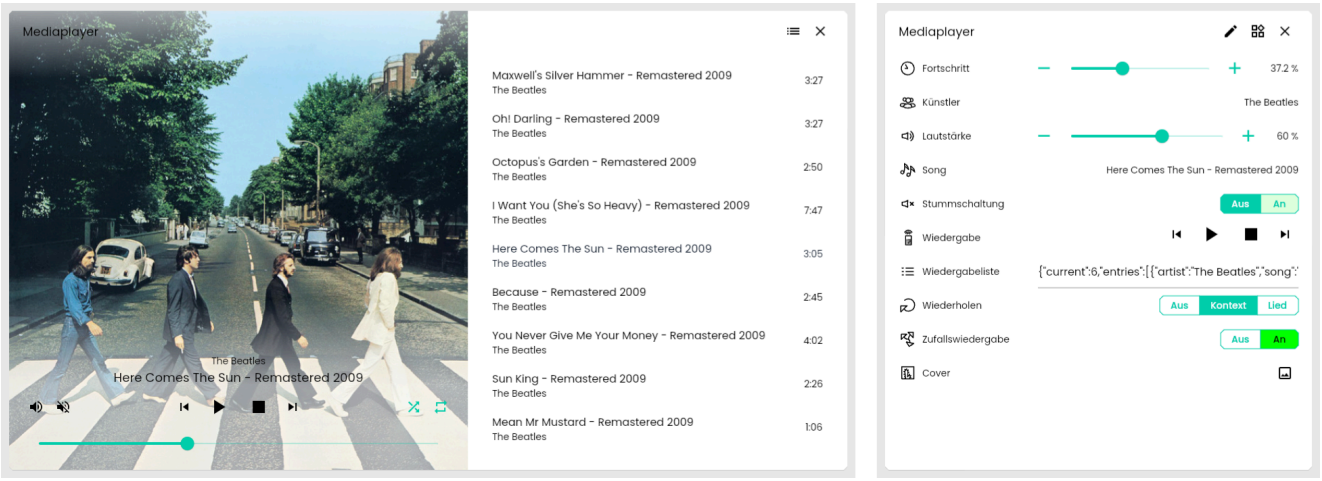
TILES

Tiles have different display modes.
Minimized view
When tiles are displayed under a category alongside others, they use the minimized view. These are described in the object-presentation described in more detail. The size of the tile can also have an influence on the content of the tile.



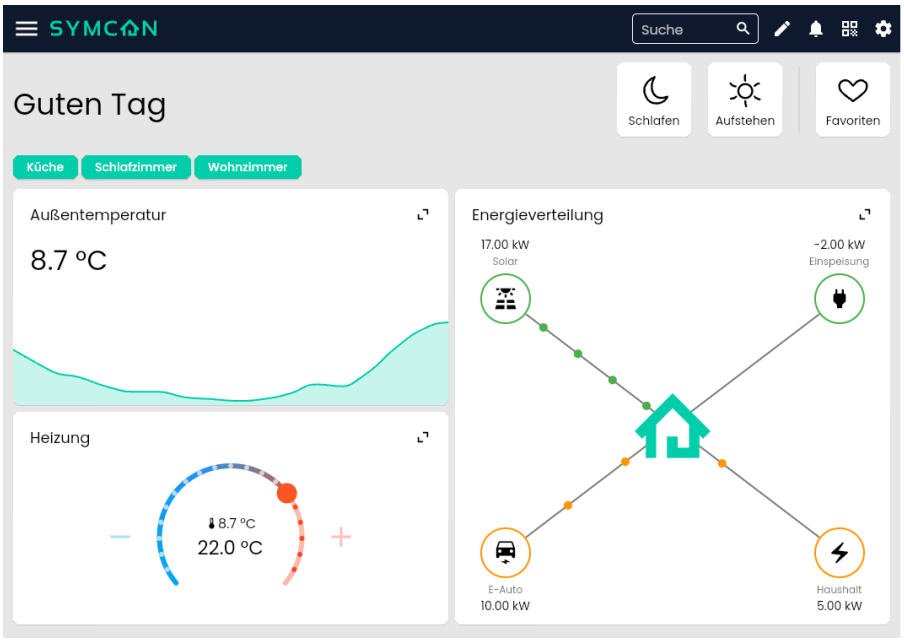
Maximized view

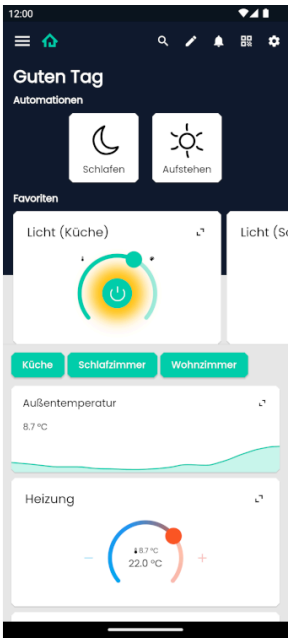
Each tile can be maximized using the two arrows in the upper right corner. Depending on the object type and configuration, a maximized view is available. If there is none, the object itself or all child objects are displayed in a list. Even if there is an extended display, you can switch to the list display at any time by clicking the list icon in the upper right corner.



ELEMENTS OF THE VISUALIZATION

The visualization is divided into different areas:





Appbar

At the top of the screen is the appbar. On the left side, an overview of the categories can be accessed via the navigation menu. On the right side is the search by default and buttons for edit mode, notifications, sharing the visualization and settings. In the instance configuration these options can be deactivated or even more presentations like the current date can be added.

Start Area

Below the appbar is the start area. The content can be changed in the instance configuration can be customized. A welcome message is displayed here by default. Additionally, a number of automations and favorites can be displayed here. For narrower screens such as smartphones, the display is slightly adjusted to make better use of the available space.

Category Bar

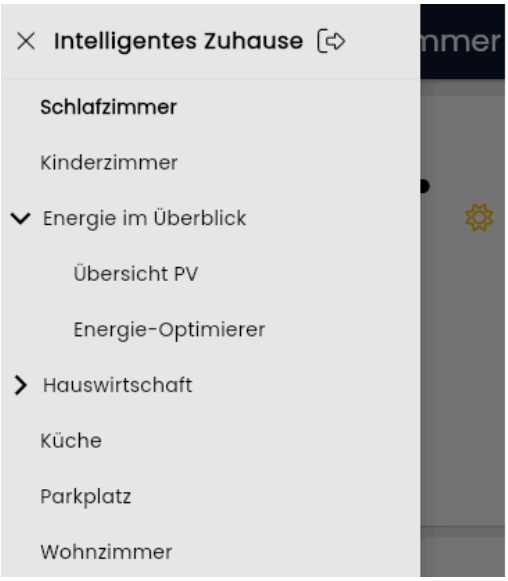
The category bar summarizes all categories that are in the current category. If needed, the bar wraps around to show all all of them. In the instance configuration, the category bar can be hidden, allowing categories to be displayed with the other objects below them in the object area.

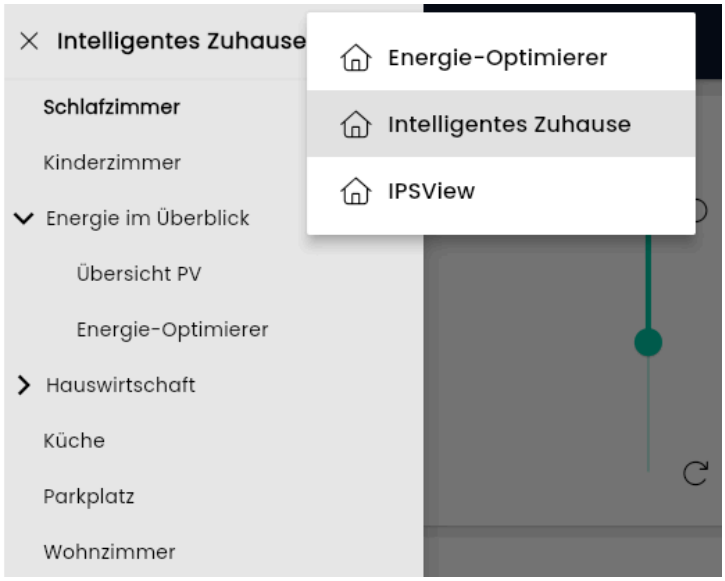
Object Area

Here all objects from IP-Symcon like variables and instances are displayed as tiles. The position and size of these can be changed at any time. Likewise the content of the individual tiles can be adapted. This information is stored in grid configurations for different screen sizes. More details can be found in section Individualization.

Sidebar

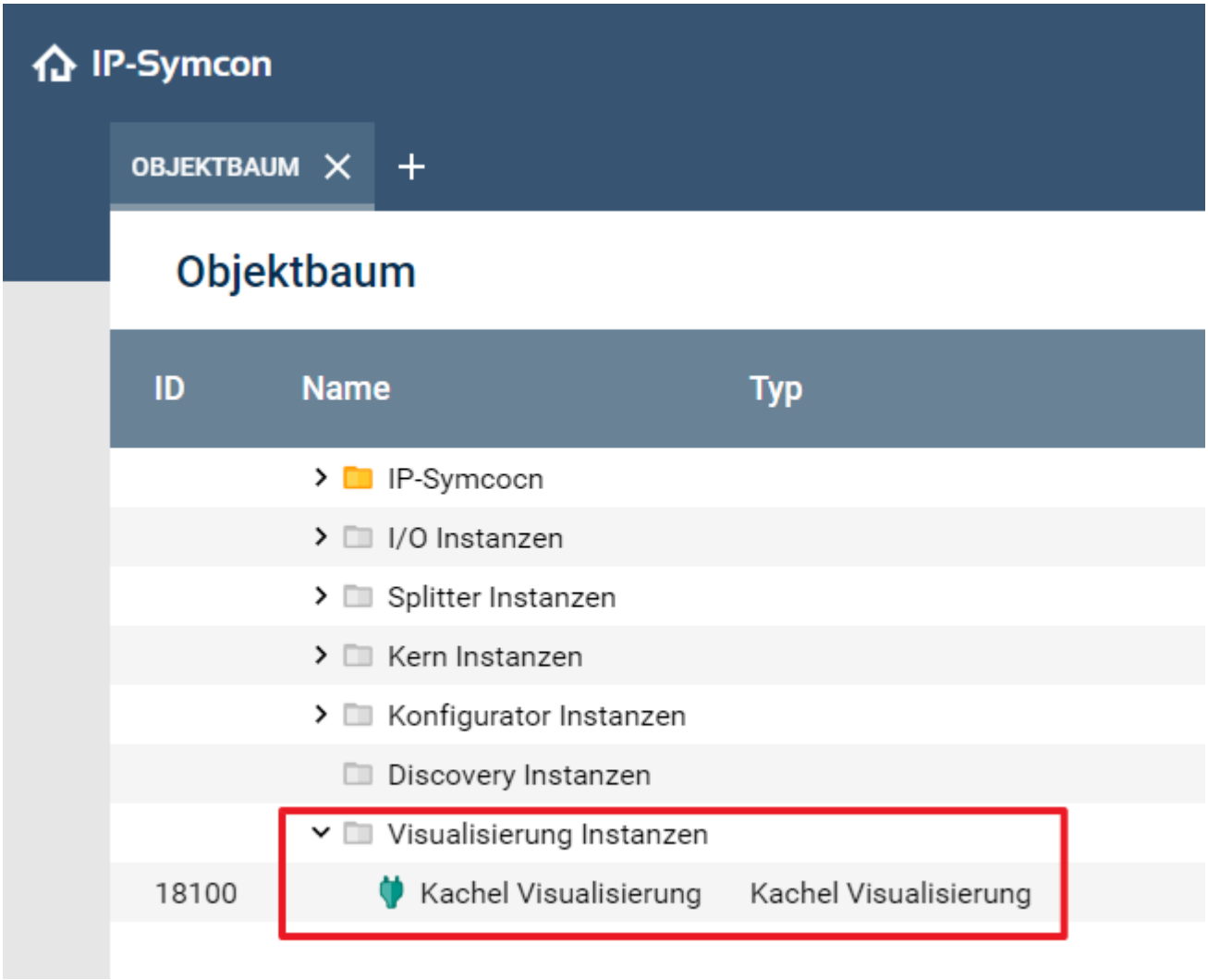
The sidebar can be opened via the icon in the top left corner. All categories are displayed here in a tree view and can be used for navigation. Next to the name of the visualization, a button can be used to switch between the available visualizations of the server.





SETUP

To set up the tile visualization, a tile visualization instance must exist. By default, an instance is created with the installation and is located in the object tree under "Visualization Instances".



If further tile visualizations are to be set up, this can be added in the object tree via the "+" -> "Instance" -> "Tile Visualization". This creates a new tile visualization, which can be

configured completely independently.

CONFIGURATION

Basically, the tile visualization runs without any further configuration. However, various settings can be made. Information about the individual settings and configuration options can be found at instance configuration. The configuration can be opened by double-clicking on the visualization tile in the object tree.

MULTIPLE VISUALIZATIONS

In IP-Symcon Basic one visualization is available. In IP-Symcon Professional Edition, a total of up to five tile visualizations can be created, which can be accessed independently via the visualization start page and secured with a password if required. A visualization can be created as a stand-alone entity - for example, for a visualization object -, represent a special view for the maintenance technician, or simply create a security-specific separation. The same options are available per visualization and can be customized as desired. In the Unlimited Edition, any number of visualizations can be created.

Individualization

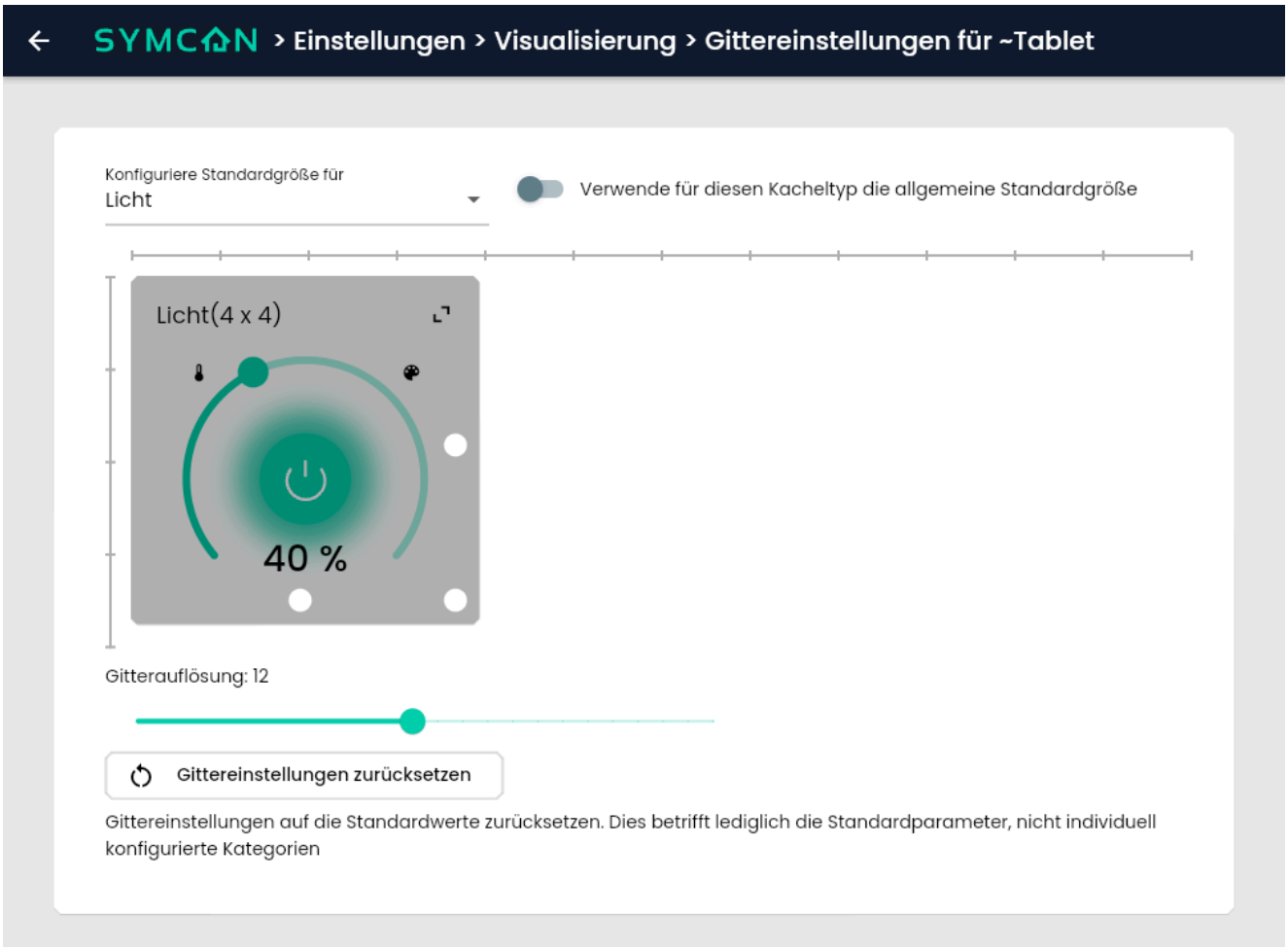
Edit layout	The size and position of tiles can be customized
Edit Tiles	The exact presentation of a tile can be customized
Designs	The basic appearance of the visualization can be changed via designs
Grid Configuration	Using different grid configurations, the same visualization can be optimized for different sized screens

Edit layout

Require: Symcon >= 7.0

DEFAULT VALUES

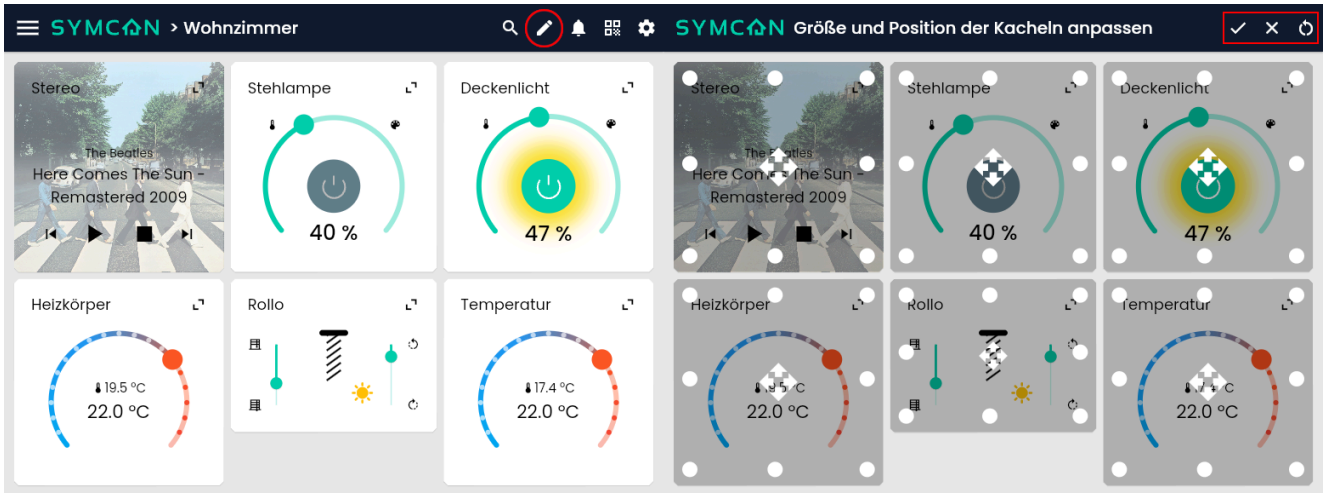
The position of the tiles results from the position of the corresponding objects in the Object Tree. The positioning runs from top left to right and from top to bottom.
All tiles have an assigned default size. The initial default sizes result from the selected grid-configuration. This is selected based on the device type of screen size and orientation to choose the best possible initial values for the tiles. These default sizes can be adjusted for each display type of a tile in the settings under Visualization>Grid Settings. The resolution of the grid on which the tiles are arranged can also be determined here.
The width of the grid is fixed. Since it is possible to scroll vertically, the height of the grid is not limited.



EDIT MODE

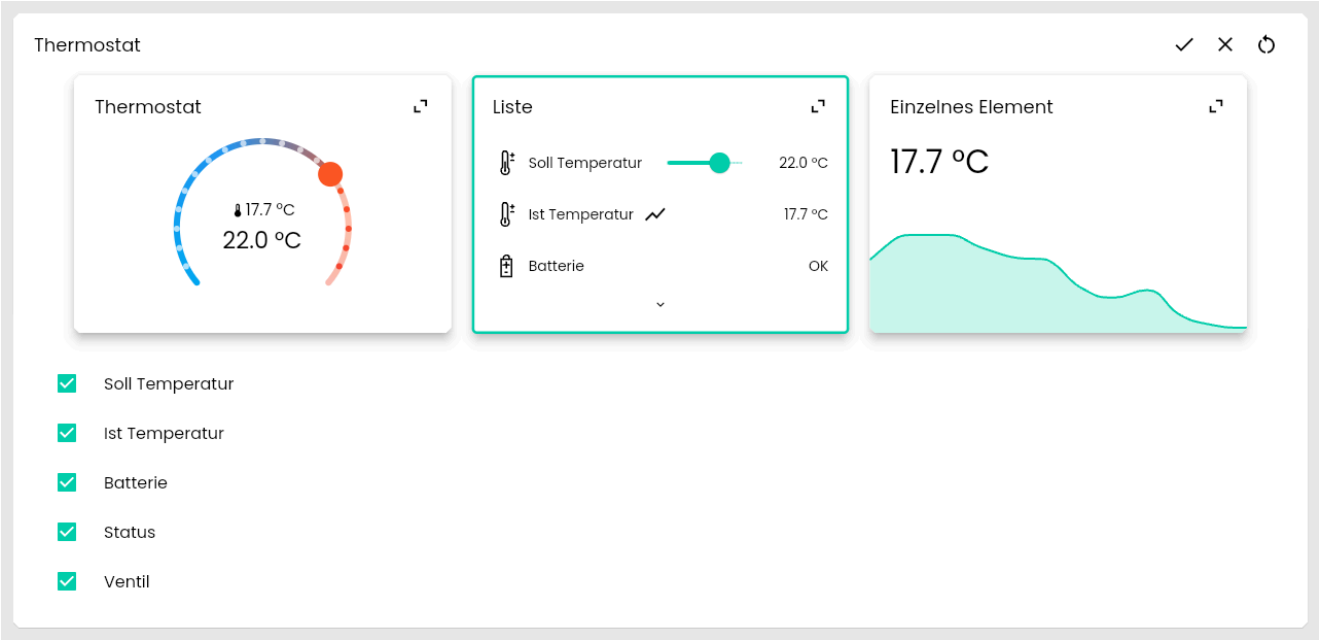
On each page of the visualization, the Edit Mode can be activated via the pencil icon in the appbar. The position of the tiles can be changed by Drag & Drop. The size can be changed by dragging the points on the edges. When the edit mode is activated, three new options are offered in the appbar. The current changes can be saved or discarded. In addition, the layout can be reset to the default sizes. The reset only applies to the currently edited category.

Once a layout has been saved, it can no longer be changed by position changes in the console. A hint to this effect appears the first time you adjust the layout. If there is empty space between the tiles, it can be removed via the recycle bin.



Edit Tiles

Require: Symcon >= 7.0
The presentation of the individual objects results in principle from the object type and with variables approximately still from the variable profile. For more details, refer to the section Object-Presentation. How instances are represented is determined by the subordinate variables. IP-Symcon automatically selects the appropriate element. To configure the appearance of a tile you have to open it and switch to the edit mode with the pencil.

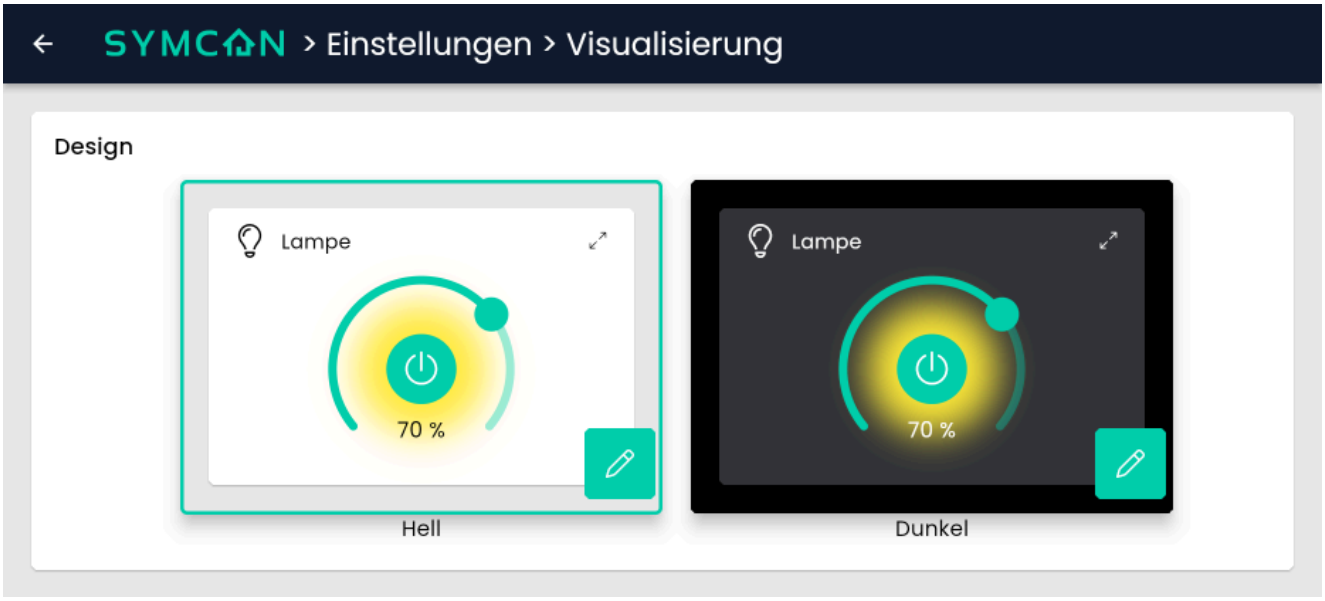


Designs

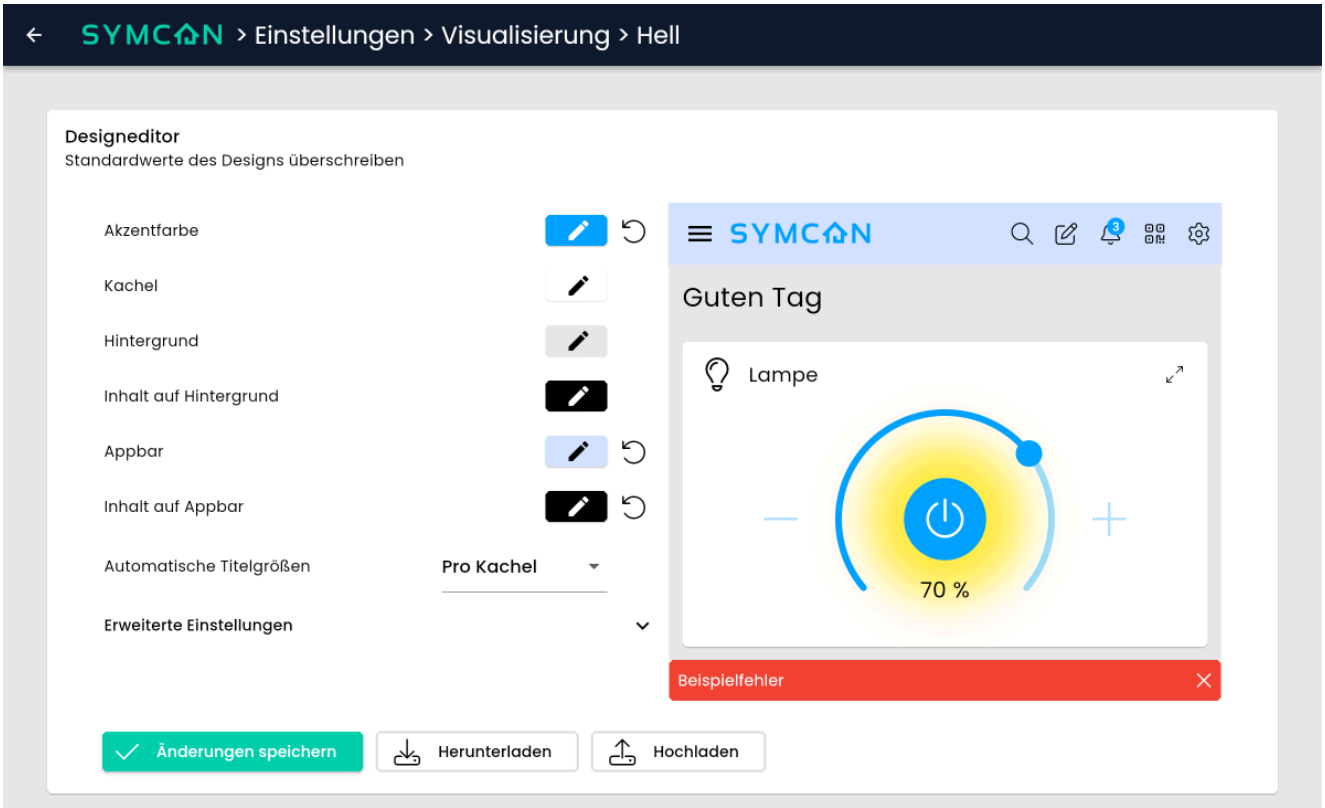
Require: Symcon >= 7.0
You can switch between different designs in the settings under Visualization using the cogwheel on the right in the appbar.



Background and text color as well as font of the HTMLBox are influenced by the design.



EDIT DESIGNS



As of version 8.1, the designs can be edited using the pencil in the corner of the tiles. Individual colors can be changed as desired and saved locally via "Save changes". The settings can be downloaded as a file and uploaded to another visualization using the corresponding buttons. Settings can be reset at any time using the arrow symbol.

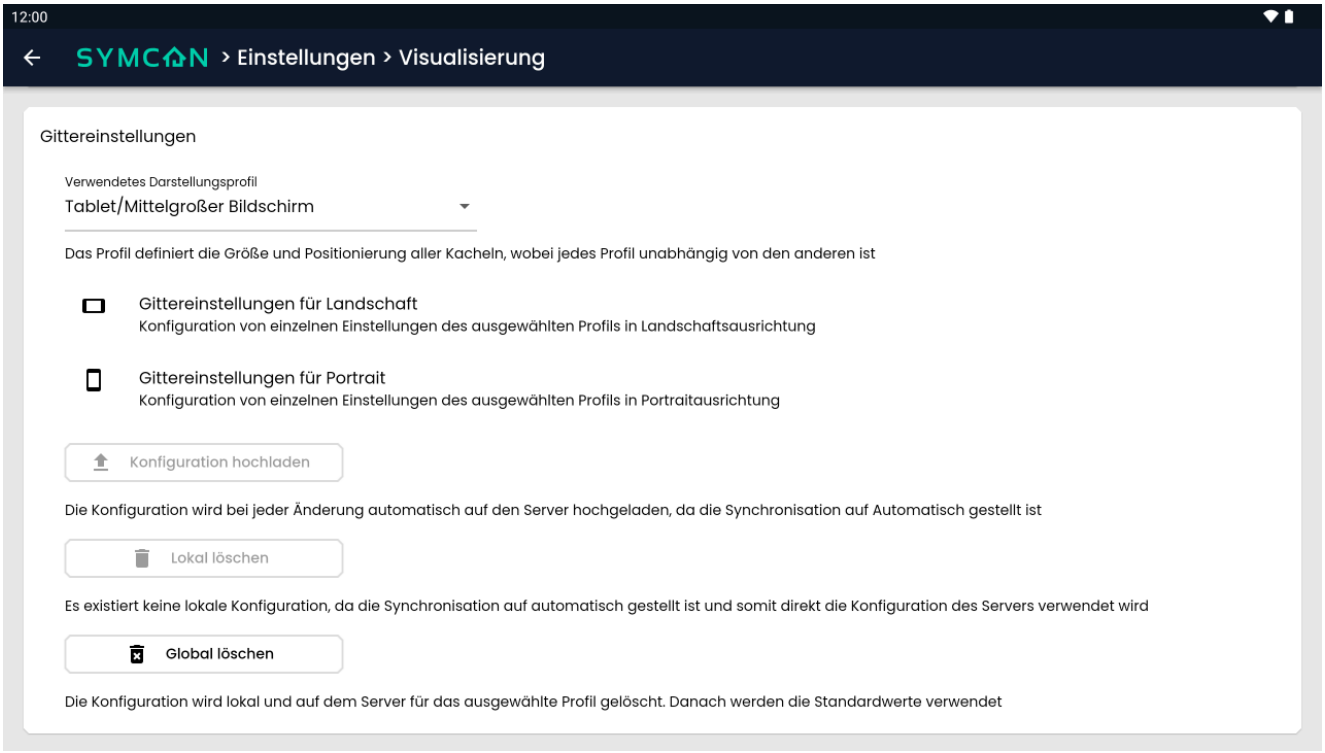
PARAMETER

COLOR	DESCRIPTION
Accent Color	The color of interactive elements (sliders, buttons)
Tile	The color of the tiles
Background	The color of the background on which the tiles are displayed
Background Text	The color of text and icons displayed on the background
Appbar	The color of the bar at the top of the visualization
Appbar Text	The color of text and icons on the appbar
Automatic title size	The size of the titles of the individual tiles (Per Tile: Each tile adjusts its title size optimally. Per Category: Within a category, all title sizes are uniform and based on the smallest title. Fixed Size: All titles have a fixed size)
Fixed title size	Visible with automatic title size “Fixed size” (Big, Medium, Small)
Accent Text	The color of text and icons on controls in the accent color
Tile Text	The color of icons and text on standard tiles
Dark Tile Text	The color of text and icons on tiles that are darker than the default due to dynamic background colors
Light Tile Text	The color of text and icons on tiles that are lighter than the default due to dynamic background colors

COLOR	DESCRIPTION
Error Color	The background color of error banners and faulty tiles
Error Text	The color of texts and icons on error banners
Corner Radius	The size of the corners of the tiles (from 0 - pointed to 10 - very flattened)

Grid Configuration

Require: Symcon >= 7.0
Grid configurations are used to display the visualization uniformly on different devices. The configurations store position and size of the tiles as well as their configuration. In the settings the tile resolution and the default sizes per grid configuration can be changed.



There are three grid configurations, each with default sizes optimized for one type of device: Computer, Tablet and Mobile. The appropriate grid configuration is selected based on the operating system and screen size. If necessary, the grid configuration can be selected manually.

ORIENTATION

All grid configurations store the settings in combination with the orientation. Thus, configurations for portrait and landscape orientation can be made for each grid configuration. In the app, the appropriate grid configuration for the current orientation is selected. In the browser, only the Landscape orientation is available.

GRID SETTINGS

The default settings for the currently selected grid configuration can be adjusted in the grid settings. In the app, both the Portrait and Landscape settings are available; in the browser, only the Landscape orientation is available.



In the dropdown at the top, you can select what the standard size should be configured for. All Object Presentations and "All Tiles" are available. When selecting "All tiles", the size of the displayed tile can be adjusted in the same way as Edit layout, the general default size for all presentations can be configured. If an object presentation is selected, an individual default size can be configured for this. The "Use the default size for this tile type" switch deactivates the individual default size and therefore uses the general default size for this object display. This results in the following size for each tile in the visualization:

If this has been set via Edit layout an individual size has been defined for the tile, this will be used

If "Use the default size for this tile type" is deactivated, the individual default size is used for this object display

If "Use the default size for this tile type" is activated, the general standard size configured via "All Tiles" is used for this object display.

The "Grid Resolution" slider can be used to define the resolution of the grid on which the tiles are placed. A higher resolution therefore enables a more detailed adjustment of width and height.

Clicking the "Reset Grid Settings" button resets the default sizes and grid resolution to the default values for the current grid configuration and orientation.

SYNCHRONIZATION

The configuration can be synchronized on all devices via the Symcon server. This way, each end device loads the same grid configuration. The synchronization can also be switched to

manual. The customized grid configuration is then synchronized via a button in the settings. If no changes are to be made to the end device, a read-only mode can also be activated. The settings for the different synchronization modes can be found in the Instance configuration.

BUTTONS FOR SYNCHRONIZATION

BUTTON	DESCRIPTION
Upload configuration	Uploads the local configuration to the server
Delete locally	Deletes the local configuration
Delete Globally	Deletes the local configuration and the server configuration

Instance Configuration

Require: Symcon >= 7.0

START CATEGORY

The Selected category defines from where the visualization is built.

SHOW WELCOME MESSAGE

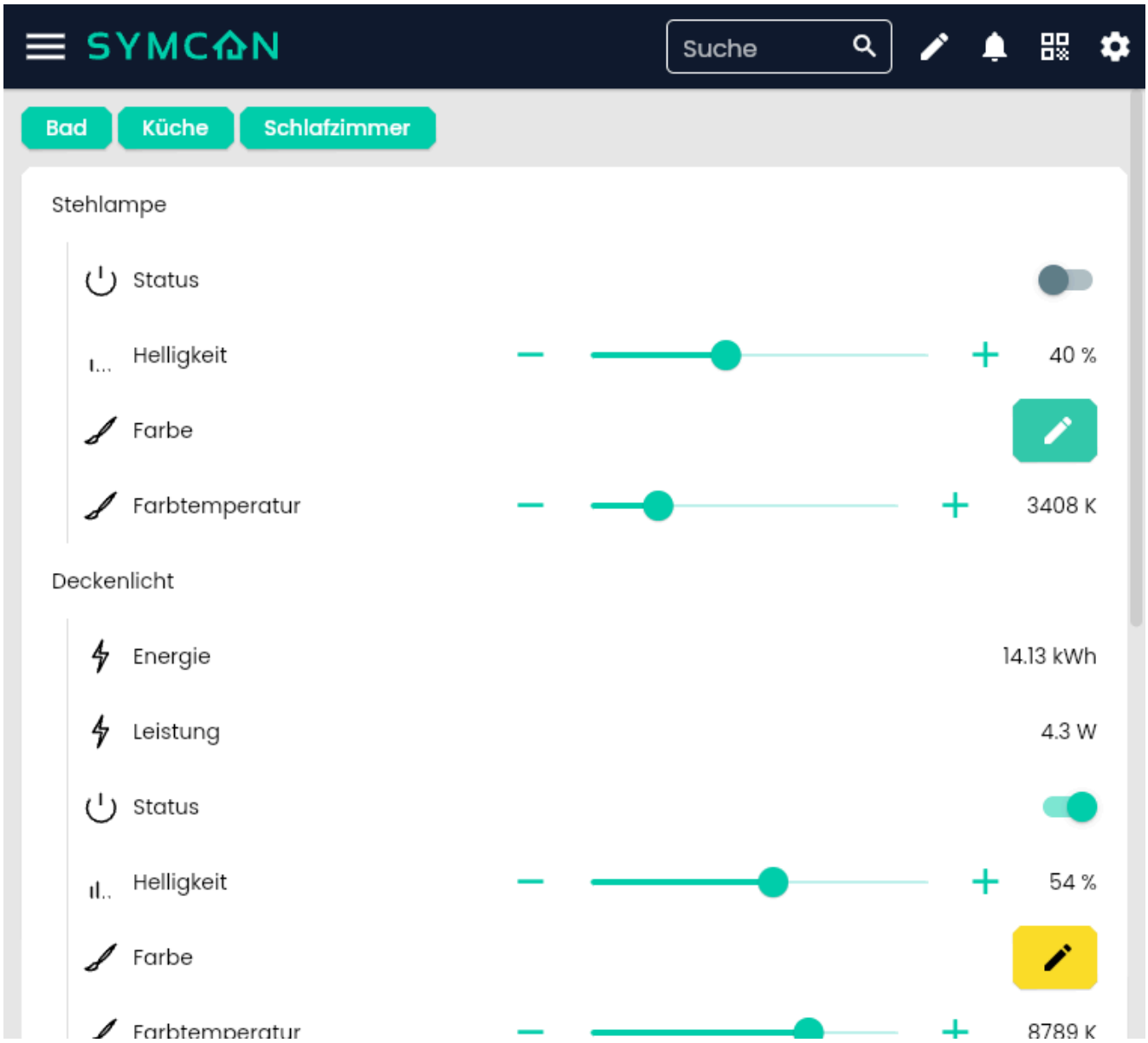
The greeting is displayed in the start area of the visualization and can be configured via various parameters.

SHOW GREETING	DESCRIPTION
Automatic	The text is adjusted based on the time of day and can optionally be extended with a name.
None	The greeting will be hidden. If no favorites or automations are available, the entire start area is hidden.
Variable	The greeting text is set to the value of a variable. The type is not relevant here.

DISPLAY MODE OF THE VISUALIZATION

DISPLAY MODE	DESCRIPTION
Tiles (Default)	All objects are displayed as tiles. Where possible, Summarized_Presentations are shown.
Tiles (List)	All objects are displayed as tiles. Initially, all instances are displayed as list.
List (Legacy)	All objects are displayed in a large list without tiles.

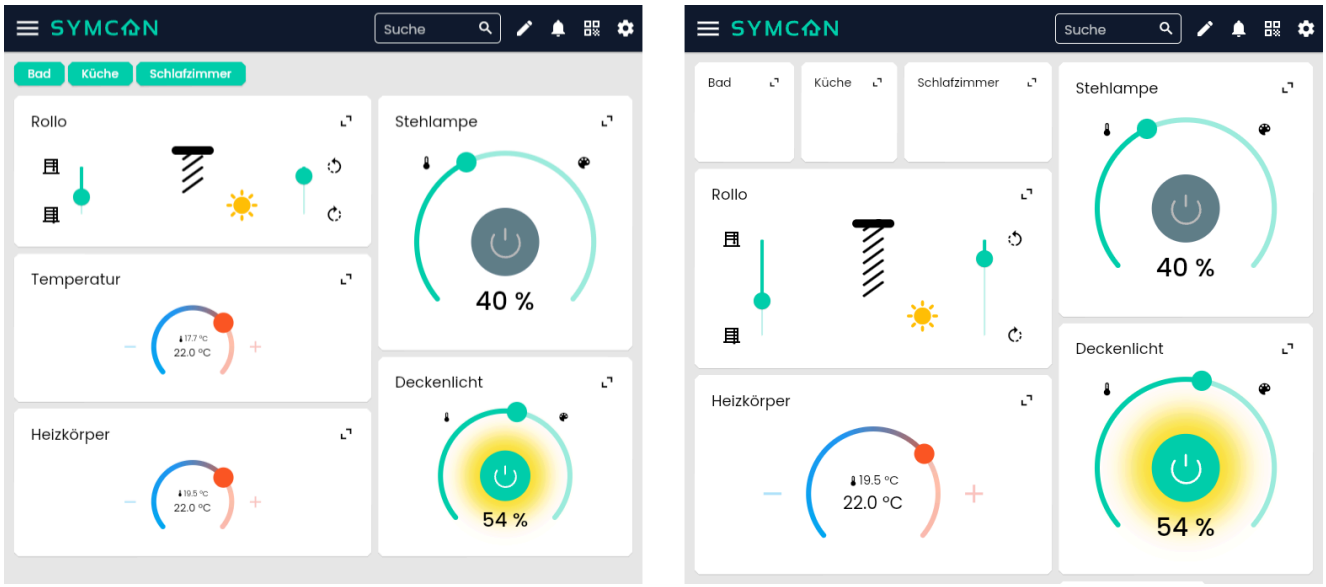
Translated with DeepL.com (free version)



CATEGORY DISPLAY

This setting serves as a basis and can be overwritten individually for each category.

PRESENTATION	DESCRIPTION
Category bar (single line)	All subcategories of the current category are displayed in one line at the top of the screen. If the number of categories exceeds the width of the screen, it can be scrolled horizontally.
Category bar (multi-line)	All subcategories of the current category are displayed in a line at the top of the screen. If the number of categories exceeds the width of the screen, the categories are displayed in several lines.
Tiles	Categories are displayed as tiles like all other objects. This means that the size and position can be changed freely.
Only in sidebar	The categories are only displayed in the sidebar



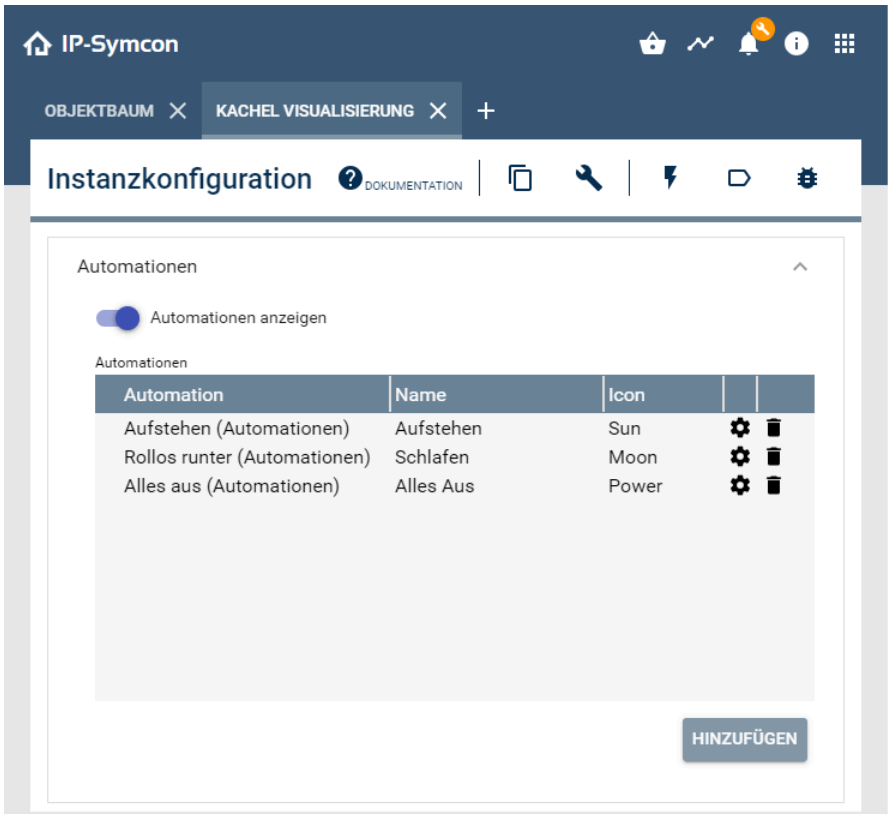
CONFIGURATION SYNCHRONIZATION

How synchronization works is explained in the section Grid Configuration.

MODE	DESCRIPTION
Automatic	Every change to the layout is immediately synchronized with the server and therefore all devices
Manual	Local changes must be uploaded in the settings so that they can also be synchronized with other devices
Read only	The configuration can only be changed locally the devices and cannot be uploaded to the server

AUTOMATIONS

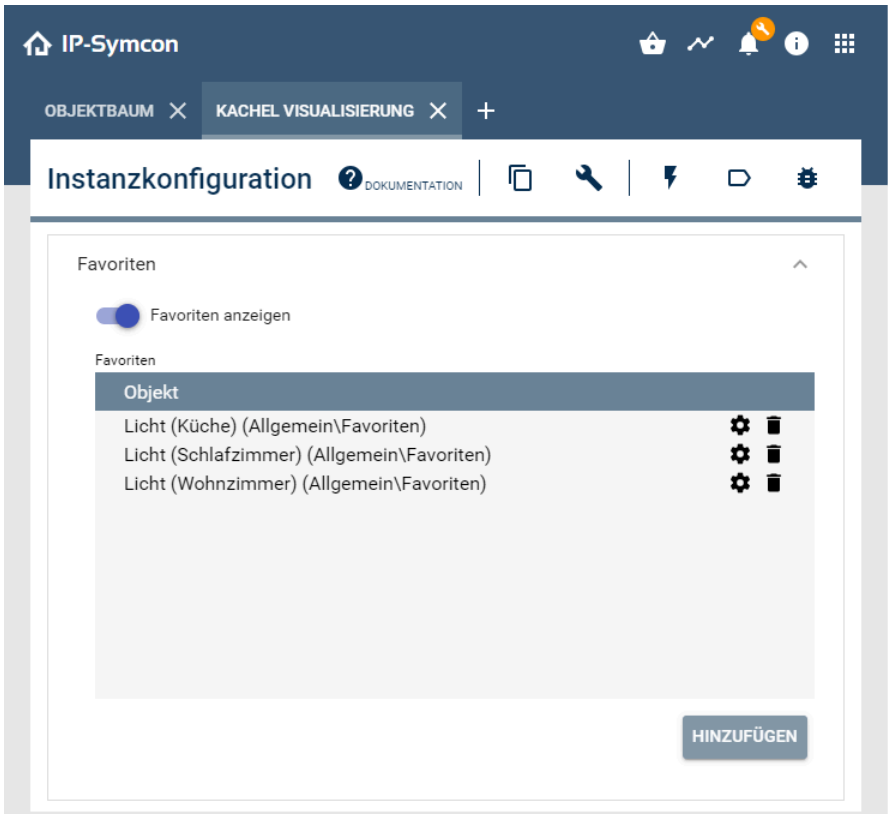
Here a list of Automations, which are then available in the start bar.
Each automation can be given its own name and icon.

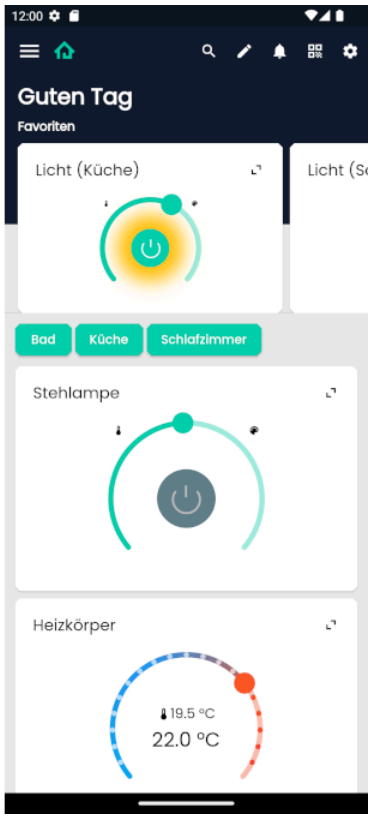




FAVORITES

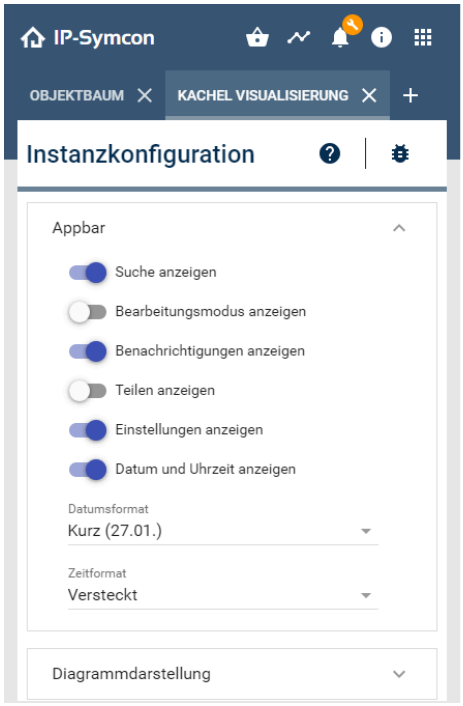
Any objects can be added as favorites. They are displayed as normal tiles in the upper part of the screen. In the version for large screens, these can be accessed via the heart button.





APPBAR

If there is not enough space for the configured elements, surplus elements are combined in a menu.



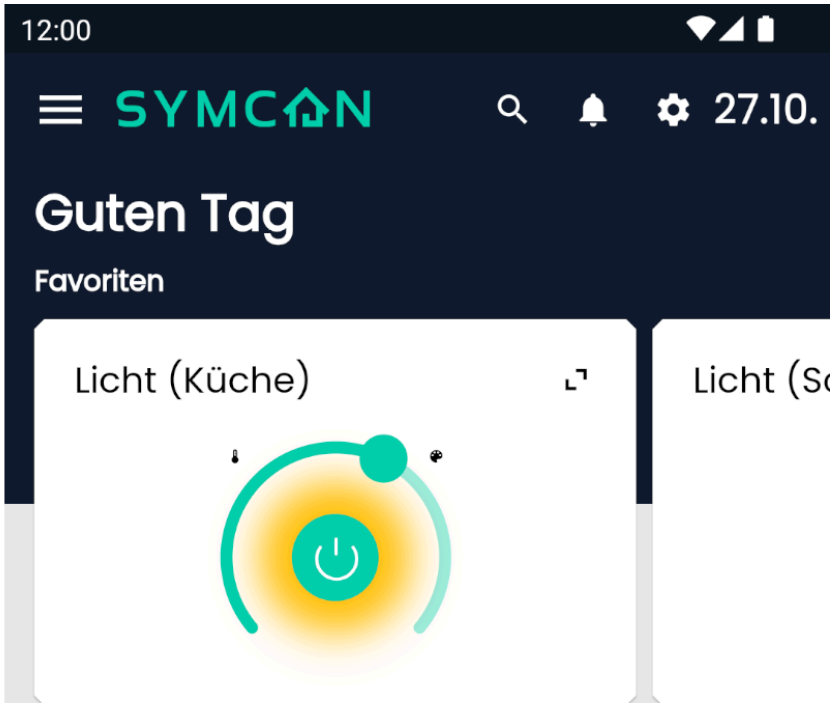
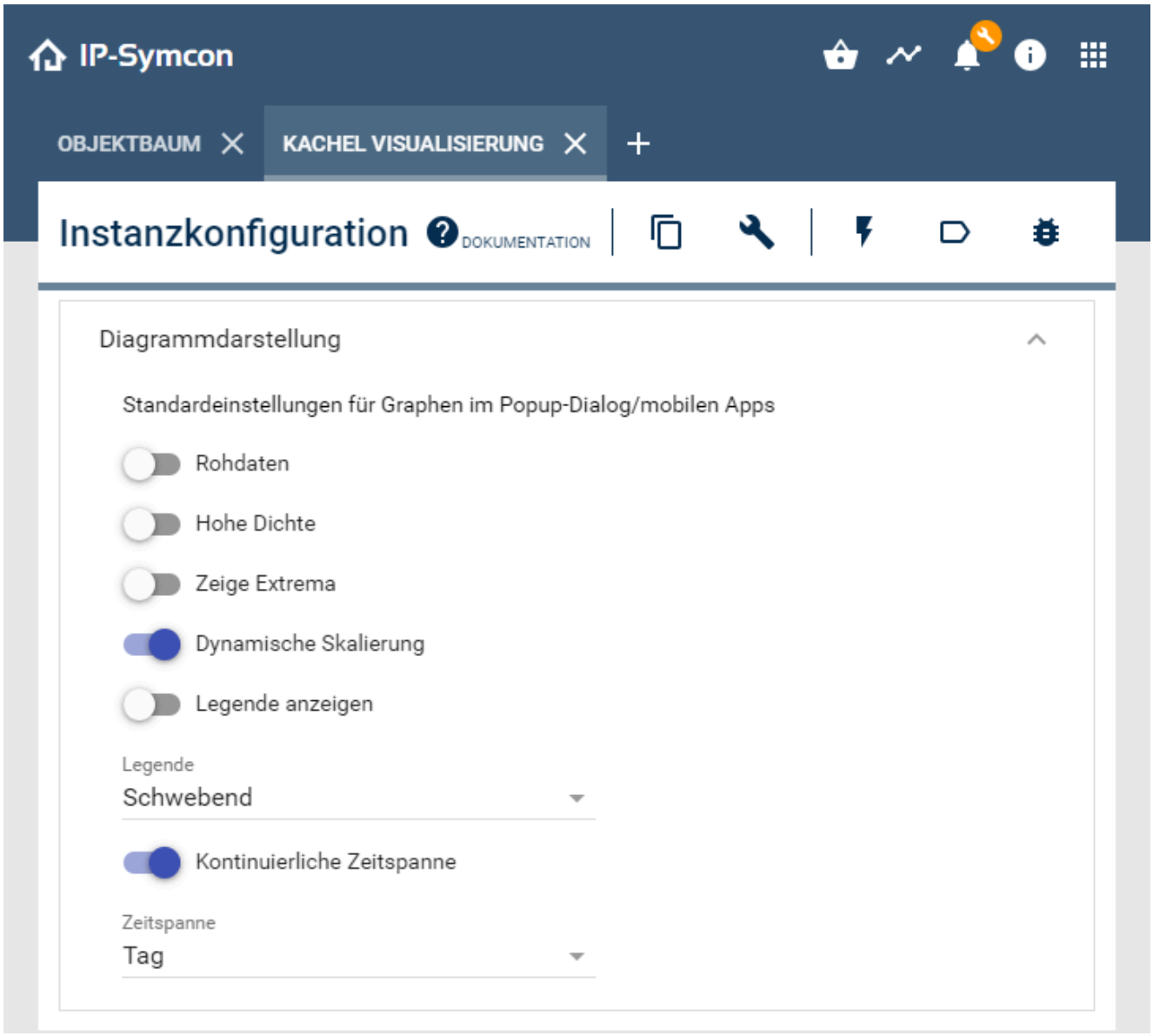


CHART DISPLAY



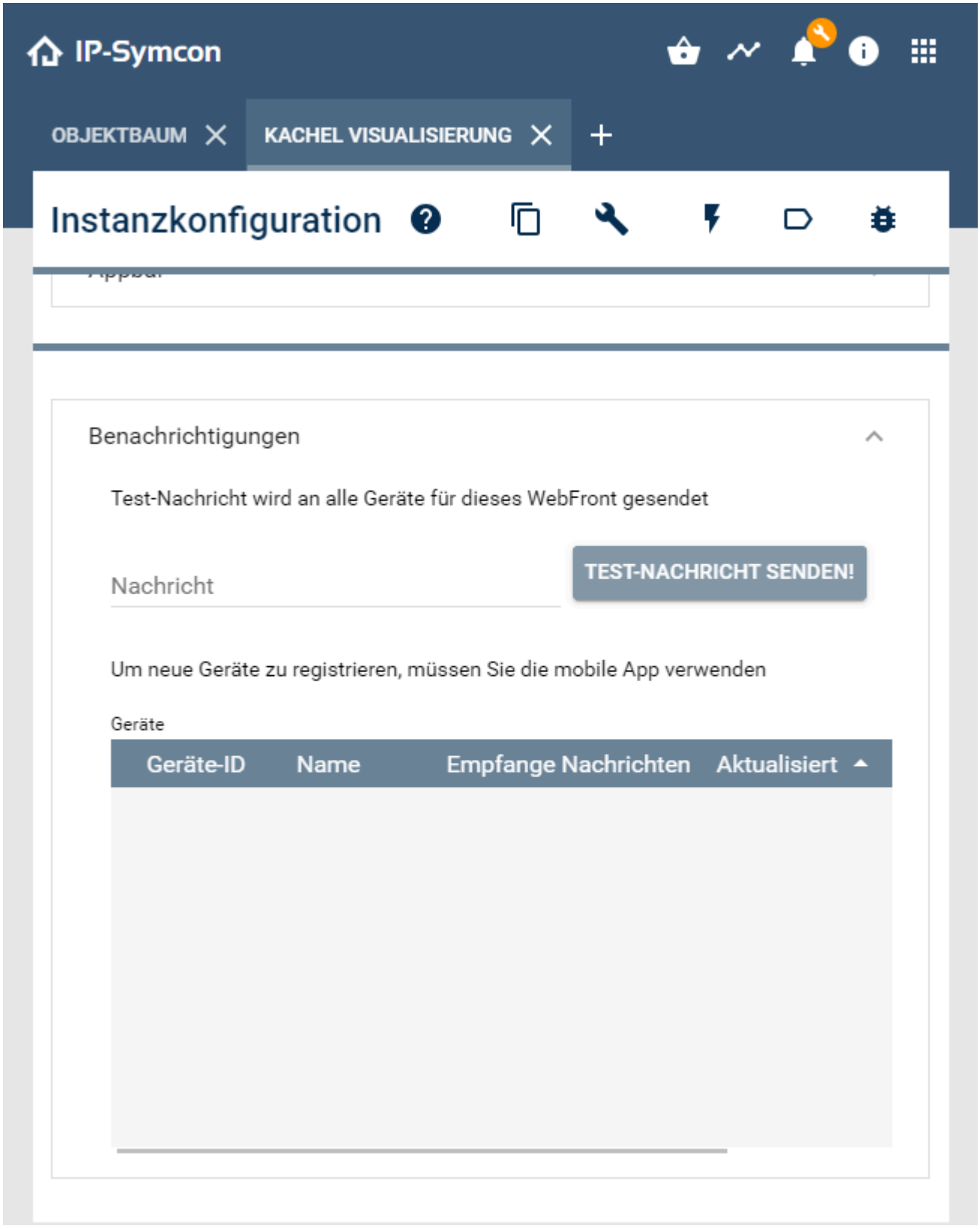
Here you can set the default settings for charts.
What exactly which term means can be found under Display types of charts.

FURTHER CONFIGURATION

PAGE	DESCRIPTION
Notifications	Allows push messages to be sent and configured devices to be managed.
Security	Allows you to set a password for access control and a special IP AutoStart function.
Virtual Hosts	Allow visualizations to be distributed across different virtual hosts

Notifications

The tab provides an overview of the registered devices. It is possible to select whether the respective device receives messages.
A test message can also be sent.



REGISTRATION TO PUSH MESSAGES SERVICE

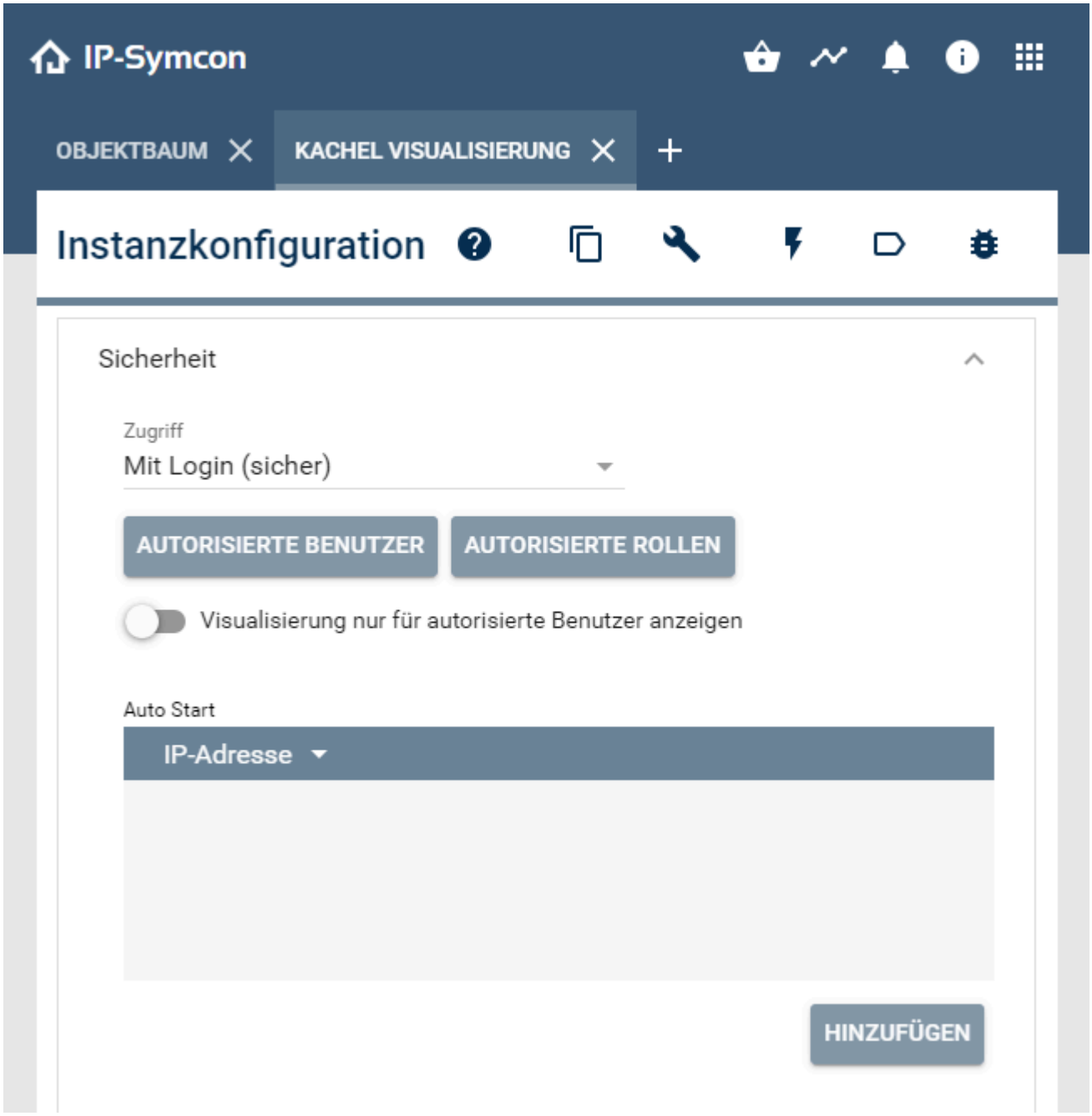
Once a mobile device successfully logs into a visualization for the first time, that device is automatically added to the push message service for that instance. If no push messages are to be sent to this device, then this device must be explicitly deactivated for reception in the list. It should be noted that the registration to the push message service remains unaffected if the password for the visualization is changed. If a device is completely removed from the list, it will be treated as an unknown device the next time it logs in and will be added to the push message service again.

NOTIFICATION CONTROL

Messages will be sent via Notification Control.
More information can be found at Notification Control .

Security

Access to a visualization can be secured in various ways. Further security information can be found at Security.



ACCESS

Each of the available access types offers different setting options.

WITH PASSWORD (SECURE)

A password can be assigned as access protection for each visualization instance. This password is required before accessing the respective visualization via browser and mobile

apps. In addition, the special option can be activated so that the password is not required within the local network. This password only applies to access to the system and its components. It is possible to query the available visualizations at any time without a password.

WITHOUT PASSWORD (INSECURE)

Without a password, external access is blocked by default. However, if the visualization should be deliberately accessible to the public, this can be deactivated with the option "Allow external access, even if no password is set".

WITH LOGIN (SECURE)



Access with login is a paid extension that can be purchased for any existing IP-Symcon license. For a suitable demo version, please contact our Support. The extension can be purchased directly in the Shop.

Access with login allows only authorized users or users with an authorized role to open the visualization. The required authorization for users and roles can be assigned via the respective buttons. To display a visualization only for authorized users, the corresponding option can be activated. Users and roles can be managed via the Authorization control.

AUTOSTART FOR IP ADDRESS

A special function is the auto-start, which opens a visualization directly without a password prompt if the IP address matches one of the IP addresses entered in this list. This function can be used, for example, to open a visualization directly on a touch panel without the normal selection dialog being displayed and without having to answer the password prompt. Please note that this function should only be used within a secure network.



The CIDR notation is also supported, so that e.g. 192.168.1.0/24 contains all addresses from 192.168.1.1 to 192.168.1.255.



This function is not available if requests are sent to IP-Symcon via a reverse proxy or via the Connect service, as the origin cannot then be verified safely and reliably. An exception is the CIDR value 0.0.0.0/0, which redirects all requests regardless of their origin and is therefore fundamentally insecure. This is also evaluated for the above exceptions and therefore forwards all requests regardless of their origin.

CALL VIA URL

There are situations (e.g. in connection with NAT) where the autostart via IP does not work. In these cases, a URL can be called directly.

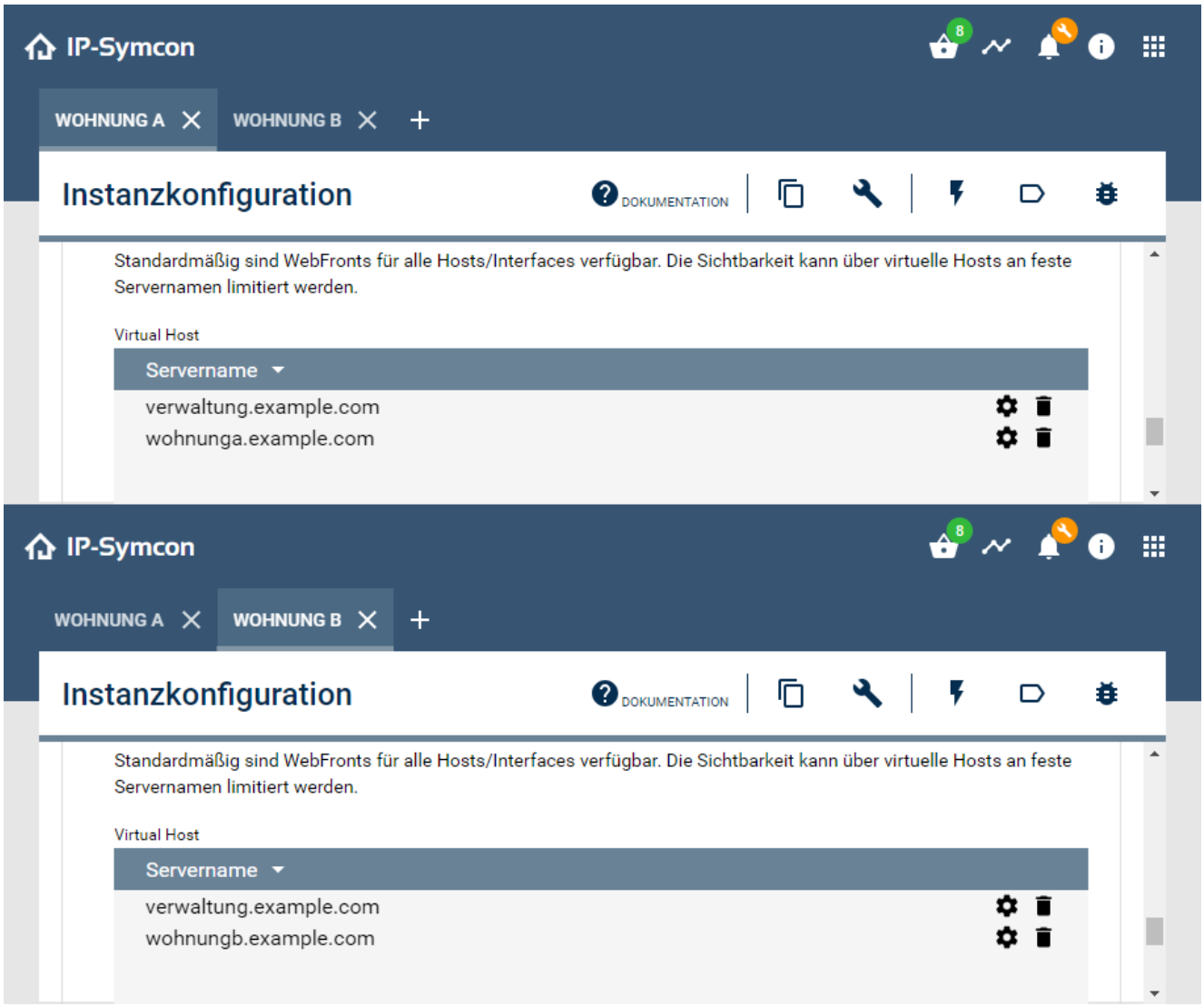
```
// Basic structure for the URL call
http://<SymconIP>:3777/#<visualizationID>

// Example as start with the Chromium Browser
// VisualizationInstanceID: = 12345
chromium-browser --start-fullscreen 192.168.1.123:3777/#12345
```

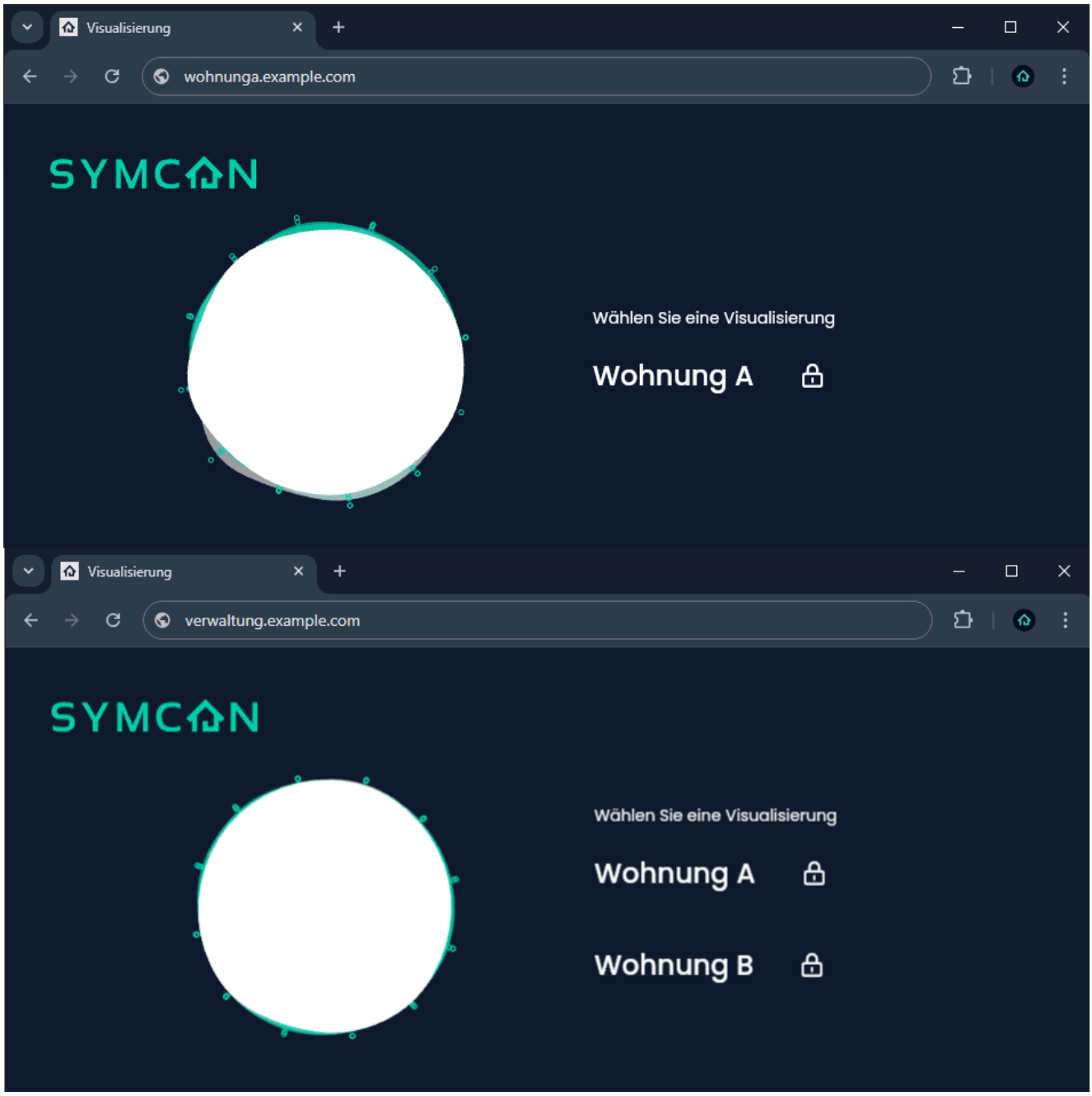
Virtual Hosts

Virtual Hosts is a paid extension that can be purchased for any existing Symcon license. For a suitable demo version, please contact our Support. The extension can be purchased directly in the Shop.

Virtual hosts can be used to define a list of hosts that can access the visualization. If the visualization is called from a host that is not in the list, the visualization is not displayed. If the list is empty, the visualization is visible for all hosts.



With the above configuration of two visualizations, the start pages are displayed as follows.


Visualisierung
wohnunga.example.com
SYMCON
Wählen Sie eine Visualisierung
Wohnung A
Visualisierung
verwaltung.example.com
SYMCON
Wählen Sie eine Visualisierung
Wohnung A
Wohnung B

Apps

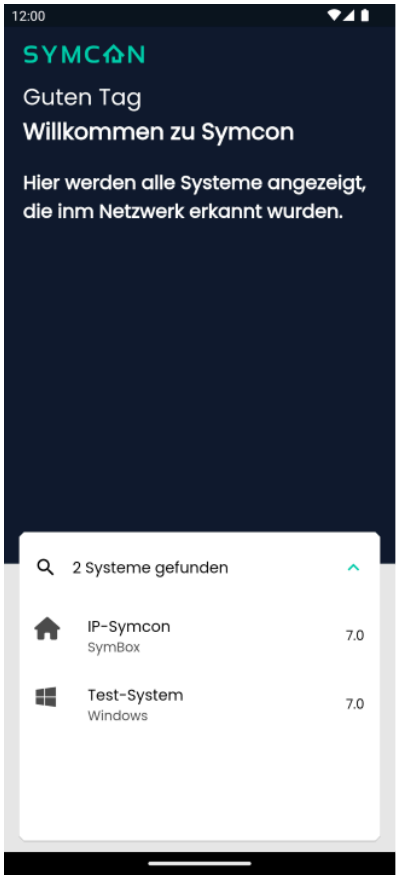
Require: Symcon >= 7.0
The apps can be downloaded from the respective appstore (iOS, Android). The mobile apps do not differ in functionality from the visualization in the browser. Only the setup process is different.

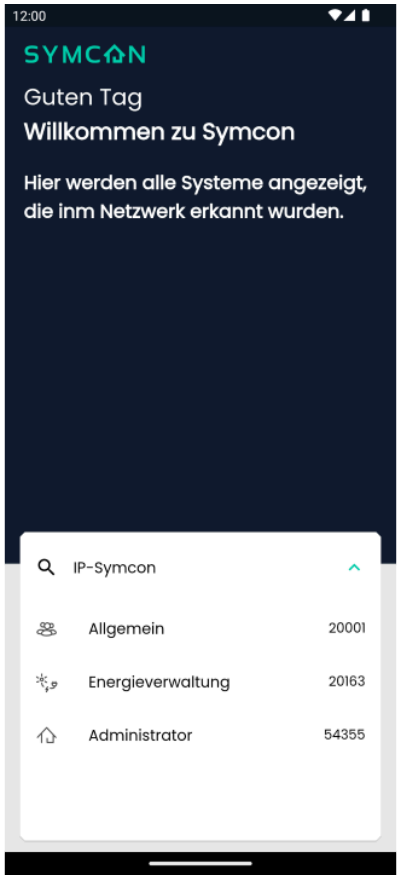
CONNECTING TO THE TILE VISUALIZATION VIA THE APPS

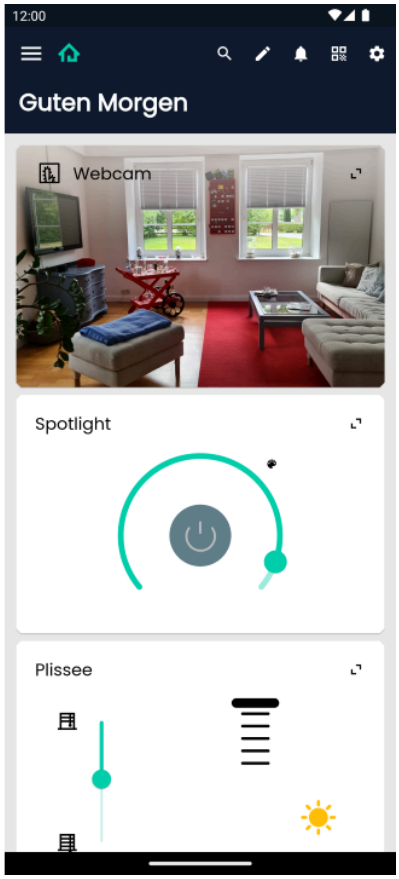
There are three ways to connect to a visualization. These are explained below. Also, the app can be opened directly in a demo visualization.

AUTOMATIC SEARCH

If the mobile device is connected to the same network as the IP-Symcon server, it can be found with the automatic search. The first tile shows the found IP-Symcon Servers. These can be tapped and then the corresponding visualization can be selected. If a password is set for the visualization, it must be entered now. If there is only one IP-Symcon server with a visualization in the network, it will be opened directly.

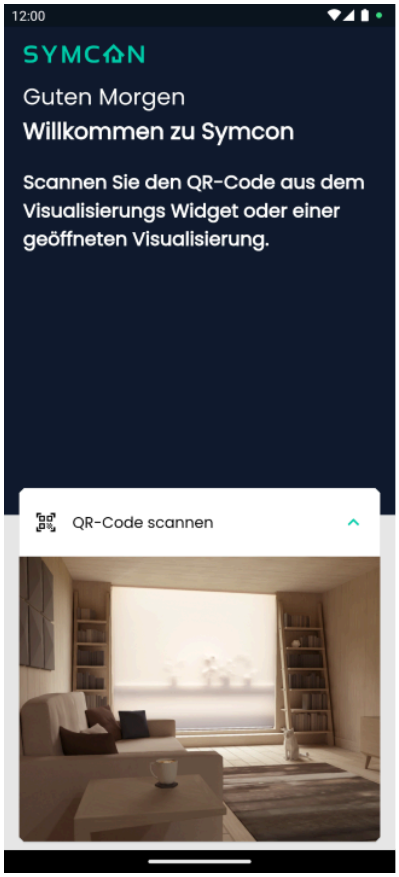


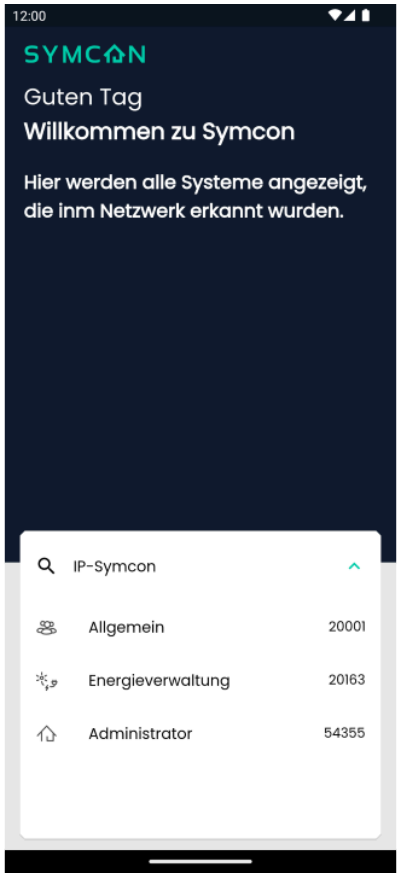


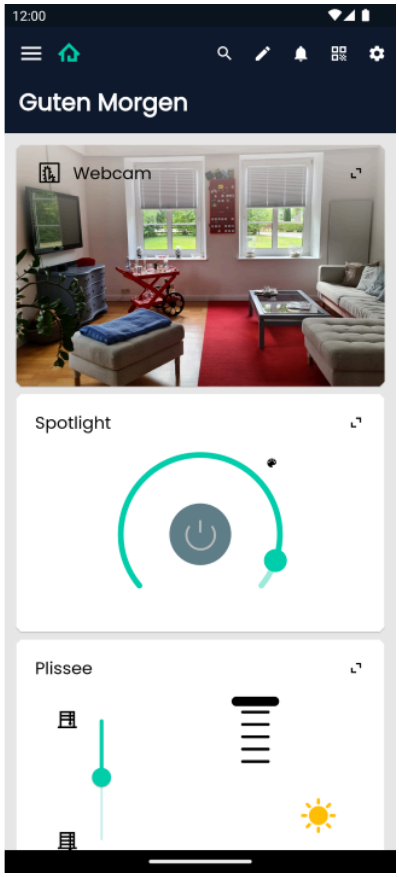


QR CODE

The QR code needed for scanning can be obtained from the visualization widget in the management console or from an already opened visualization. To scan the code, the necessary permissions must be granted in the app. After the code is scanned, the desired visualization can be selected and the password can be entered.



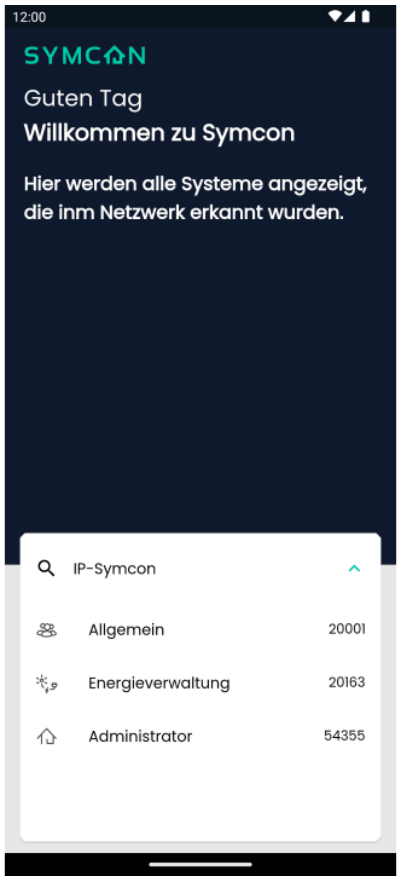


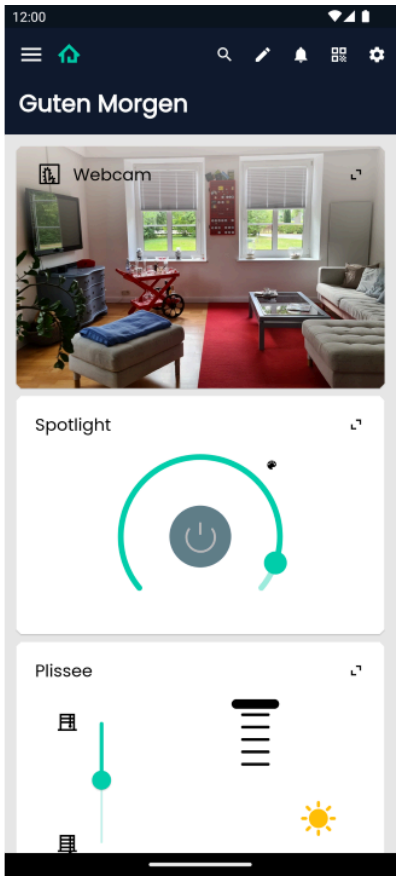


ADD MANUALLY

To add an IP-Symcon server manually just enter the IP address and port of the server into the input field. Alternatively you can use the connect address or your own DynDNS address.

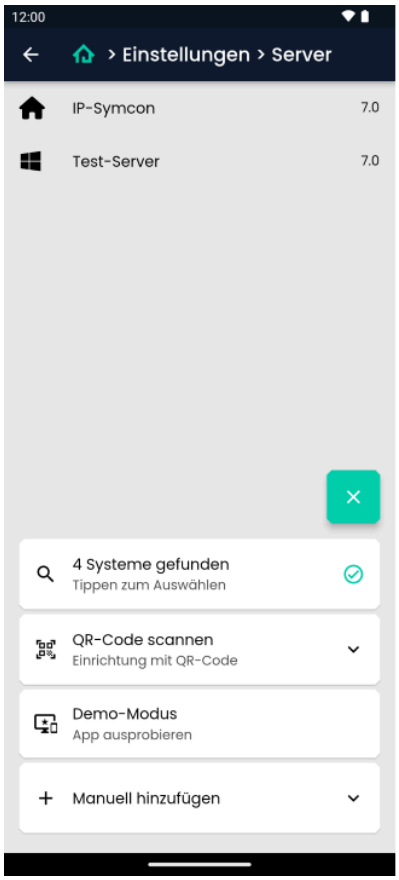


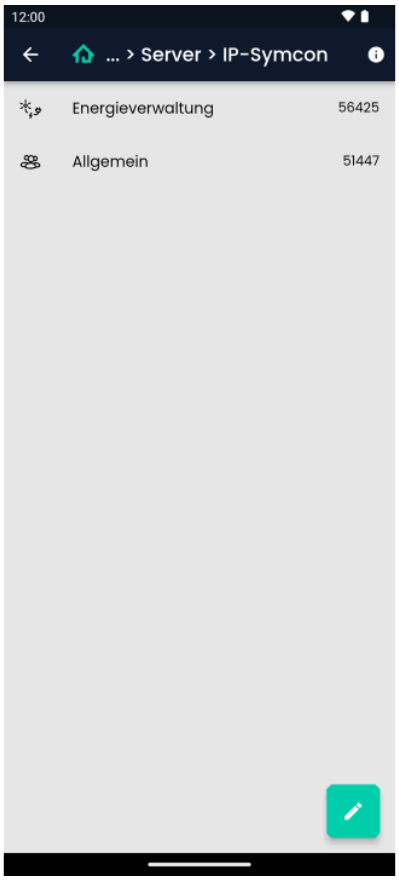


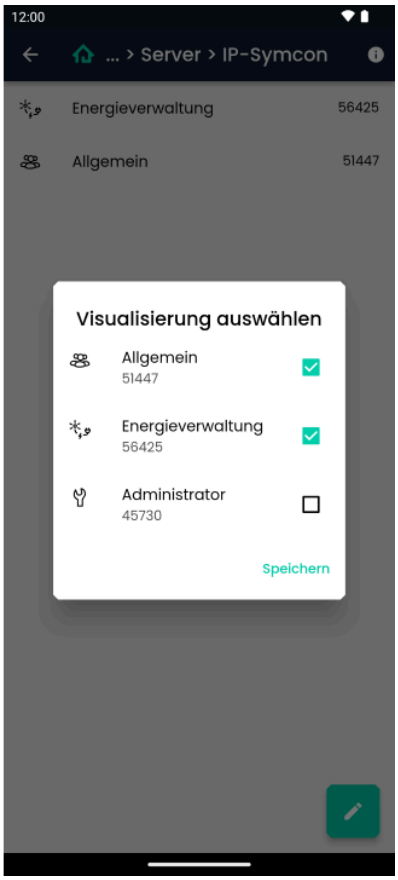


SETTINGS

All configured servers and visualizations are saved on the device. New servers can be set up via the gear in the appbar under the item 'Server'. If a server is selected, the pencil in the lower right corner can be used to edit the saved visualizations.


12:00
> Einstellungen > Server
IP-Symcon 7.0
Test-Server 7.0
4 Systeme gefunden
Tippen zum Auswählen
QR-Code scannen
Einrichtung mit QR-Code
Demo-Modus
App ausprobieren
Manuell hinzufügen


12:00
... > Server > IP-Symcon
Energieverwaltung 56425
Allgemein 51447


12:00
... > Server > IP-Symcon
Energieverwaltung 56425
Allgemein 51447
Visualisierung auswählen
Allgemein
51447
Energieverwaltung
56425
Administrator
45730
Speichern

Browser

Require: Symcon >= 7.0

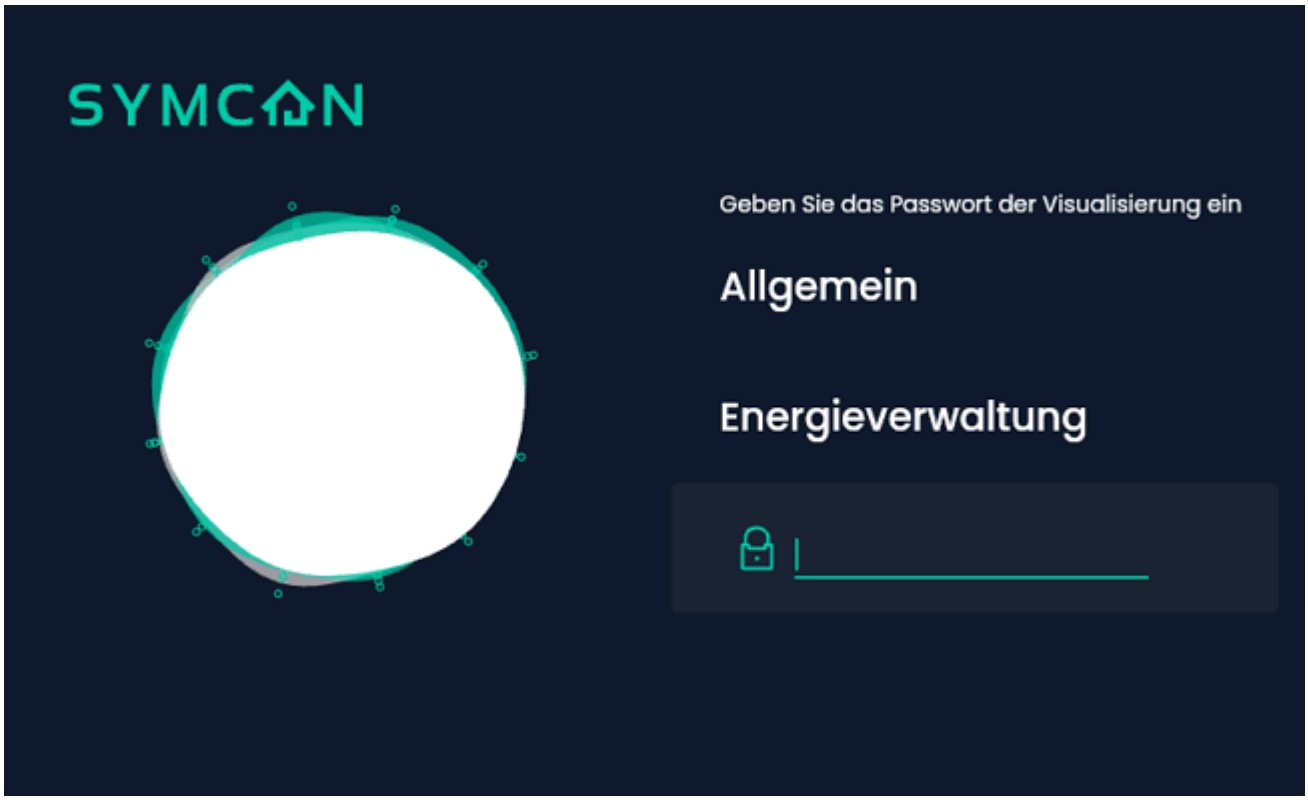
CONNECTING TO THE TILE VISUALIZATION IN THE BROWSER

The tile visualization can be opened via "[ServerIP]:3777/" in a browser. If the WebFront visualization is still set as default, the Tile-Visualization can be accessed via "[ServerIP]:3777/tile/".

SELECT VISUALIZATION



A list of the visualizations available on this server will then appear in the browser. If a visualization is password protected, this is symbolized by a lock icon. With one click the visualization to be opened can be selected. If the visualization is not password protected, it will be opened directly.



Otherwise a text field appears for entering the password. There the password must be entered and finally confirmed with Enter. If the password is correct, the visualization will be opened.

WebFront Visualization

The WebFront is still supported. However, since version 7.0, the Tile Visualization was introduced as modern visualization. Even though the WebFront is still usable and is maintained regarding errors and compatibility, there will be no new features. It is recommended to use the Tile Visualization.

The WebFront does not have to be configured and can be used completely without any advanced settings. All possibilities presented here are optional and can be used by experienced users, in order to better adapt visualizations to their needs.

The WebFront offers a simple and attractive way to display and switch devices, display images, and run scripts.

WEBFRONT VERSIONS

VERSION	DESCRIPTION
WebFront (incl. Editor)	The HTML5 Version of the WebFront can be opened and operated with the current versions of common web browsers. These include: *Google Chrome (25+) [officially recommended]* Mozilla Firefox (34+) *Apple Safari (OS X 10.9+)* Opera (12+, CSS3 animations only with Opera Next) * Microsoft Internet Explorer (10+)
WebFront (without Editor)	*Apple iOS (7.0+)* Google Android (4.0+)
Mobile (iOS)	IP-Symcon Mobile is a native app that can be downloaded directly from the AppStore. All iOS devices with iOS 6.x or later are supported. IP-Symcon Mobile for iPhone, iPod touch and iPad in the iTunes App Store
Mobile (Android)	The Android counterpart to IP-Symcon Mobile is also a native app that can be downloaded directly via Google Play. IP-Symcon Mobile for Android devices at Google Play

INSTALLATION

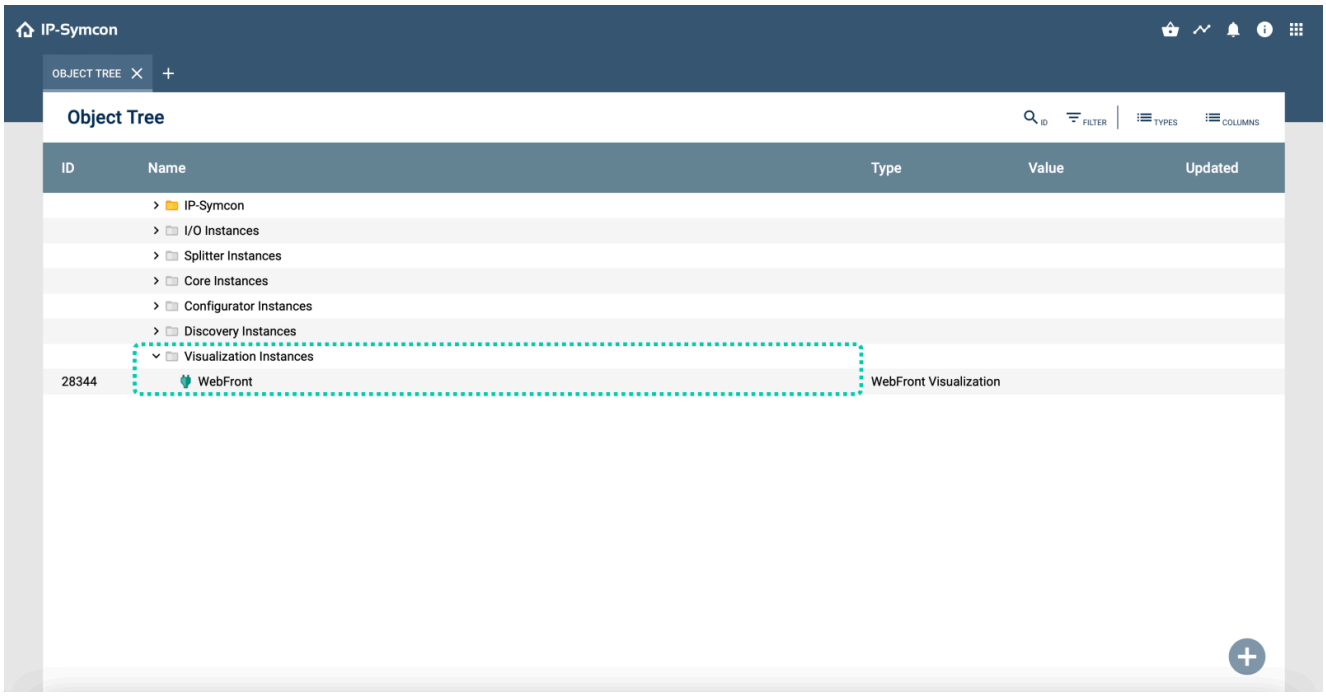
The WebFront is immediately available and can be opened by clicking "Open Visualization". The WebFront is also be available to all other devices with browsers in the network. If the computer on which the IP-Symcon Server is installed is called Home Server, the WebFront is available under the address http://Home Server:82/ in the local network. If the computer's IP address, e.g., 192.168.1.2, is known, the WebFront can also be opened under the address http://192.168.1.2:82/.

CONFIGURATION

The WebFront generally runs without any further necessary configuration. However, various settings can be changed. The relevant documentation can be taken from the subpages of this document. The general, version-comprehensive configuration is described in more detail below.
A choice of icons for objects in WebFront can be found under Icons.

OPEN CONFIGURATION

Configuration can be started directly from the welcome page of the IP-Symcon Management Console. The configuration can be opened via the menu item 'Configure WebFront'.



CONFIGURATORS

IP-Symcon Basic contains a configurator that only supports limited changes. Up to five different configurators can be created in the IP-Symcon Professional Edition that can be retrieved on their own using the WebFront start page and be secured with a password, if desired. A configurator can be created as an independent unit, e.g. for a visualizations object, represent a special view for a maintenance technician, or simply create a security-specific separation. The same options are available for each configurator, which can be adapted as desired. As many configurators as you like can be created in the Unlimited Edition.

Web

Require: Symcon >= 3.0

The **IP-Symcon Professional** or **IP-Symcon Unlimited** edition is required to configure WebFronts

Only objects, which were previously created in the Object Tree of the Management Console, can be added and displayed in the WebFront. The WebFront Editor cannot create new categories or objects. It is only used to arrange existing objects according to your individual preferences.

Scripts that are executed via the WebFront provide these System Variables

OPERATION

At the top of the screen you can select elements that were set up in the WebFront. By default, these are IP-Symcon for building control and weather to display the current weather data and forecasts of the German Weather Service. Clicking a module name displays its content in the middle section. The current module name is highlighted. The middle section displays the main elements of a module. The other bars at the top of the screen offer control options for the currently selected category.

WEBFRONT-EDITOR

CONFIGURING

The configuration is divided into several parts. Different settings can be adjusted in each tab. This article primarily presents the settings options for the regular WebFront.
The following settings can be adjusted in the individual tabs:

TAB	DESCRIPTION
Editor (since 5.0)	Starts the WebFront editor
Appearance	Configuration of a title and icon in the login screen
Security	A password for access control and a special IP AutoStart function can be set
Notifications	Allows sending test messages to configured devices and removing registered devices

TAB	DESCRIPTION
Structure (up to 4.4)	Configuration of the individual elements of a WebFront

SETTING UP ANOTHER WEBFRONT

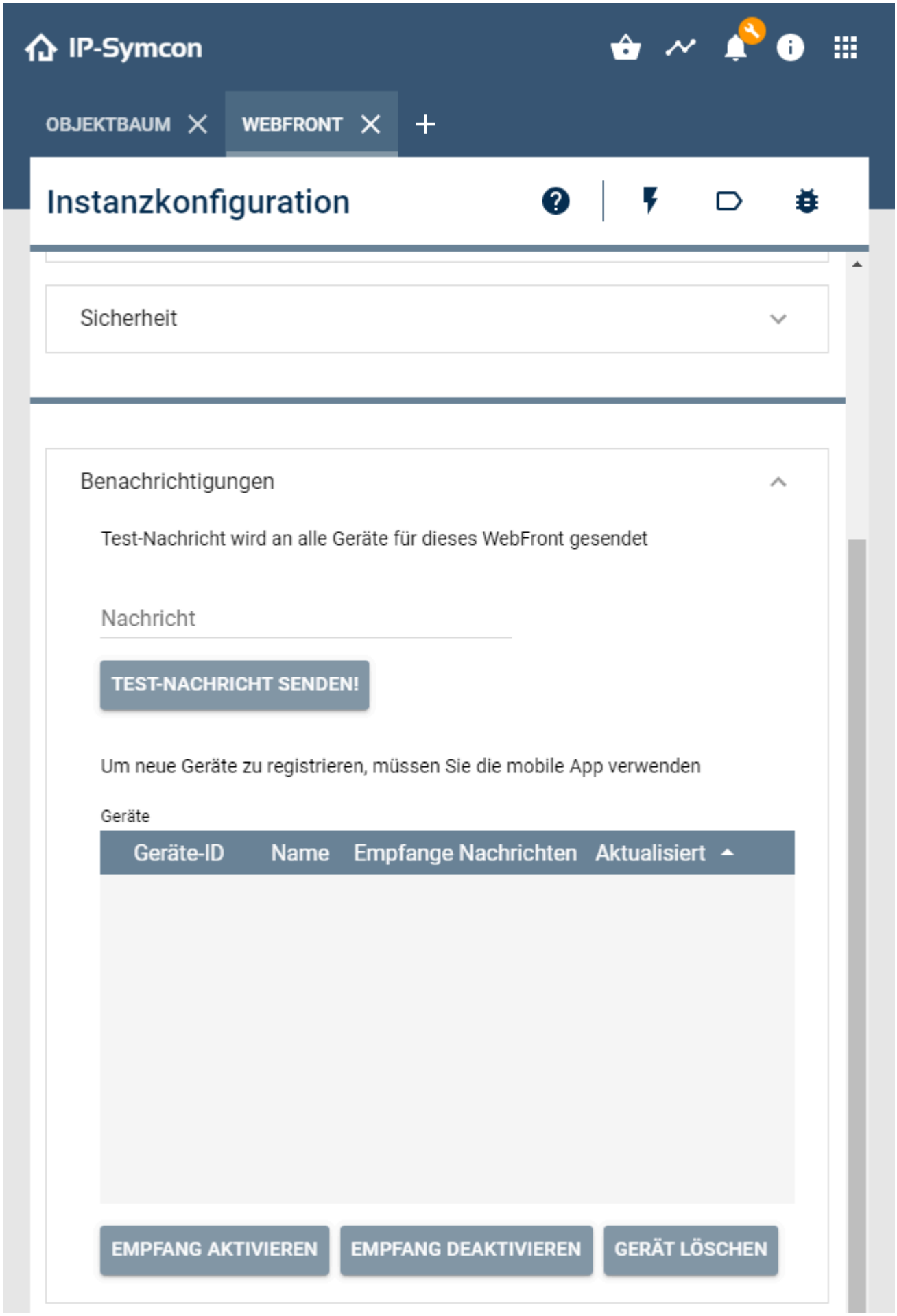


To use more than one WebFront, you need an **IP-Symcon Professional** or **IP-Symcon Unlimited** edition.

In the “Visualization” section of the management console, you can display an overview of all WebFront configurators via “configure visualization”. To create a new configurator and thus a new WebFront, you have to click "New". The name of the configurator instance is also the name of the WebFront in the browser.

Notifications

The tab provides an overview of the registered devices. It can be selected whether the respective devices receive messages.
Also, a test message can be sent.



REGISTERING TO THE PUSH MESSAGE SERVICE

The first time a mobile device successfully logs in to a WebFront, that device will automatically be added to the push message service for this Configurator. If no push messages are to be sent to this device, then this device must be explicitly deactivated for reception in the configurator. Note that if the password for the configurator is changed or the mobile access is completely disabled, the registration to the push message service remains untouched. If a device is completely removed from the list, it will be handled as an unknown device at the next login and added to the push message service again.

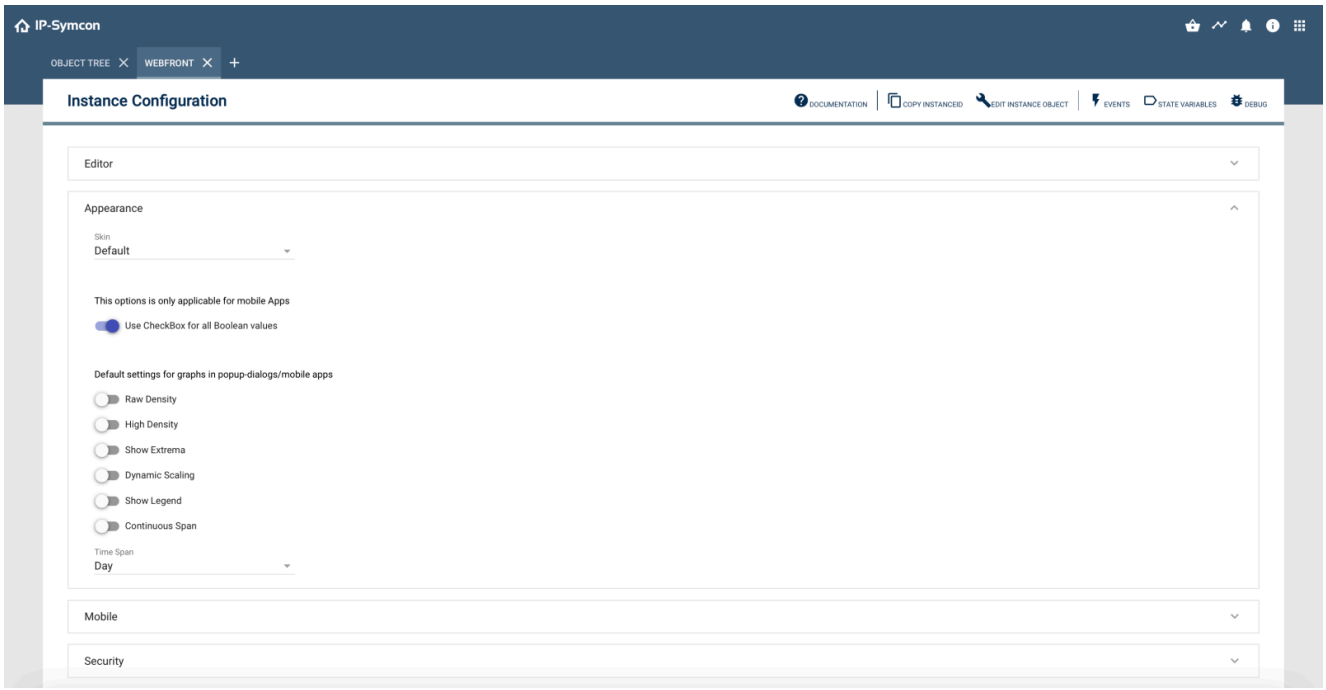
NOTIFICATION CONTROL

The messages are sent via Notification Control.
Further information can be found under Notification Control.

Appearance

APPEARANCE

The appearance tab takes care of the visual presentation and possible default settings of the WebFront.



NAME, ICON, VISIBILITY

Via "Configure" various default settings can be selected.

SKIN

Since IP Symcon 3.0 it is possible to design the WebFront via skins. Further information is available in the Skin Control.

NESTING

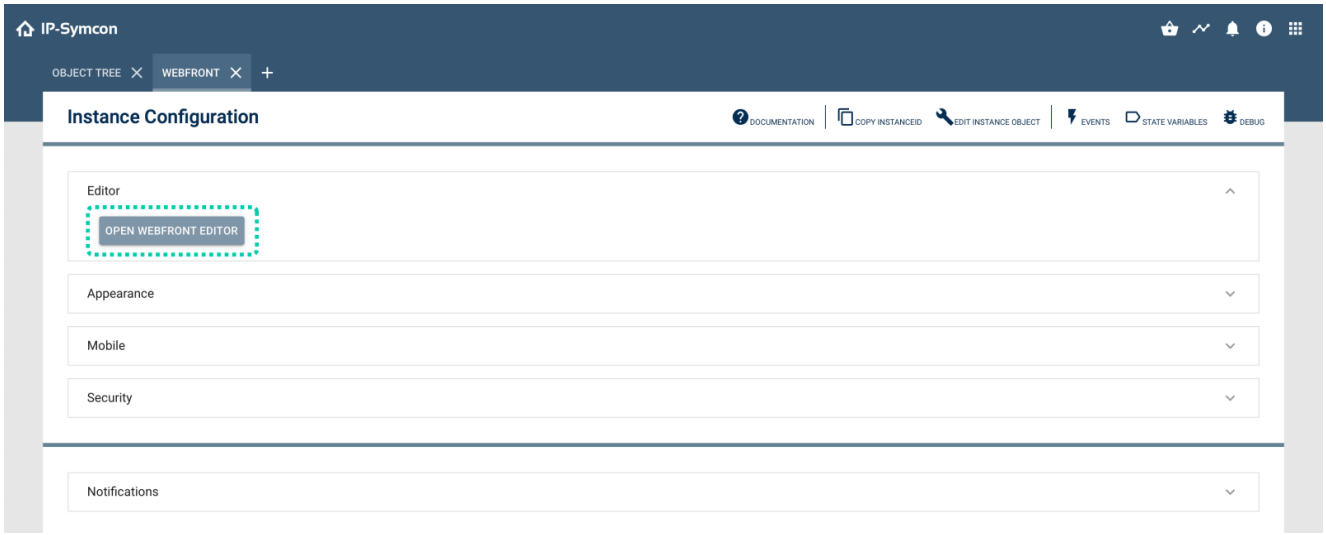
Since IP-Symcon 5.5 this option is enabled by default und enables the presentation of nested elements, e.g., instances below instances. The mobile apps with version 5.5 or newer also support this option. After activation, this option cannot be disabled again as it should removed in the long term.

DEFAULT POPUP GRAPH SETTINGS

Default settings for popup graphs can be defined here. The exact meaning of the options is explained in Display types of charts

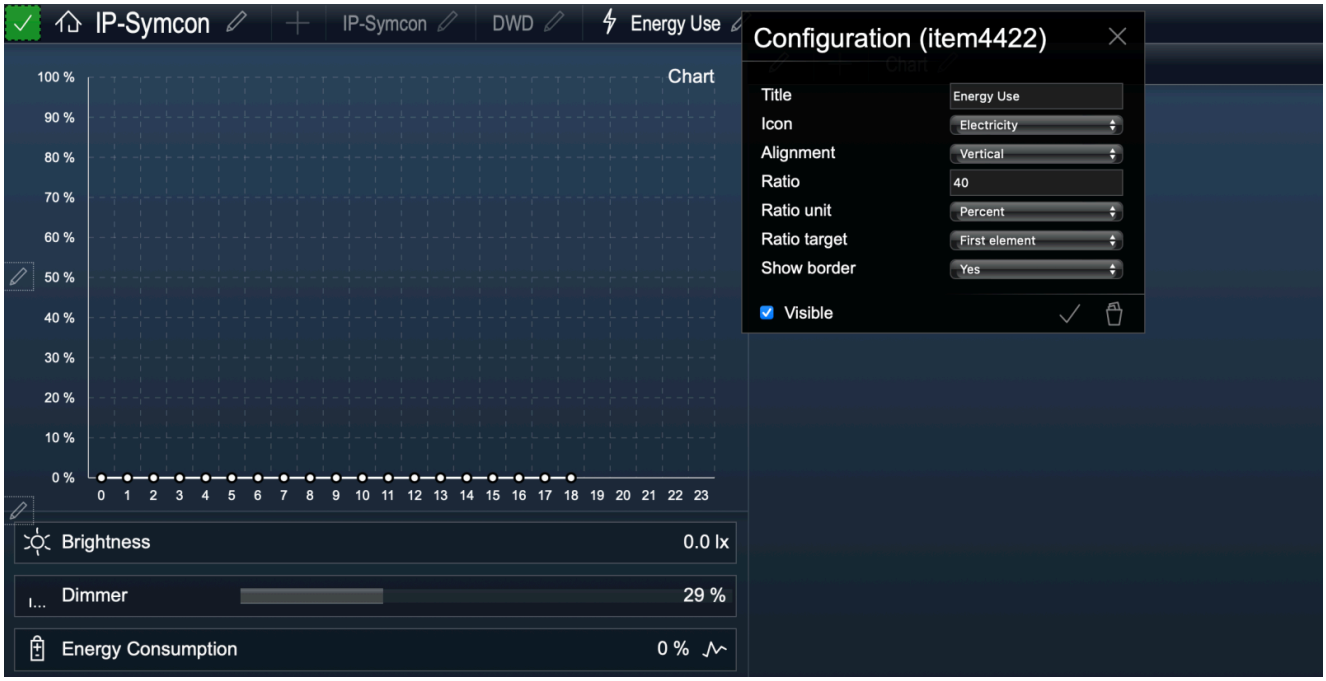
Editor

The editor can be opened from the WebFront configurator via the button "Open WebFront Editor" under "Editor".



A Professional license or higher is required to use all features of the WebFront editor. Otherwise, the extent is severely limited.

USAGE OF THE EDITOR



Within the editor, the WebFront is displayed as usual and is extended by configuration elements. With a click on the plus, an element can be added. Added elements can be configured via a click on the pen. A dialog is opened where the parameters of the element

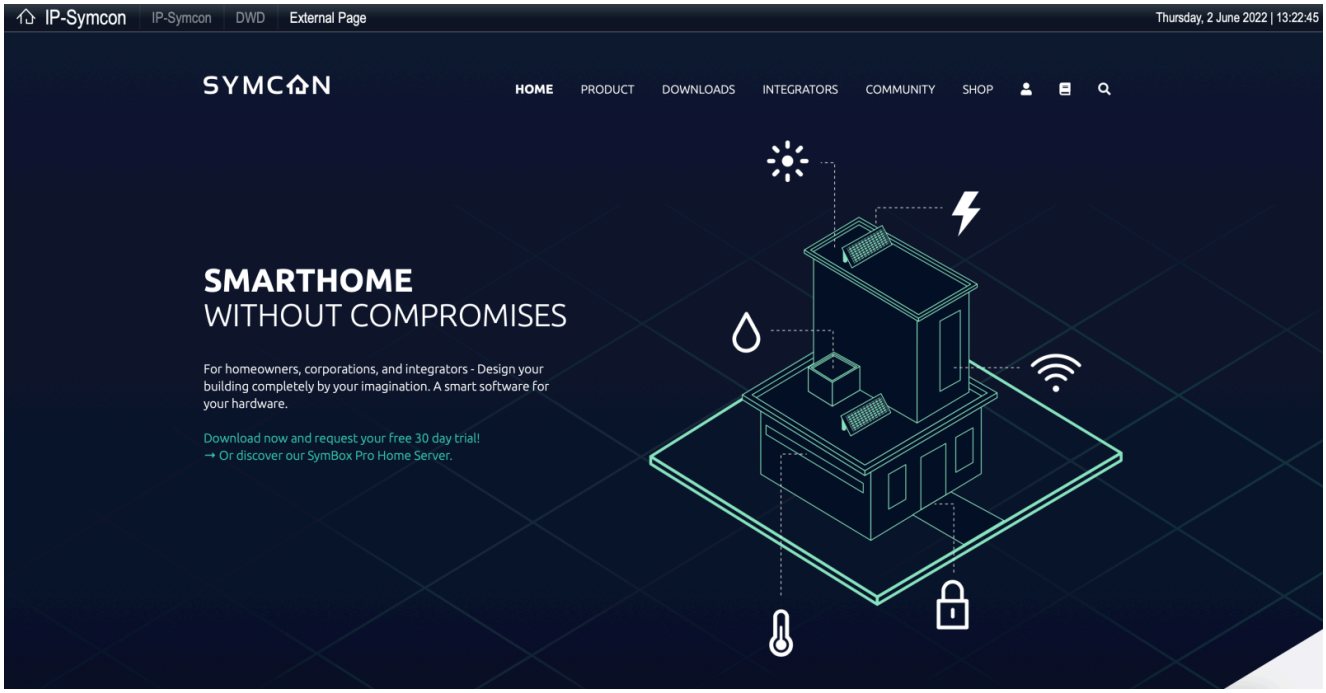
can be configured. The individual parameters of the different elements are explained in their individual documentation pages. After adjusting the parameters, these are confirmed by clicking the check mark. Alternatively, an element can deleted by clicking the trash can in the dialog. Finally, the changes need to be confirmed by clicking the green check mark at the top left. Otherwise, the changes are not saved.

OVERVIEW OF AVAILABLE ELEMENTS

ELEMENT	DESCRIPTION
Category	This element displays a category and all its sub-objects in the WebFront. The ID of the category to be displayed can be changed through the start category parameter.
Graph	With this element, the graphs set in the Archive Module are displayed directly. Normally, graphs can be called using the icon located at the left of the value of the variable. If graphs should be embedded directly, for example in TabPanes or SplitPanes, this can be done using the graph element.
External Page	This element inserts an external site in the WebFront via an IFrame. The URL parameter and interval are available.
Content Changer	The Content Changer allows switching between multiple Images, Streams, HTML Boxes, Text Boxes or mail instances (IMAP, POP3) within a single page.
SplitPane	Using the SplitPanes, the WebFront view can be split into two parts and any other element can be inserted in each part. Thus, if SplitPanes are inserted in SplitPanes, a QuadView can be simulated.
TabPane	TabPanes can be used to create custom menu bars at the upper area, which refers to other items.
Weather (DWD)	Displays the weather of the DWD with the ability to select specific areas of Germany and to switch between a normal image or a radar image.
Time Widget	Displays the current time.
Info Widget	Can either be used to display a static icon or linked to a variable which content / icon is displayed.
Logout Widget	This element adds a widget with which you can return to the WebFronts selection screen.
Idle Widget	This element adds an invisible widget, which switches to a specific page after a period of inactivity.

External Page

This item allows the display of external websites. Both locally stored pages and external websites can be displayed. It is also possible to have the page updated automatically at a certain interval.



EXTERNAL WEBSITES

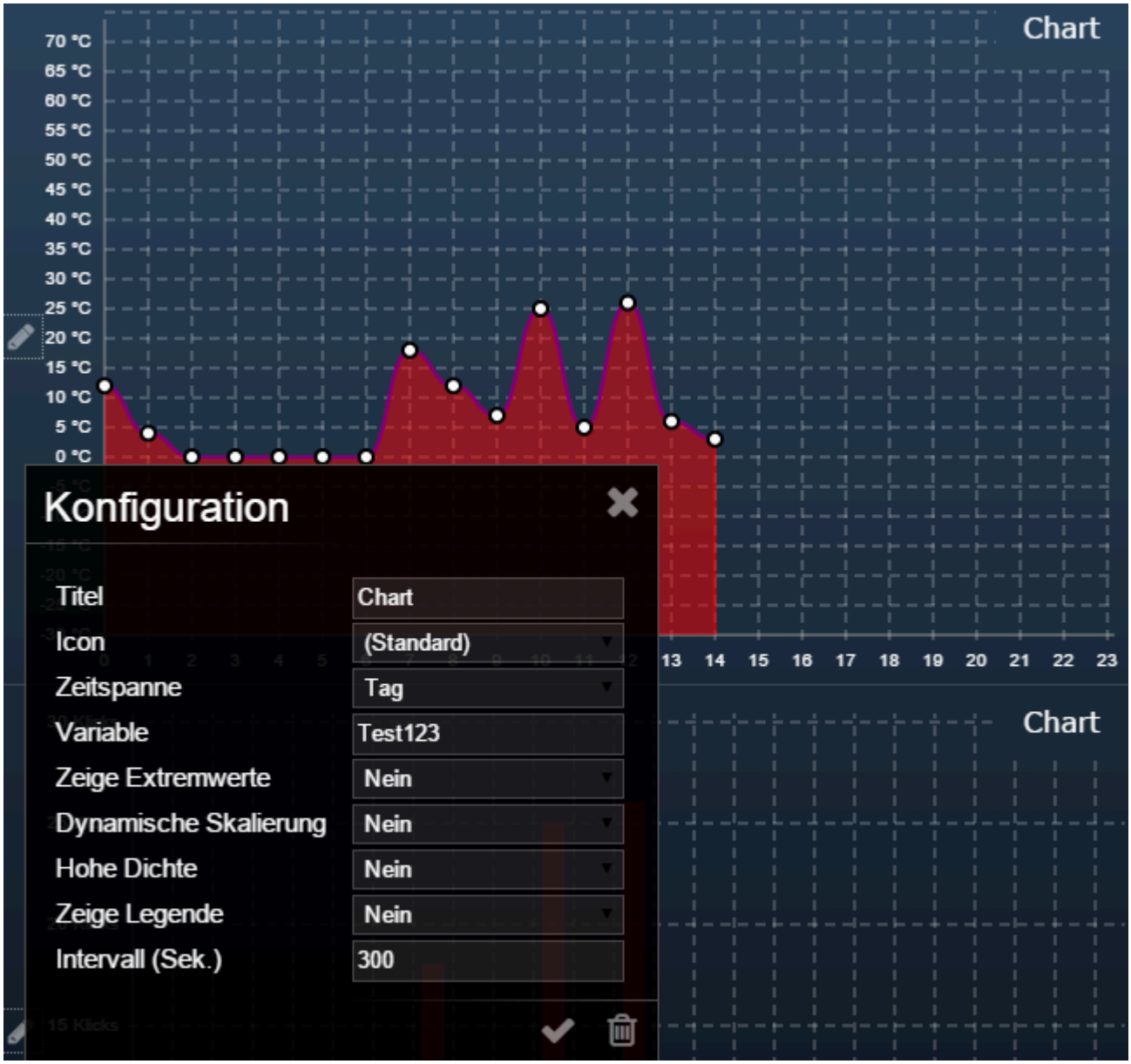
External websites are specified in the normal URL format, e.g.: https://www.symcon.de.

PROPERTIES

Title: Title for the item in the TabPane navigation. By default, "External Page" will be displayed.
Icon: Icon that appears next to the title in the TabPane navigation. By default, no icon will be displayed.
URL: Specify the address of the page which content will be displayed.
Interval: Optionally, an automatic refresh interval for the page can be specified in seconds.
Security: Optionally, the Sandboxing option can be disabled. In some cases, this may cause the external page to disable your WebFront and to present itself in full size.

Graph

A full-screen graph will be shown directly in the WebFront.
A single logged variable or a chart object can be presented.



PROPERTIES

Title: Title for the item in the TabPane navigation. If no individual name is specified, the object name of the selected variables will be displayed.
Icon: Icon that appears next to the title in the TabPane navigation. By default, no icon will be displayed.
Period of Time: The display area of the graph can be set to hour, day, week, month, or year.
Variable: Here a logged variable or chart media file to be displayed can be selected.
Show extreme values: This option specifies whether the graph is displayed with or without extreme value curves, if available.

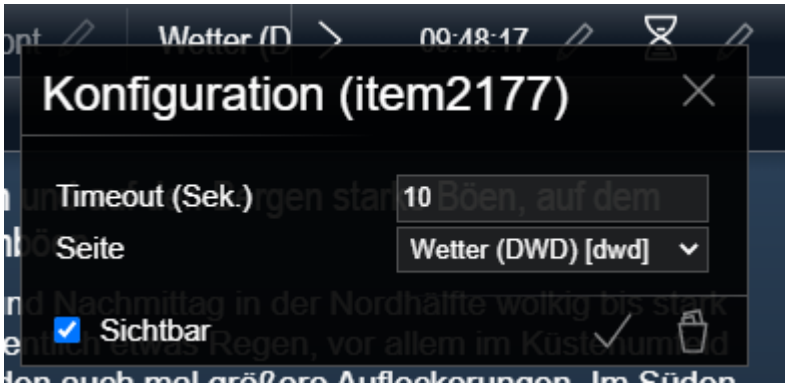
Dynamic Scaling: If the dynamic scaling is enabled, the graph will be dynamically scaled according to the minimum and maximum values in the time range. Otherwise, scaling will be performed according to the minimum and maximum specified in the variable profile of the selected variable.
High Density: Increases the data point density, which makes the graph more accurate. Works only with line charts or logged variables on "Standard" -Logging.
Show legend: Turn the legend on / off.
Intervall (sec): The displayed graph will be automatically updated in seconds according to the interval set here. By default, this happens every 300 seconds.

Idle Widget

The Idle Widget offers the possibility to create an invisible timer in the widget area of a TabPane. This is visible only in the editor mode and changes the displayed page after a defined inactivity time.

SplitPanes and the TabPane itself can not be jumped directly. These are purely cosmetical elements. However, elements within the SplitPane or TabPane can be selected as target, e.g., added categories.

PROPERTIES



Timeout (sec): The current page will be changed to the target page after the defined time of inactivity.
Page: The target page that is selected when the Idle Widget triggers.

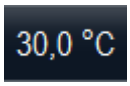
Info Widget

The Info-Widget offers the possibility to display in the widget - area of a TabPane, a fixed icon, a dynamic icon for a variable or a textual variable value. In addition, a script can be selected that will be executed by clicking on the widget.

DISPLAY VARIANTS

ICON - DISPLAY TEXT PRESENTATION





A fixed icon can be selected in order to display the Icon property. It should be noted that neither type nor variable may be specified.
A dynamic icon will be displayed when a variable is specified and type is set to Icon. Which icons will be displayed and which icons are available by default is described under Icons. If the "transparent" icon is displayed, then the widget is hidden. In this way, information - icons can be dynamically inserted, e.g.: when a device needs a battery change.
The current variable value is formatted textually using the variable profile associated with the variable if a variable is specified and the type is set to text.

PROPERTIES

Icon: Here a fixed icon can be selected for display.
Type: The type determines whether a variable should be displayed as text or as an icon.
Variable: Here you can select a variable that will be displayed according to the selected type.
When clicking: A script, which is executed by clicking on the widget, can be set here.

Content Changer

The Content Changer is used to display the following objects:
Email-instances (IMAP, POP3)
String-variables with variable profile ~TextBox or ~HTMLBox
Media-objects of the type Image or Streams

PRESENTATION



If a displayable object is selected as source object in the configurator, only that object will be displayed. Otherwise, all displayable immediate child objects of the selected source object will be selected for display. If more than one displayable child object is found, a selection list for switching between the child objects will be displayed. For editable string variables of Type ~TextBox, a button will also be displayed to call the on-screen keyboard in order to edit the text. Switching between the objects to be displayed can also take place automatically if a change interval is specified.

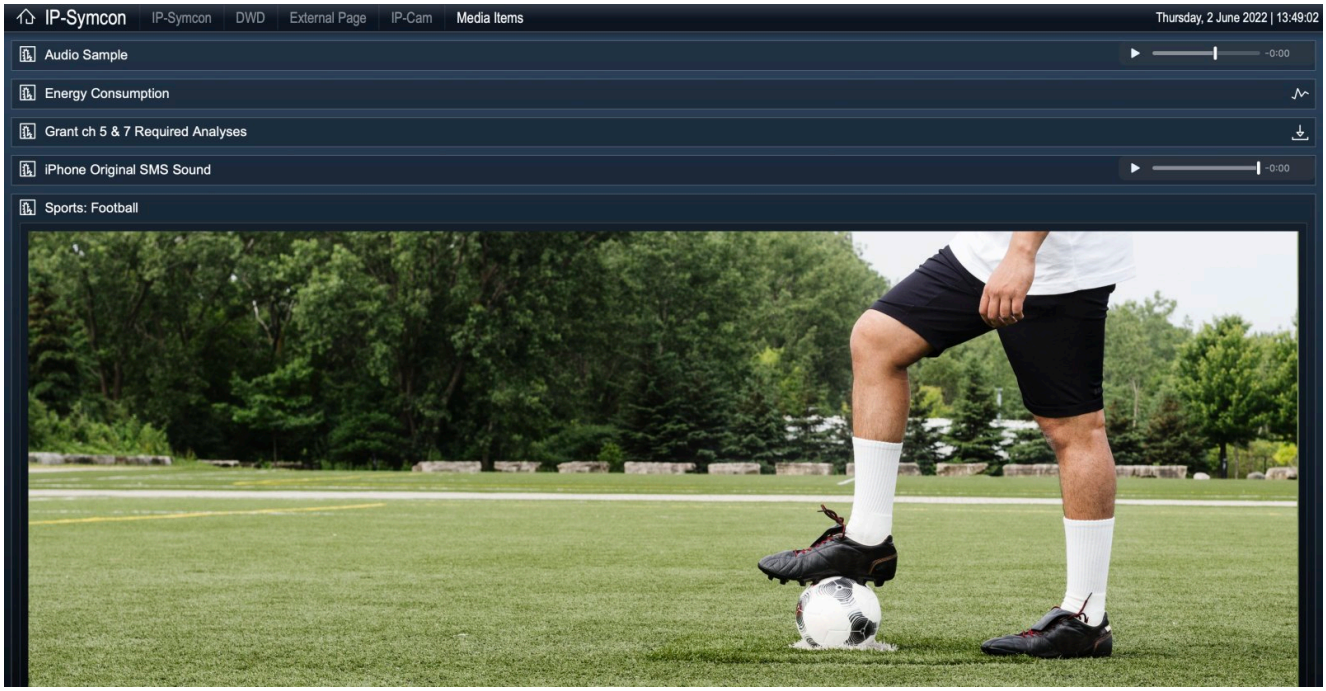
PROPERTIES

Title: Title for the item in the TabPane navigation. If no individual value is specified, the name of the selected object is shown.
Icon: Icon that appears next to the title in the TabPane navigation. By default, no icon will be displayed.
Source object: The object that is displayed directly or whose child objects are shown.
Change interval: Interval in seconds, in which automatic switching takes place between the displayed objects. The default value "0" disables the function.
Upscale: _This option indicates, whether images to be displayed are automatically scaled up when space is available. By default, it is not scaled up.

Downscale: If this option is activated, images will be automatically downscaled if there is not enough space available. The option is enabled by default.
Maintaining the aspect ratio: If scaling up or downscaling is enabled, this option allows you to set whether or not to maintain the aspect ratio of the scaled images when scaling. By default, the aspect ratio will be maintained.

Category

The category element allows the display of objects that can be found in a category as well as the navigation in subcategories.



OBJECT AREA

The object area displays the child objects of the selected category. Further information and possible display modes can be found in the Object Presentation area.

NAVIGATION BAR

If the displayed category has visible subcategories, a navigation bar with subcategories will be displayed automatically. If a subcategory is selected in the navigation, its content will be displayed in the object area. If a category displayed in this way also has visible subcategories, the navigation will be expanded automatically. Naturally, categories that were linked via Link object will also be displayed in the navigation bar. The "Show Navigation" property can be used to hide the category bar if required.

PROPERTIES

Title: Title for the item in the TabPane navigation. If no individual value is specified, the name of the selected object will be displayed automatically. If the value is set, but the text is left in blank, the title will not be displayed.
Icon: Icon that appears next to the title in the TabPane navigation. By default, no icon will be displayed.
Start category: Category whose content will be represented by the element. By default, the main category will be displayed as "ID 0".

Show Navigation: This option defines that the navigation bar is displayed, provided there are subcategories below the start category. By default, the navigation bar will be displayed.

SPECIAL FEATURES

If the title is set to an empty text and no icon is specified, the category module won´t be displayed in the navigation, but it remains accessible via WFC_SwitchPage.
If a category should be displayed in WebFront with icon but without title, a space must be entered in the title (" ").

Logout Widget

The Logout Widget offers the option of displaying a button in the widget area of a TabPane. When used, one will be brought back to the WebFront login.

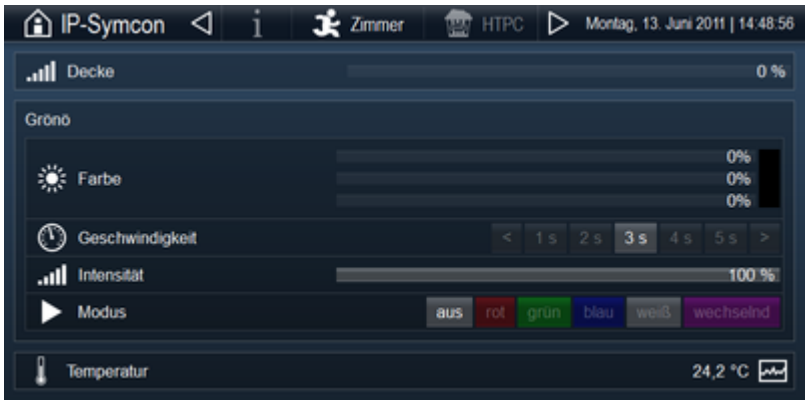
PROPERTIES

Timeout (sec.): Timeout, which automatically logs out in the event of inactivity. If the value = 0, it is switched off.

TabPane

The TabPane is an element that allows navigation between all other types of elements as well as displaying widgets.

REPRESENTATION



At the top of the TabPane there is a navigation bar, with a content area below. In the navigation bar on the left side, if selected, the *"sub-icon"* and the *"subtitle"* are displayed first. Right next to it follows the navigation area, which displays the clickable fields for all TabPane child elements with their icon and title, if set. Clicking on a field, displays the content of the associated element in the content area. At the left edge of the navigation bar the is the widget area, which displays all widgets that are children of the TabPane.
The TabPane is typically used as the topmost element of a WebFront configuration because of the navigation capabilities it provides. Of course, a TabPane can also include more TabPanes.
If the image width is insufficient to display all navigation fields, clickable arrows will be displayed automatically to scroll within the navigation area. In this case, the width of the navigation area is reduced to 100 pixels. Next, the widget area is shortened.

PROPERTIES

Title: Title of the TabPane in the navigation of possible parent TabPanes. By default, "Tab Pane" will be displayed.
Icon: Icon that is displayed next to the title in the navigation of possible parent TabPanes. By default, no icon will be displayed.
Subtitle: Text that appears at the left edge of the navigation bar. By default, no text is displayed.
Subicon: Icon that appears at the left edge of the navigation bar. By default, no icon is selected.

SplitPane

The SplitPane is a layout element that can split two sub-elements horizontally or vertically. In this way it is possible to display, for example a Content Changer with camera views next to a category. As SplitPanes and other layout elements such as other SplitPanes and TabPanes can be inserted below, infinite and individual layout adjustments are possible.



SUBDIVISION OF SUB-ELEMENTS

The "Subdivision" property allows you to set whether the two sub-elements of the SplitPane should be displayed one above the other (horizontal division) or side by side (vertical division). The available space for each sub-element is defined by the properties *"Size"*, *"Size unit"* and *"Ratio Target"*. *"Ratio Target"* defines whether the first or second sub-element should be configured via *"Size"* and *"Size unit"*. The target element is then assigned the space specified by the *"Size"* property in the set *"Size unit"*.

EXAMPLE

An image with a picture width of 640 pixels should be displayed to the right side of a category. In order to do this, you assign the vertical value to the SplitPane property, set the *"Ratio target"* to the second element, *"Size"* to 640, and *"Size unit"* to pixel. Now create a sub-element of type category and a sub-element of type content changer in the WebFront configurator below the SplitPane. For the sub-elements, the position should be set so that the category Position 0 and the content changer are given Position 1. Now the sub-elements can be configured to display the desired category and image.

PROPERTIES

PROPERTY	DESCRIPTION
Title	Title for the item in the TabPane navigation. By default, "SplitPane" will be displayed.
Icon	Icon that appears next to the title in the TabPane navigation. By default, no icon will be displayed.
Subdivision	Here you can select whether the division of the two sub-elements should be horizontal or vertical. By default, it is divided horizontally.
Size	The size of the sub-element selected for the division according to the size unit. The default value is 50.

PROPERTY	DESCRIPTION
Size unit	Here you can select whether the size refers to percent or to pixels. By default, percent will be taken.
Target ratio	This property allows you to choose whether the first or the second sub-element of the SplitPane will be displayed according to the specified size specification. By default, the first item will be selected.
Show margin	With this property, it can be defined whether if a margin should be shown to separate the sub-elements. By default, the margin is shown.

Time Widget

The Time Widget allows you to view the current date and time in the widget area of a TabPane. If no custom display format is set, the presentation depends on the language set in the web browser. You can choose between the standard formats "Hidden", "Short", "Medium", "Long" and "Complete" for the date format and the time format. A custom format can be entered, according to the datePattern-table at dojotoolkit.org/reference-guide/dojo/date/locale/format.html.



PROPERTIES

Date format: A predefined pattern for date formatting the can be selected here.
Time format: Like date formating, a predefined pattern for time formatting can be selected here.
User-defined format: Optionally, a custom format can be specified here.

Weather (DWD)

The weather element allows displaying the weather forecast of the German Weather Service for Germany and the individual federal states in text and image.



NAVIGATION BAR

The navigation bar is located in the upper area. If the property "Region" is set, you can switch between the weather report for Germany and the selected state. In addition, the forecast period can be changed between today, tomorrow, 3rd day and 4th day.

FORECAST AREA

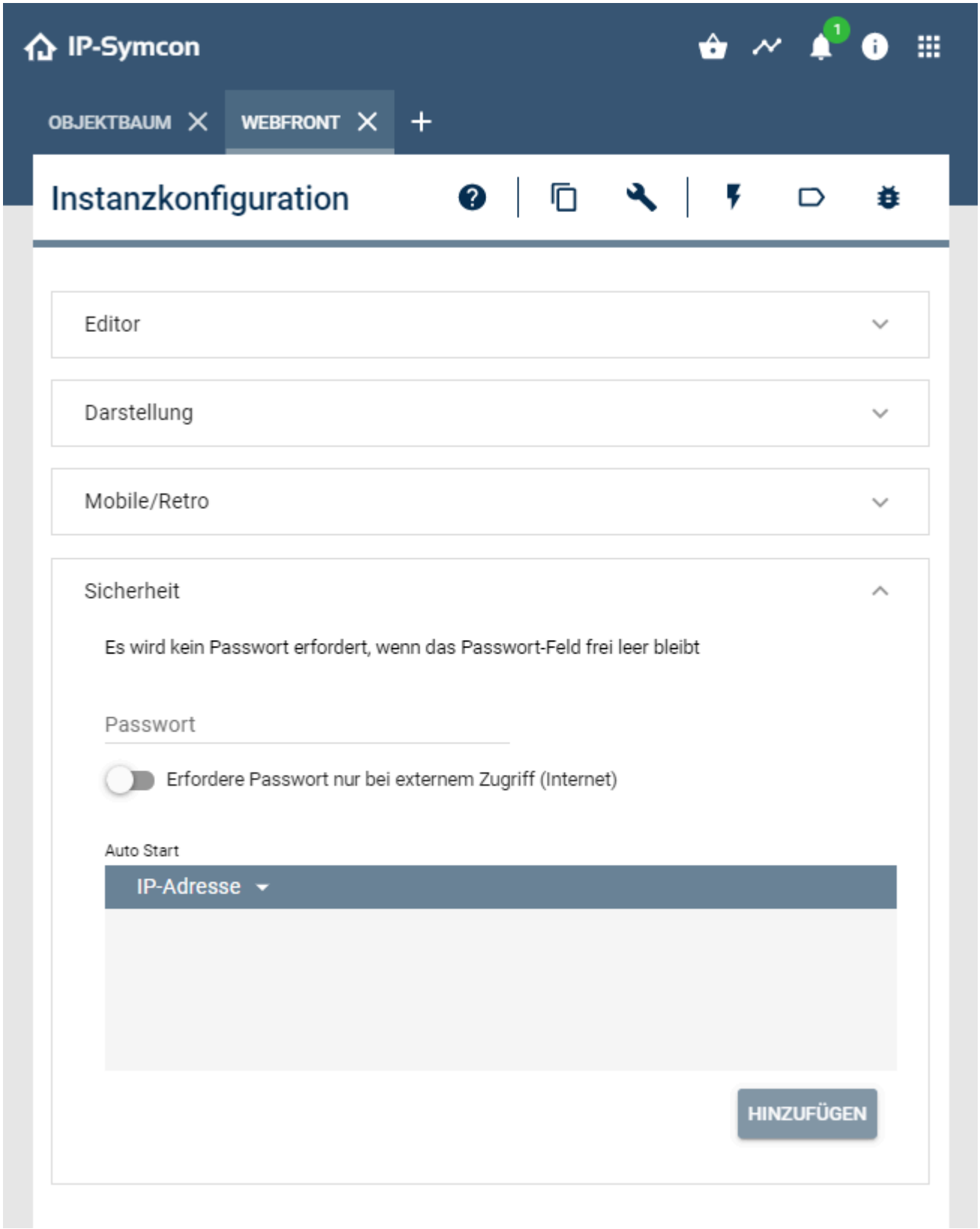
In the forecast area, the forecast text is shown on the right and an appropriate weather map on the left.

PROPERTIES

Title: Title for the item in the TabPane navigation. By default, "Weather" will be displayed.
Icon: Icon that appears next to the title in the TabPane navigation. By default, no icon will be displayed.
Region: Optionally, a federal state can be selected here, its weather forecast can be displayed instead of the Germany-wide forecast in the navigation bar.

Security

In this tab a password query for the WebFront and its editor can be set up.
Further security information can be found under Security .



ENABLE EDITOR

Activates the WYSIWYG editor. This option is enabled by default for a fast initial setup and configuration. If the setup is finished, the editor should also be disabled.
The editor will be likewise protected by the password set here.

Hidden objects are invisibly loaded into the WebFront and can therefore be viewed, for example to via the developer console of the web-browser with the appropriate know-how. This is not an error, but a necessity to realize a rapid change in the visibility through the command IPS_SetHidden. As soon as the WYSIWYG editor is activated in the WebFront Configurator, the complete object tree is transferred, for example to realize the start/home category selection. This enables anyone with access to this WebFront to change the start category. After a successful use of the WYSIWYG editor, this should therefore be deactivated in the WebFront configurator.

REQUIRED PASSWORD

For each configurator, a password can be assigned as access protection. This password is required before accessing the respective configurator, or during the configuration within the mobile apps. In addition, the special option can be activated, in which the password is not required within the local network. This password is only valid for accessing the system and its components. The query of the available configurators (Mobile) and the WebFront start page (overview) it is available at any time without a password.

AUTOSTART FOR IP-ADDRESS

A special function is the autostart, which opens a configurator directly without a password request if the IP address matches one of the entered IP addresses in this list. This function can be used for example to open a WebFront directly on a touch panel without having to display the normal selection dialog and without having to answer the password prompt. It is important to remember to use this feature only within a secure network.

Since IP-Symcon 5.5 the CIDR notation is also supported. For example, 192.168.1.0/24 includes all addresses from 192.168.1.1 to 192.168.1.255.

This function is not available if IP-Symcon is accessed via a Reverse Proxy or the Connect Service as the origin cannot be verified securely. An exception is the CIDR value Wert 0.0.0.0/0, which forwards all requests, independently of the origin.

CALL VIA URL

There are some situations (e.g., in connection with NAT) when autostart via IP is not working. In these cases a URL with the WebFront password can be used.

```
// Basic framework for a call via URL
http://<YourIP>:3777/?password=<Password>#<WebFrontID>

// Example by starting with the Chromium Browser
// Password: "secret", WebFrontInstanceID: = 12345
chromium-browser --start-fullscreen 192.168.1.123:3777/?
password=secret#12345
```



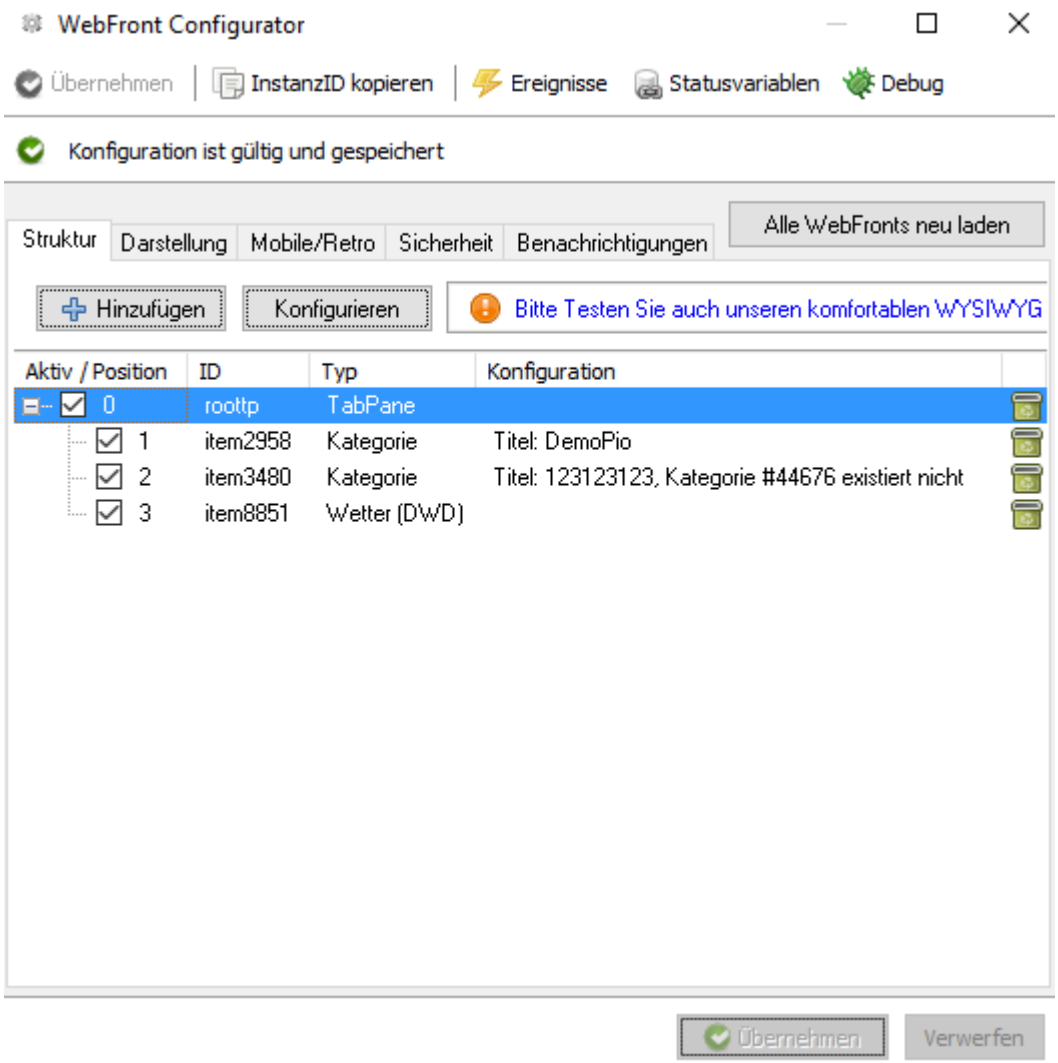
There is a security risk, because the URL is visible and includes the WebFront password as plain text.

Structure (deprecated)



This functionality is only avaiable until version 4.4 avaiable. It was replaced in total by the editor.

The configurator can contain **elements**, which start in the visualization as a tab from the top left margin and additionally contain **widgets**, which start in an extra area at the top right margin. This area is similar to the SysTray of Windows.
An element or widget can be inserted via the **Add** button and configured accordingly. An element consists of a unique ID that is automatically assigned, a position, and a configuration corresponding to the element.
The ID can be defined by the user when the same is created, so that the element can be better recognized during actions. The ID has no relation to the objectIDs of the instances or variables.
The position indicates the position in the tab or the widget bar and must have a value greater than or equal to zero. If two elements have the same position number, the order is defined by the title of the element.



OVERVIEW OF AVAILABLE ELEMENTS

ELEMENT	DESCRIPTION
Category	This element displays a category and all its sub-objects in the WebFront. The ID of the category to be displayed can be changed through the start category parameter.
Graph	With this element, the graphs set in the Archive Module are displayed directly. Normally, graphs can be called using the icon located at the left of the value of the variable. If graphs should be embedded directly, for example in TabPanes or SplitPanes, this can be done using the graph element.
External Page	This element inserts an external site in the WebFront via an IFrame. The URL parameter and interval are available.
Content Changer	The Content Changer allows you to switch multiple media files, strings, HTMLBoxes, StringBoxes, mail instances, or external sites within a single page.
SplitPane	Using the SplitPanes (formerly Container), the WebFront view can be split into two parts and any other element can be inserted in each part. Thus, if SplitPanes are inserted in SplitPanes, a QuadView can be simulated.
TabPane	TabPanes can be used to create custom menu bars at the upper area, which refers to other items.
Weather (DWD)	Displays the weather of the DWD with the ability to select specific areas of Germany and to switch between a normal image or a radar image.
Time Widget	Displays the current time.
Info Widget	Can either be used to display a static icon or linked to a variable which content / icon is displayed.
Logout Widget	This element adds a widget with which you can return to the WebFronts selection screen.
Idle Widget	This element adds an invisible widget, which switches to a specific page after a period of inactivity.

Mobile

The mobile visualization for mobile devices can be downloaded from the respective AppStore. There, the latest description of the functions of the app can be found as well. (**Android:** (link: https://play.google.com/store/apps/details?id=de.symcon.visualization text: IP Symcon Mobile on Google Play) / **iOS:** (link: https://apps.apple.com/us/app/symcon-visualization/id1566124129 text: IP Symcon Mobile for iPhone, iPod touch, and iPad in the iTunes App Store))
This page is intended to be a description and reference for frequently asked questions and functions that differ between the apps. Furthermore, no app is eligible for full compatibility with the WebFront. These serve as an extension to IP Symcon for quick and easy access to the basic functions of the system. For a complete visualization the WebFront can and must be used. No change is planned here as the mobile applications would become slower with ever-increasing functions. But that would collide with the main feature "speed".
The documentation for the push messages can be found in the section Notification Control



It should be noted that access to the system via the mobile apps is deactivated by default. It is indicated by the error message "No valid configurator!". In the desired WebFront Configurator mobile use must be activated.

FUNCTIONS THAT ARE INTENTIONALLY UNAVAILABLE

User-defined icons are not loaded, as this would mean a considerable loss of speed, which would be further decreased as several icon sizes would be required, for example, for the Retina display, which only few users have at hand.
Multiple nesting is not available. The aim is to make optimum use of the display area on mobile devices.

FUNCTIONS THAT DIFFER FROM THE WEBFRONT

Boolean-variables are shown as on/off (switch, checkbox) by default. This can be managed in through the WebFront visualization instance, so only ~Switch profils have this representation.

WEBFRONT RELEASE LINK

The app can be started directly and linked to a WebFront via WebFront release links.
If the corresponding WebFront does not exist in the app, a configurator will be set up automatically.
There are 2 types of links:
For servers without SSL: **symcon://www.webfront.info/#98765**
For servers with SSL: **symcons://my.ssl-server.com/#12345**
If the server/configurator is not in the app list, it will be added and opened automatically.
Thus, for example, without manual settings being necessary, the visitor at home or even the conference participant can be given quicker access to the local WebFront.

KNOWN BUGS

When SSL is enabled, externally loaded content (such as images) can not be loaded if the certificate is not accepted by the browser.
Certain profiles and status/variables (for example HomeMatic Window Contacts or LCN Relays) can be switched in the Mobile App. This possibility of the action is displayed in color (green/red) in the WebFront and means that either an action script is defined or that a standard action is available. If the behavior is to be changed, this topic gives the corresponding answer in the forum: Changeability of Fixed Variables

HINTS

If a JSON error occurs while loading, check if at least IP-Symcon 3.1 #3367 has been installed.

Legacy Icons



The WebFront uses the Legacy Icons and not the usual icons of the tile visualization.

ICON TYPE

It is differentiated between ‘normal’ and ‘adaptive’ icons. Normal icons do not change their appearance, whereas adaptive icons adapt their appearance to the set variable, depending on the value.
In this way, adaptive icons orientate themselves around the min/max values set in the profile. These are handled as 0-100%. Should it be handled as 100-0%, the term ‘.Reversed’ must be entered into the profile name.
Individual adaptive icons (e.g. Intensity-25), can also be entered directly and are then integrated as a ‘normal’ icon.

CUSTOMIZED ICONS INCLUDED

From version IP-Symcon 3.0, customized icons are integrated via skins.
Further information can be found in the developer’s area as SDK: Skins SDK

SELECTION IN THE MANAGEMENT CONSOLE

The icon selection in the management console is optimised for the current icons. Still, the legacy icons can be chosen by entering the icon name manually into the corresponding field.

ICON OVERVIEW

The following icons are contained in WebFront and can be selected at the relevant locations in IP-Symcon. A special case is the icon **Transparent**. If that icon is selected, other icons of lower in rank (see above) are not displayed. This style element can be used in order to, e.g., not display an icon of a linked category object, even though the category has an icon.

ICONS

Aircraft

Alert

ArrowRight

Backspace

Basement

Bath

Battery

Bed

Bike

Book

Bulb

Calendar

Camera

Car

Caret

Cat

Climate

Clock

Close

Closeall

Cloud

Cloudy

Cocktail

Cross

Database

Dining

Distance

DoctorBag

Dog

Doll

Dollar

Door

Download

Drops

Duck

Edit

Electricity

EnergyProduction

EnergySolar

EnergyStorage

ErlenmeyerFlask

Euro

Execute

Eyes

Factory

Favorite

Female

Fitness

Flag

Flame

FloorLamp

Flower

Fog

Garage

Gas

Gauge

Gear

Graph

GroundFloor

Handicap

Heart

Help

HollowArrowDown

HollowArrowLeft

HollowArrowRight

HollowArrowUp

HollowDoubleArrowDown

HollowDoubleArrowLeft

HollowDoubleArrowRight

HollowDoubleArrowUp

HollowLargeArrowDown

HollowLargeArrowLeft

HollowLargeArrowRight

HollowLargeArrowUp

Hourglass

HouseRemote

Image

Information

Intensity

Internet

Ips

Jalousie

Key

Keyboard

Kitchen

Leaf

Light

Lightning

Link

Lock

LockClosed

LockOpen

Macro

Mail

Male

Melody

Menu

Minus

Mobile

Moon

Motion

Move

Music

Network

Notebook

Ok

Pacifier

Paintbrush

Pants

Party

People

Plug

Plus

Popcorn

Power

Presence

Radiator

Raffstore

Rainfall

Recycling

Remote

Repeat

Return

Robot

Rocket

Script

Shift

Shower

Shuffle

Shutter

Sink

Sleep

Sleet

Snow

Snowflake

Sofa

Speaker

Speedo

Stars

Sun

Sunny

TV

Talk

Tap

Teddy

Tee

Telephone

Temperature

Thunder

Title

TopFloor

Tree

TurnLeft

TurnRight

Umbrella

Unicorn

Ventilation

WC

Warning	**Wave**
Wellness	**WindDirection**
WindSpeed	**Window**
XBMC	

ADAPTIVE ICONS

Battery	**0%**	**50%**	**100%**			
Door	**0%**	**100%**				
EnergyStorage	**0%**	**25%**	**50%**	**75%**	**100%**	
Eyes	**0%**	**100%**				
FloorLamp	**0%**	**100%**				
Garage	**0%**	**25%**	**100%**			
Hourglass	**0%**	**30%**	**60%**	**100%**		
Intensity	**0%**	**25%**	**50%**	**75%**	**100%**	
Jalousie	**0%**	**50%**	**100%**			
Light	**0%**	**1%**	**25%**	**50%**	**75%**	**100%**
Lock	**0%**	**100%**				
Presence	**0%**	**100%**				
Raffstore	**0%**	**50%**	**100%**			
Speaker	**0%**	**1%**	**25%**	**50%**	**100%**	
Speedo	**0%**	**25%**	**50%**	**75%**	**100%**	
Temperature	**0%**	**25%**	**50%**	**75%**	**100%**	
WindSpeed	**0%**	**30%**	**60%**	**100%**		
Window	**0%**	**100%**				

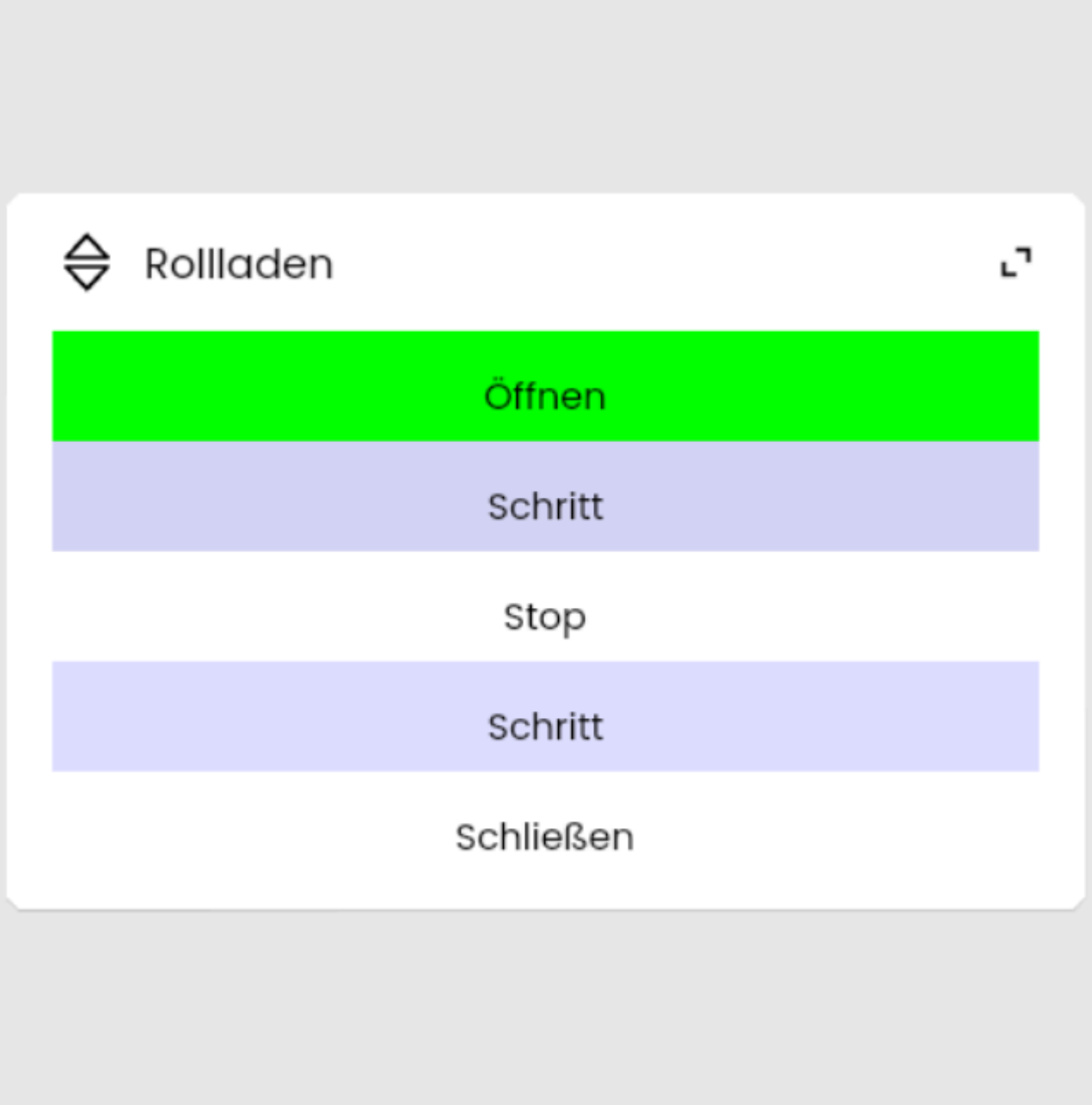


The following icons are only available from IP-Symcon 5.0/Symcon Mobile 5.0: Bath, Bike, Book, Cloudy, Doll, Door, Download, EnergyProduction, EnergySolar, EnergyStorage, Favorite, Fitness, FloorLamp, Gas, Handicap, Heart, Help, Link, Menu, Pants, Party, People, Presence, Raffstore, Remote, Sink, Sleet, Sunny, Teddy, Tee, Thunder, Umbrella

Object Presentation

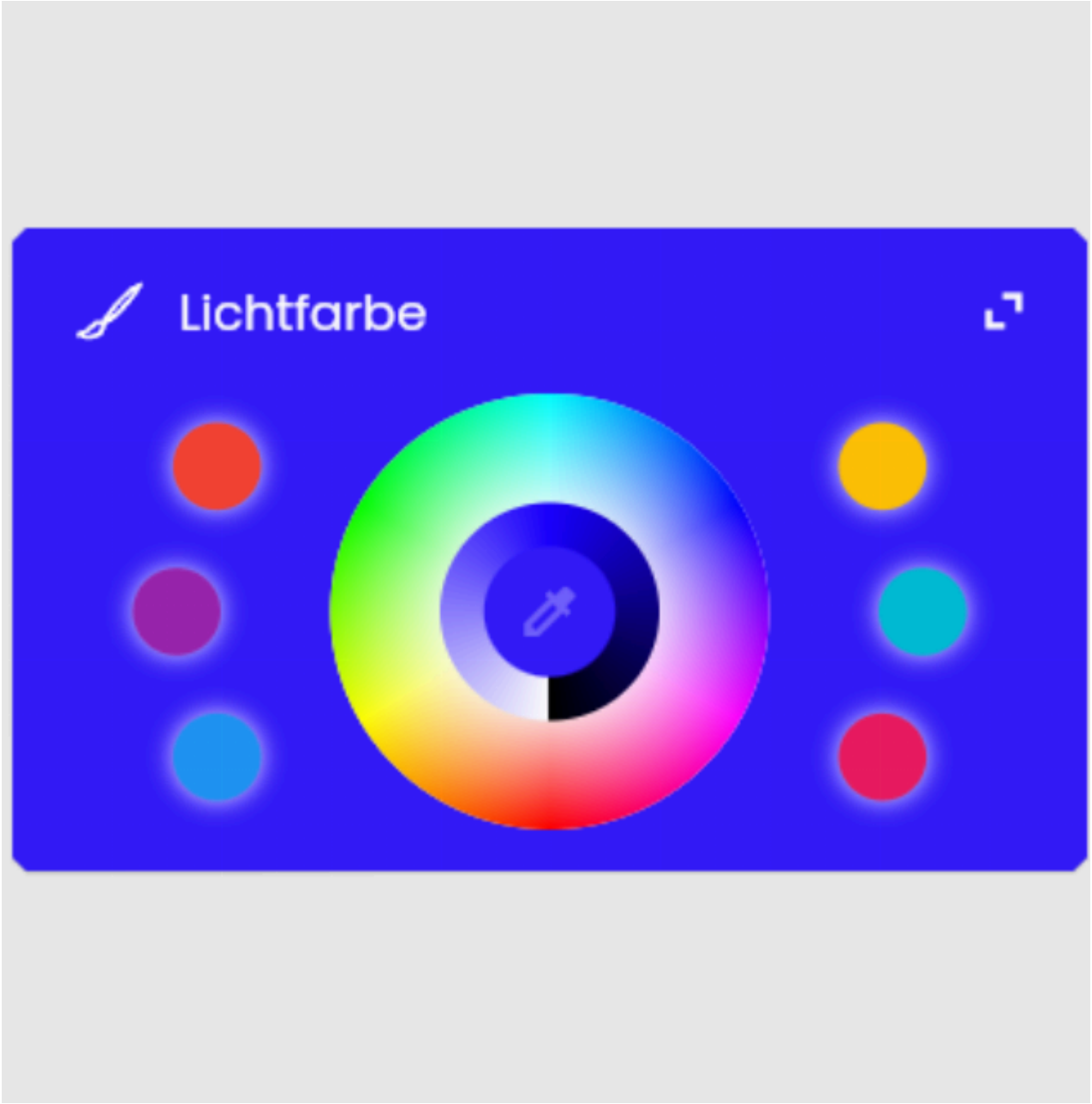
ACTIVE ELEMENTS

Individual switchable elements can be presented in different ways.



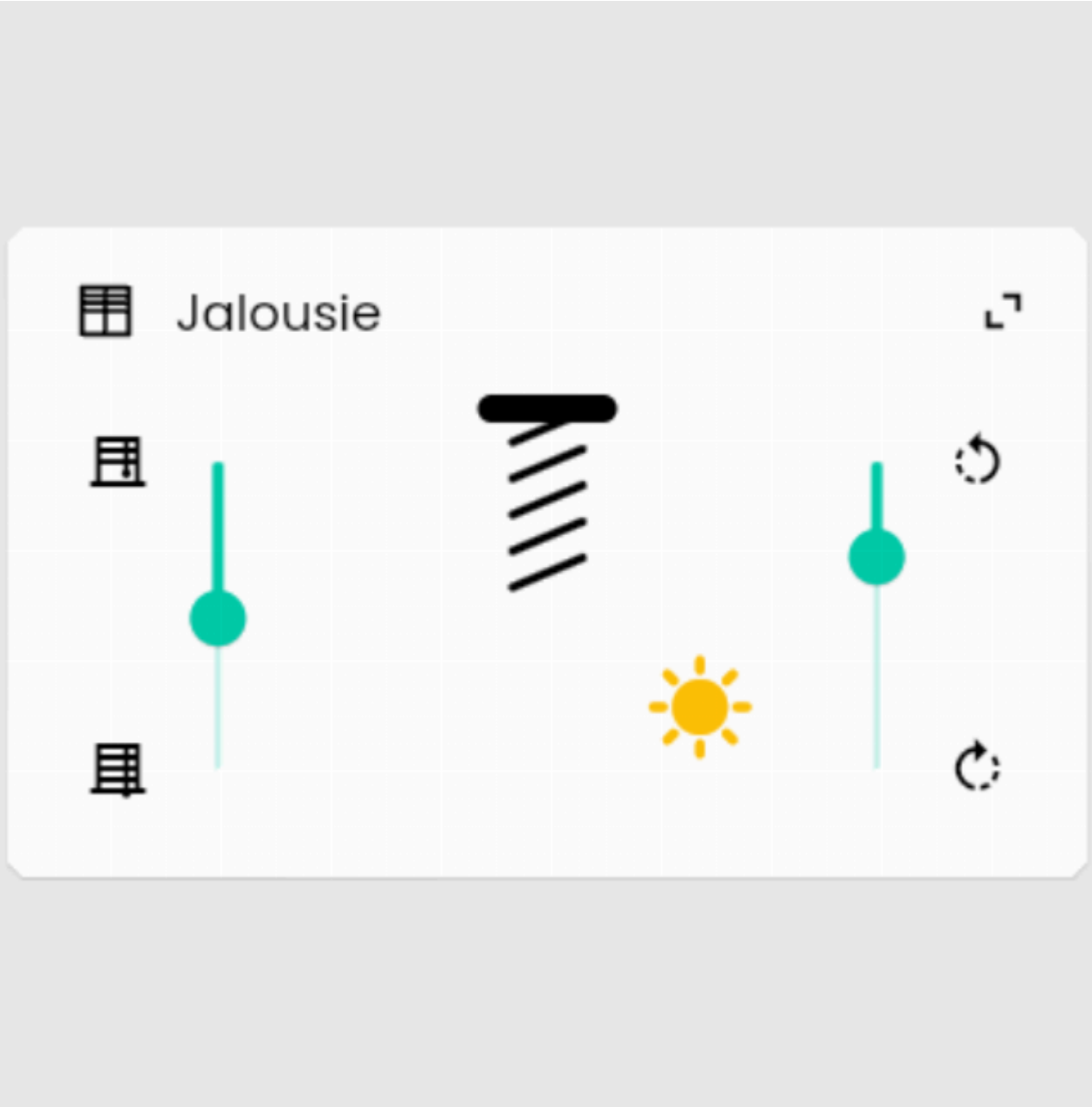
ENUMERATION

An Enumeration is a list of values that can be activated.



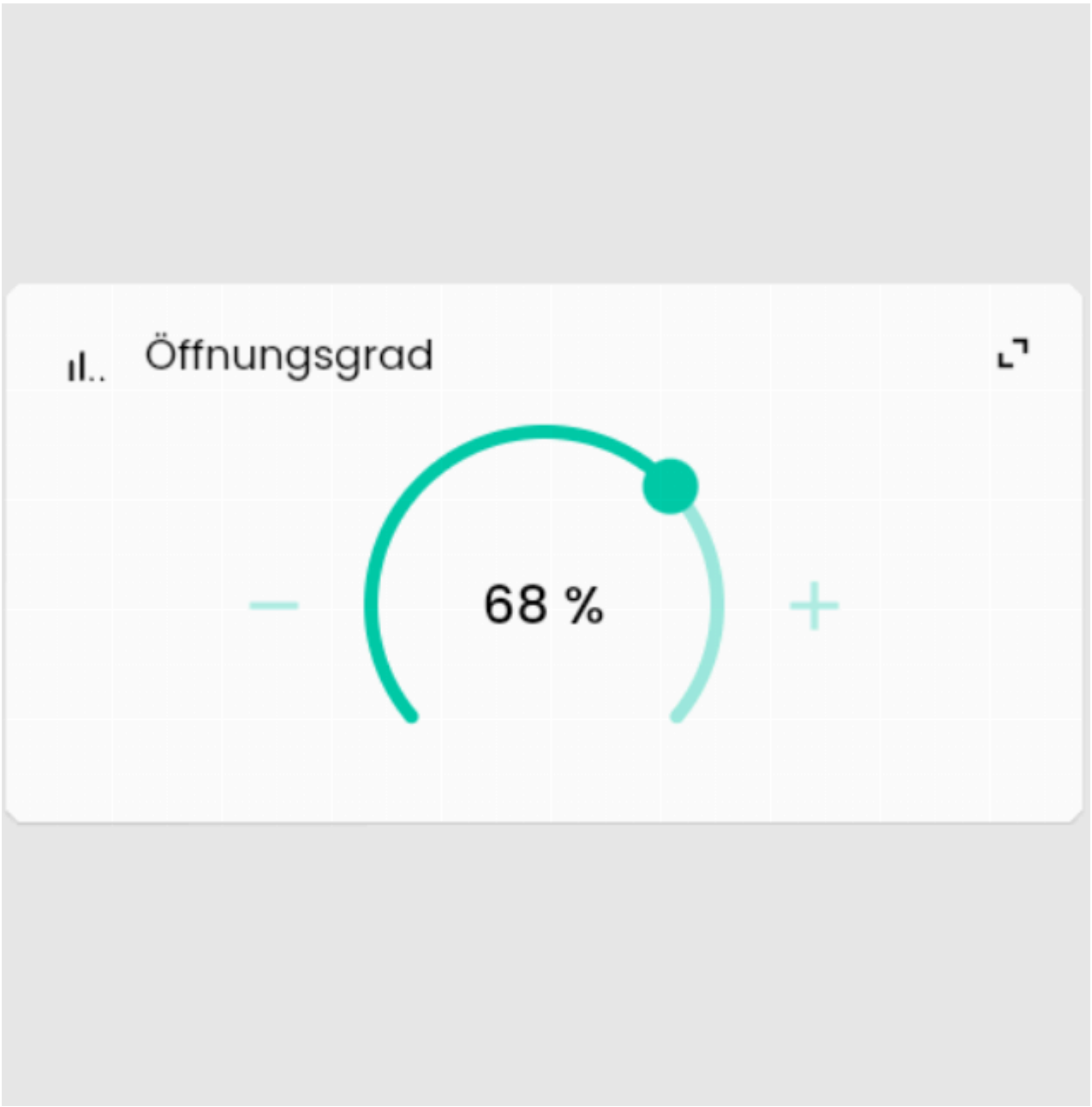
COLOR SELECTION

A color can be displayed and configured with a Color Selection.



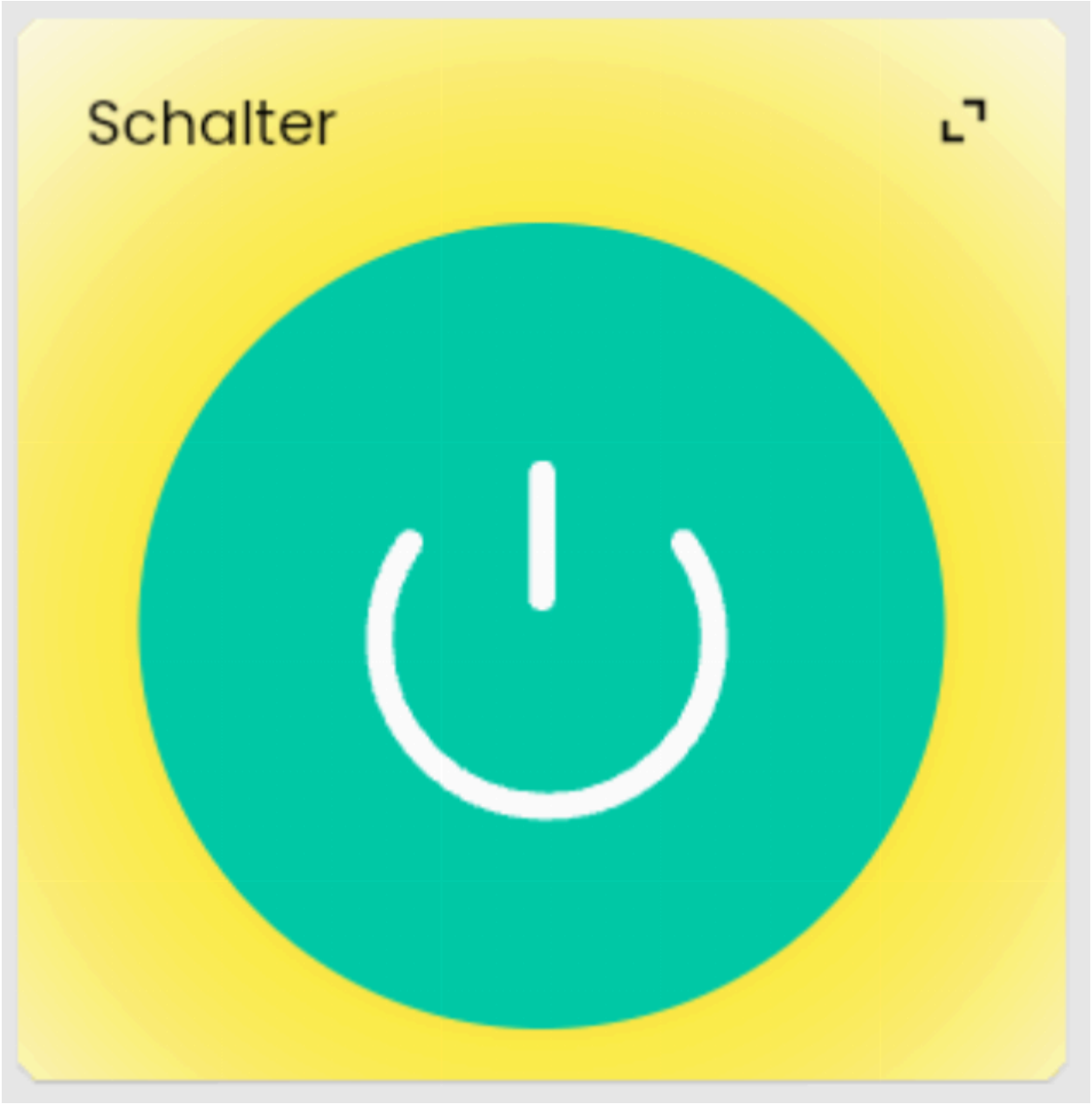
SHUTTER

A Shutter can set the degree of opening or inclination of the slats.



SLIDER

A value can be checked and configured with a Slider.



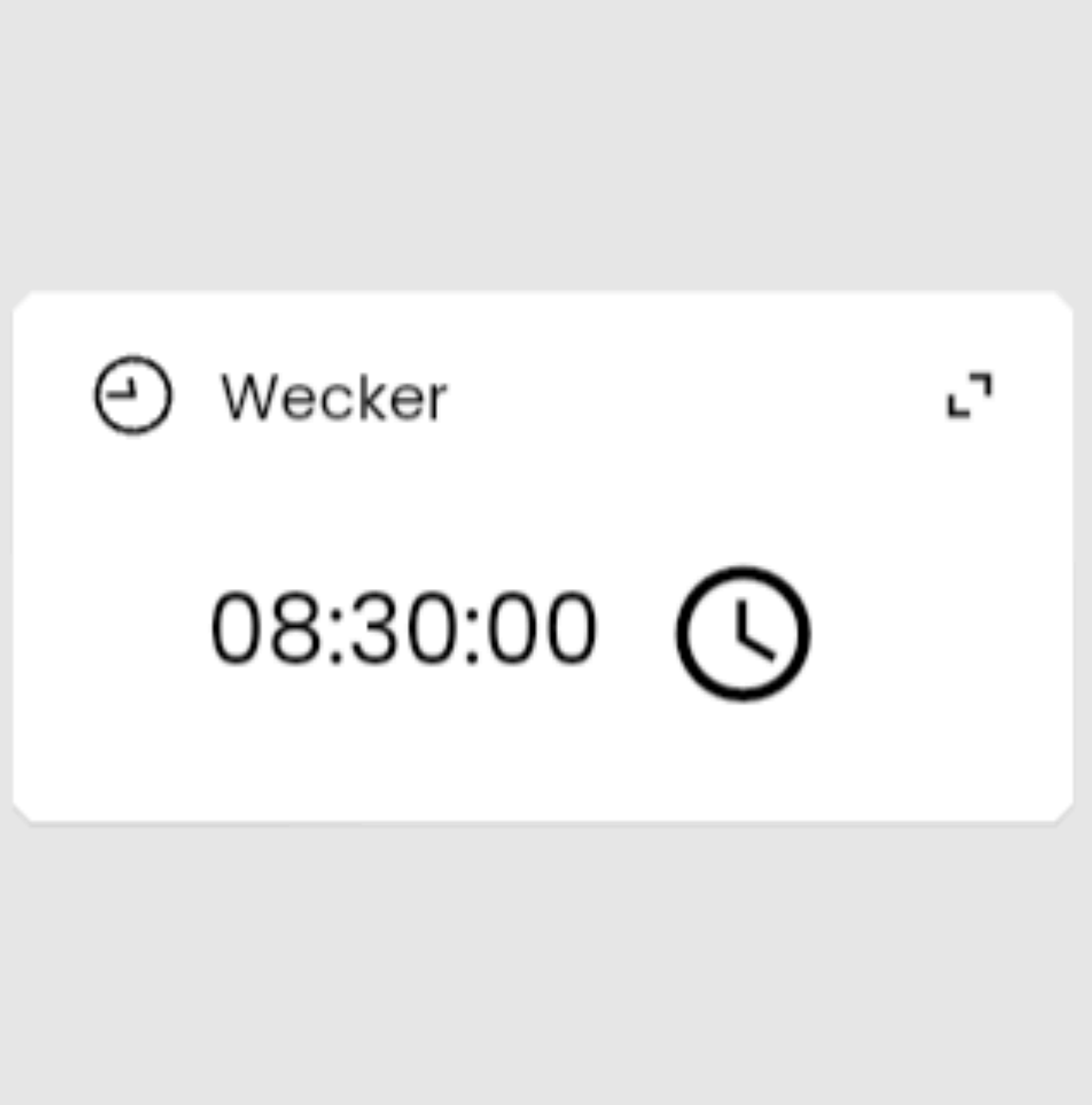
SWITCH

A Switch can be switched on (true) or off (false).



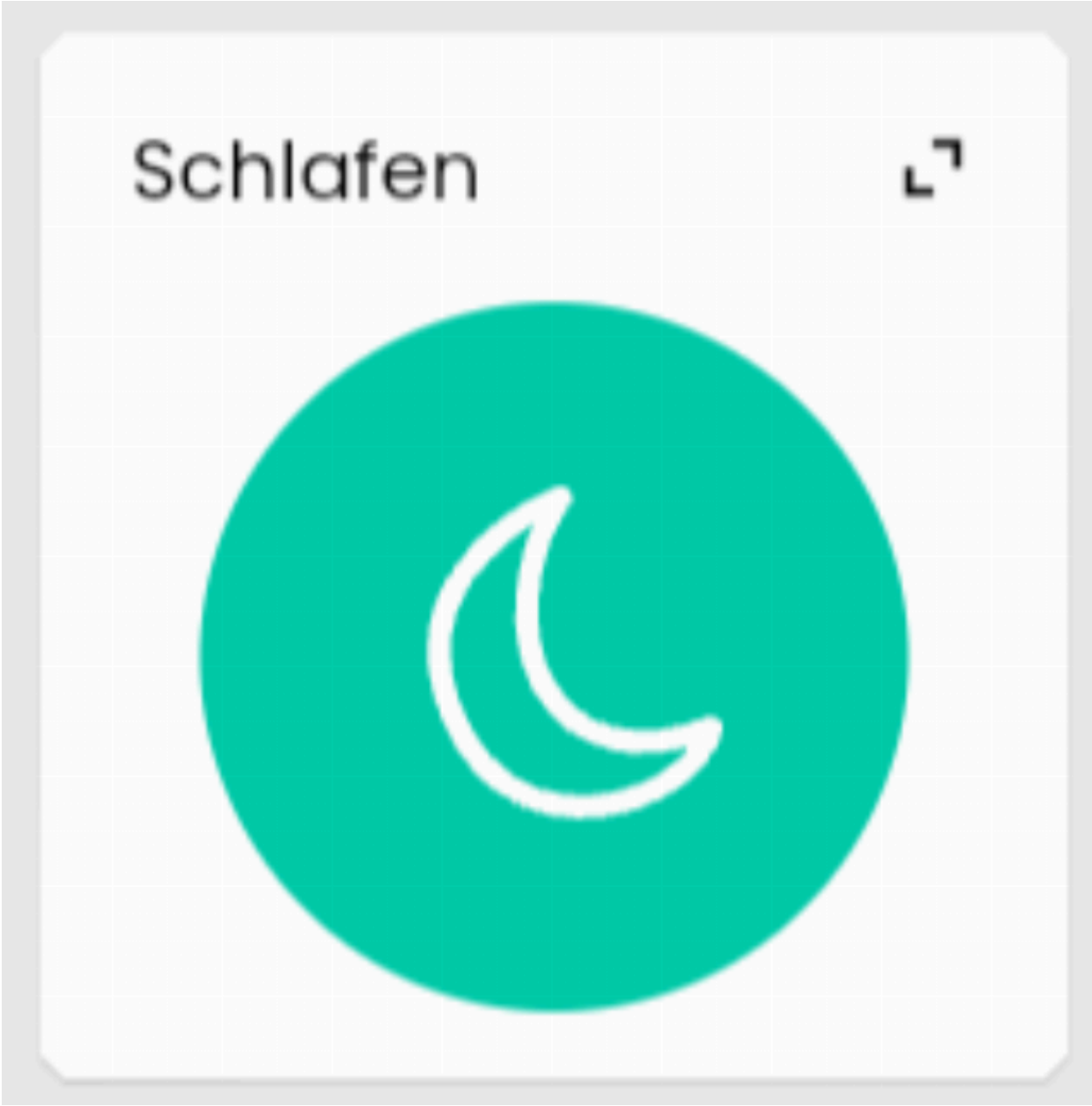
VALUE INPUT

A Value Input allows you to change values as free text.



DATE/TIME

With Date/Time, a point in time can be displayed and adjusted if necessary.

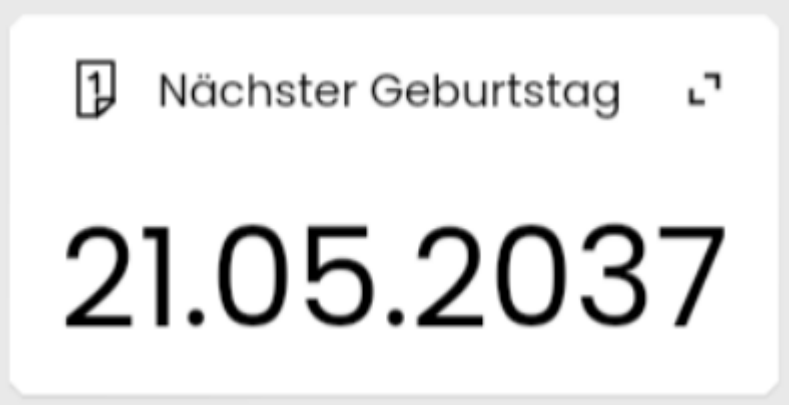


AUTOMATION

Automations can be executed via a button.

PASSIVE ELEMENTS

Individual elements that cannot be switched have different presentations.



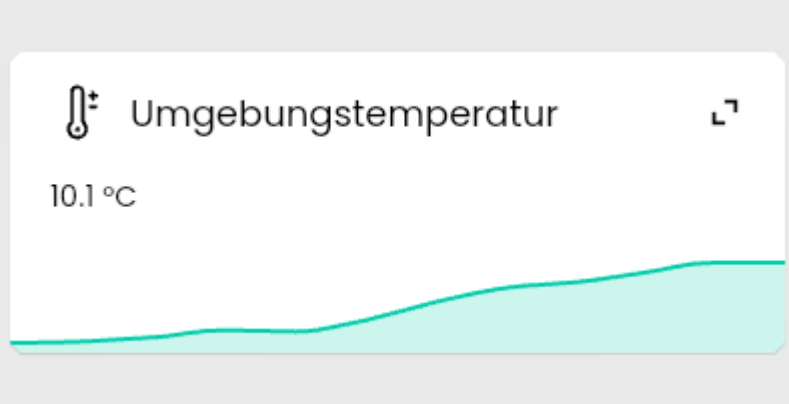
DATE/TIME

With Date/Time, a point in time can be displayed.



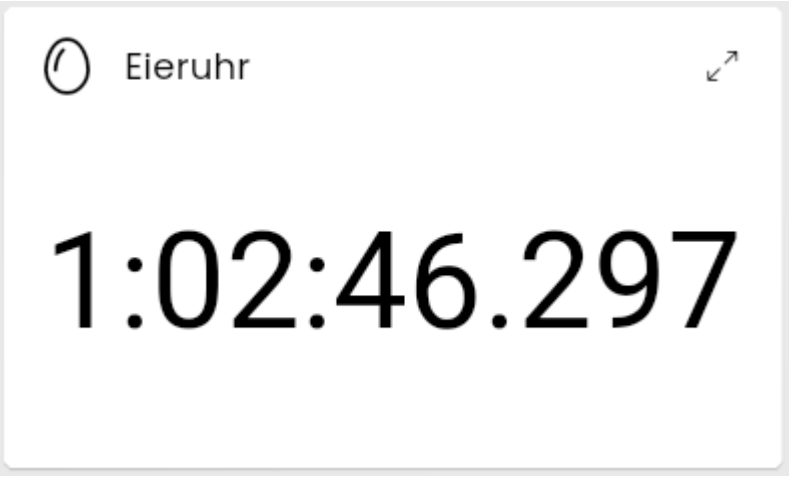
COLOR PRESENTATION

A Color Presentation displays a color.



VALUE PRESENTATION

Displays a value, optionally with background color or value progression.

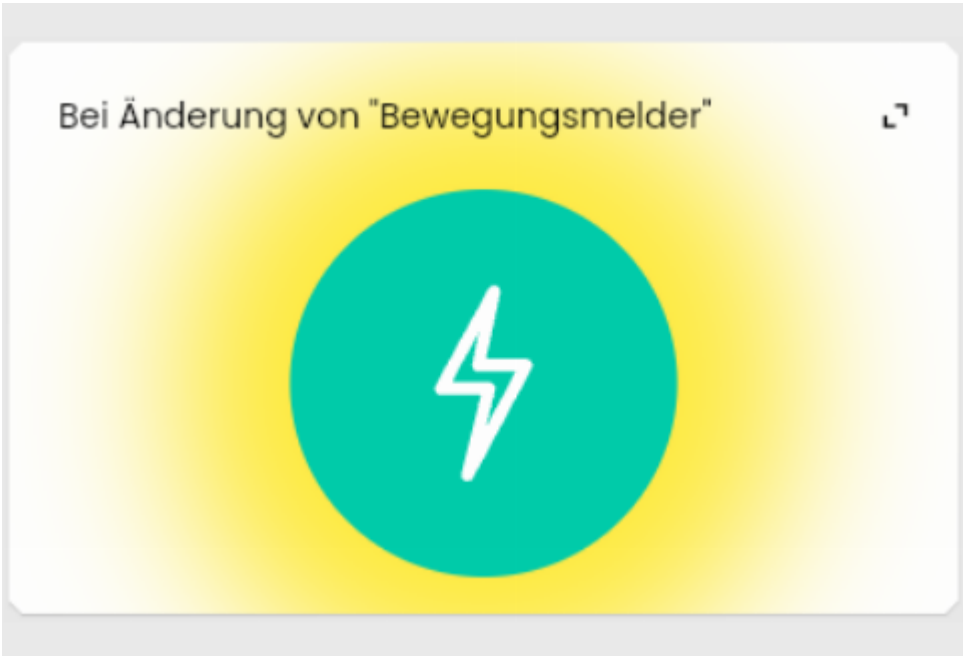


DURATION

Displays the remaining time until a point in time.

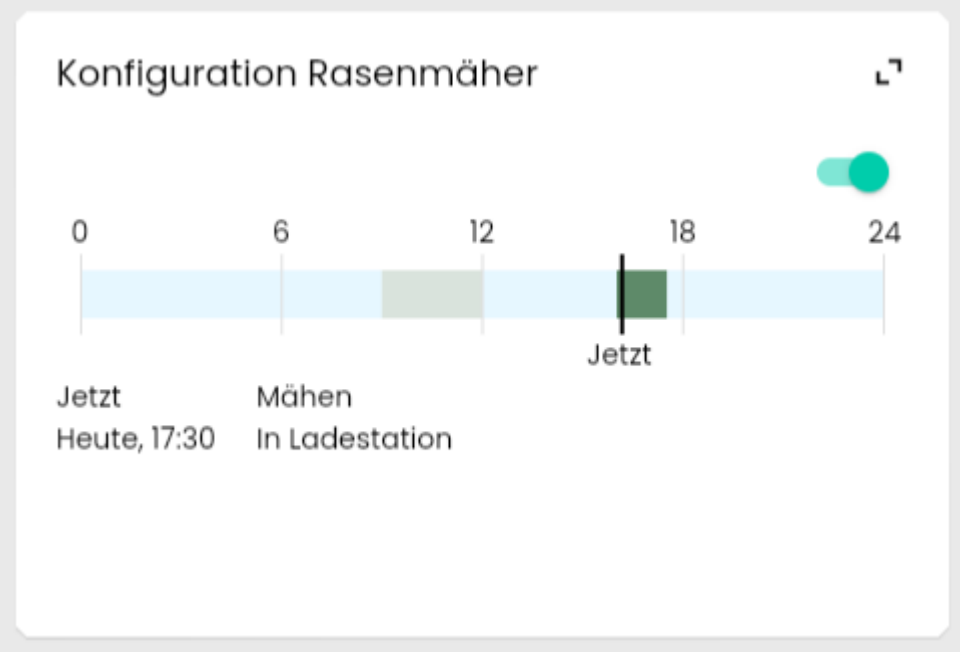
EVENTS

Allows the execution of logic, depending on date and time or the update of other objects.



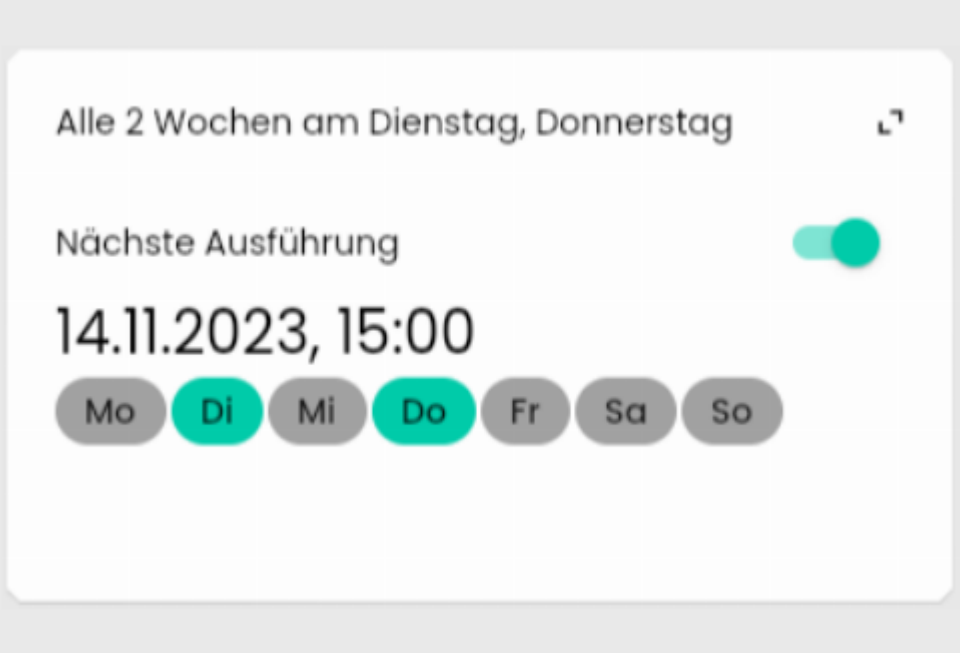
TRIGGER EVENT

A triggered event can be activated or deactivated in the visualization.



SCHEDULE EVENT

A Schedule Event can be viewed and configured in the visualization.



CYCLIC EVENT

A Cyclic Event can be viewed and configured in the visualization.

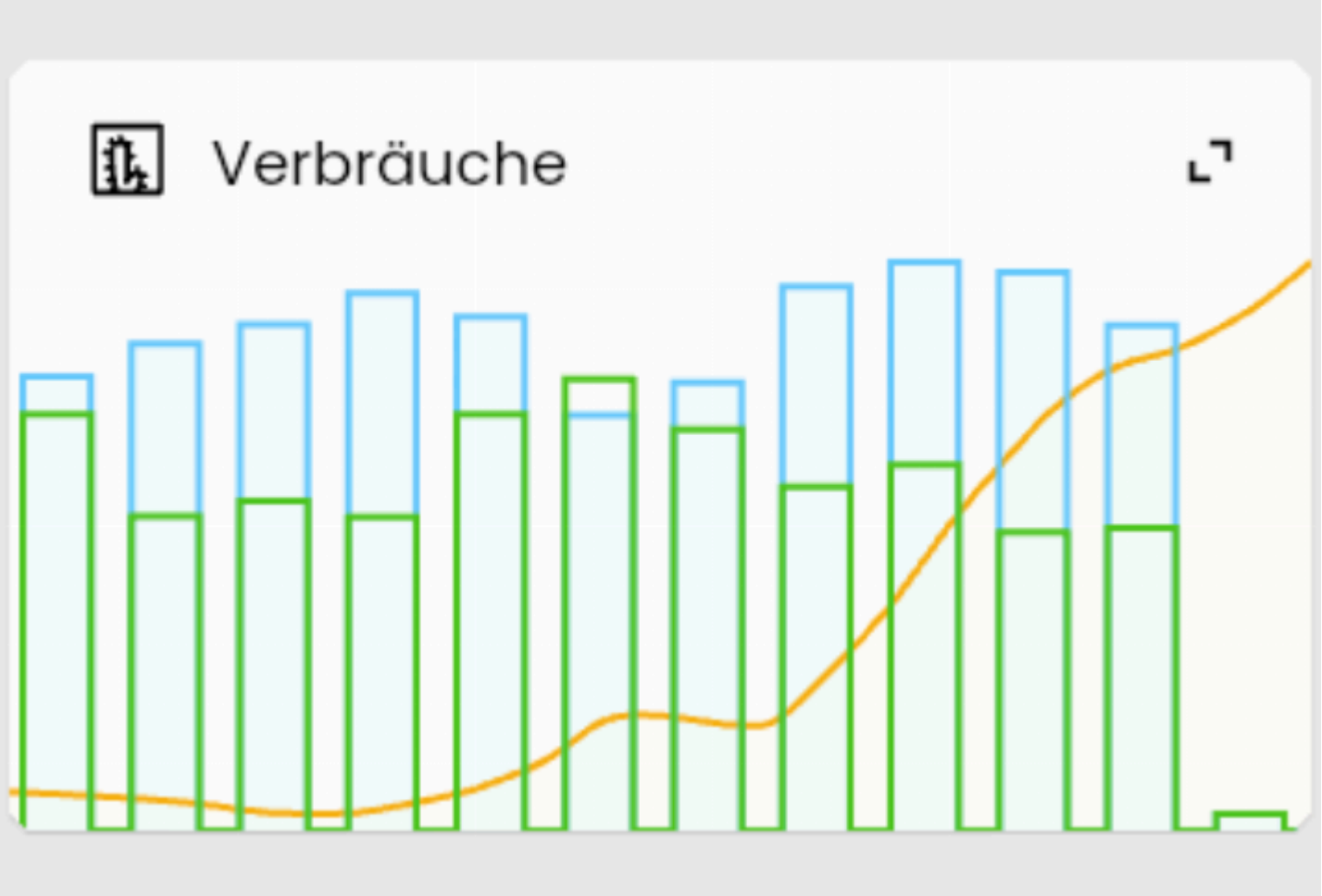
EXTERNAL ELEMENTS

External files like **Media** can be displayed within the visualization.



IMAGE

Any images can be integrated into the visualization.



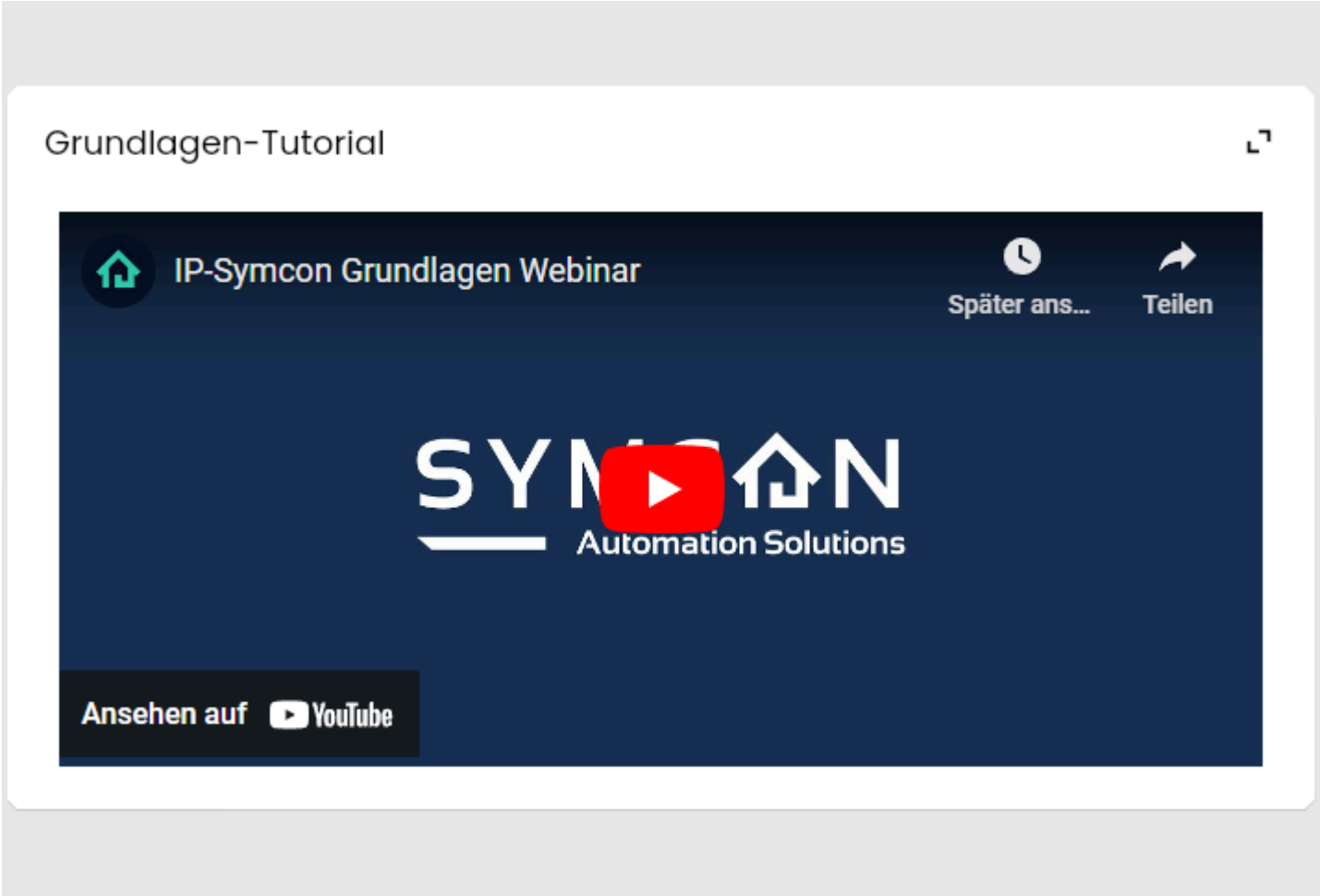
CHART

The course of several variables can be displayed in a Chart.



DOCUMENT

A Document can be downloaded by selecting the button.



WEB CONTENT

Web Content offers experts the possibility to display any HTML content or embed a website.



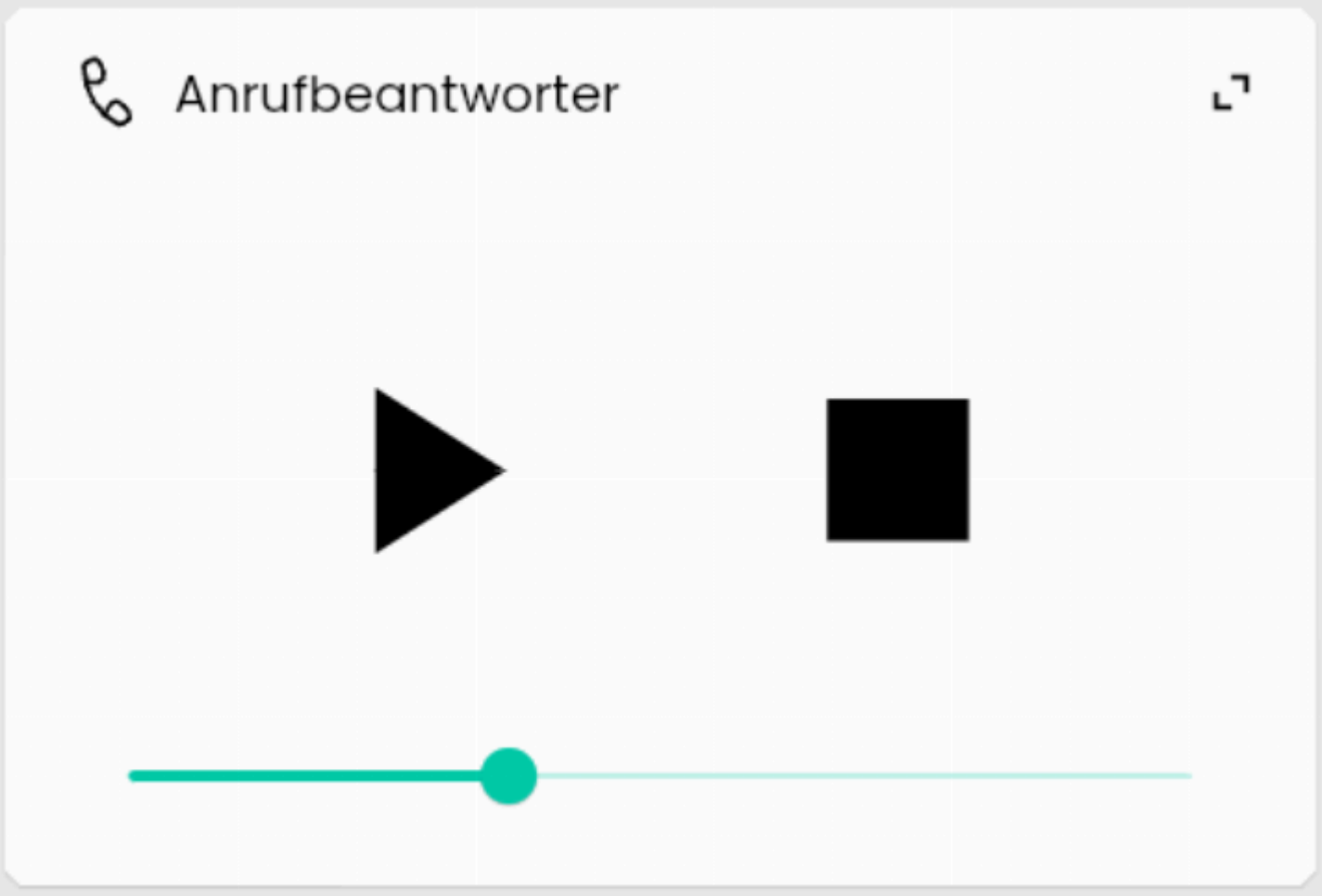
IPSVIEW

IPSViews can be seamlessly integrated into the visualization.



STREAM

Streams represent the video output, for example from a surveillance camera.



SOUND

Audio files can be integrated and played back in the visualization.

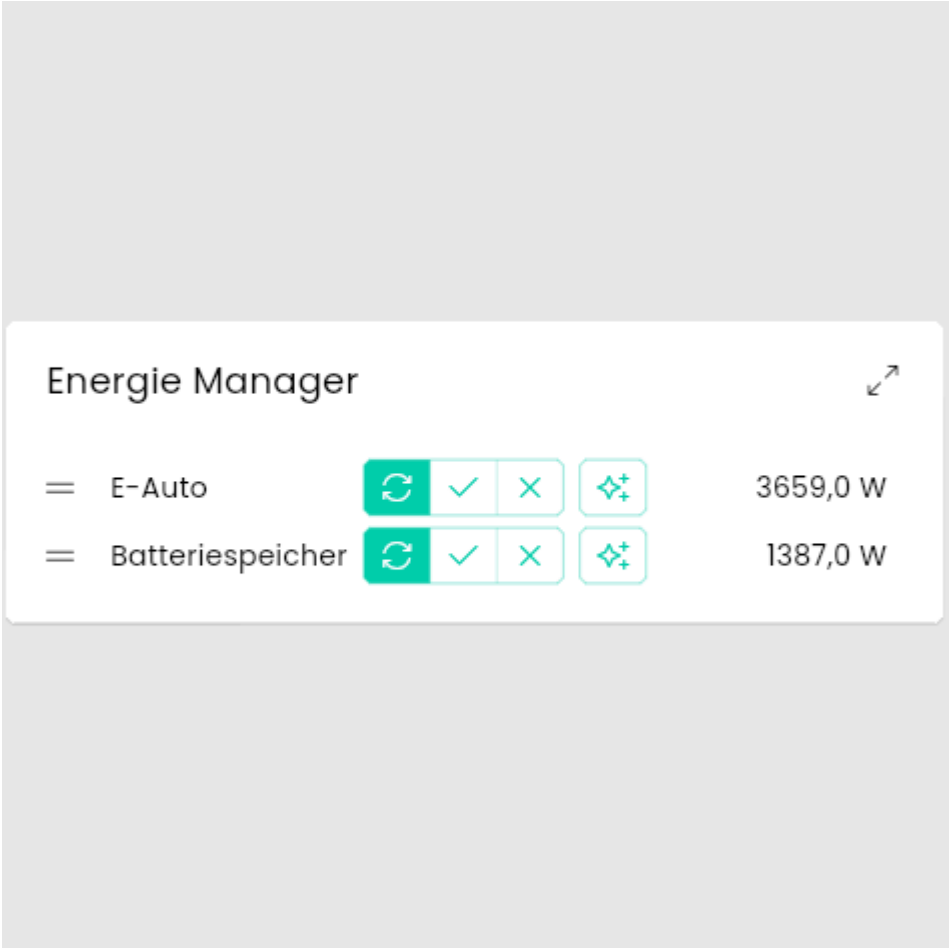
SUMMARIZED PRESENTATIONS

Multiple elements can be summarized intelligently to a single presentation.



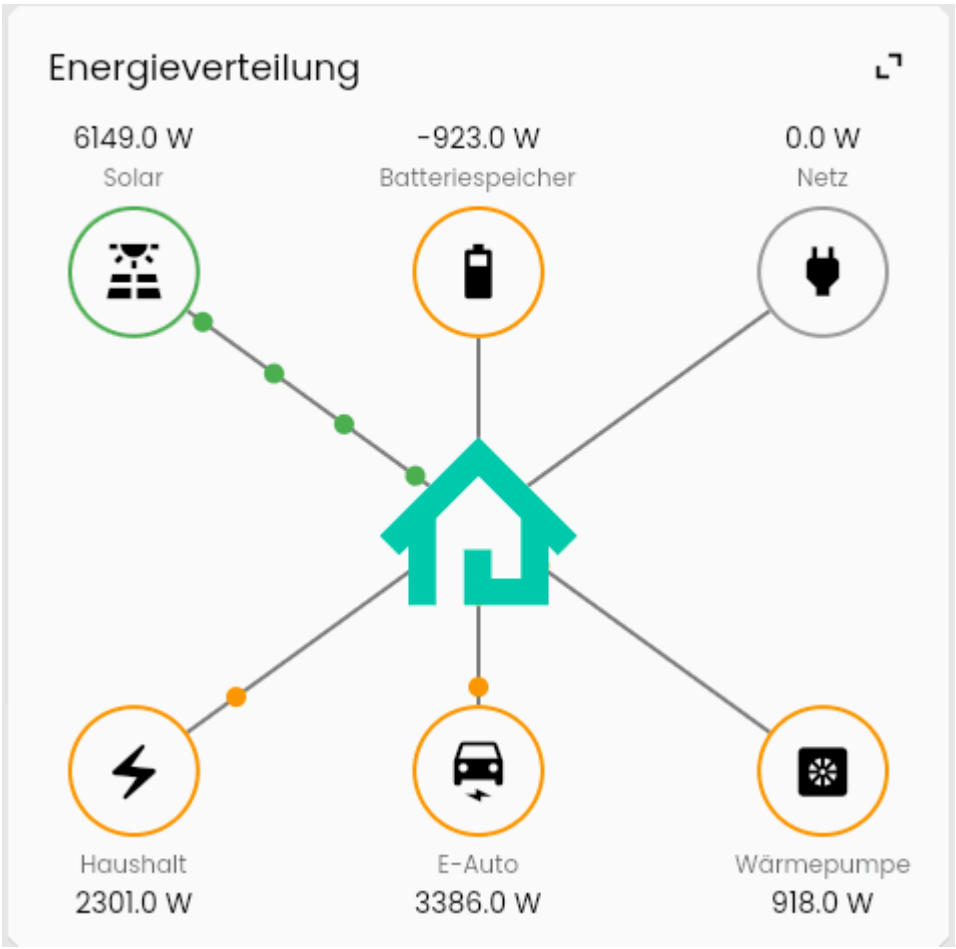
SINGLE ELEMENT

Enables the display of an element from a List as a single tile.



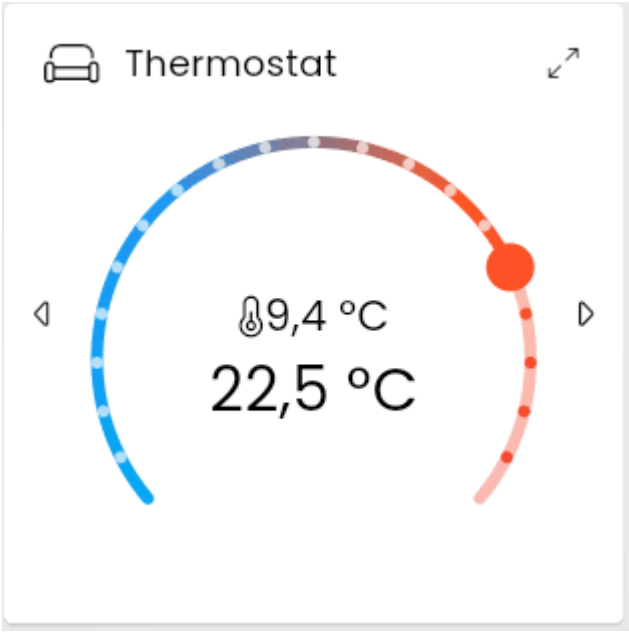
ENERGY MANAGER

The Energy Manager switches devices depending on available energy.



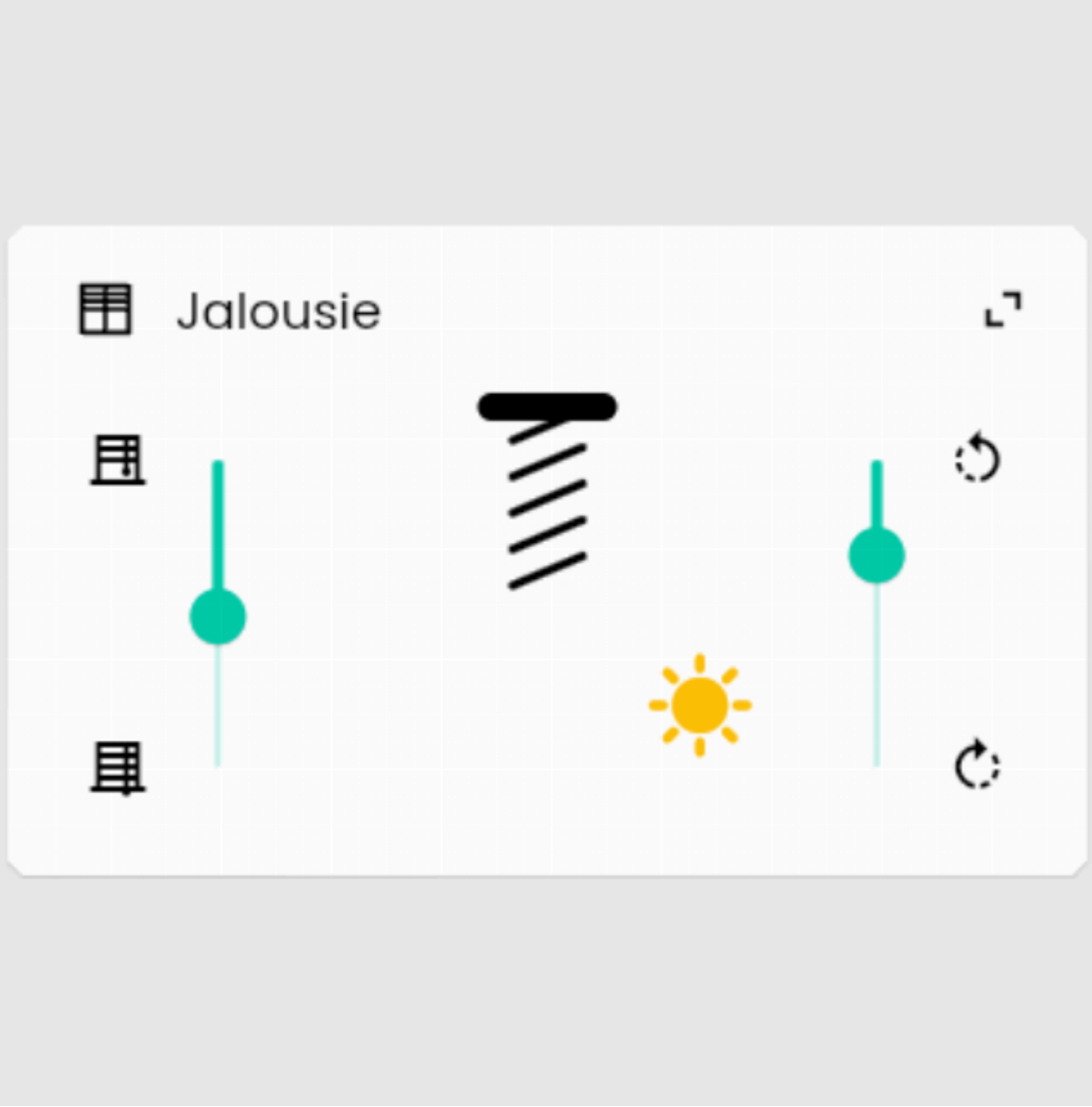
ENERGY DISTRIBUTION

The Energy Distribution shows the flow of generated and consumed energy.



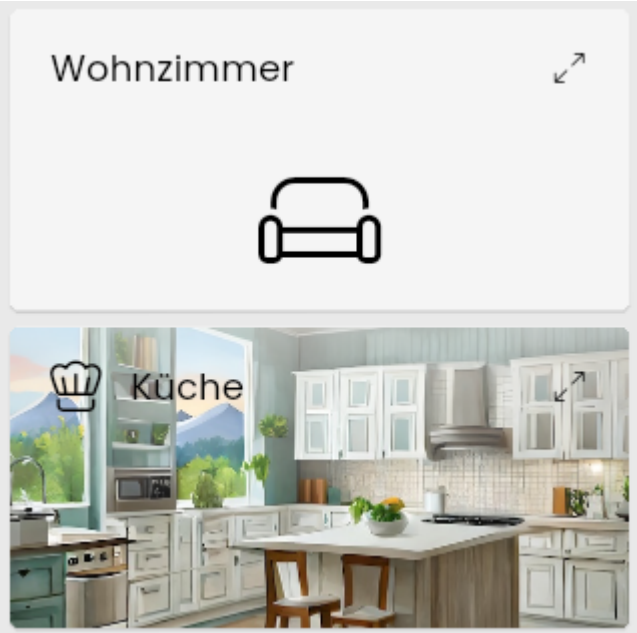
CONTENT CHANGER

Allows to switch between multiple elements in one tile



SHUTTER

For a Shutter, the degree of opening and optionally the degree of inclination of the slats can be set.



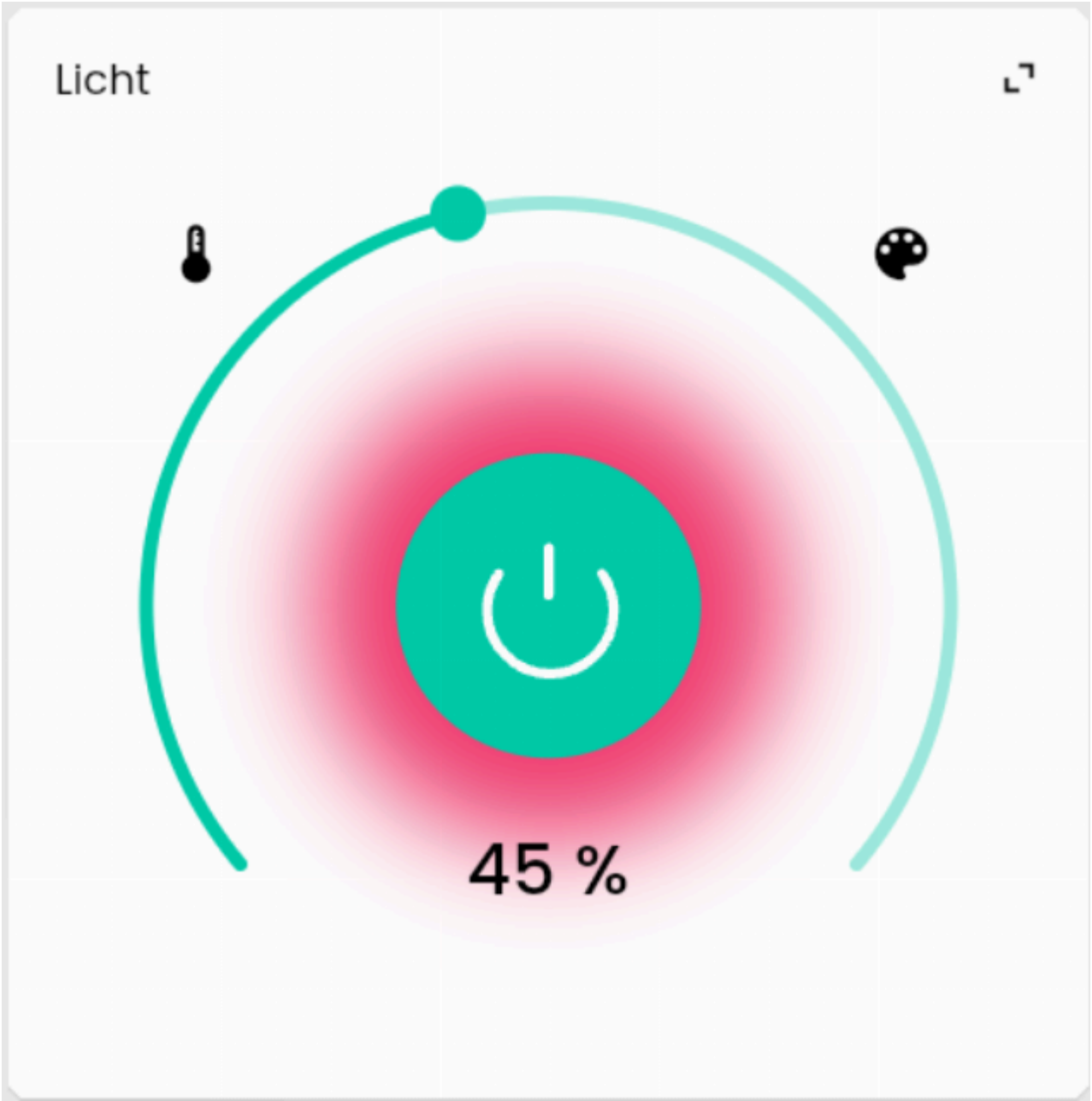
CATEGORY

A Category can be opened to view the objects it contains.



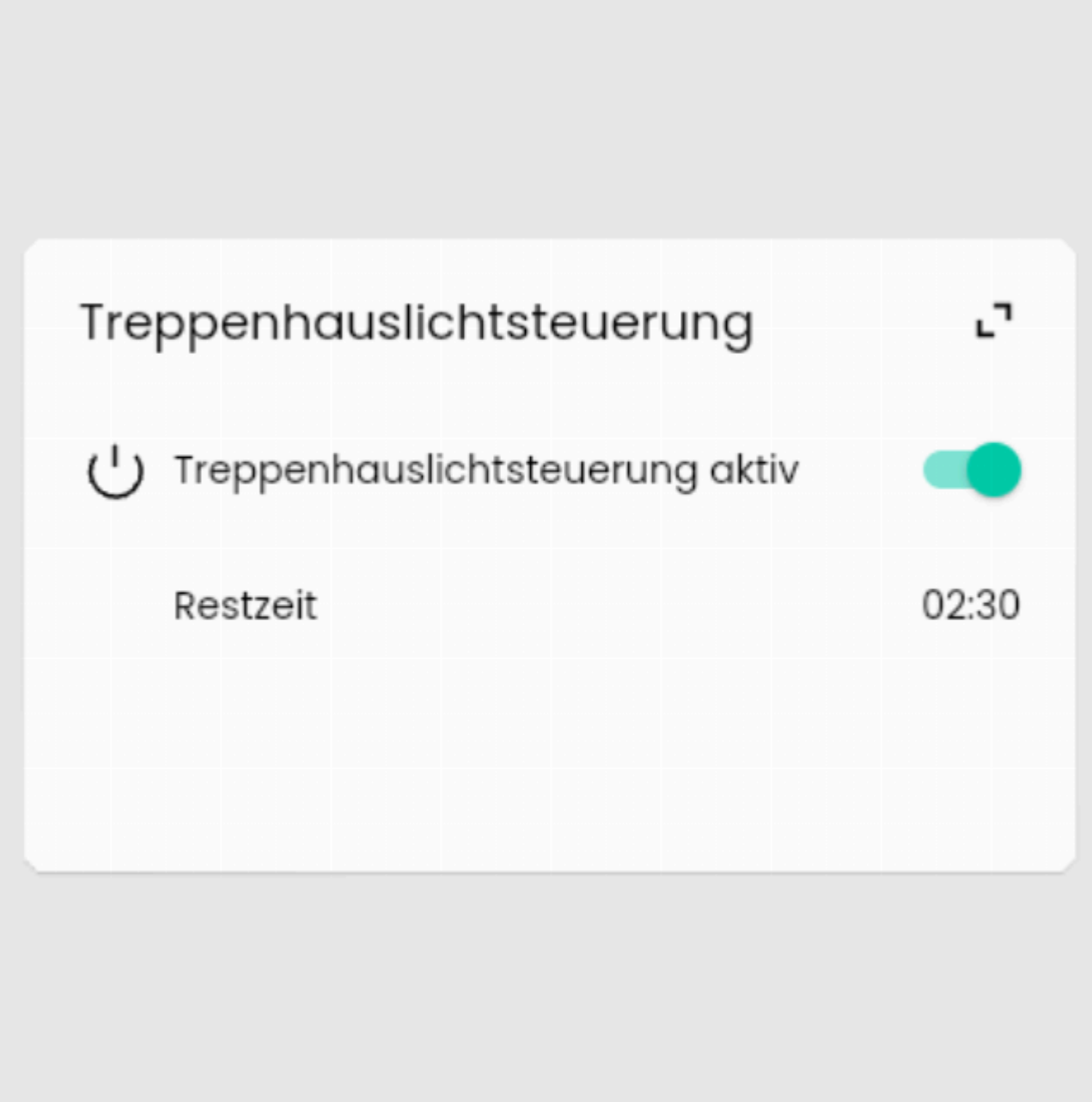
CATEGORY BAR

Shows categories in a bar at the top of the screen



LIGHT

In the presentation Light, status, brightness, tuneable white and color can be visualized in a single tile.



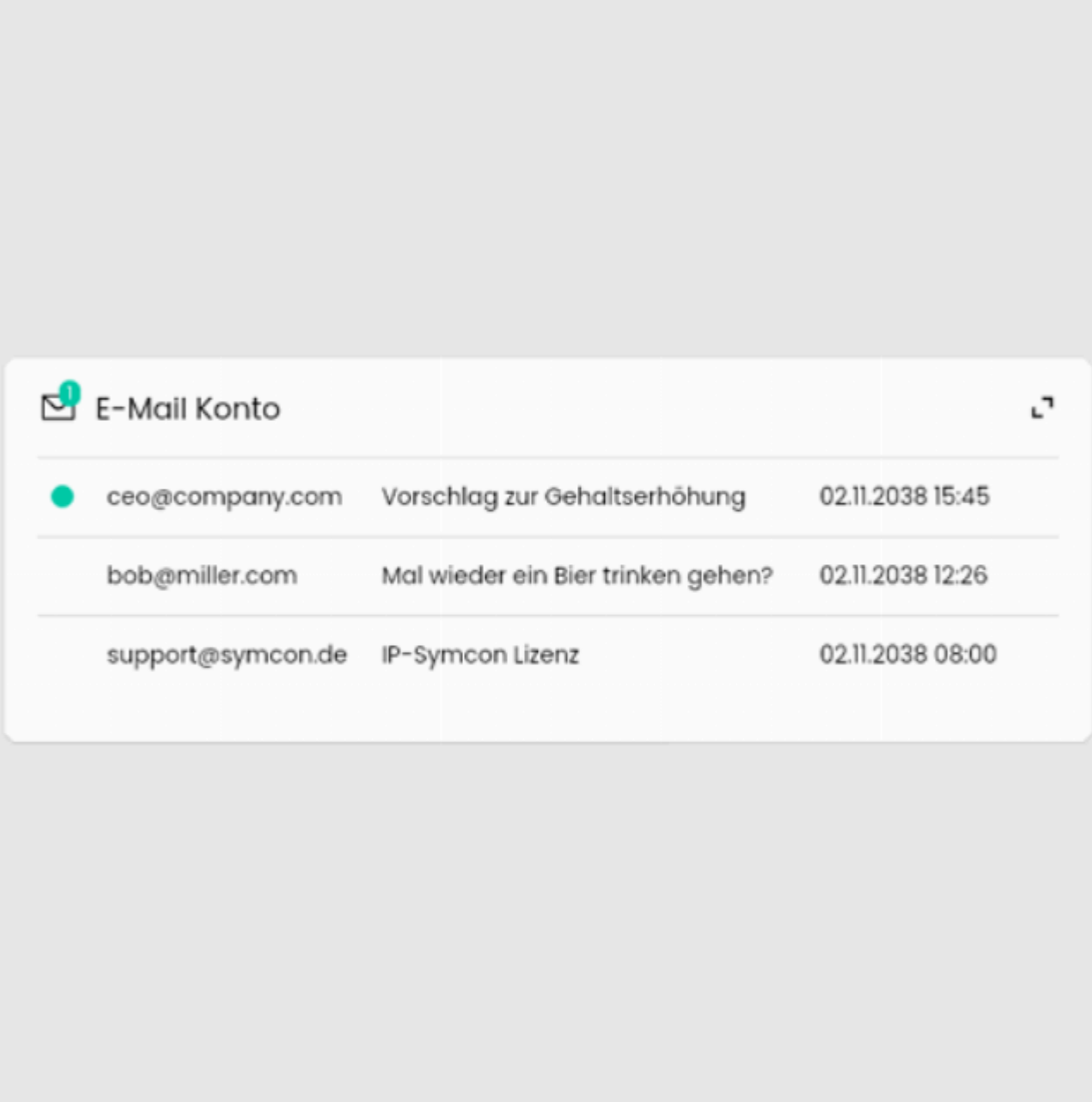
LIST

A list view can display any objects one below the other.



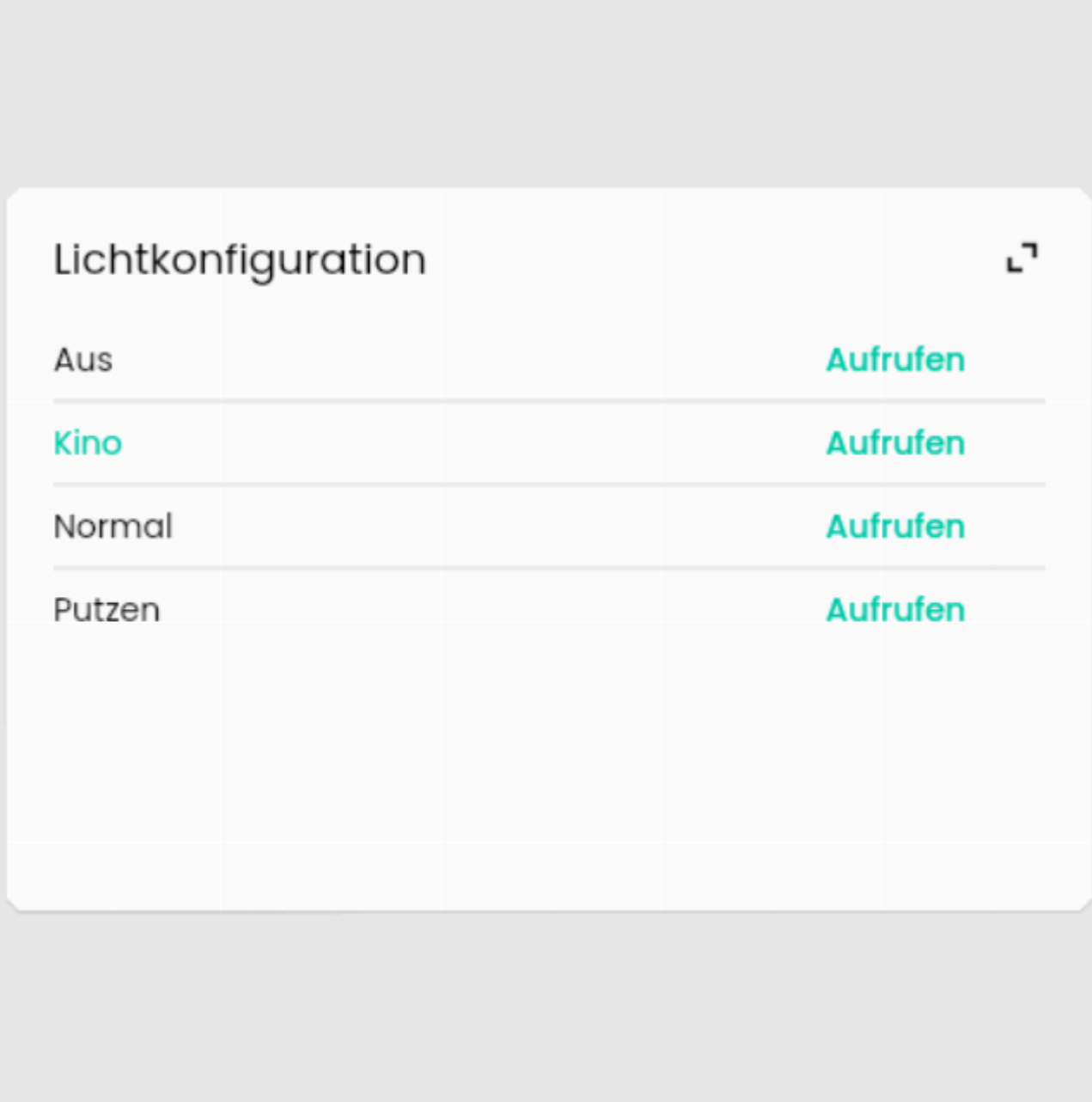
MEDIA PLAYER

A Media Player can display various parameters for media playback and switch them.



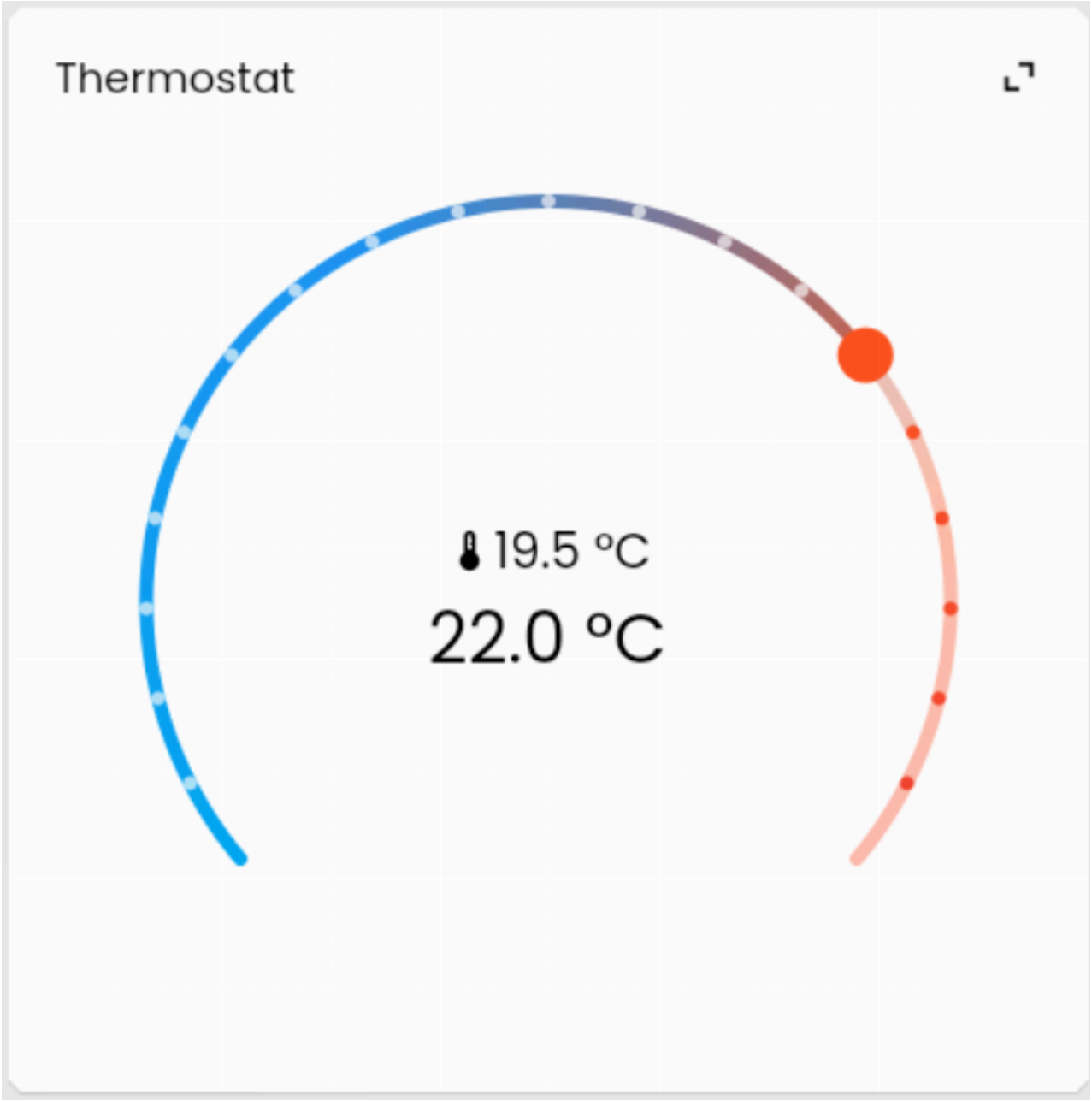
MESSAGES

Messages visualizes the current messages and the number of unread.



SCENES

Various device settings can be saved together as a scene and called up again.

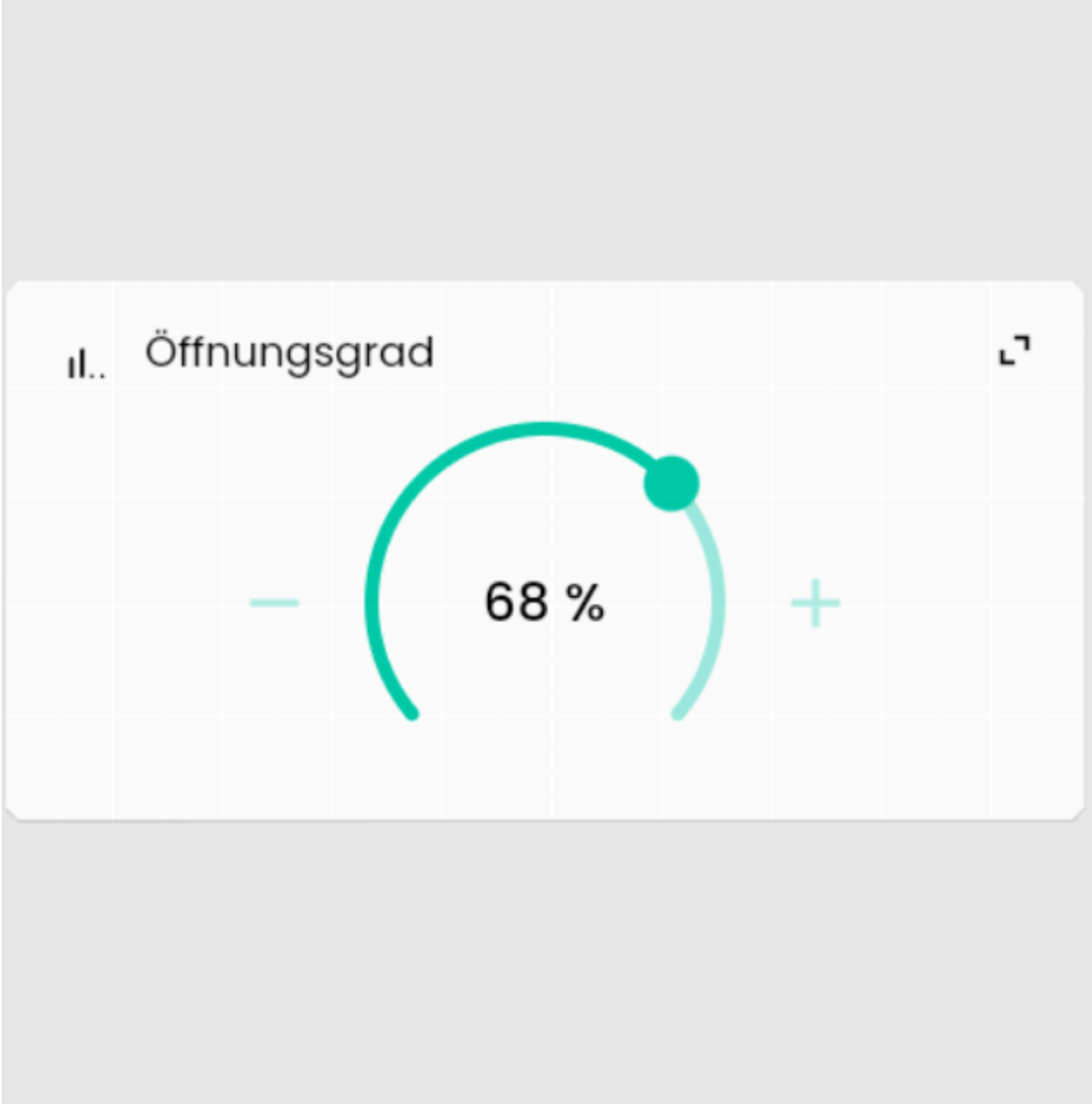


THERMOSTAT

A Thermostat allows setting a desired temperature combined with a display of the current temperature.

LEGACY-PRESENTATIONS

These are outdated presentations which redirect to their actual presentations.



LEGACY-PROFILE

Variable profiles can be used as presentations.

Enumeration

An Enumeration is a list of values that can be selected.

REQUIREMENTS

An Enumeration is a Variable with the presentation Enumeration. To use this presentation, the variable must fulfill the following conditions:
configured Variable action

PARAMETER

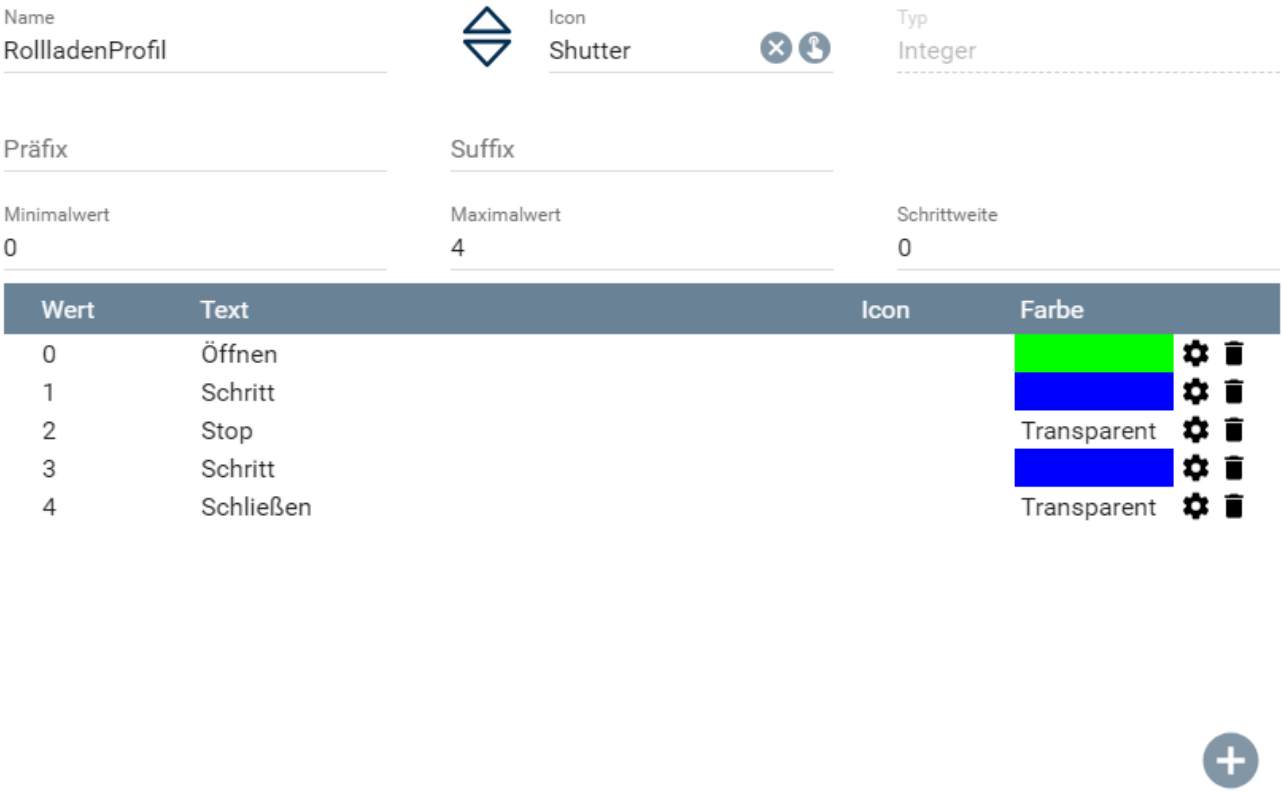
PARAMETER	DESCRIPTION
Default Icon	The Icon used for the variable
Options	The individual Options of the enumeration
Layout	The layout of the options in a tile. Possible layouts are Column, Row, and Grid. Grid is not available for variables of type Boolean. This parameter is only evaluated for appearance as own tile, not for appearance within a list.
Display...	The displayed content for each option. "Caption", "Icon", and "Caption and Icon" can be selected. (since Symcon 8.2)

Options

PARAMETER	DESCRIPTION
Value	The option is applied for this variable value
Label	Text displayed for this option
Overwrite icon	If set, the default icon for this presentation is overwritten if the variable has the value of this option
Icon	Icon for the variable if the variable has the value of this option (only available if "Overwrite icon" is set)
Color	Color that is used to display this option inside the enumeration

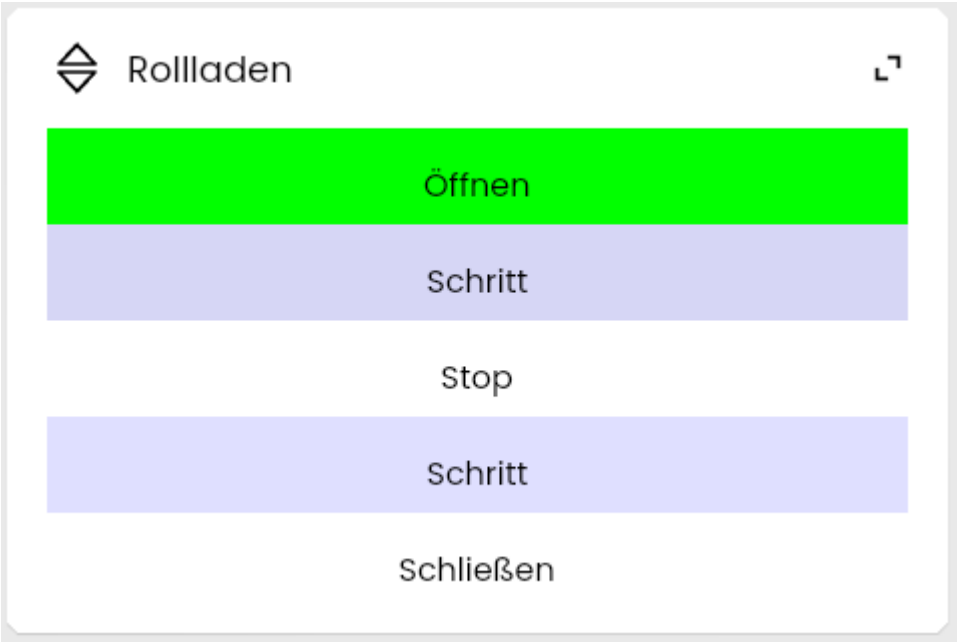
DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as an Enumeration if it also fulfills the following conditions:
Variable profile with the following parameters:
step size to 0 (only for integer/float, otherwise no option for step size)
at least 1 association set
minimum value and maximum value irrelevant



APPEARANCE IN TILE VISUALIZATION

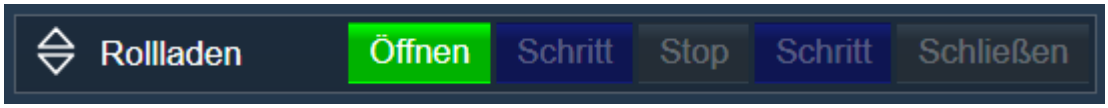
AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Trigger Event

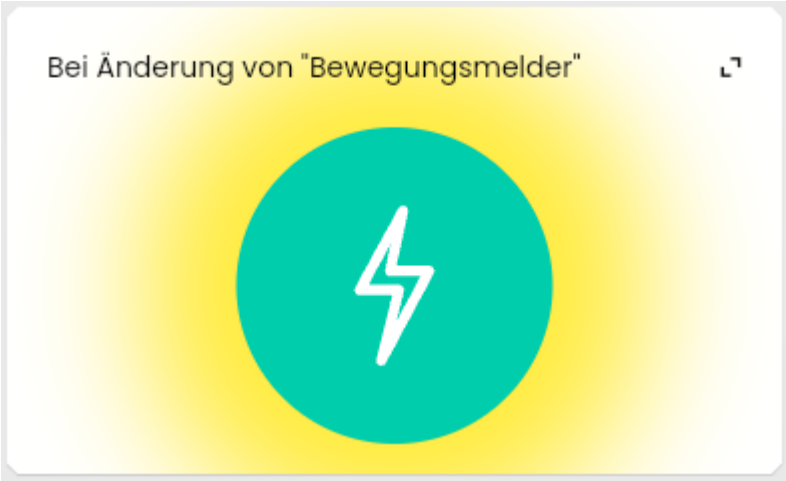
Require: Symcon >= 7.0
A triggered event can be activated or deactivated in the visualization.

REQUIREMENTS

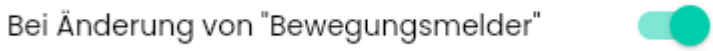
A triggered event is a Event of the type Trigger.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT

Trigger events are not displayed in the WebFront.

Automation

Automations can be executed via a button.

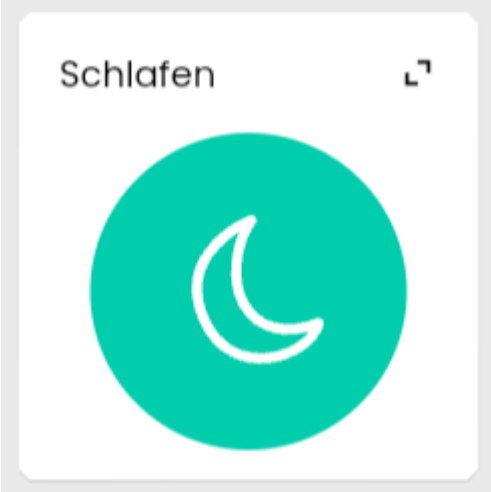
REQUIREMENTS

A configured Automation can be executed in the visualization

> In the display as a separate tile, the icon of the button is based on the Icon of the Automation

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Image

Any images can be integrated into the visualization.

REQUIREMENTS

An Image is a Medium of type Image.

APPEARANCE IN TILE VISUALIZATION

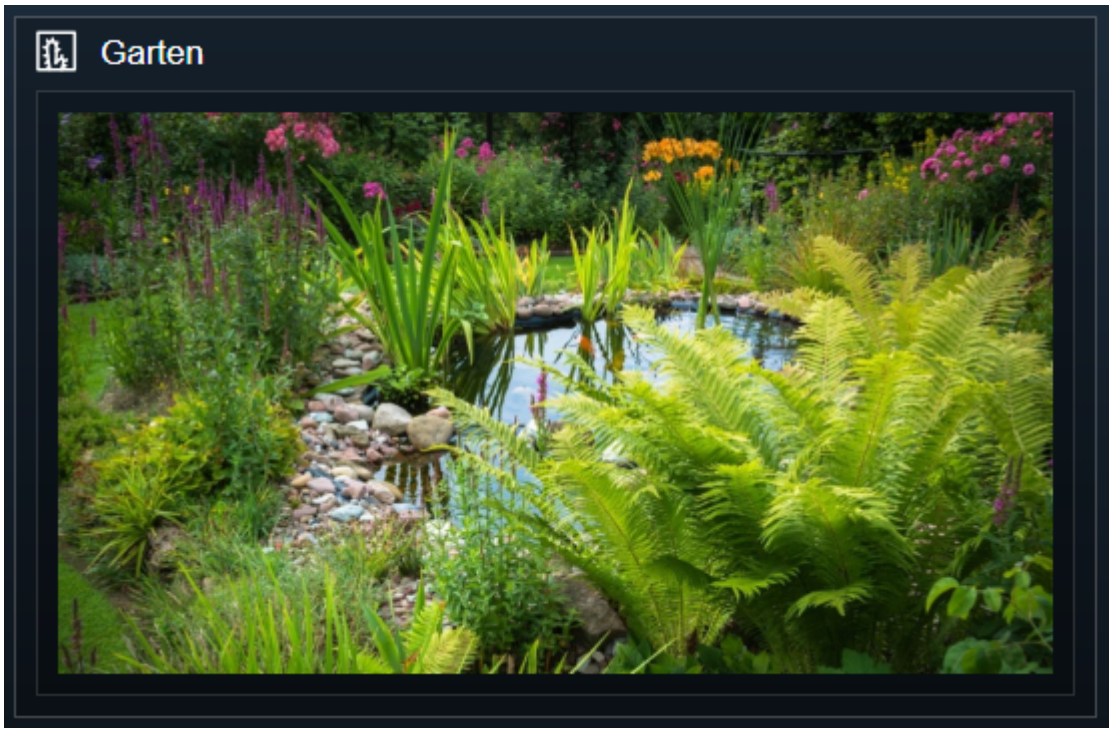
AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT


Garten

Date/Time

With Date/Time, a point in time can be displayed and adjusted if necessary.

REQUIREMENTS

A Date/Time is a variable with the presentation Date/Time. To use this presentation, the variable must fulfill the following conditions:
integer type

PARAMETER

PARAMETERS	DESCRIPTION
Date Display	Determines how the date is formatted. Possible options are: "None", "Year, Month and Day", "Month and Day", and "Year and Month"
Format Month as	If the month is to be displayed in the date, its formatting can be adjusted. The following are possible: "Text" and "Number"
Show Day of the Week	Determines whether the day of the week is also displayed in the format
Time Display	Determines how the time should be formatted. The following are possible: "None", "'Hours and Minutes" and "Hours, Minutes and Seconds"

DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as a date/time if one of the following Variable profiles is selected:
~UnixTimestamp
~UnixTimestampDate
~UnixTimestampTime
The different profiles correspond to fixed parameter combinations.

~UNIXTIMESTAMP

Date Display: Year, Month and Day
Format Month as: Text
Show Day of Week: No
Time Display: Hours and Minutes

~UNIXTIMESTAMPDATE

Show date: Year, Month and Day
Format Month as: Text
Show Day of the Week: No
Show Time: None

~UNIXTIMESTAMPDATE

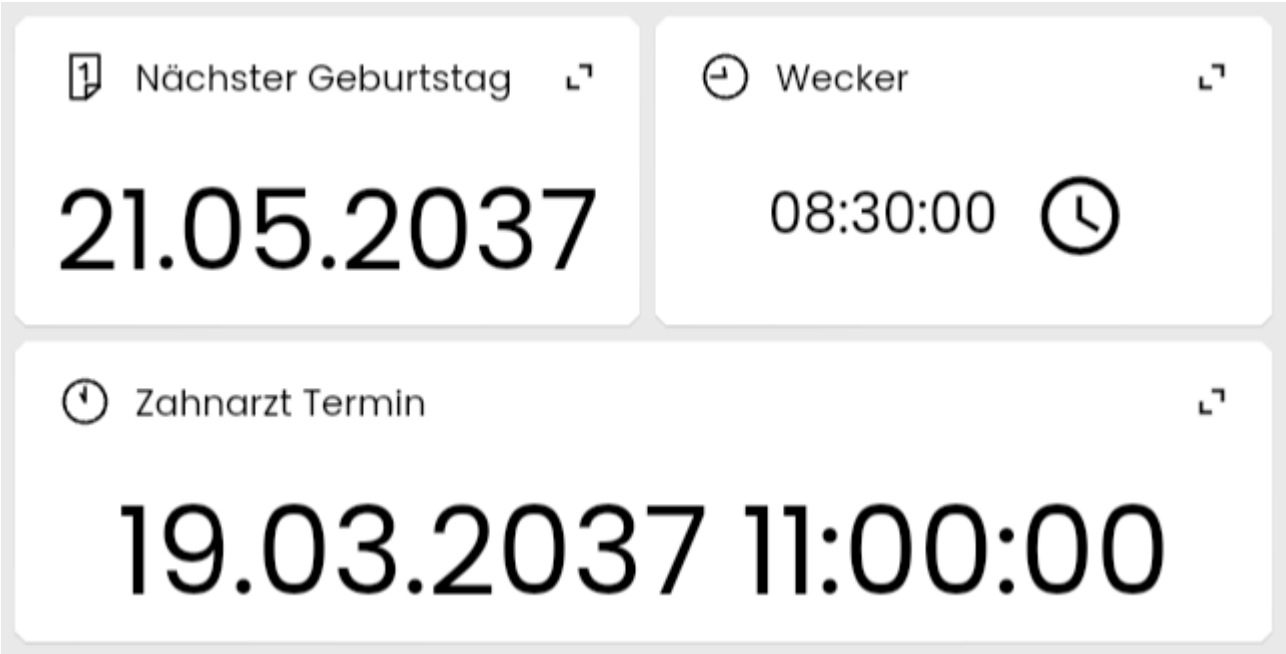
Date Display: None
Time Display: Hour and Minutes
If the variable has a variable action, the time can be adjusted in the visualization

APPEARANCE IN TILE VISUALIZATION



In the grid settings, the various displays can be configured separately as "Date", "Time" and "Date and Time"

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT


Nächster Geburtstag
21. Mai 2037
Wecker
08:30
Zahnarzt Termin
19. März 2037 11:00

Duration

Require: Symcon >= 8.0
Duration can be used to format the time until/since a point in time or a simple value as a duration.

REQUIREMENTS

Duration is a variable with the presentation duration. To use this presentation, the variable must fulfill the following conditions:
type integer or float
no configured variable action

PARAMETER

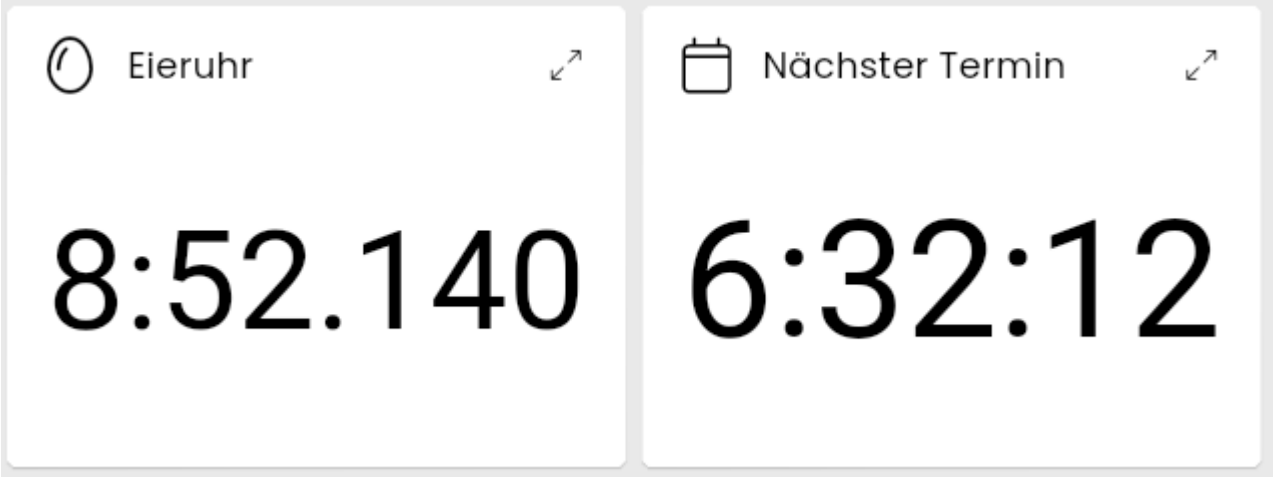
PARAMETER	DESCRIPTION
Display Type	Defines how the displayed value is created. **Value in variable:** The duration in seconds which should be formatted **Duration until value in variable:** The duration until the time coded as a Unix timestamp **Duration since value in variable:** The duration since the time coded as a Unix timestamp
Format	The format for in which the duration is formatted. The following are possible: "seconds only", "minutes and seconds", and "hours, minutes and seconds"
Show Milliseconds	Determines whether milliseconds should also be displayed



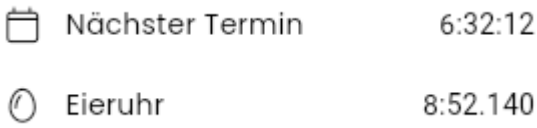
If the display type includes a point in time, the displayed value is updated automatically.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT

This display is not supported by the WebFront. Instead, the unformatted value of the variable is displayed.

Chart

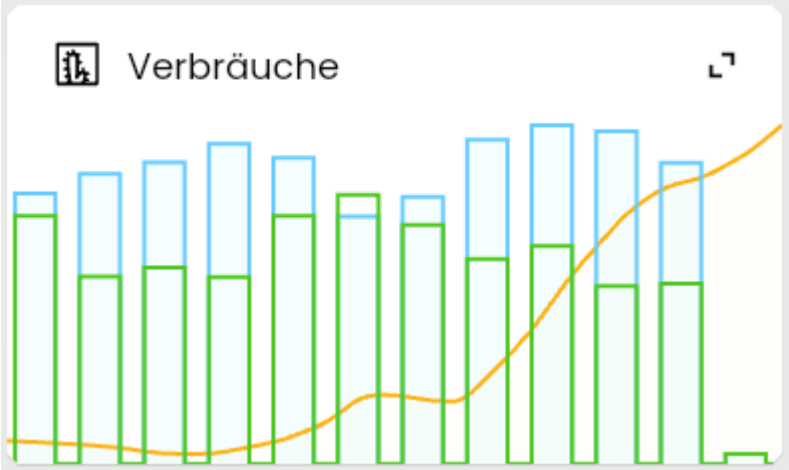
The course of several variables can be displayed in a Chart.

REQUIREMENTS

A chart is a Medium of the type Chart.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



If a Tile is sufficiently large, it is displayed analogous to the fullscreen display instead

IN FULLSCREEN

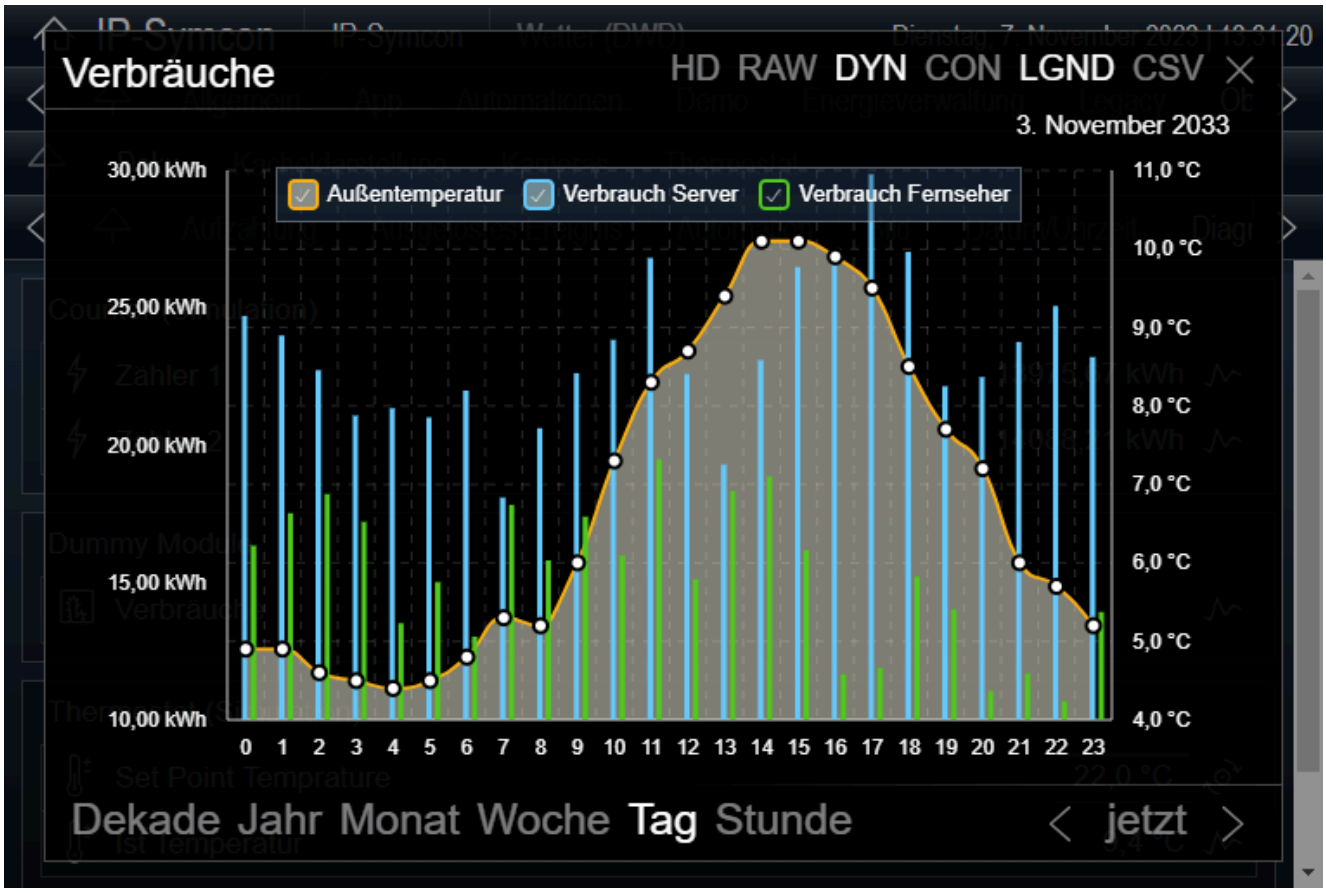


WITHIN A LIST



APPEARANCE IN THE WEBFRONT




Verbräuche
HD RAW DYN CON LGND CSV
3. November 2033
30,00 kWh
25,00 kWh
20,00 kWh
15,00 kWh
10,00 kWh
11,0 °C
10,0 °C
9,0 °C
8,0 °C
7,0 °C
6,0 °C
5,0 °C
4,0 °C
Außentemperatur
Verbrauch Server
Verbrauch Fernseher
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Dekade Jahr Monat Woche Tag Stunde
jetzt

Document

A Document can be downloaded by selecting the button.

REQUIREMENTS

A Document is a Medium of type Document.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Single Element

Require: Symcon >= 7.0
Enables the display of an element from a List as a single tile.

REQUIREMENTS

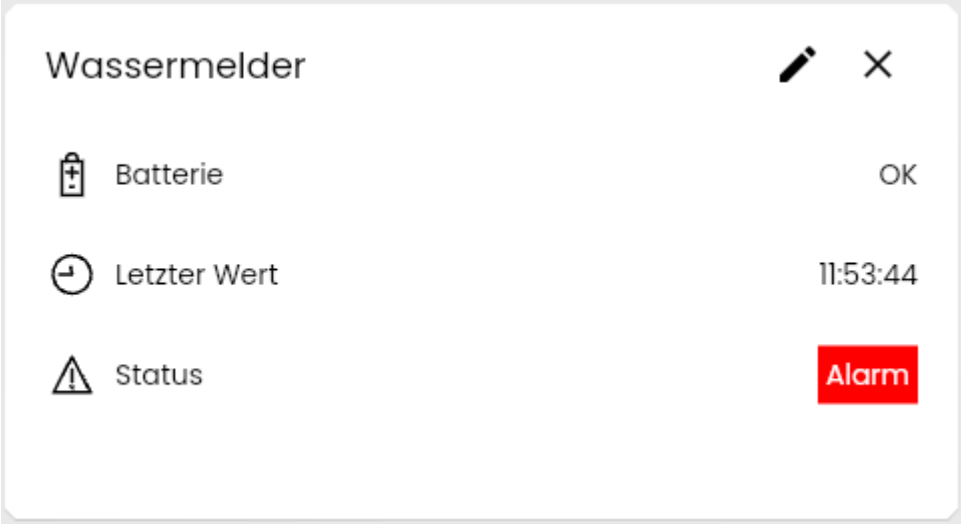
Each Instance can be displayed as a Single Element. The presentation is automatically selected if an instance only has one sub-object. Alternatively, it can be selected for each instance with multiple objects via "Edit Tiles".

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



If the Tile is opened, all objects of the instance are displayed as List.



APPEARANCE IN THE WEBFRONT

This display is not supported by the WebFront. Instead, the instance is displayed as List.

Energy Manager

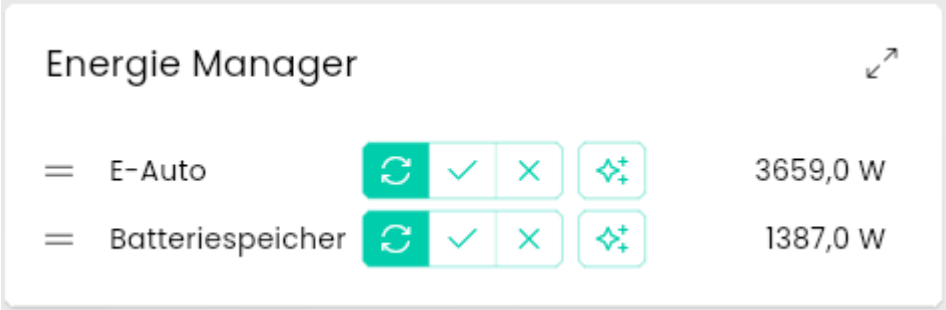
Require: Symcon >= 7.0
The Energy Manager intelligently switches devices when enough energy is available. This presentation makes it possible to change the priority of individual appliances using drag & drop or to exclude them from the calculation. In addition, the intelligent optimization like Overnight Charge or Cheap Charge can be configured within a dialog.

REQUIREMENTS

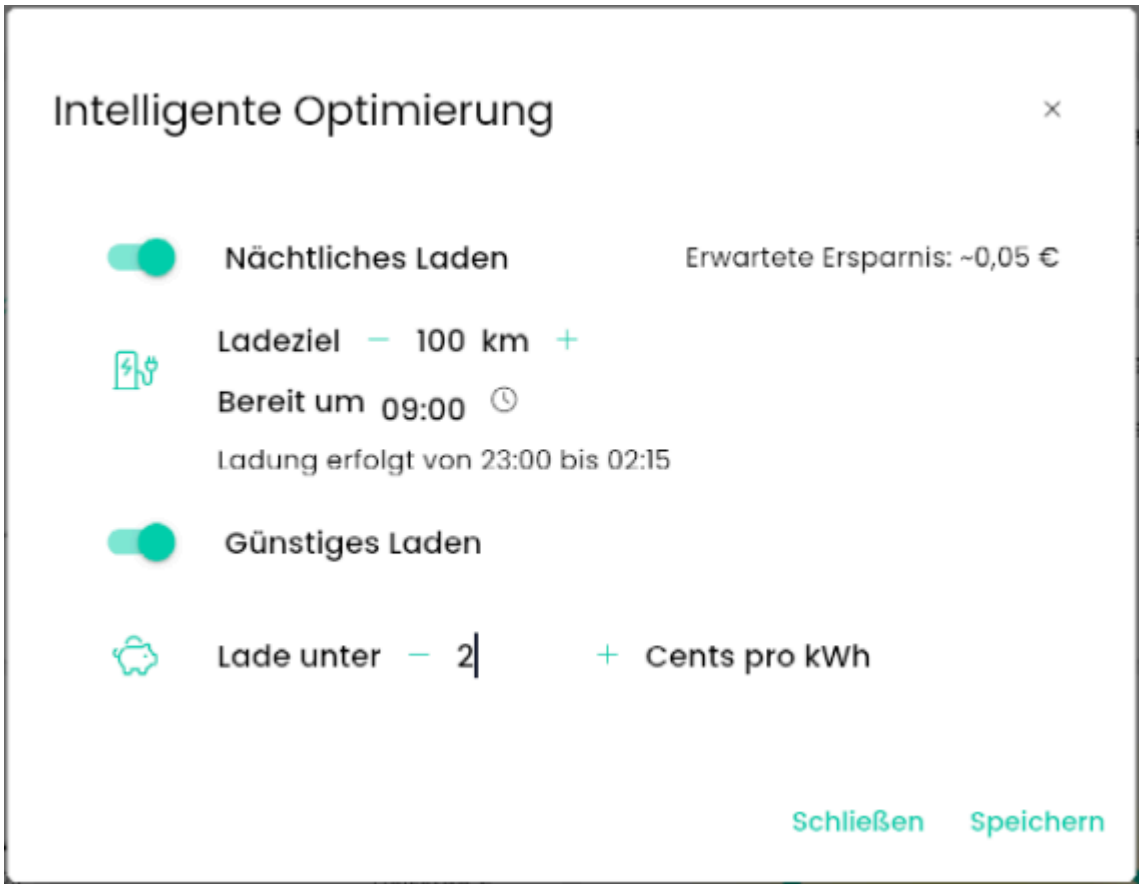
The Energy Manager requires an Instance of the module Energy Manager.

APPEARANCE IN TILE VISUALIZATION

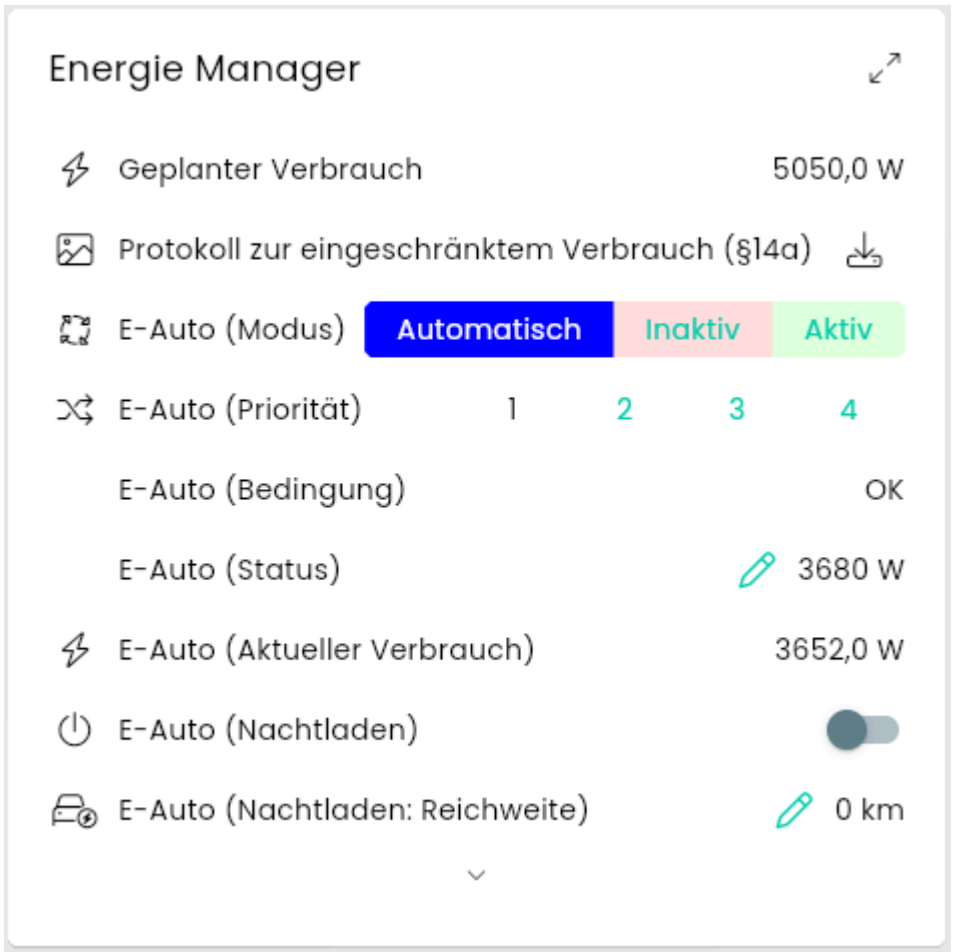
AS OWN TILE



DIALOG FOR INTELLIGENT OPTIMIZATION



WITHIN A LIST



APPEARANCE IN THE WEBFRONT

This display is not supported by the WebFront. Instead, the instance is displayed as List.

Energy Distribution

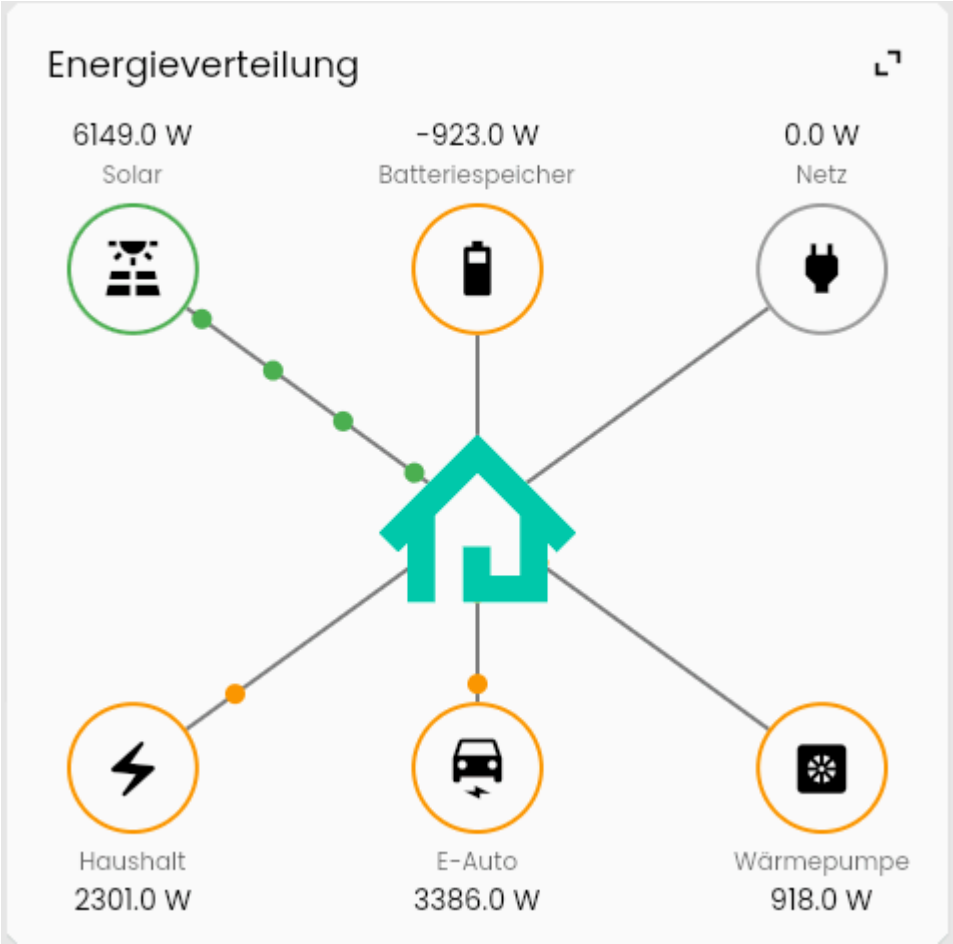
Require: Symcon >= 7.0
The Energy Distribution shows the flow of generated and consumed energy.

REQUIREMENTS

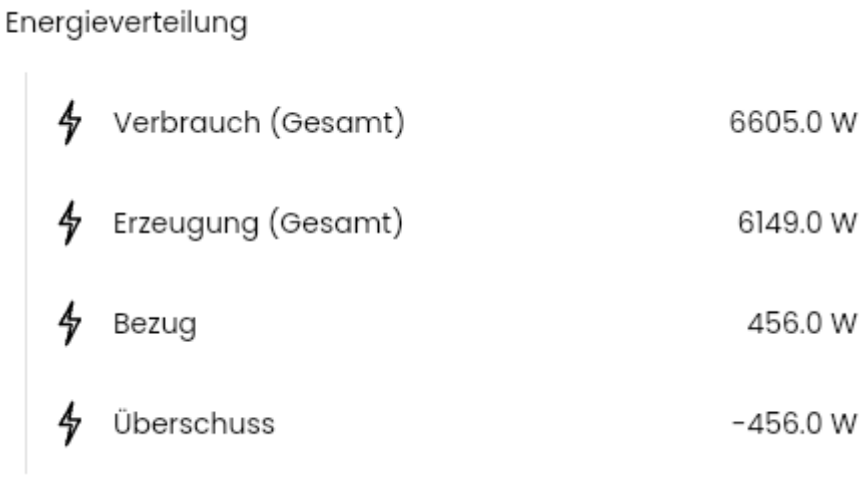
The Energy Distribution display requires a Instance of the module Energy Distribution.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT

This display is not supported by the WebFront. Instead, the instance is displayed as List.

Color

A color can be displayed and possibly configured with a Color.

REQUIREMENTS

An Color is a Variable with the presentation Color. To use this presentation, the variable must fulfill the following conditions:
Type Integer or String

PARAMETER

PARAMETER	DESCRIPTION
Encoding	Used encoding for the color, only adjustable for variables of type String
Default Values	List with default values that are offered in a selection, only available for variables with Variable Action. Not available if 'xy' encoding is selected
Color Sapce	The color space from which a color can be selected. This defines a triangle
Custom Color Space	List in which the red, green, blue value and white point of a custom color space can be defined.
Color Curve	If a color curve is selected, the selection of the color is limited to this
Custom Color Curve	A list of coordinates that define the

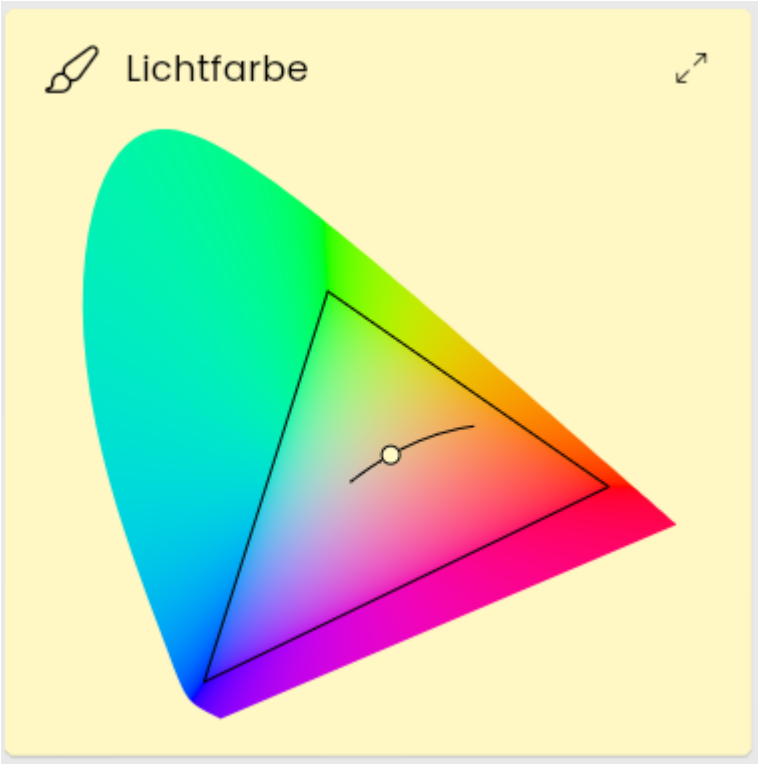
DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as a color if it also fulfills the following conditions:
selected Variable profile "~HexColor"

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE WITH VARIABLE ACTION





When selecting the pipette in the middle of the circle, a dialog opens to set the color alternatively in detail

AS OWN TILE WITHOUT VARIABLE ACTION



WITHIN A LIST WITH VARIABLE ACTION



When the pen is selected, a dialog opens, analogous to the display as a separate tile.

WITHIN A LIST WITHOUT VARIABLE ACTION

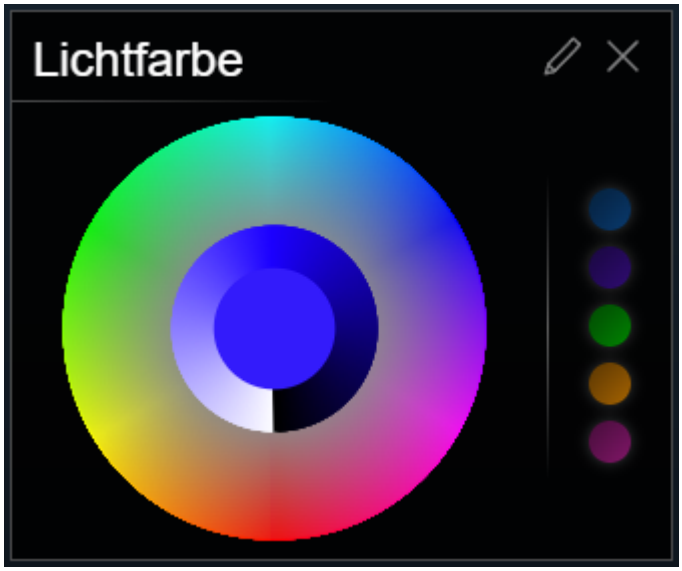


APPEARANCE IN THE WEBFRONT

WITH VARIABLE ACTION



Selecting the brush on the right opens a color selection dialog:



WITHOUT VARIABLE ACTION



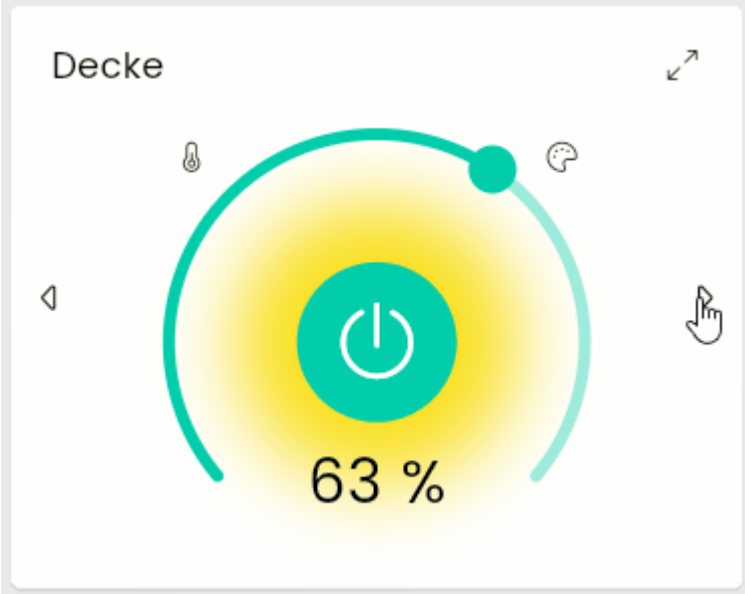
Content Changer

A content changer allows subordinate objects to be displayed within a single tile. The individual objects can be changed by clicking or automatically after a certain time.

REQUIREMENTS

An object of the type Category for which the content changer presentation was selected in the visualization in edit mode.

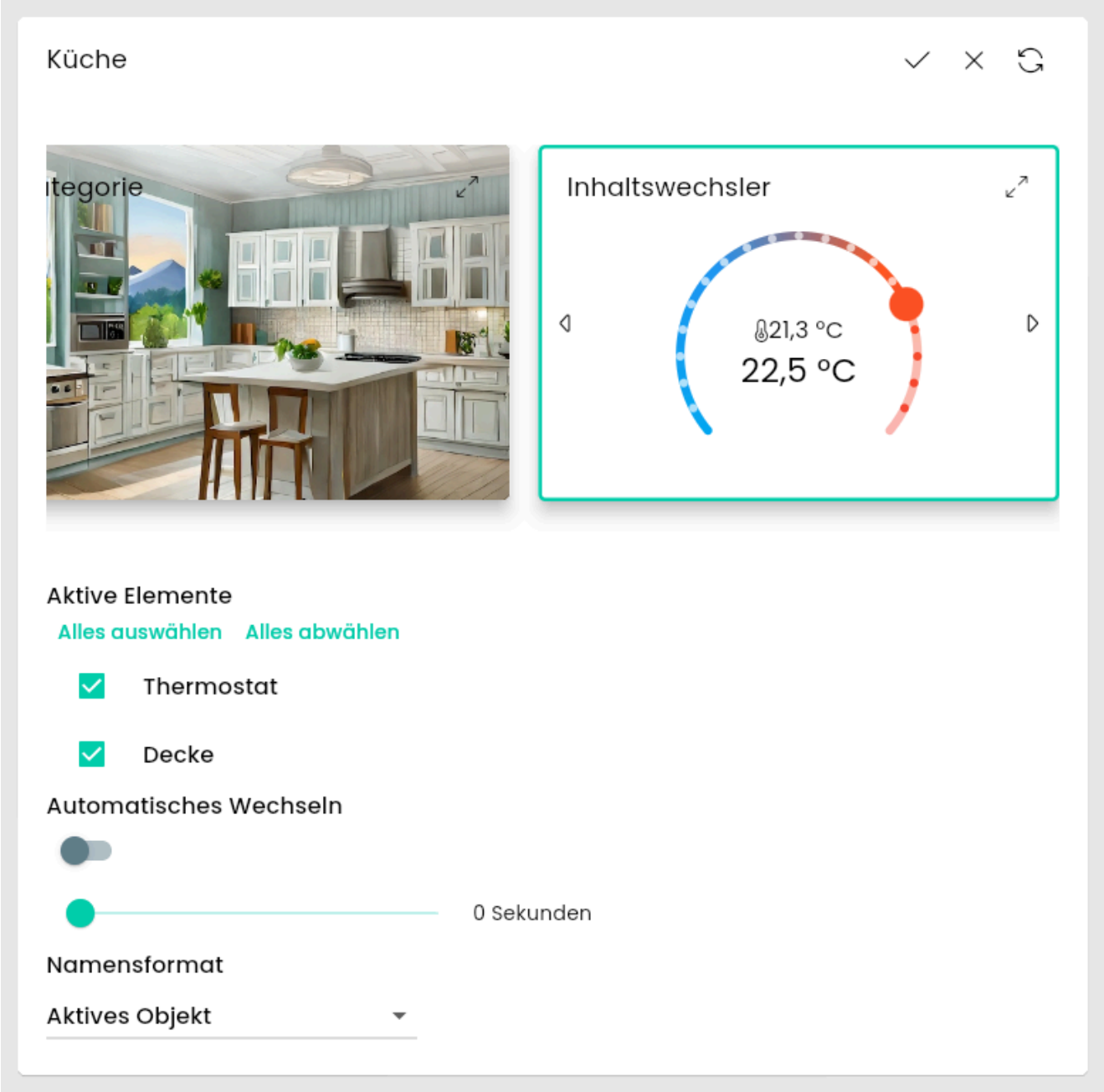
APPEARANCE IN TILE VISUALIZATION



CONFIGURATION

When a content changer is edited in the visualization, the following parameters are available.

NAME	DESCRIPTION
Active Elements	Here you can select from all visible elements of the category which should be displayed.
Automatic Change	If active, the content changer jumps to the next element after a specified duration. The maximum duration is 100 seconds.
Name Format	The name to be displayed as the title of the tile. (Active object, category, active object and category)



APPEARANCE IN THE WEBFRONT

This presentation is displayed like a normal category.



Something similar can be achieved via the Content changer in the WebFront Editor.

IPSView

IPSViews can be seamlessly integrated into the visualization.

> The extension IPSView must be purchased separately for use

REQUIREMENTS

An IPSView is a Medium of type IPSView.

APPEARANCE INTILE VISUALIZATION

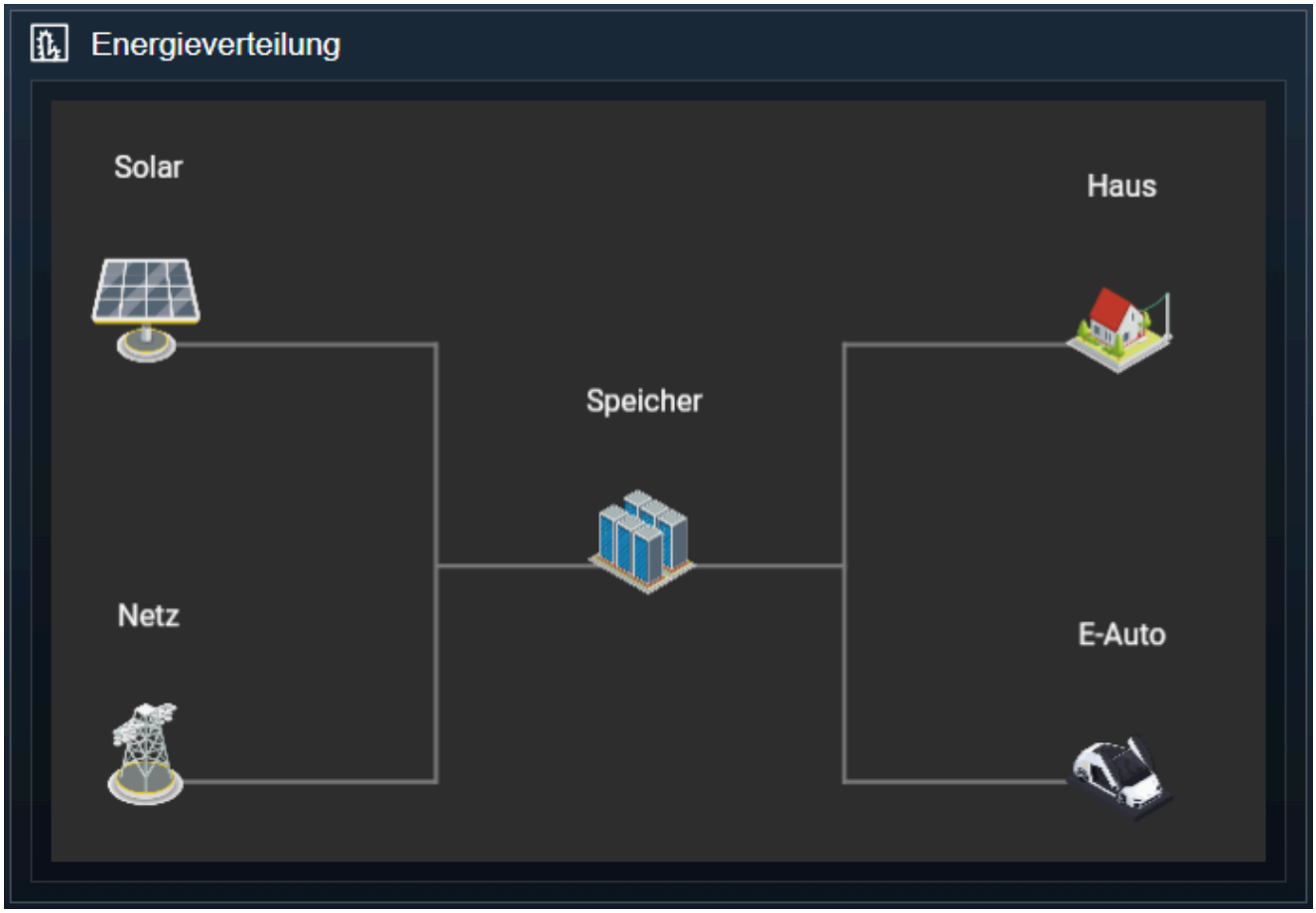
ALS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Shutter

For a Shutter, the degree of opening and optionally the degree of inclination of the slats can be set.

REQUIREMENTS

A Shutter can be set either as Variable or as Instance.

AS A VARIABLE

A variable with the presentation shutter must fulfill the following requirements:
Integer type
Configured Variable action

PARAMETER

PARAMETER	DESCRIPTION
Usage type	Determines whether the slider is used to control the rotation of slats or the opening degree of a shutter. The Usage Type parameter determines which other parameters are available.
Open at	The value at which the shutter is considered open (only for the Open Usage Type)
Closed at	The value at which the shutter is considered closed (only for Open Usage Type)
Inside at	The value at which the slats are considered to be turned inwards (only for Rotation Usage Type)
Outside at	The value at which the slats are considered to be rotated outwards (only for Rotation Usage Type)
Rotation	The fixed rotation of the displayed slats (only for Open Usage Type)
Maximum Rotation Inside	The maximum inward rotation value of the slats in degrees (only for Rotation Usage Type)
Maximum Rotation Outside	The maximum outward rotation value of the slats in degrees (only for Rotation Usage Type)
Sun Position	Determines whether and where a sun should be displayed next to the shutter. The height of the sun depends on the time of day. A moon is displayed at night.

AS INSTANCE

An instance can be displayed as a blind if the following child objects are present:
Position (optional): Variable as described above with the ~Shutter profile
Degree of rotation (optional): Variable as described above with the ~Lamella profile

At least one optional variable must exist.



The default size of Shutters can be configured separately in the grid settings for variables as "Shutter Variable" and instances as "Shutter"

DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as a shutter if one of the following Variable profiles is selected:
~Shutter
~Lamella
The different profiles correspond to fixed parameter combinations.

~SHUTTER

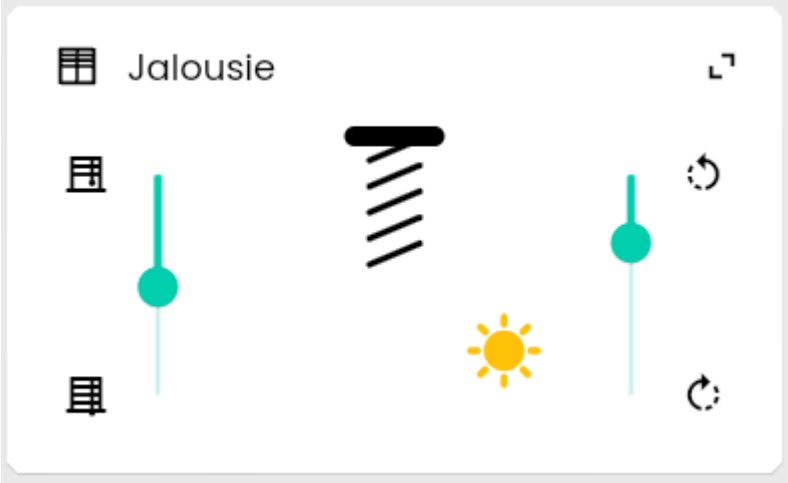
Usage Type: Open
Open At: 0
Closed At: 100
Rotation: -55°
Sun Position: Right

~LAMELLA

Usage Tyoe: Rotation
Inside At: 0
Outside At: 100
Maximum Rotation Inside: -55°
Maximum Rotation Outside: 55°
Sun Position: Right

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST

In the list view, the visualization falls back to the presentation List.



APPEARANCE IN THE WEBFRONT

This presentation is not supported by the WebFront. Instead, variables are displayed as Slider and instances are displayed as List.

Category

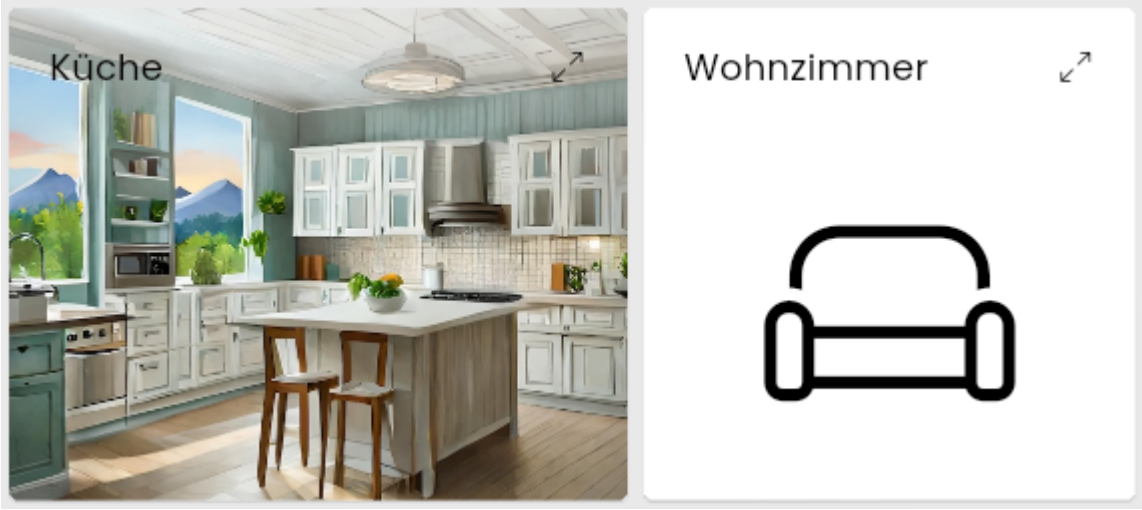
A category can be opened to view the objects it contains.

REQUIREMENTS

An object of the type Category.

APPEARANCE IN TILE VISUALIZATION

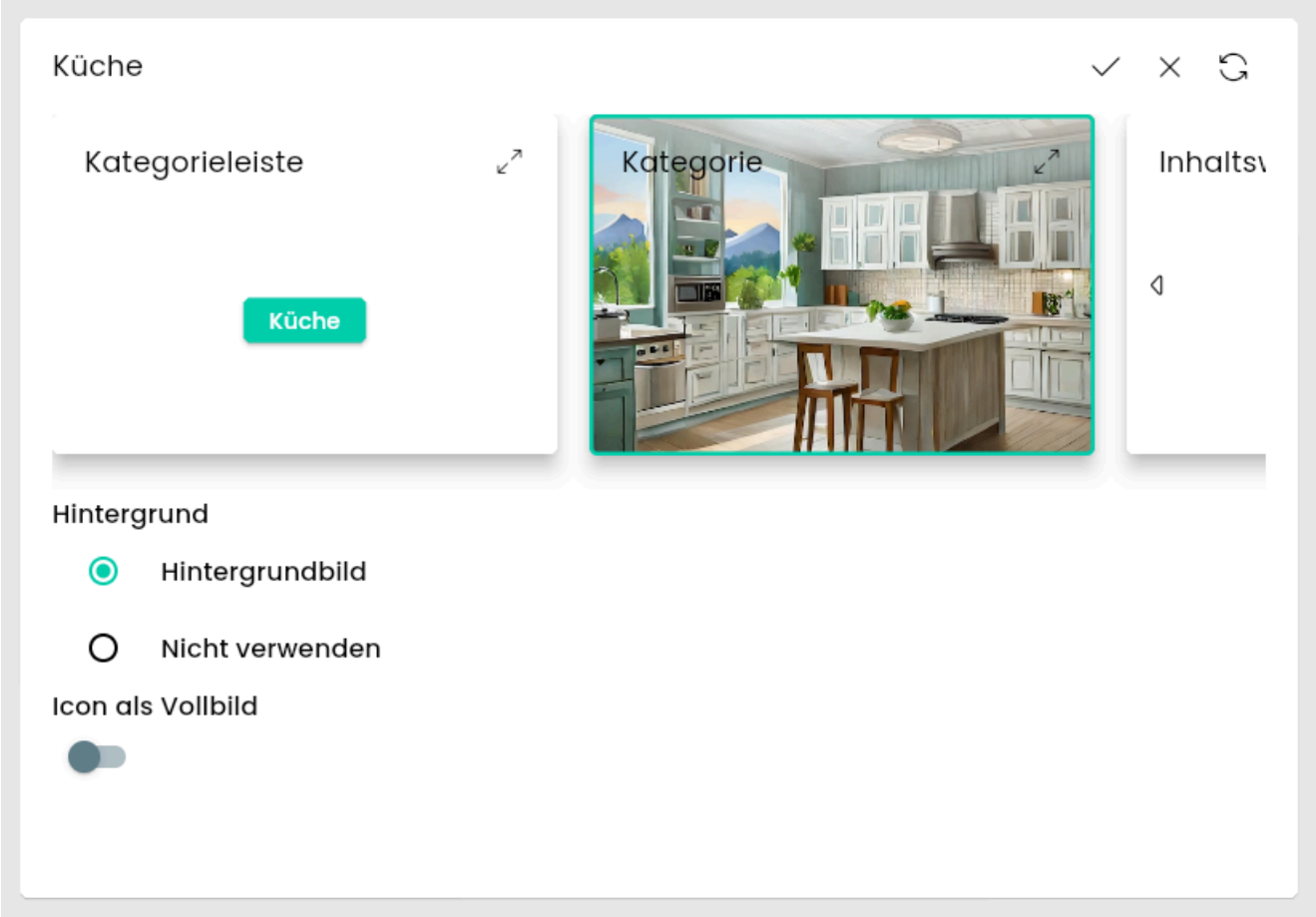
In the Instance configuration 'Tiles' can be selected for 'Default category display'. This default setting can be overwritten in edit mode in the visualization for each object.



CONFIGURATION

When a category is edited in the visualization, the following parameters are available.

NAME	DESCRIPTION
Background	If the category contains an image as Media object, this can be selected here as the background. The media object can also be made invisible via the 'Visual settings' in the console.
Icon as full screen	If active, the Icon of the category object is displayed in the center of the tile



APPEARANCE IN THE WEBFRONT



Category Bar

A category can be opened to view the objects it contains.

REQUIREMENTS

An object of the type Category.

APPEARANCE IN TILE VISUALIZATION

In the Instance configuration 'Category Bar (Single Line)' or 'Category Bar (Multiline)' can be selected for 'Default Category Display'. This default setting can be overwritten for each object in edit mode in the visualization. In addition to the object name, the Icon of the category is also displayed.



APPEARANCE IN THE WEBFRONT



Legacy Profile

This Variable presentation evaluates a variable profile and translates it to the actual presentation.

PREREQUISITES

One variable uses the "Legacy Profile" variable presentation. The presentation has no further prerequisites.

PARAMETER

PARAMETER	DESCRIPTION
Profile	The underlying variable profile

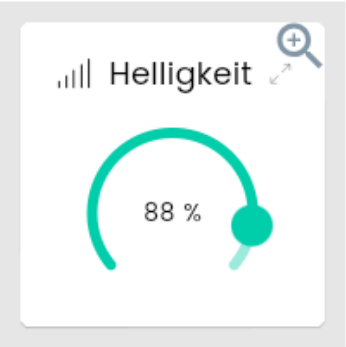
The following variable presentations can be realized via Legacy Profile. The exact requirements can be found on the individual pages for the corresponding presentation:

- Enumeration
- Date/Time
- Color
- HTML Box
- Shutter
- Switch
- Slider
- Value display
- Value input

CONVERT

In the "Edit variable" dialog, a legacy presentation can be converted to its actual presentation via the corresponding button under the Variable presentation item.

Variablendarstellung



AUF STANDARD ZURÜCKSETZEN

Darstellung
Legacy Profil

Vorlage
(Benutzerdefiniert)

Darstellungsparameter

Profil
~Intensity.100

KONVERTIERE LEGACY PROFIL ZU NEUER DARSTELLUNG

Light

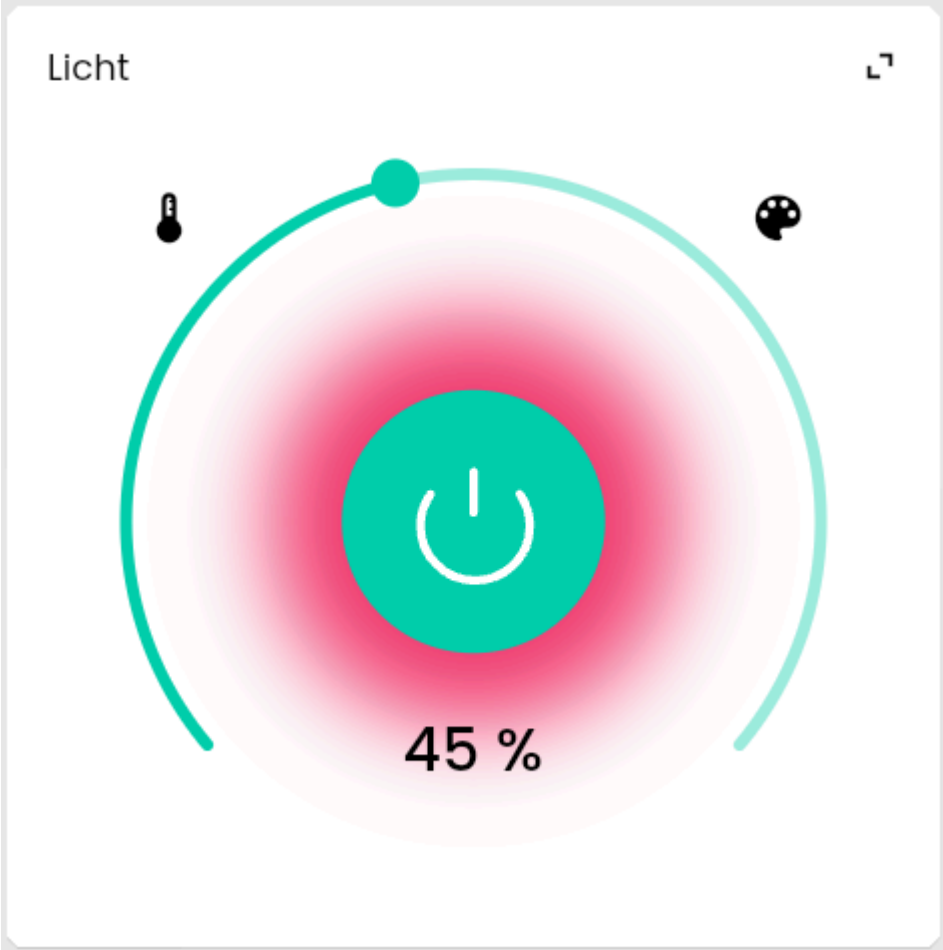
Require: Symcon >= 7.0
In the presentation Light, status, brightness, tuneable white and color can be visualized in a single tile.

REQUIREMENTS

An Instance can be displayed as light if the following child objects are present:
Status:
Uses the variable presentation Switch
Variable Usage of the variable is "On/Off"
Brightness (optional):
Uses the variable presentation Slider
Variable Usage of the variable is "Intensity"
Tuneable White (optional):
Uses the variable presentation Slider
Variable Usage of the variable is "Color Temperature"
Color (optional):
Uses the variable presentation Color
configured Variable action
At least one optional variable must exist.

APPEARANCE IN TILE VISUALIZATION

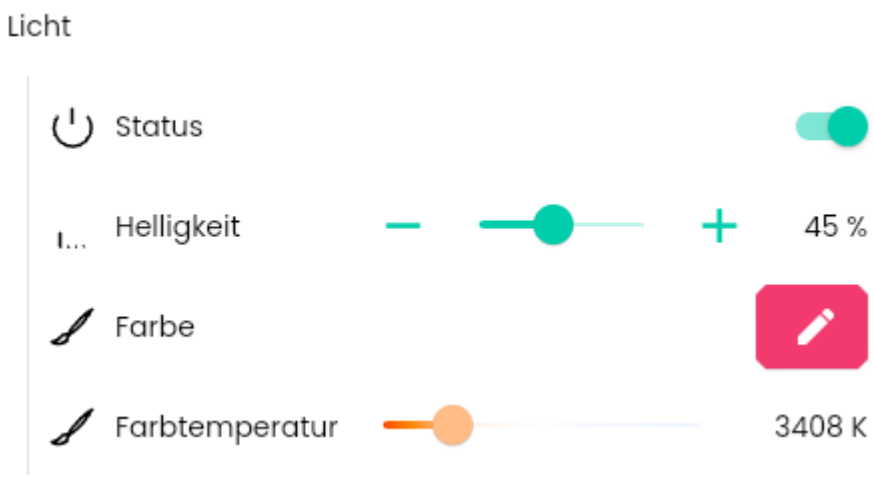
AS OWN TILE


Licht
45 %


Licht

WITHIN A LIST

In the list view, the visualization falls back to the presentation List.



APPEARANCE IN THE WEBFRONT

This presentation is not supported by the WebFront. Instead, the instance is displayed as List.

List

A list view can display any objects one below the other.

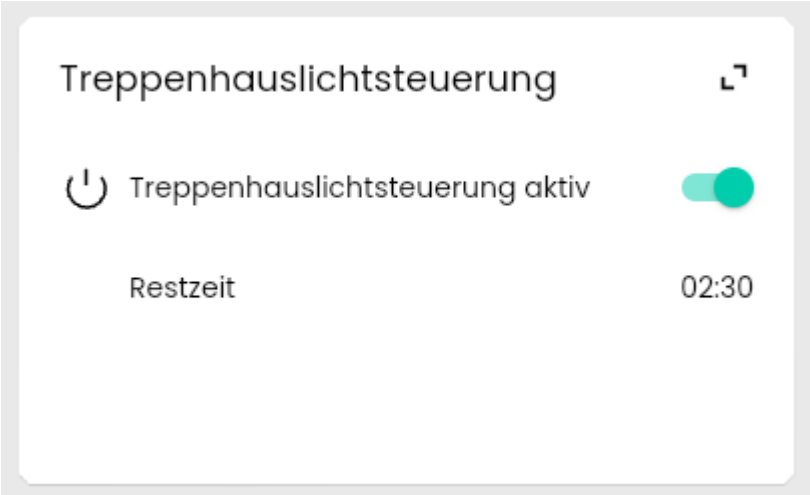
REQUIREMENTS

Each Instance can be displayed as a List.

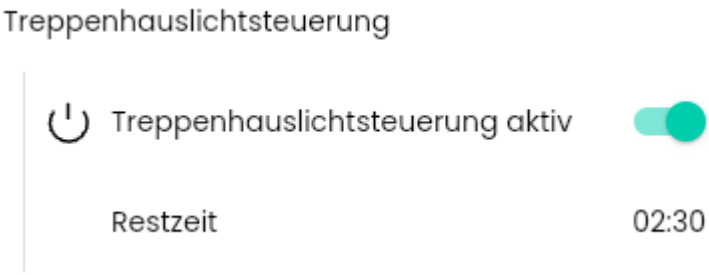
> If values are simply to be summarized without there being a suitable parent instance, a Dummy module can be used for structuring

APPEARANCE IN TILE VISUALIZATION

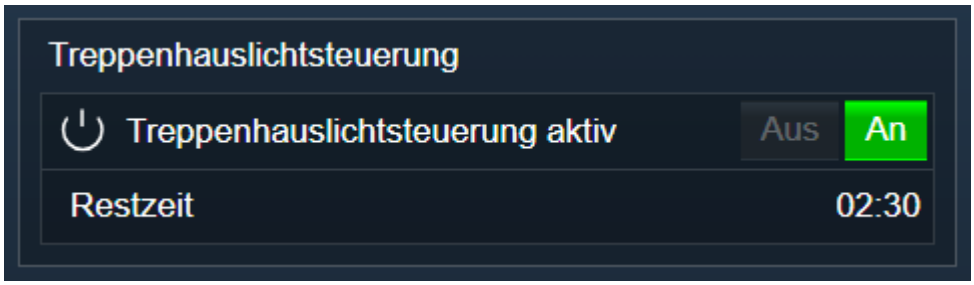
AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT


Treppenhauslichtsteuerung
Treppenhauslichtsteuerung aktiv
Aus
An
Restzeit
02:30

Media Player

Require: Symcon >= 7.0
A Media Player can display various parameters for media playback and switch them if necessary:
- Play and pause media playback
- Stop media playback (optional)
- Jump to the next or previous element (optional)
- Display and adjust the progress of the current element (optional)
- Set volume (optional)
- Mute (optional)
- Set repeat (optional)
- Activate shuffle playback (optional)
- Display cover (optional)
- Show artist (optional)
- Show title (optional)
- Display and adjust current playlist including current position (optional)

REQUIREMENTS

A Media Player can be created either as Variable or as Instance.

AS A VARIABLE

A variable that is to be displayed as a Media Player must fulfill the following requirements:
- Type Integer
- Set up Variable action
- selected Variable profile:
  - ~PlaybackNoStop: Only supports play and pause
  - ~Playback: Supports playing, pausing and stopping
  - ~PlaybackPreviousNextNoStop: Supports playing, pausing and jumping to the next or previous element
  - ~PlaybackPreviousNext: Supports playing, pausing, stopping and jumping to the next or previous element
  - All other functions of the media player are not supported when displayed as variable.

AS INSTANCE

An instance can be displayed as a Media Player if the following child objects are present:
- Playback:
  - Variable as described above
- Progress (optional):
  - Variable of type integer
  - Uses the variable presentation Slider
  - usage type of the variable is "Progress"
  - If the variable has a variable action, the position can be adjusted, otherwise it is only displayed

Volume (optional):
Uses the variable presentation Slider
Variable Usage of the variable is "Volume"
Mute (optional):
Uses the variable presentation Switch
Variable Usage of the variable is "Mute"
Repeat (optional):
Enumeration
Variable type Integer
Variable profile "~Repeat"
Shuffle (optional):
Enumeration
Variable type Boolean
Variable profile "~Shuffle"
Cover (optional):
Image
Artist (optional):
Variable of type String
Variable profile "~Artist"
Title (optional):
Variable of type String
Variable profile "~Song"
Playlist (optional):
Variable of type String
Variable profile "~Playlist"
If the variable has a variable action, the position is customizable, otherwise it is only displayed
The value for the playlist is a JSON-encoded object with the following parameters:

PARAMETER	TYPE	DESCRIPTION
entries	Array	Entries of the playlist
current (optional)	Integer	The index of the currently active entry

Structure of the entries

PARAMETERS	TYPE	DESCRIPTION
song (optional)	String	Title of the entry
artist (optional)	String	Artist of the entry
duration (optional)	Integer	Duration of the entry in seconds
Additional entries (optional)	any	Any other parameters can be added for the functionality. However, these are not displayed

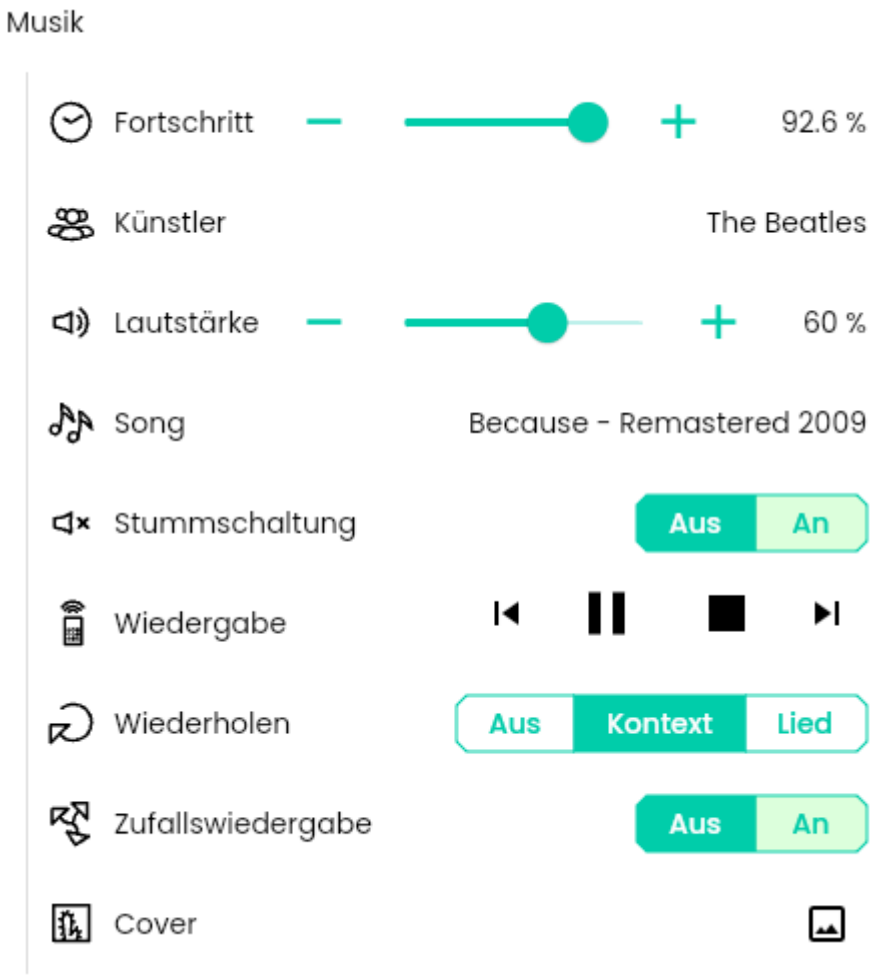
APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST

In the list view, the visualization falls back to the presentation List.



APPEARANCE IN THE WEBFRONT

This display is not supported by the WebFront. Instead, variables are displayed as Enumeration and instances are displayed as List.

Messages

Messages visualizes the current messages and the number of unread.

REQUIREMENTS

The presentation Messages requires an Instance of the module IMAP or POP3.

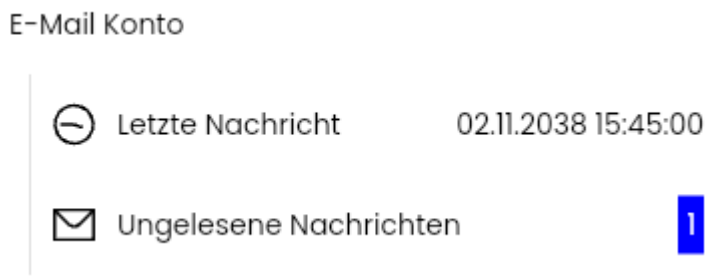
APPEARANCE IN TILE VISUALIZATION

AS OWN TILE

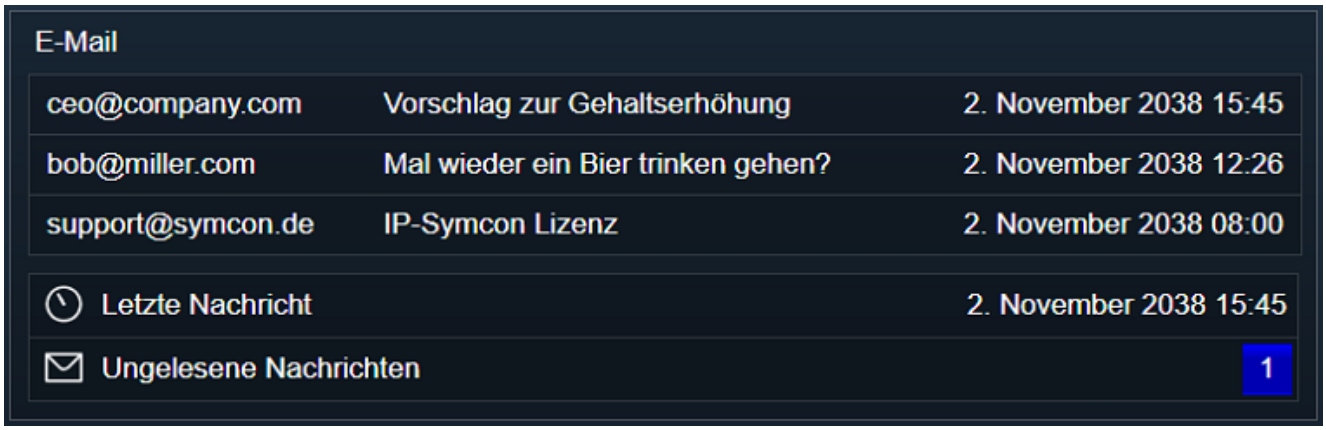


WITHIN A LIST

In the list view, the visualization falls back to the presentation List.



APPEARANCE IN THE WEBFRONT


E-Mail
ceo@company.com
Vorschlag zur Gehaltserhöhung
2. November 2038 15:45
bob@miller.com
Mal wieder ein Bier trinken gehen?
2. November 2038 12:26
support@symcon.de
IP-Symcon Lizenz
2. November 2038 08:00
Letzte Nachricht
2. November 2038 15:45
Ungelesene Nachrichten
1

Popup

Several objects can be displayed together in a Popup.

PREREQUISITES

A Popup is set up as Popup Module Instance. Displayed objects are placed or linked below the popup module.

APPEARANCE IN TILE VISUALIZATION

A popup display in the visualization tile will follow in future versions, but is currently not available.

APPEARANCE IN THE WEBFRONT



Switch

Require: Symcon >= 5.0
A Switch can be switched on (true) or off (false).

REQUIREMENTS

A Switch is a variable with the presentation Switch. To use this presentation, the variable must fulfill the following conditions:
Type Boolean
Set up Variable action

PARAMETER

PARAMETER	DESCRIPTION
Individual icons based on value	If set, individual icons can be selected for true and false. Otherwise, a single icon is selected, which is always displayed
Icon	Icon which is permanently displayed on the switch (only available if "Individual icons based on value" is not set)
Icon for true	Icon which is displayed on the switch if the variable has the value true (only available if "Individual icons based on value" is set)
Icon for false	Icon which is displayed on the switch if the variable has the value false (only available if "Individual icons based on value" is set)
Color of glow while active	Configures the colour of the glow around the switch when it is active, i.e., when the variable has the value true
Intensity of glow while active	Configures the intensity of the glow around the switch when it is active, i.e., when the variable has the value true
Variable Usage	Use of the variable in summarized presentations. The value does not have a direct effect on the display of the variable, but only on its use in summarized displays.

DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as a Switch if it also fulfills the following conditions:
no Variable profile set or variable profile "~Switch" or "~Mute"
When using a variable profile, some parameters of the slider are derived

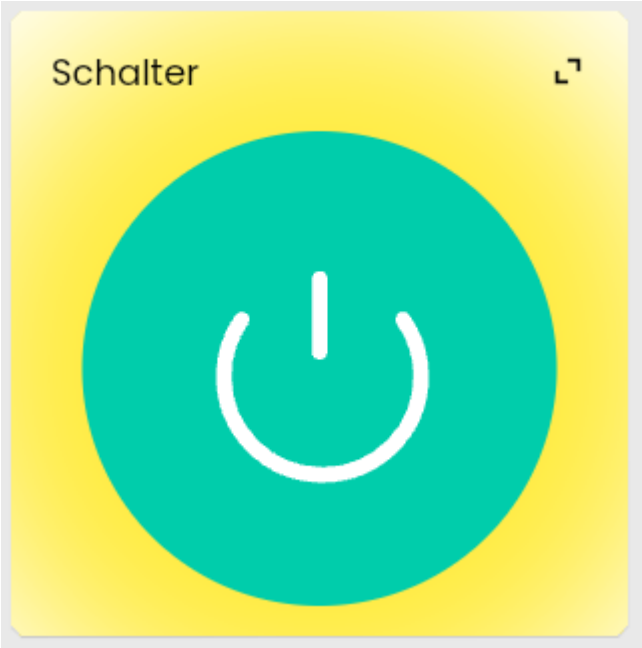
VARIABLE USAGE

The use depends on the profile name.
On/Off for profile name
~Switch
No profile

Mute Switch for profile name "~Mute"

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT

Falls die Darstellung über ein Legacy-Profil erfolgt, unterscheidet das WebFront ob kein Variablenprofil gesetzt ist oder das Profil "~Switch" und hat für jeden Fall eine eigene Darstellung.
If the appearance occurs via Legacy profile, the WebFront distinguishes whether no variable profile is set or the "~Switch" profile and has a separate display for each case

WITH PRESENTATION "SWITCH" OR PROFILE ~SWITCH



WITHOUT PROFILE


Schalter
true

Slider

A slider can be used to set a specific value.

PREREQUISITES

An Slider is a Variable with the presentation Slider. To use this presentation, the variable must fulfill the following conditions:

Type Integer or Float
configured Variable action

PARAMETER

PARAMETER	DESCRIPTION
Minimum Value	Minimum value of the slider
Maximum Value	Maximum value of the slider
Step Size	Size of a single step in the slider. At 0, the slider has no fixed step size.
Gradient	Defines the appearance of the gradient
Custom Gradient	Colors can be selected for specific values in the list. The gradient of the slider then consists of a gradient over the defined colors. (only visible and usable if gradient = "Custom")
Variable Usage	Use of the variable in summarized presentations. The value does not have a direct effect on the display of the variable, but only on its use in summarized displays.
Prefix	Is written in front of the variable value during formatting
Suffix	Is written after the variable value during formatting
Display Type	With "Percentage", the values are displayed as a percentage. This means that the minimum value is displayed as 0 and the maximum value as 100, regardless of their absolute values. Other values are also scaled accordingly. No conversion takes place with "Absolute".
Thousands Separator	Defines the thousands separator used when formatting the value
Digits	The number of decimal places that are displayed when formatting the value (only available for variables of type Float)
Decimal Separator	Defines the decimal separator used when formatting the value (only available for variables of type Float)
Icon	The Icon that is used for the variable
Use updated parameters for specific intervals	Activates a list of intervals that can override the formatting in specific value ranges (not usable if Display Type = "Percentage")
Intervals	A list of intervals for overwriting the formatting in specific value ranges

Intervals

PARAMETERS	DESCRIPTION
Interval Start	The interval is applied from this value
Interval End	The interval is applied up to this value. If there is an interval with a start at this value, the other interval is used for the exact value for End and the defined interval is only used for values that are smaller than the end.
Display	If "Formatted Value" is selected, the value is formatted with adjusted parameters; if "Constant" is selected, the display is replaced by a constant value.
Constant	Defines the text that is to be displayed without further formatting for values within this interval. (only available if Display = "Constant")
Conversion Factor	Before formatting, the value is divided by the conversion factor (only available if display = "Formatted Value")
Overwrite Prefix	If the switch is activated, the prefix defined in the interval is used instead of the prefix of the main configuration (only available if display = "Formatted Value")
Prefix	Prefix used within the interval (only available if the "Overwrite Prefix" switch is active)
Overwrite Suffix	If the switch is activated, the suffix defined in the interval is used instead of the suffix of the main configuration (only available if display = "Formatted Value")
Suffix	Suffix used within the interval (only available if the "Overwrite Suffix" switch is active)
Overwrite Digits	If the switch is activated, the digits defined in the interval is used instead of the digits of the main configuration (only available if display = "Formatted Value")
Digits	Decimal places used within the interval (only available if the "Overwrite Digits" switch is active)
Overwrite Icon	If the switch is activated, the icon defined in the interval is used instead of the icon of the main configuration
Icon	Icon used within the interval (only available if the "Overwrite icon" switch is active)

DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as a slider if it also fulfills the following conditions:

Does not fulfill the conditions of any of the following presentations:

- Shutter
- Enumeration

Variable profile with the following parameters:

- Max value > Min value
- Any increment, but has an influence on the increment of the slider

When using a variable profile, some parameters of the slider are derived

GRADIENT

The gradient depends on the profile name.

Temperature for profile name
~Temperature
~Temperature.Difference
~Temperature.Fahrenheit
~Temperature.Room
Color temperature for profile name "~TWColor"
Default otherwise

VARIABLE USAGE

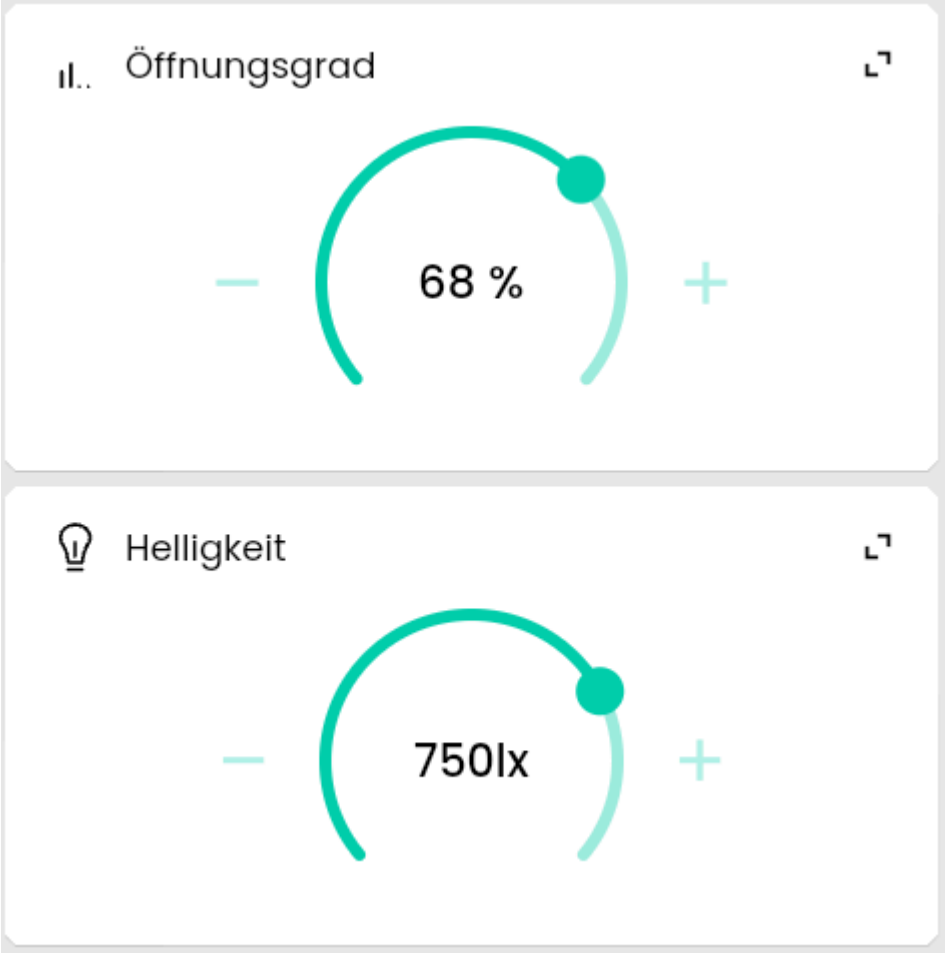
The use depends on the profile name.
Temperature for profile name
~Temperature
~Temperature.Difference
~Temperature.Fahrenheit
~Temperature.Room
Color temperature for profile name "~TWColor"
Intensity if profile name begins with "~Intensity." begins
Volume for profile name "~Volume"
Progress for profile name "~Progress"
None of these otherwise

DISPLAY TYPE

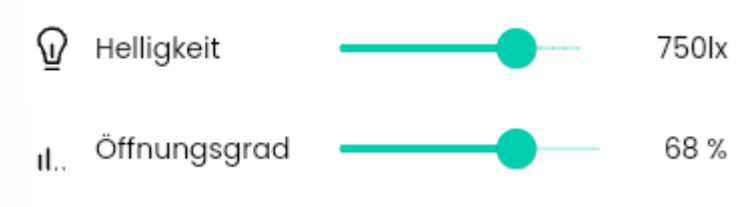
"Percentage" if suffix "%", otherwise "Absolute"

APPEARANCE IN VISUALIZATION TILE

AS OWN TILE



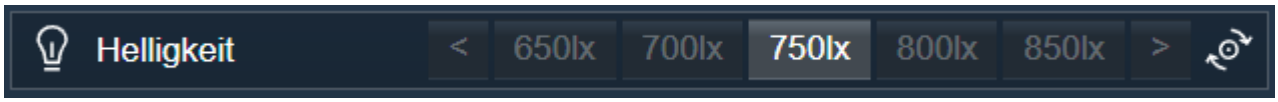
WITHIN A LIST



APPEARANCE IN THE WEBFRONT

The WebFront differentiates according to the display type and has a separate display for each case:

DISPLAY TYPE "ABSOLUTE"



DISPLAY TYPE "PERCENTAGE"


Öffnungsgrad
68 %

Stream

Streams represent the video output, for example from a surveillance camera.

REQUIREMENTS

A Stream is a Medium of type Stream.

APPEARANCE IN TILE VISUALIZATION

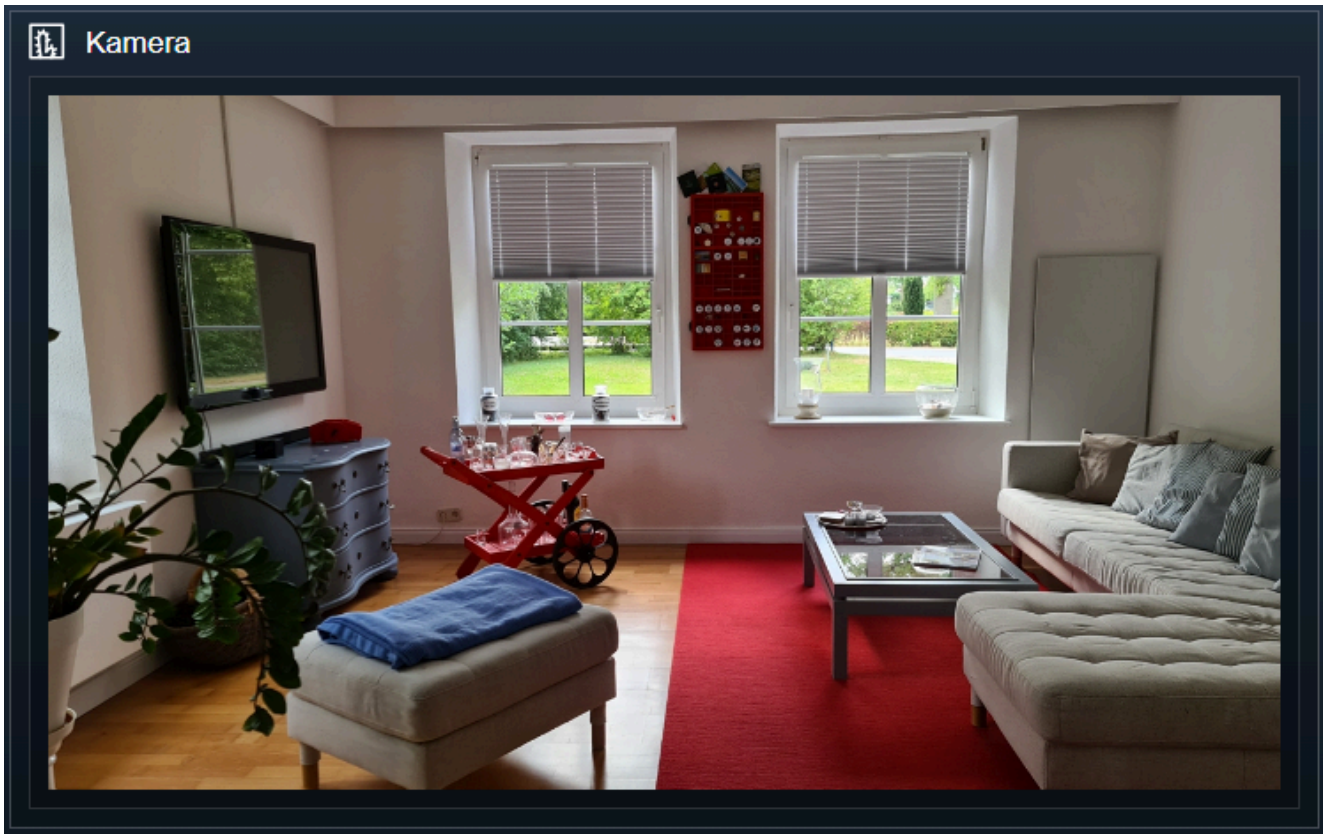
AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT


Kamera

Scenes

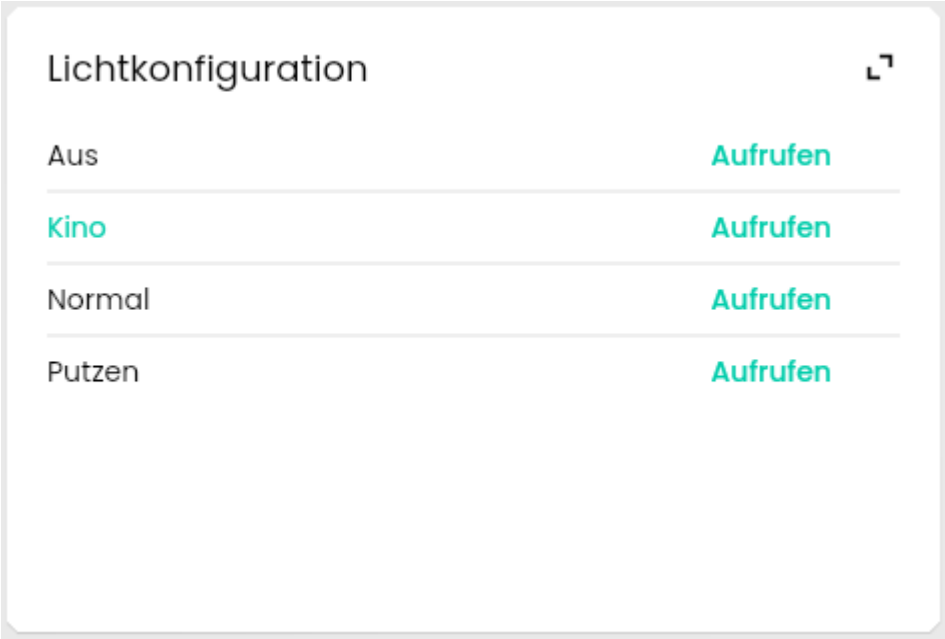
Require: Symcon >= 7.0
Various device settings can be saved together as a scene and called up again.

REQUIREMENTS

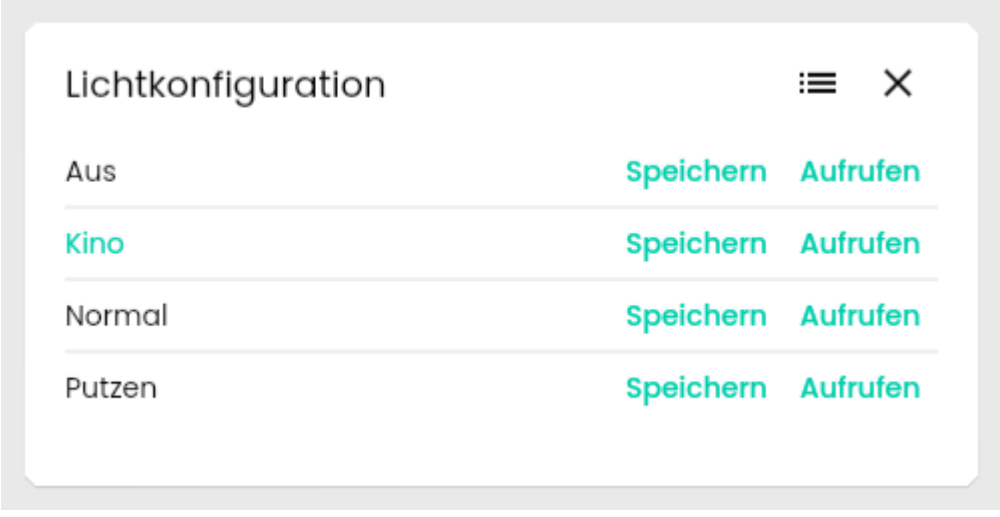
Scenes require an Instance of the module Scene Control.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE

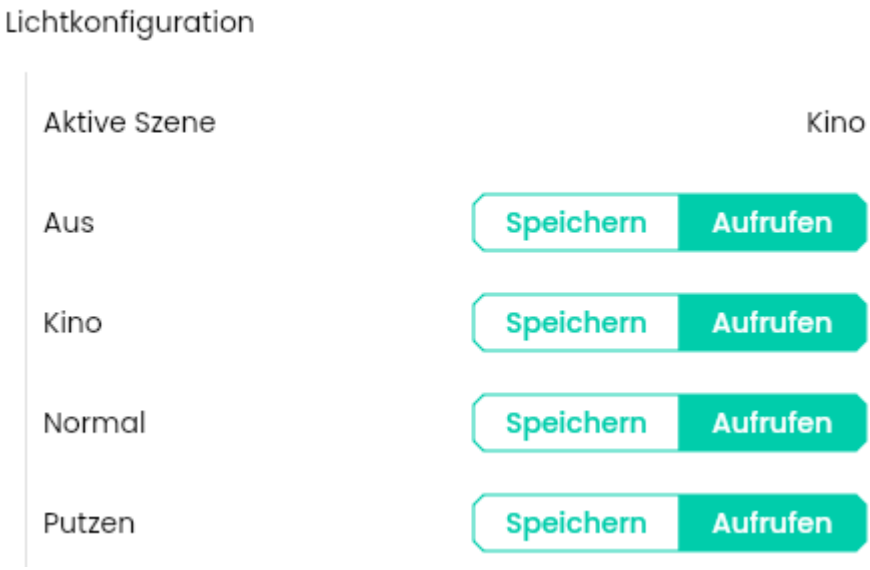


IN FULL SCREEN



WITHIN A LIST

In the list view, the visualization falls back to the presentation List.



APPEARANCE IN THE WEBFRONT

This presentation is not supported by the WebFront. Instead, the instance is displayed as List.

Thermostat

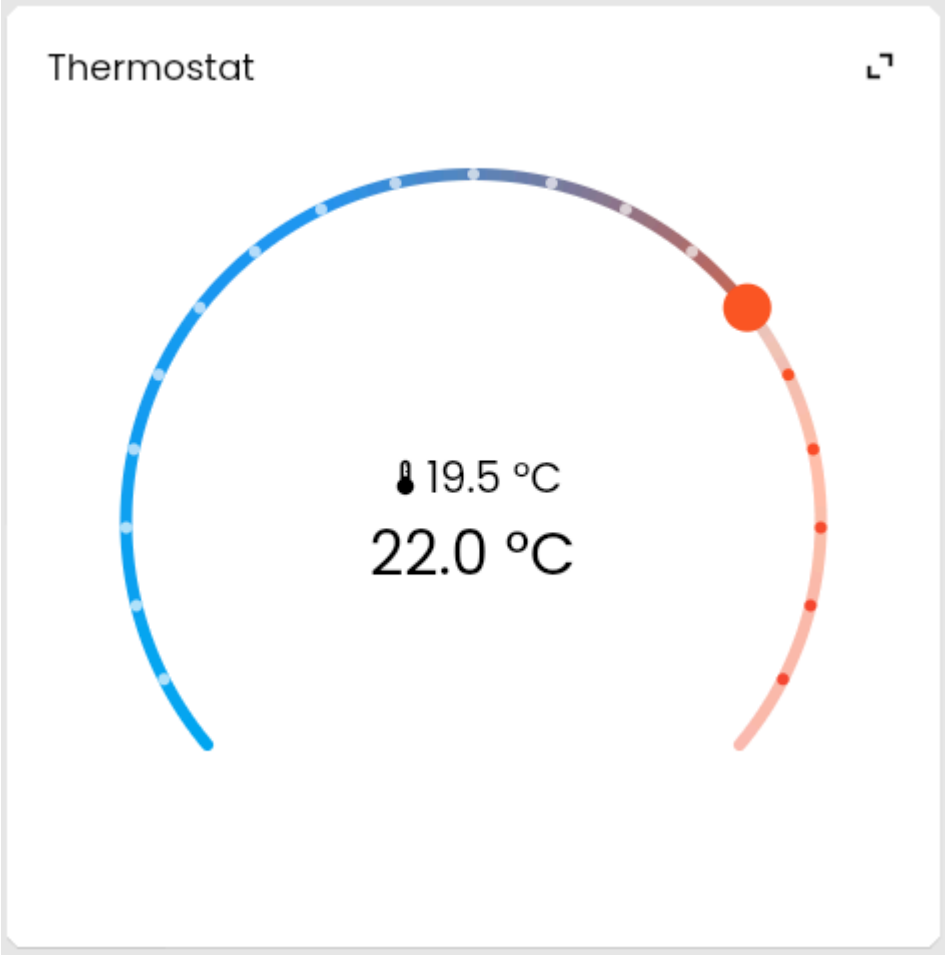
A Thermostat allows setting a desired temperature combined with a display of the current temperature.

REQUIREMENTS

An Instance can be displayed as a Thermostat if the following child objects are present:
Set temperature:
uses the variable presentation Slider
usage type of the variable is "Temperature"
Current temperature:
Uses the variable presentation Value Presentation
usage type of the variable is "Temperature"

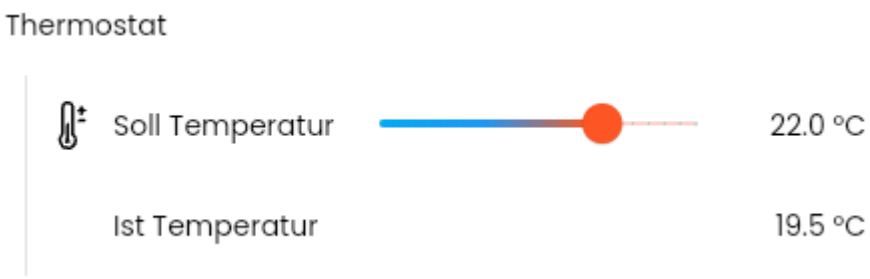
APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST

In the list view, the visualization falls back to the presentation List.



APPEARANCE IN THE WEBFRONT

This presentation is not supported by the WebFront. Instead, the instance is displayed as List.

Sound

Audio files can be integrated and played back in the visualization.

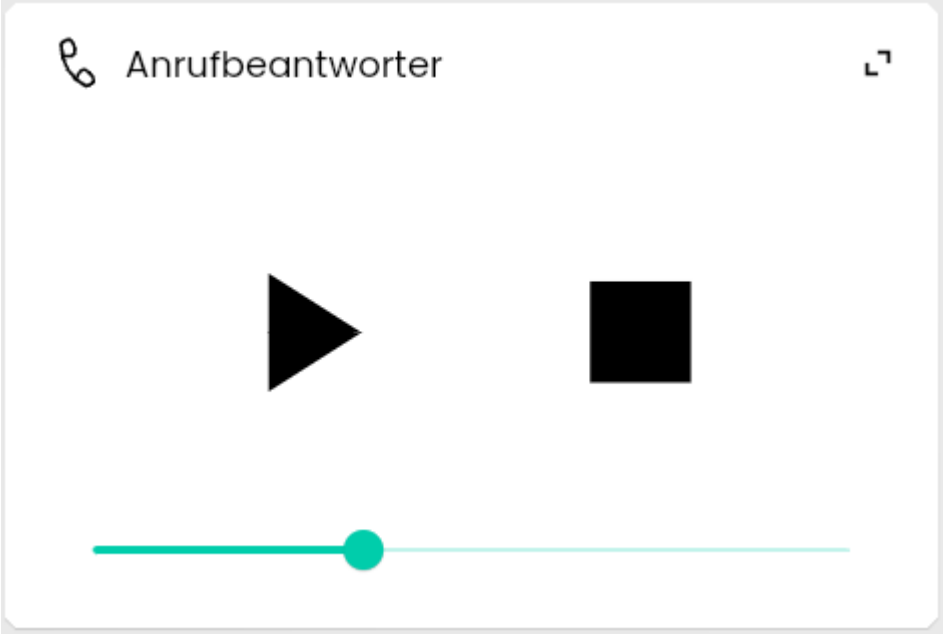
REQUIREMENTS

A Sound is a Medium of the type Sound.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE

The progress bar and the stop button can be omitted depending on the size of the tile.



WITHIN A LIST



APPEARANCE IN THE WEBFRONT

The display may vary depending on the browser.



Web Content

Web Content can display any HTML content and therefore offers experts the option of displaying individual elements. By default, CSS displays text within the presentation in the same way as the rest of the visualization. In addition, the address of any website can be entered to display it in the visualization.

REQUIREMENTS

Web Content is a Variable with the presentation Web Content. To use this presentation, the variable must fulfill the following conditions:

Type String
No configured Variable action

PARAMETERS

PARAMETERS	DESCRIPTION
Display Type	For "HTML content", the variable value is interpreted and displayed as HTML. For "Website", the address of a website can be specified as long as it does not block embedding.
Remove Padding	If activated, the borders of the tile are removed.

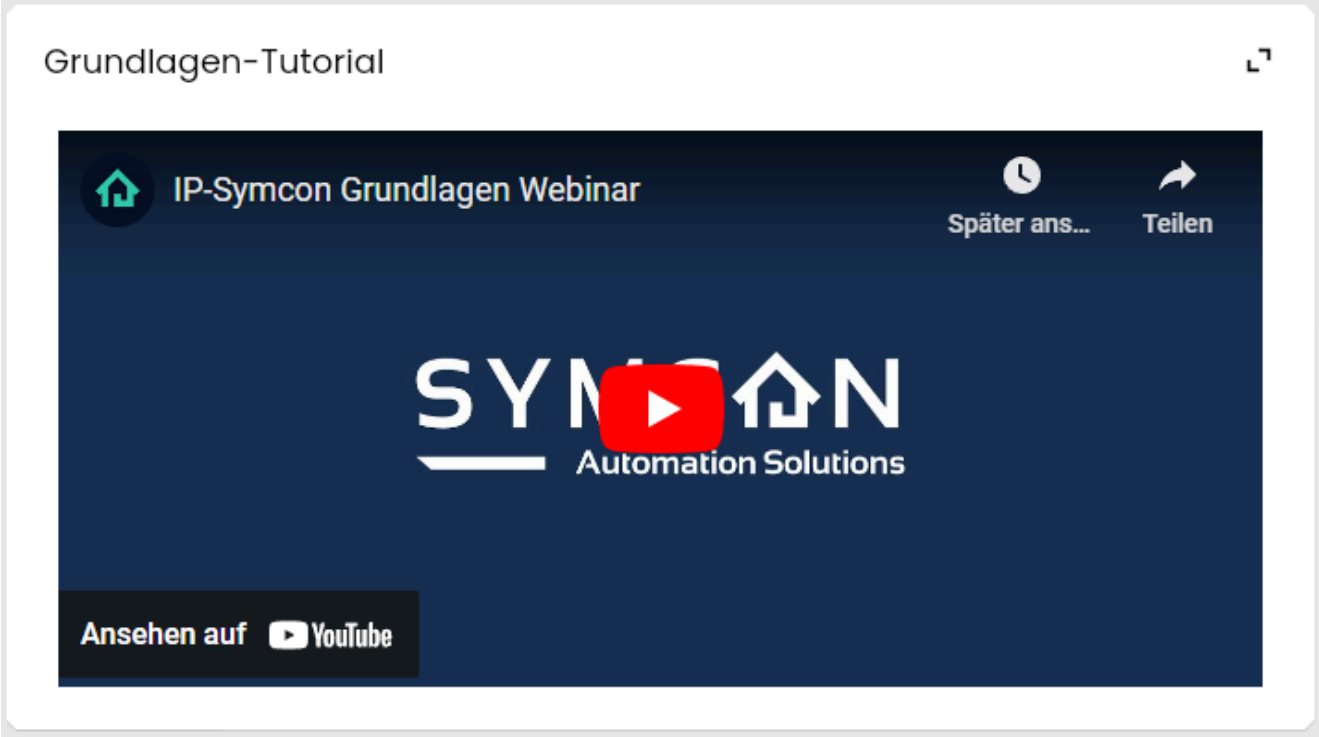
DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as Web Content if it also fulfills the following conditions:

Selected variable profile ~HTMLBox

APPEARANCE IN TILE VISUALIZATION

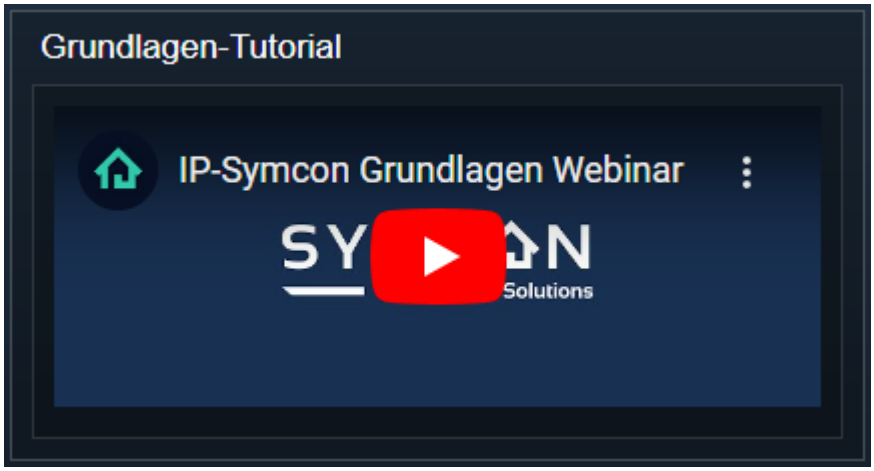
AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Value Presentation

Displays a value, optionally with background color or value progression.

REQUIREMENTS

A value presentation is a variable with the presentation value presentation. To use this presentation, the variable must fulfill the following conditions:

No set up Variable action

PARAMETERS

Depending on the variable type (Boolean, Integer, Float, String), different parameters are available. The following apply to all of them.

PARAMETER	DESCRIPTION
Default icon	The icon that is displayed. It can be overwritten by other parameters
Default color	The color with which the value is highlighted
Prefix	Is written in front of the variable value during formatting (Not available for Boolean)
Suffix	Is written after the variable value during formatting (Not available for Boolean)

FLOAT AND INTEGER

PARAMETER	DESCRIPTION
Usage type	Use of the variable in summarized presentations. The value does not have a direct effect on the display of the variable, but only on its use in summarized displays.
Display type	With "Percentage", the values are displayed as a percentage. This means that the minimum value is displayed as 0 and the maximum value as 100, regardless of their absolute values. Other values are also scaled accordingly. No conversion takes place with "Absolute".
Thousands separator	Defines the thousands separator used when formatting the value
Minimum value	Minimum value of the variable (only usable if display type = "Percentage")
Maximum value	Maximum value of the variable (only usable if display type = "Percentage")
Decimal places	The number of decimal places that are displayed when formatting the value (only available for variables of type Float)
Decimal separator	Defines the decimal separator used when formatting the value (only available for variables of type Float)

PARAMETER	DESCRIPTION
Icon	The Icon that is used for the variable
Use updated parameters for specific intervals	Activates a list of intervals that can override the formatting in specific value ranges (not usable if display type = "Percentage")
Intervals	A list of Intervals for overwriting the formatting in specific value ranges

Intervals

PARAMETERS	DESCRIPTION
Start of interval	The interval is applied from this value
End of interval	The interval is applied up to this value. If there is an interval with a start at this value, the other interval is used for the exact value for End and the defined interval is only used for values that are smaller than the end.
Display	If "Formatted value" is selected, the value is formatted with adjusted parameters; if "Constant" is selected, the display is replaced by a constant value.
Constant	Defines the text that is to be displayed without further formatting for values within this interval. (only available if display = "Constant")
Conversion factor	Before formatting, the value is divided by the conversion factor (only available if display = "Formatted value")
Overwrite prefix	If the switch is activated, the prefix defined in the interval is used instead of the prefix of the main configuration (only available if display = "Formatted value")
Prefix	Prefix used within the interval (only available if the "Overwrite prefix" switch is active)
Overwrite suffix	If the switch is activated, the suffix defined in the interval is used instead of the suffix of the main configuration (only available if display = "Formatted value")
Suffix	Suffix used within the interval (only available if the "Overwrite suffix" switch is active)
Overwrite decimal places	If the switch is activated, not the decimal places of the main configuration are used, but the decimal places defined in the interval (only available if display = "Formatted value" and variable of type Float)
Digits	Decimal places used within the interval (only available if the "Overwrite decimal places" switch is active)
Overwrite icon	If the switch is activated, the icon defined in the interval is used instead of the icon of the main configuration
Icon	Icon used within the interval (only available if the "Overwrite icon" switch is active)
Overwrite color	If the switch is activated, the color defined in the interval is used instead of the color of the main configuration
Color	Color used within the interval (only available if the "Overwrite color" switch is active)

BOOLEAN AND STRING

PARAMETERS	DESCRIPTION
Multiline	(String only) Displays the variable value in multiple lines
Options	The individual Options of the display, which allow specific texts, colors and icons to be displayed for different variable values

Options

PARAMETER	DESCRIPTION
Value	The option is applied for this variable value
Label	Text displayed for this option
Overwrite icon	If set, the default icon for this presentation is overwritten if the variable has the value of this option
Icon	Icon for the variable if the variable has the value of this option (only available if "Overwrite icon" is set)
Color	Color that is used to display this option inside the enumeration

VALUE PROGRESSION

If the Logging is activated for the variable, a preview of the value history is displayed.

DISPLAY AS LEGACY PROFILE

If a variable uses the presentation Legacy profile, it is displayed as a value presentation if it fulfills the following conditions:

Does not fulfill the conditions of one of the following presentations:

- Date/Time
- Color Presentation
- Multi-row Textbox
- HTML Box

When using a variable profile, some parameters of the value display are derived

USE OF THE VARIABLES

The usage type depends on the profile name.

Temperature for profile name

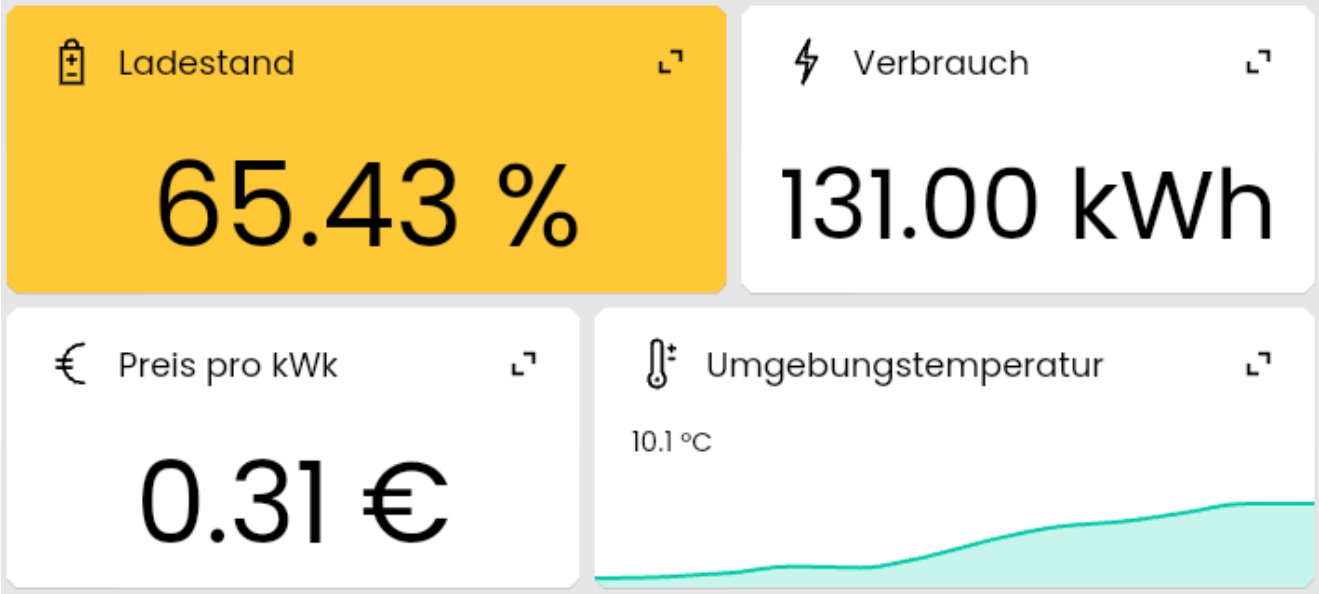
- ~Temperature
- ~Temperature.Difference
- ~Temperature.Fahrenheit
- ~Temperature.Room

DISPLAY TYPE

“Percentage” if suffix ‘%’, otherwise ”Absolute”

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE

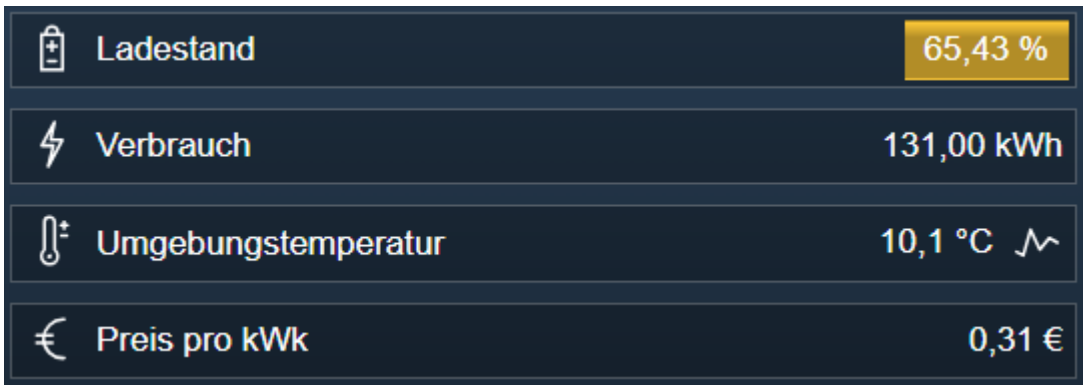


WITHIN A LIST



APPEARANCE IN THE WEBFRONT

No preview of the value history is displayed in the WebFront.



Value Input

Require: Symcon >= 5.0
A Value Input allows you to change values as free text.

REQUIREMENTS

A Value Input is a variable with the presentation Value Input. To use this spresentation, the variable must fulfill the following conditions:
configured Variable action
Variable type Integer, Float, or String

PARAMETERS

PARAMETER	DESCRIPTION
Prefix	Is written in front of the variable value during formatting
Suffix	Is written after the variable value during formatting
Multiline Input	If active, input over multiple lines is possible. Otherwise, the input is restricted to a single line (only available for variables of type String)



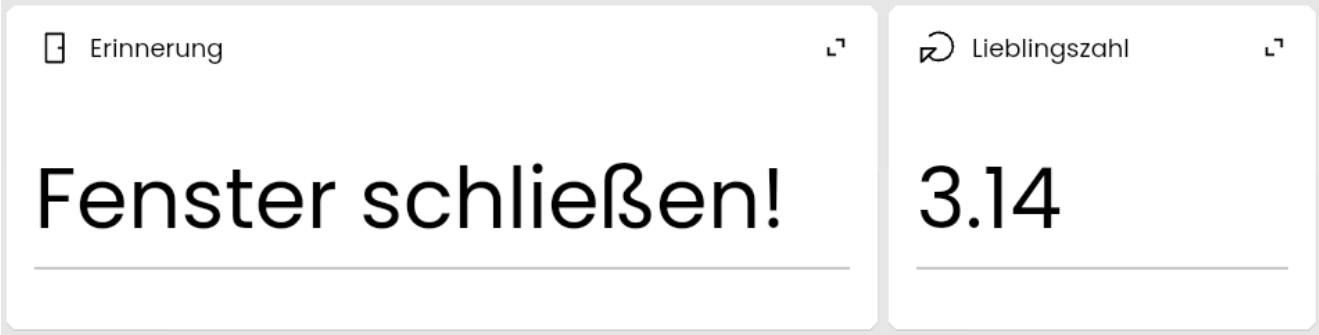
If the variable is numeric, inputs that are not numbers are filtered out

DISPLAY AS LEGACY PROFILE

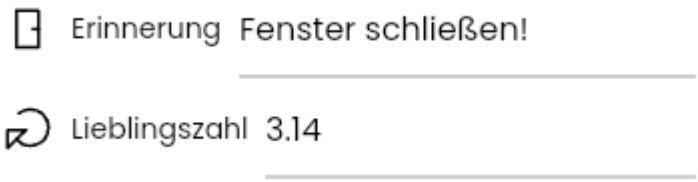
If a variable uses the presentation Legacy profile, it is displayed as a Value Input
no Variable profile or variable profile "~TextBox"
When using a variable profile, the parameter "Multiline Input" is active for the variable profile "~TextBox" and inactive otherwise.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Schedule Event

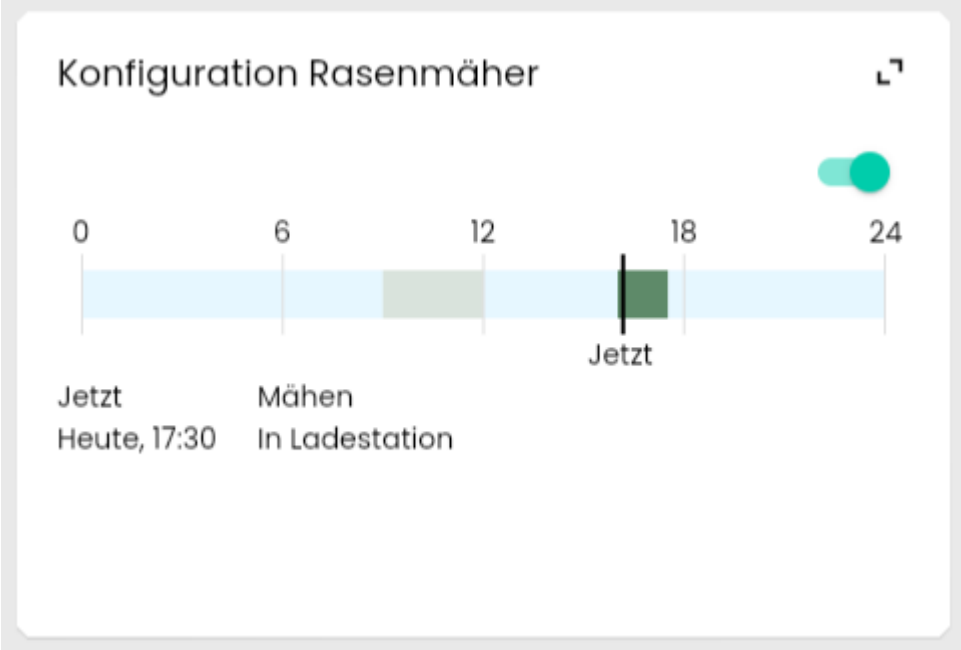
A Schedule Event can be viewed and configured in the visualization.

REQUIREMENTS

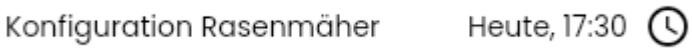
A Schedule Event is an Event of type Schedule.

APPEARANCE IN TILE VISUALIZATION

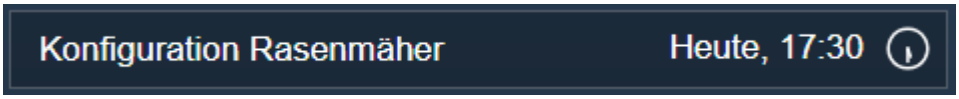
ALS OWN TILE



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Cyclic Event

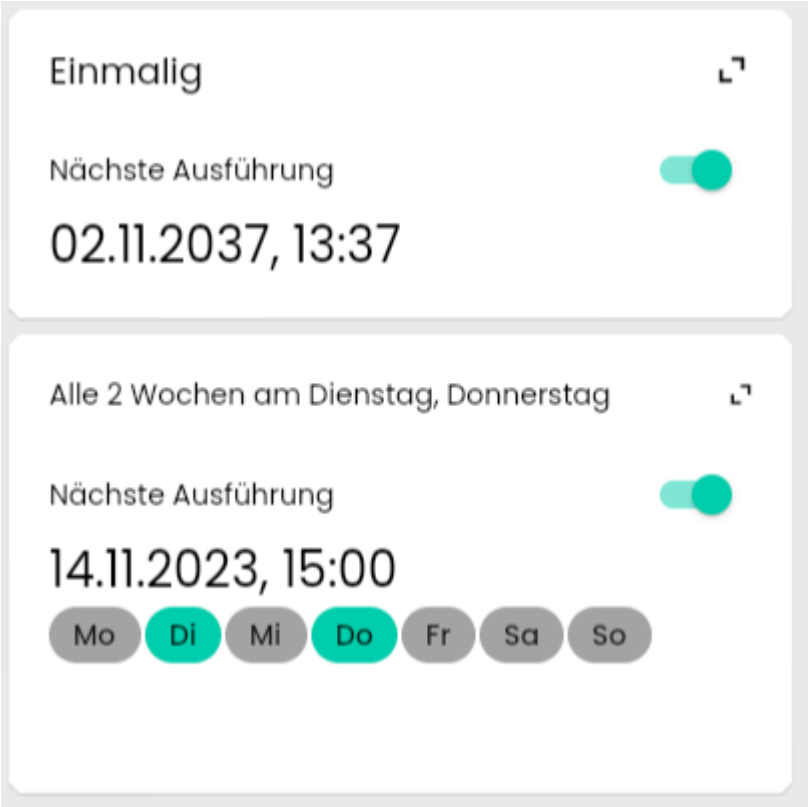
A Cyclic Event can be viewed and configured in the visualization.

REQUIREMENTS

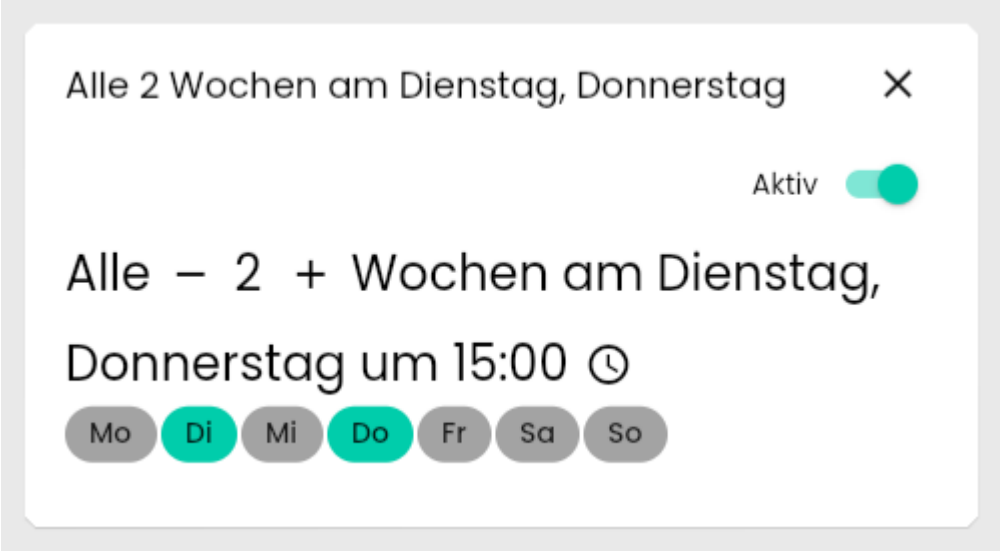
A cyclic event is an Event of type Cyclic.

APPEARANCE IN TILE VISUALIZATION

AS OWN TILE



IN FULL SCREEN



WITHIN A LIST



APPEARANCE IN THE WEBFRONT



Icons

Each object in Symcon can have an icon. In addition, icons can be present within Variable presentations or inherited through linked structures (see Links).
Icons are prioritized in the following order:

the icon assigned to the link object (when linking)
the icon assigned to the (linked) object
depending on the object type:
    For automations: the icon *caret-right*
    For media: the icon *image*
    For variables: Icon based on the presentation

Icons that are set based on the variable presentation are automatically updated when the variable value changes.
If an inherited icon is to be explicitly removed, the special icon *Transparent* can be selected.

ICON OVERVIEW

The more than 3000 classic icons from Font Awesome can be used in the tile visualization: Usable icons. These icons are displayed in the Light style. In addition, all Brand icons can be used. Finally, there are a number of icons that have been specially created for use in Symcon:

heatpump

marquee-closed

marquee-half

marquee-open

terrace-door-closed

terrace-door-open

terrace-door-tilted

volant-closed

volant-half

volant-open

window-left-closed-right-closed

window-left-closed-right-open

window-left-closed-right-tilted

window-left-closed

window-left-open-right-closed

window-left-open-right-open

window-left-open-right-tilted

window-left-open

window-left-tilted-right-closed

window-left-tilted-right-open

window-left-tilted-right-tilted

window-left-tilted

window-right-closed

window-right-open

window-right-tilted

window-roof-closed

window-roof-open

The Legacy icons can be used in the WebFront.

THE ICON *TRANSPARENT*

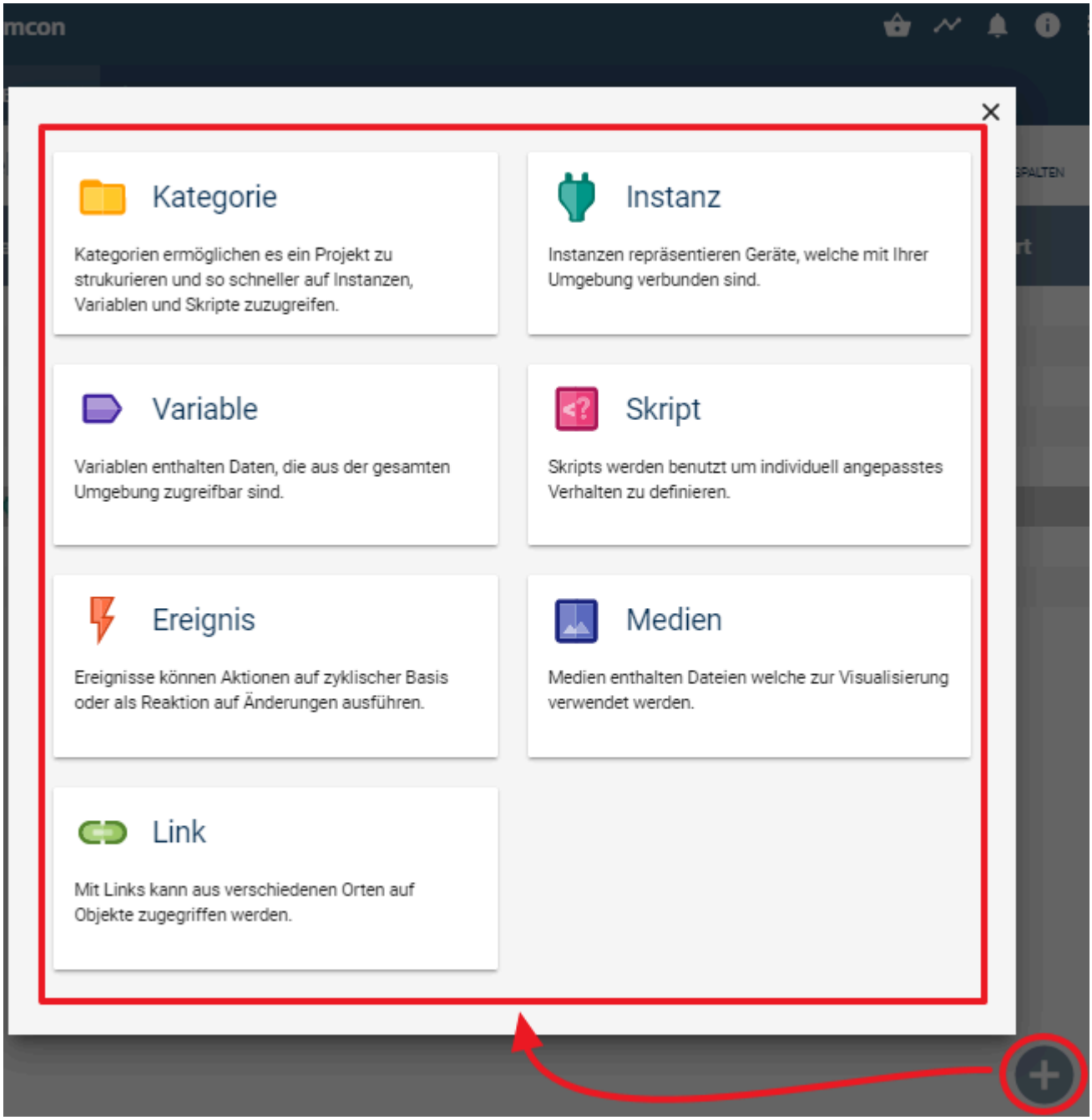
The *Transparent* icon is a special case. If this icon is selected, another icon that is lower in the hierarchy (see above) can be overwritten so that no icon is displayed. This can be used as a style element, e.g. to not display an icon for linked object categories, even though the category had an icon assigned.

Procedures

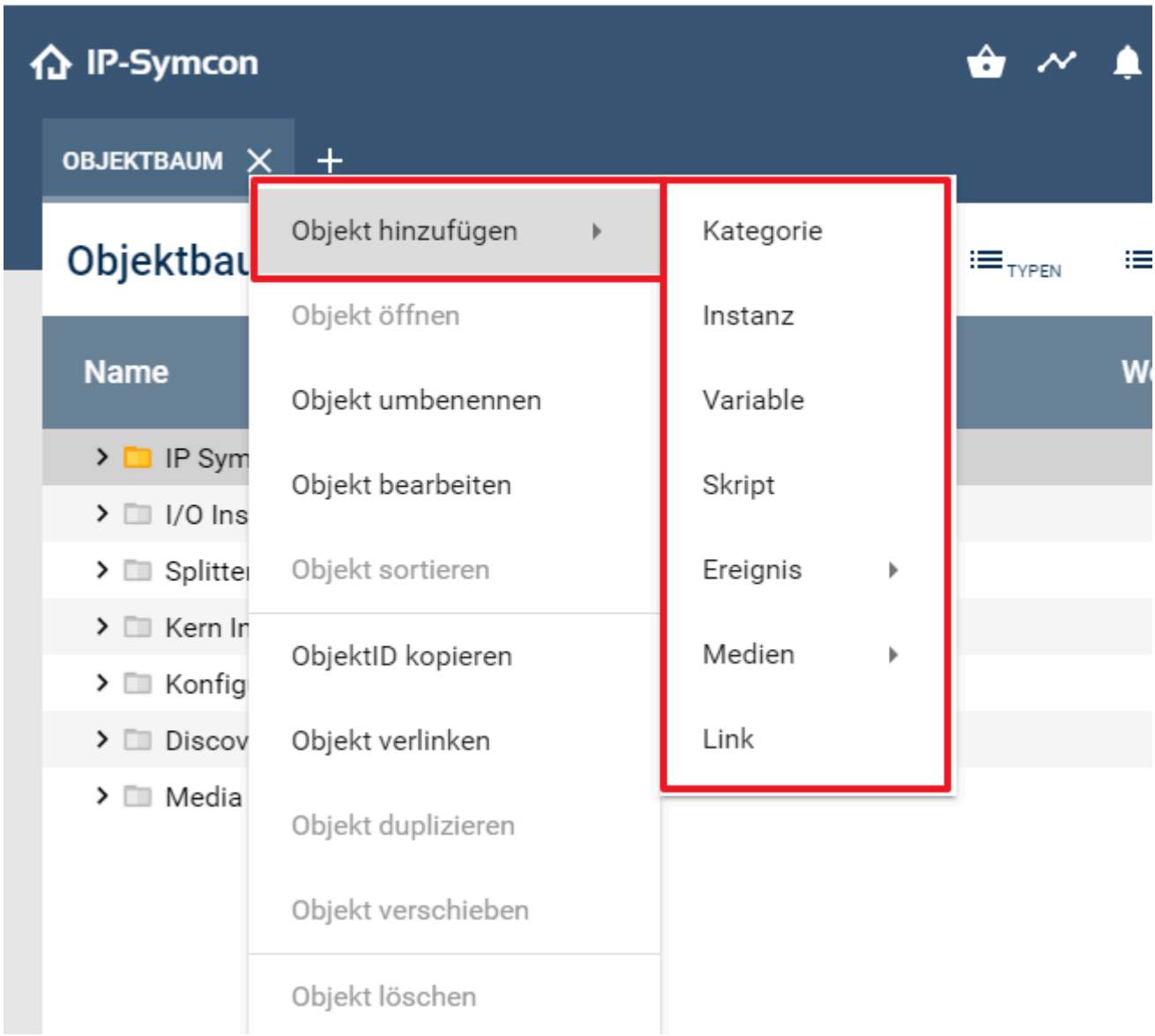
In the following section, basic and often used functions are explained.
A general start is found at Quick Start.

Add object

In order to "Add a new object", the "+" in the lower right corner of the object tree must be clicked. The following dialog and selection of various objects appears:



Alternatively, "right mouse button" can be used within the object tree to click on the desired position in the tree. The following context menu opens:


IP-Symcon
OBJEKTBAUM
Objektbau
TYPEN
Name
IP Sym
I/O Ins
Splitte
Kern In
Konfig
Discov
Media
Objekt hinzufügen
Objekt öffnen
Objekt umbenennen
Objekt bearbeiten
Objekt sortieren
ObjektID kopieren
Objekt verlinken
Objekt duplizieren
Objekt verschieben
Objekt löschen
Kategorie
Instanz
Variable
Skript
Ereignis
Medien
Link

Replace devices

If a device is defective and has to be physically replaced, this can be implemented in IP-Symcon without creating a new instance. This has the advantage that the object IDs of the instance and the associated variables do not change. This means that any linked scripts do not have to be adapted to the new IDs.

REPLACE DEVICE WITHIN AN INSTANCE

Devices are addressed and integrated in IP-Symcon via so-called instances.
System-specific addresses are often available on the configuration page of the respective entity.
This can be changed without changing the object ID.

> If the system used has a configurator, it can be updated and also adopts the changes

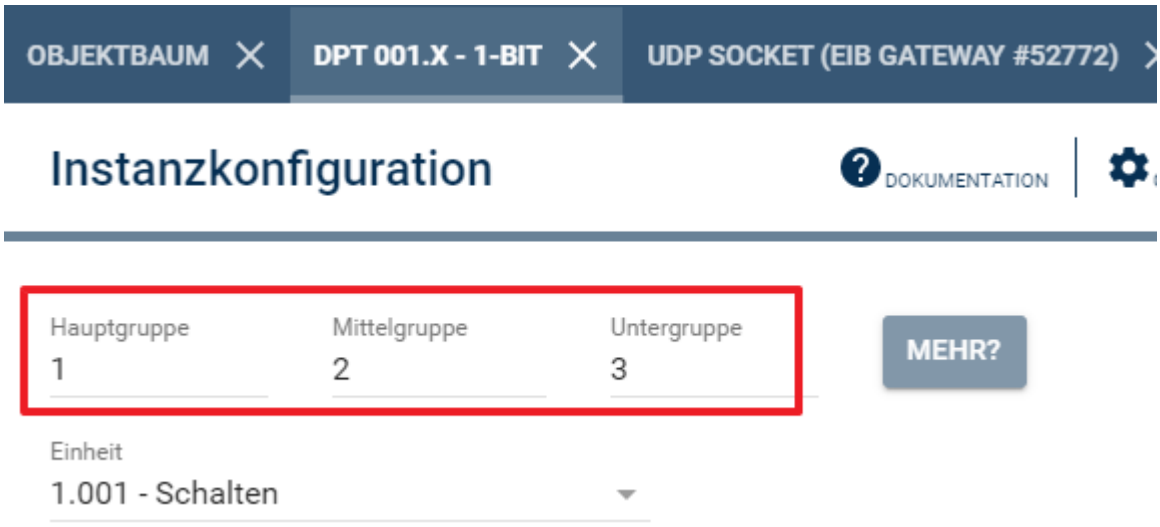
EXAMPLES

KNX

For replacement, the configuration of the defective device must be opened in IP-Symcon and the three-level address of the old device must be exchanged with that of the new one. The address can be checked in the configurator.

Open the KNX configurator
"Search" but do not "Create" an instance
Memorise the new address and open the configuration of the old device
Enter new address (see screenshot) and save with "Apply".
Return to the configurator and update via "Search".

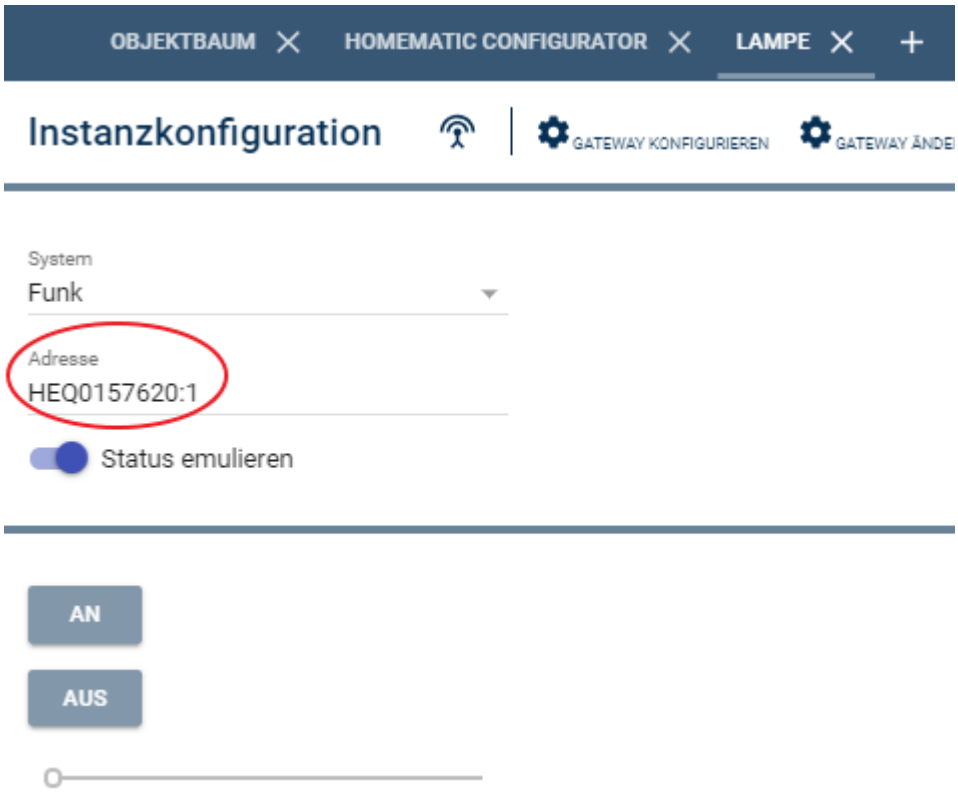
The new address should now have an InstanceID.



The three-level address in the marked area must be replaced and saved with "Apply".

HOMEMATIC

For replacement, the configuration of the defective device must be opened in IP-Symcon and the address of the old device must be replaced with that of the new one. The address is either on the device itself or can be selected via "Search".



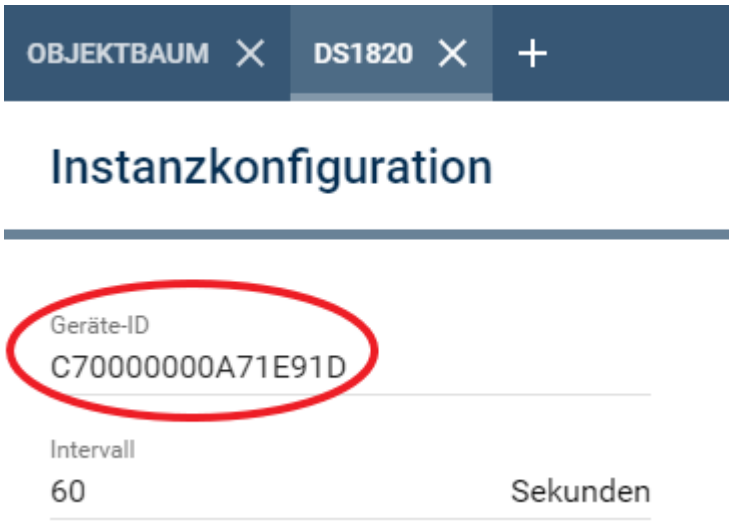
The address in the highlighted area must be replaced and saved with "Apply".

ONEWIRE

For replacement, the configuration of the defective device must be opened in IP-Symcon and the 16-digit address of the old device must be replaced for that of the new one. The address can be checked in the configurator.

Open OneWire configurator
"Search" but do not "Create" an instance
Memorise the new address and open the configuration of the old device
Enter new address (see screenshot) and save with "Apply".
Return to the configurator and update via "Search".

The new address should now have an InstanceID.



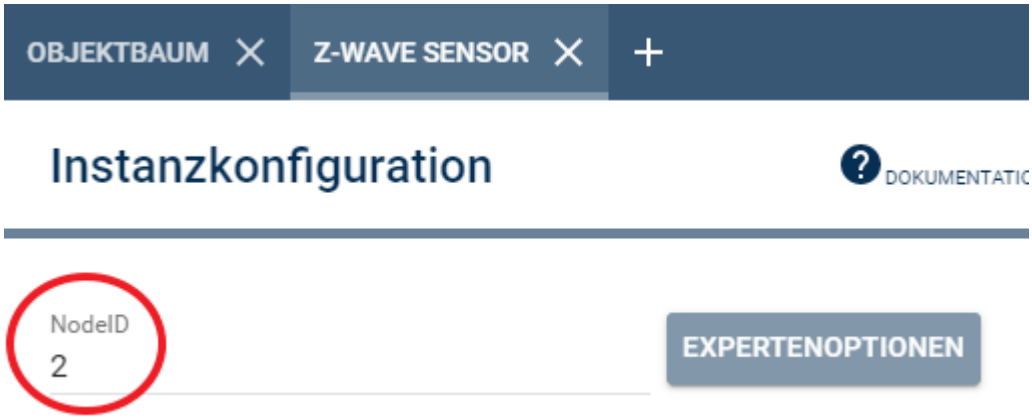
The 16-digit address in the marked area must be replaced and saved with "Apply".

Z-WAVE

For replacement, the configuration of the defective device must be opened in IP-Symcon and the NodeID of the old device must be replaced for that of the new one. The NodeID can be checked in the configurator.

Open Z-Wave Configurator
Teach device but do not "Create" an instance
Memorise NodeID and open the configuration of the old device
Enter new NodeID (see screenshot) and save with "Apply".
Return to the configurator and "Refresh"

The new NodeID should now have an InstanceID.



The NodeID in the marked area must be replaced and saved with "Apply".

Integrate devices

Devices are addressed in IP-Symcon via so-called instances.



The principle of creating the devices has changed fundamentally from version 1 onwards. I/O instances, splitters or similar no longer have to be created/connected. IP-Symcon does this automatically. IP-Symcon also creates the status variables automatically.

CREATE INSTANCE

Configurators are available for the following systems for a more convenient configuration of devices. The appropriate configurator can be added to the object tree within the web-based Management Console using the "+" button.

- digitalStrom
- Eaton xComfort
- KNX
- HomeMatic
- LCN
- MQTT
- Siemens OZW
- Z-Wave
- 1-Wire

For all other systems, the respective device must be added by creating a suitable instance. The creation dialog can be accessed in the object tree using the "+" -> "Instance" button. IP-Symcon takes care of creating the required gateways, splitters and I/O's.
In the "Messages" tab it can be checked whether an instance has an error. A message with a yellow or red background is interesting. This means there is still one setting missing for an instance to work. A "double click" on the respective message leads directly to the section of IP-Symcon where the said setting is still required.
Access to the gateway and I/O interfaces belonging to each device is possible via the "Configure gateway" or "Configure interface" gear wheel in the device "Configuration" tab. The required settings can be checked there and improved if necessary.
The "Module reference " area contains the setup instructions for the respective devices/instances, either as a description or as a video tutorial.

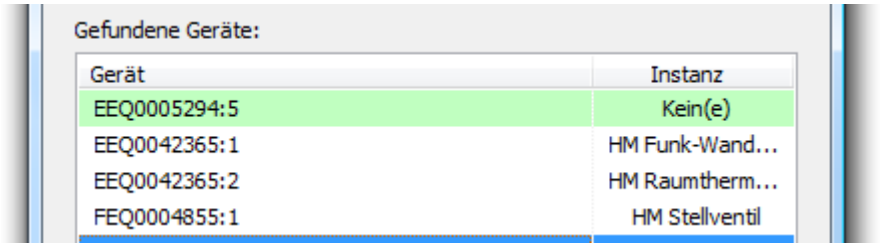
Search devices

IP-Symcon has a very convenient function that allows the user to quickly and easily add new devices to the system. The prerequisite is that the new, to be taught devices send messages of their own accord (e.g. wireless temperature sensors). Alternatively, the user can - or must - press a "teach button" or simply a button on the remote control. With some systems, such as the 1-Wire system, IP-Symcon can specifically search for connected, unknown devices. According to the motto: "Hello, who is it?". And the devices respond with: "It's me, the brightness sensor".

Devices can be searched for via the Configurators of the respective systems.

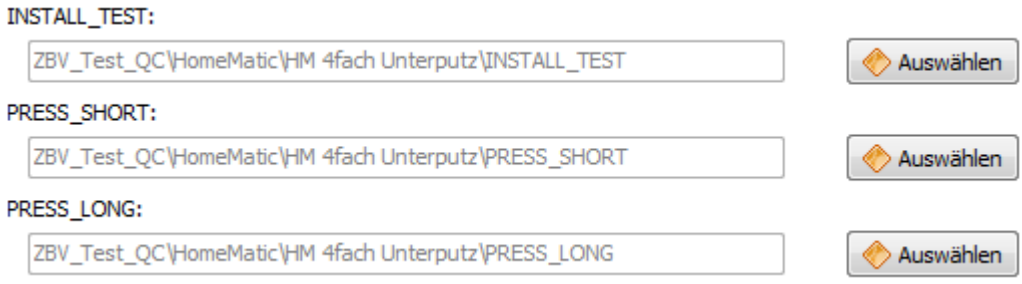
EXAMPLES

The following example shows the received devices of the HomeMatic system:

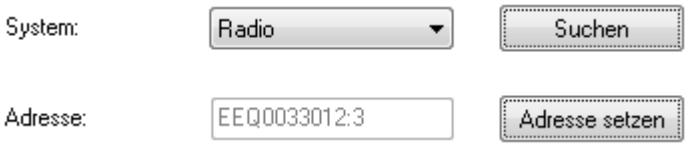


It may take several minutes for all devices to appear in the list!

A green background indicates a new module. It can then be selected by double-clicking. Depending on the properties of the device, IP-Symcon automatically creates variables that reflect the functionality. The names and the location can be changed at any time.
The following example shows the three variables of a wireless switch or remote control of the HomeMatic system:



With the HomeMatic system, it must also be specified for the search whether wireless or wired devices are to be searched for.
If the BidCoS serial number is known, it can also be entered directly:



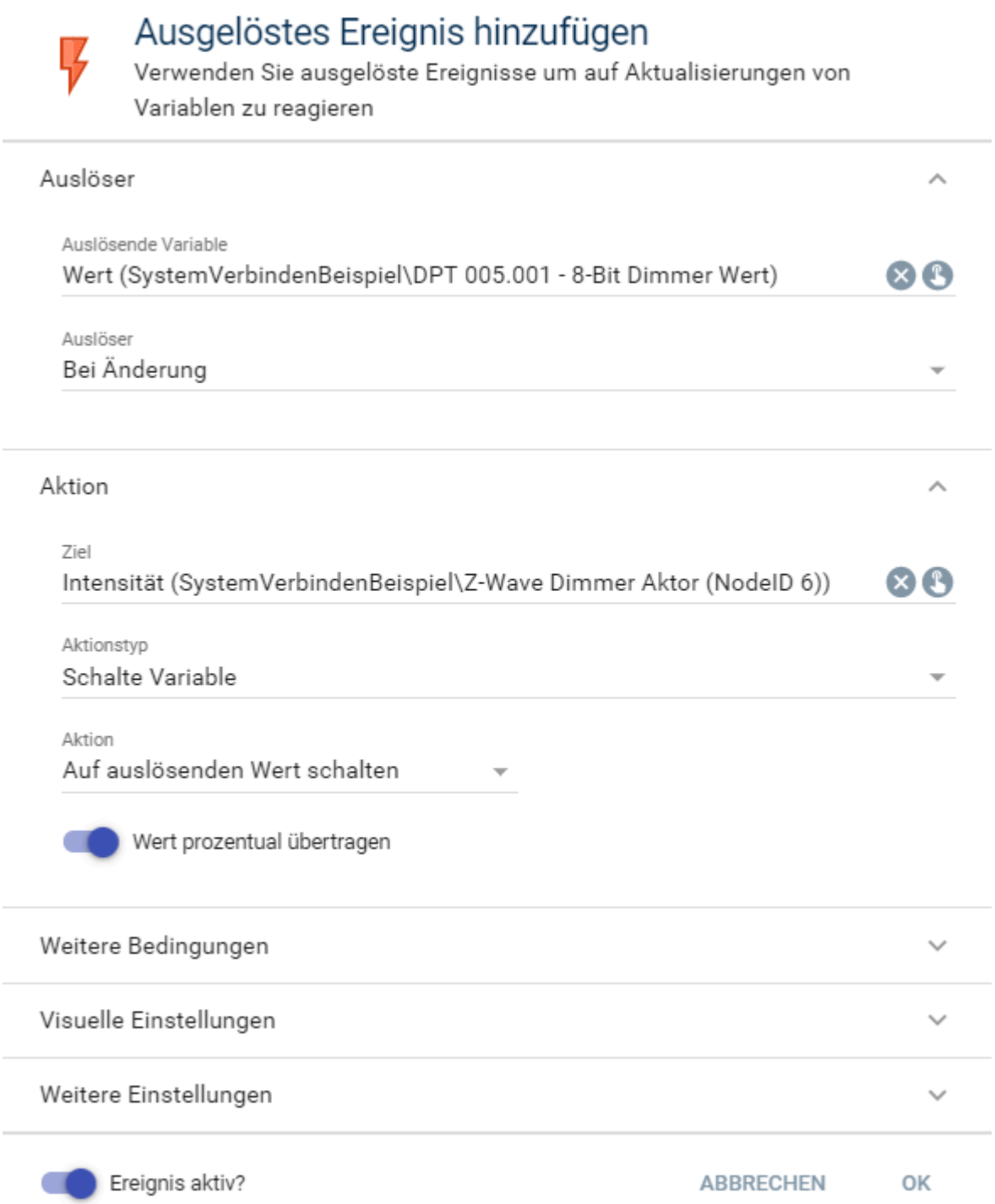
Please note that its variables are only filled with values the second time a data record is received from the respective device.

Connect Systems via Events

IP-Symcon offers the very comfortable option to transfer values from one system directly to another. For this purpose an Event can be added, which can set another value to the trigger value when the trigger value is changed. It is irrelevant whether the two values belong to different systems. In the example below, a KNX dimmer value is transferred to a Z-Wave dimmer.

EXAMPLE

A KNX dimmer value can be transmitted to a Z-Wave lamp with dimmer value. For this purpose, a change of the KNX value is reacted to and the new value is transferred to the dimming value of the Z-Wave lamp which is then being switched.
In the object tree, the two device instances have already been created and are functioning properly on their own. Now an event must be created. To do this, an event can be added via the "+" at the bottom right of the object tree.
The event must be configured as follows.
The dimming value, whose value is to be transferred on change, is selected as the triggering variable. In this example it is the value of DPT 005.001.
Under Action, the intensity of the Z-Wave lamp with dimmer actuator is selected as "Target".
As action type "Switch variable" and as action "Switch to triggering value" must be selected.
By clicking "OK" the event is created and the configuration is saved.



From now on, every time the KNX dimmer value changes, the intensity of the Z-Wave instance will be dimmed to the same value.
The object tree should then look like this.

24669	DPT 005.001 - 8-Bit Dimmer Wert	KNX DPT 5		
25908	Wert	Integer	50 %	14:08:41
53115	Z-Wave Dimmer Aktor (NodeID 6)	Z-Wave Module		
16977	Intensität	Integer	50 %	14:08:41
37072	Ereignis: Bei Änderung von "Wert (SystemVerbindenBeispiel\DP...	Ereignis		14:08:41
49228	Status	Boolean	An	14:08:41
10917	Schalte alle (Modus)	Integer	Aus	23.06....
25280	Farbe (Warmweiß)	Integer	100 %	23.06....
20917	Farbe (Kaltweiß)	Integer	0 %	23.06....
38443	Farbe (Rot, Green, Blau)	Integer		23.06....

Reuse scripts



This is an example that can also be mapped with events. This article is more about System variables and reusability of script sections.

This example is about a corridor light control that is the same on all floors. If possible, this should not be programmed three times.
One possibility would be to create 3 scripts and then copy and paste the content of one script into the individual scripts. This is certainly possible for a quick fix. However, this also leads to a certain redundancy and if something needs to be improved at one point in the script, this does not have to be done once but three times.
As an example for code reusing, a script with the name "light control (xComfort, Corridor, Motion detector, Brightness control)" can be created. The name should be descriptive and include what systems, devices and status it uses.
The following code could be the content:

```
//If no movement - see "System Variables" documentation for meaning of
$_IPS['VALUE']
if(!$_IPS['VALUE']) {
MXC_SwitchMode($lampID, false);
}
//If motion was detected
else {
//Only after sunset
if(!GetValueBoolean($istTag)) {
//If it is between 10:30 p.m. and 06:00 a.m
| if((time() > strtotime("22:30")) || (time() &lt; strtotime("06:00")))
{ |
MXC_DimSet($lampID, 15);
}
else {
MXC_DimSet($lampID, 50);
}
}
}
```

This script must be called by a triggering event - here usually by one/several motion detectors. The light is then set to a different dimming level depending on the time of day. As soon as the motion detector no longer detects any movement and sends the FALSE impulse, the device switches off.
The logic is thus outsourced and only scripts have to be created that set the missing variables and are called by an event. Below is an example:

```
//Triggering of the event by motion detector variable!

//Unique Device-ID
$lampID = 54321 /*[ground floor\corridor\ceiling lamp]*/;
//Day/night variable from the location control
```

```
$isDay = 56789 /*[IsDay]*/;

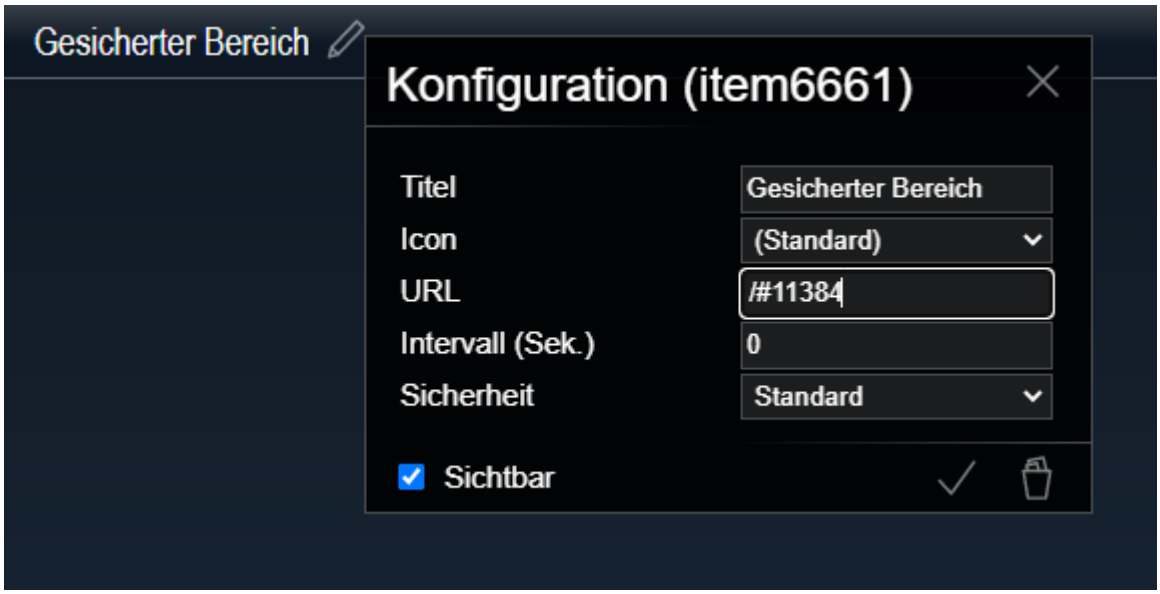
includeScript(12345 /*[scenarios\light control]*/);

//Function to insert script content
function includeScript($scriptID) {
$s = IPS_GetScript($scriptID);
include($s['ScriptFile']);
}
```

In this way, "function templates" can be created and then fed with the necessary IDs. In these scripts then are the IDs of the devices/variables for which IP-Symcon creates complete and meaningful names (provided the object tree structure was created properly).
If required, the script can also be adapted for other systems and the correct function can be called dynamically using the transferred InstanceID or the system-specific changes can be made.

Password protected category in WebFront

In order to protect certain categories from the object tree in the WebFront with a password, the following steps must be observed.
The category to be saved must be moved away from the rest of the WebFront in the object tree .
Then a new WebFront must be set up. Once this is done, any password can be set up in the Security of the WebFront .
In the second WebFront, everything must be removed via the Editor and the category that is to display the password-protected content needs to be added.
The second WebFront must then be added to the first WebFront as an external page via a path.

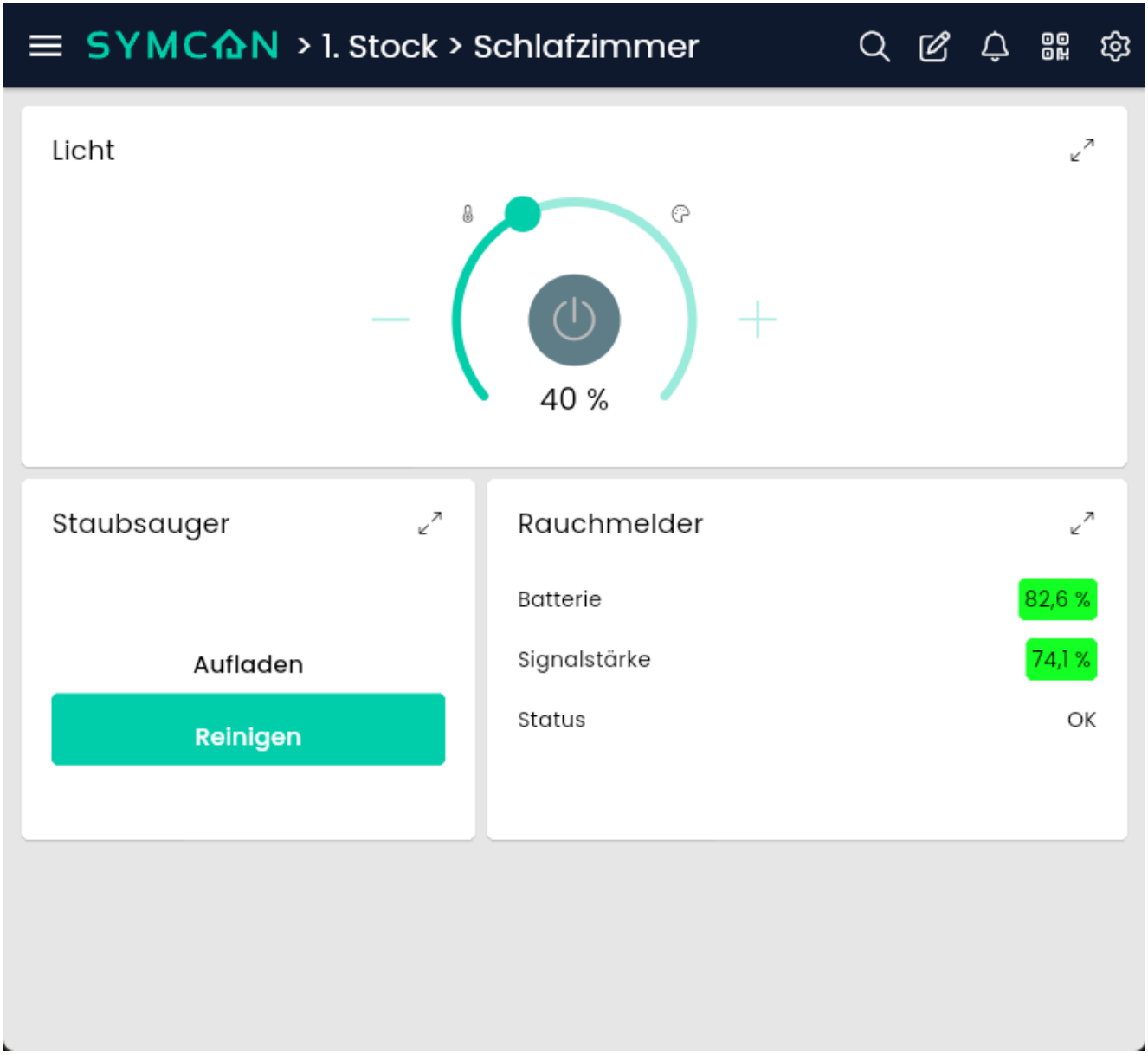


The call to the external page then looks as follows.

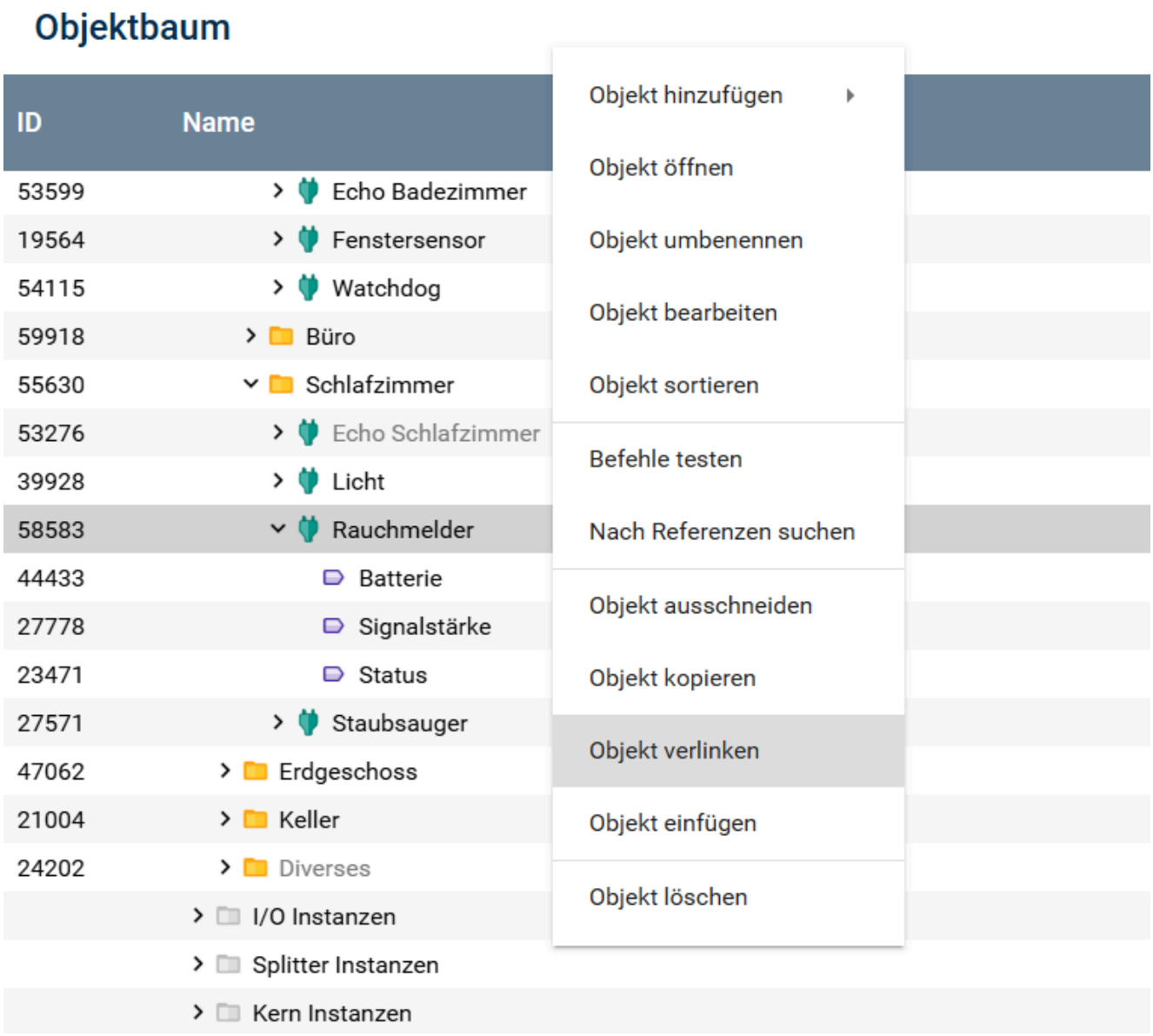

IP-Symcon
WebFront
Wetter (DWD)
Gesicherter Bereich
14:48:18
SYMCON
WebFront
Benötige Kennwort

Use links

The following example illustrates the principle of Links. For example, a number of smoke detectors are installed and connected in a house.
For safety reasons, the smoke detectors are now to be checked every three months. An initial check was quite time-consuming. When structuring the visualization, the structure was sensibly designed according to floor -> room -> device. However, each room now has to be checked individually in order to check each smoke detector. This seems too time-consuming.



A category called "Smoke detectors" is now created next to the floors and all smoke detectors are moved to this category using "drag and drop".
Because all smoke detectors have been assigned to a new category in Symcon, they are no longer visible in the individual rooms in the visualization. Unfortunately, this means that the smoke detectors are now missing in the individual rooms in which they are installed.
Symcon offers the option of creating links to individual objects. To create these links, right-click on a selected object in the object tree - in this case, a smoke detector.



As soon as "Link object" is clicked, the object has been moved to the clipboard for linking. A link to the originally selected object can now be created anywhere in the object tree by right-clicking and selecting "Insert object". This can be repeated with all smoke detectors and the smoke detectors can be easily checked in the visualization without constantly clicking back and forth. Nevertheless, they can still be found in their respective rooms. In addition, links can be given their own name and icon by right-clicking on "Edit object".

Send push notifications to different user groups

The procedure for sending push notifications to different user groups is explained below. In this example, a user group consists of several terminal devices (smartphones) registered to a WebFront.

SETUP

User groups for push notifications are implemented in IP-Symcon via separate WebFronts or WebFront visualizations.



In order to be able to set up more than one WebFront, an **IP-Symcon Professional** or **IP-Symcon Unlimited** version is required.

How an additional WebFront can be set up can be seen under "Set up additional WebFront ". These WebFronts can then be set up separately according to personal needs.



In order to be able to use the same instances in different WebFronts, Links are required. See "Use links " for an example.

REGISTRATION IN THE USER GROUPS

As described above, user groups are implemented via WebFronts. It is sufficient to register once in the respective WebFront with the app on the smartphone that is to be registered. This automatically adds the device to the respective user group. From this point on, the smartphone app automatically receives every push notification. In this way, the smartphone can be registered with any number of user groups.



Logging into the respective WebFront via the mobile phone browser does not result in an entry in the user group.

LOGOUT FROM A USER GROUP

Logout from individual user group:
In the WebFront visualization, the registered devices can be managed in the "Notifications" tab. This means that a single smartphone can be deregistered from a user group.
Logout from all user groups:
If a smartphone is to be deregistered from all user groups, this can be done via the "Notification Control" core instance.
All mobile devices that are registered in a WebFront can be seen in the configuration tab. Is the device deleted here, the smartphone is logged off from all user groups.

SENDING PUSH NOTIFICATIONS

A push notification can be created with WFC_PushNotification. This needs the ObjectID of the respective WebFront to which the push notification is to be sent. All smartphones registered on this WebFront then receive the notification via the app.

TIPS & TRICKS

If a password is set up for the respective WebFront via the "Security" tab, it is easy to prevent "accidentally" registering for a user group.

APPLICATION EXAMPLES

A household with multiple WebFronts.

A WebFront for administrators: Only administrators are in this user group and are notified of supposed system problems.

A WebFront for a single family member: Includes personal interests and sends messages that only interest that person.

A WebFront for all family members: The refrigerator reports "The milk is empty, someone has to go shopping", The weather module reports "It's about to rain" or a burglar alarm.

How can I...?



Many of these scripts use special IP-Symcon functions.
The Command Reference/Module Reference entail further information on how exactly these functions work.

... switch on a device and switch it off again after 60 seconds
... get a list of module names and GUID
... configure an instance from PHP
... find out the remaining number of seconds of a ScriptTimer
... directly include a script by ID
... Output UpdateTime in a separate string variable (1 script - n variables)
... create a timer & variable from PHP
... download a file from the Internet
... load a folder recursively into the MediaPlayer playlist
... export a variable profile

... SWITCH ON A DEVICE AND SWITCH IT OFF AGAIN AFTER 60 SECONDS

```
if($_IPS['SENDER'] == "TimerEvent")
{
//From function
...

//Turn off the timer
IPS_SetScriptTimer($_IPS['SELF'], 0);
} else {
//To function
...

//Turn on the timer
IPS_SetScriptTimer($_IPS['SELF'], 60);
}
```

... GET A LIST OF MODULE NAMES AND GUID

```
foreach(IPS_GetModuleList() as $mid)
{
$m = IPS_GetModule($mid);
echo $mid."=".$m['ModuleName']."\n";
}
```

... CONFIGURE AN INSTANCE FROM PHP

```
//Change property
WWWReader_SetPage($id,"http://www.google.com");

//Apply changes
IPS_ApplyChanges($id);

//Get new URL
WWWReader_UpdatePage($id);
```

... FIND OUT THE REMAINING NUMBER OF SECONDS OF A SCRIPTTIMER

```
echo GetTimeRemaining($_IPS['SELF']); //Find out by yourself
function GetTimeRemaining($id)
{
$eid=@IPS_GetEventIDByName("ScriptTimer", $id);
if($eid === false) {
return -1;
} else {
$e=IPS_GetEvent($eid);
if($e['NextRun'] == 0)
{
return -1;
} else {
return $e['NextRun'] - microtime(true);
}
}
}
```

... DIRECTLY INCLUDE A SCRIPT BY ID

```
//Include script with ID 14871
include(IPS_GetScriptFile(14871));
```

... OUTPUT UPDATETIME IN A SEPARATE STRING VARIABLE (1 SCRIPT - N VARIABLES)

```
//Evaluate event
if($_IPS['SENDER'] != "Variable")
return;
```

```
SetValue(CreateVariableIDByName($_IPS['VARIABLE'], 'Updated',
VARIABLETYPE_STRING), date("d.m.y H:i:s"));

function CreateVariableIDByName($id, $name, $type)
{
$vid = @IPS_GetVariableIDByName($name, $id);
if($vid===false) {
$vid = IPS_CreateVariable($type);
IPS_SetParent($vid, $id);
IPS_SetName($vid, $name);
IPS_SetInfo($vid, "This Variable was created by Script
#".$_IPS['SELF']);
}
return $vid;
}
```

... CREATE A TIMER & VARIABLE FROM PHP



When the script runs, it sets a timer that starts every six hours and then puts a variable with the time of day as a value between 0-3 into the variable.

```
//NOTE:
//~~~~~~~~
//This script sets itself up automatically when run
//
//- A variable is set depending on the time of day (0-3)
// 0 = 0-6
// 1 = 6-12
// 2 = 12-18
// 3 = 19-24

//------------------------------------------------------------------------
--------
//From this point nothing needs to be changed
//------------------------------------------------------------------------
--------
if($_IPS['SENDER'] == "Execute")
{
$eventid = @IPS_GetEventIDByName("Timer", $_IPS['SELF']);
if($eventid === false)
{
$eventid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetEventActive($eventid, true);
IPS_SetName($eventid, "Timer");
IPS_SetEventScript($eventid, $_IPS['SELF']);
IPS_SetEventCyclic($eventid, EVENTCYCLICDATETYPE_NONE, 0, 0, 0,
EVENTCYCLICTIMETYPE_HOUR, 6);
}
$variableid = @IPS_GetVariableIDByName("Daytime", $_IPS['SELF']);
if($variableid === false)
{
```

```
$variableid = IPS_CreateVariable(VARIABLETYPE_INTEGER);
IPS_SetName($variableid, "Daytime");
IPS_SetParent($variableid, $_IPS['SELF']);
}
}
SetValue(IPS_GetVariableIDByName("Daytime", $_IPS['SELF']),
floor(date("H") / 6));
```

... DOWNLOAD A FILE FROM THE INTERNET

```
$remoteImage = "https://www.google.com/images/srpr/logo3w.png";
$localImage = IPS_GetKernelDir()."\\media\\image.jpg";

//Download
$content = @file_get_contents($remoteImage);
if((strpos($http_response_header[0], "200") === false))
{
return;
}

//Save to computer
file_put_contents( $localImage, $content );
```

... LOAD A FOLDER RECURSIVELY INTO THE MEDIAPLAYER PLAYLIST

```
function WAC_PlayDir($id, $dir)
{
function ReadRecursive($dir, $subdir = "") {
$result = Array();
$files = scandir($dir."/".$subdir);
foreach($files as $file)
{
if(($file != ".") &amp;&amp; ($file != "..")) {
if(is_dir($dir."/".$subdir."/".$file)) {
$res = ReadRecursive($dir, $subdir."/".$file);
$result = array_merge($res, $result);
} else {
$filedir = $subdir."/".$file;
$filedir = substr($filedir, 1, strlen($filedir));
$result[] = $filedir;
}
}
}
return $result;
}

$allowed = Array("mp3", "wma");
$files = ReadRecursive($dir);

//Use PHP's random number generator
```

```
//shuffle($files);

WAC_ClearPlaylist($id);
foreach($files as $file)
{
$ext = pathinfo($dir."/".$file, PATHINFO_EXTENSION);
if(in_array(strtolower($ext), $allowed))
{
WAC_AddFile($id, $dir."/".$file);
}
}
WAC_Play($id);
}
```

... EXPORT A VARIABLE PROFILE

```
getVariableProfileCreationCode("~Temperature.FHT");
getVariableProfileCreationCode("~Temperature.FHT", "TemperatureTest");

// first function parameter: Profile name, second parameter (optional):
new profile name
function getVariableProfileCreationCode ($profileName, $newProfileName
= "")
{
$profile = IPS_GetVariableProfile($profileName);
if ($profile !== false)
{
$profileName = (strlen($newProfileName) > 0) ? $newProfileName :
$profileName;
echo 'IPS_CreateVariableProfile("'.$profileName.'",
'.$profile['ProfileType'].');'."\n";
echo 'IPS_SetVariableProfileText("'.$profileName.'",
"'.$profile['Prefix'].'", "'.$profile['Suffix'].'");'."\n";
echo 'IPS_SetVariableProfileValues("'.$profileName.'",
'.$profile['MinValue'].', '.$profile['MaxValue'].',
'.$profile['StepSize'].');'."\n";
echo 'IPS_SetVariableProfileDigits("'.$profileName.'",
'.$profile['Digits'].');'."\n";
echo 'IPS_SetVariableProfileIcon("'.$profileName.'",
"'.$profile['Icon'].'");'."\n";
foreach ($profile['Associations'] as $association)
{
echo 'IPS_SetVariableProfileAssociation("'.$profileName.'",
'.$association['Value'].', "'.$association['Name'].'",
"'.$association['Icon'].'", '.$association['Color'].');'."\n";
}
echo "\n";
}
}
```

Keyboard Shortcuts

GENERAL KEYBOARD SHORTCUTS

SHORTCUT	DESCRIPTION
Esc	Close the current tab

KEYBOARD SHORTCUTS IN SCRIPT EDITOR

SHORTCUT	DESCRIPTION
F3	Search next
Shift + F3	Search previous
Ctrl + A	Select all
Ctrl + C	Copy text via Ctrl + V
Ctrl + E	Execute
Ctrl + F	Search
Ctrl + H	Replace
Ctrl + O	Select object
Ctrl + R	Rename script
Ctrl + S	Save
Ctrl + D	Display events
Ctrl + X	Move text via Ctrl + V
Ctrl + Shift + K	Delete row
Ctrl + V	Insert text from clipboard
Ctrl + Z	Undo
Ctrl + Shift + Z, Ctrl + Y	Redo
Ctrl + Space	Display function list
Ctrl + Shift + F	Search in all scripts
Ctrl + Shift + H	Replace in all scripts

TASTENKOMBINATIONEN IM OBJEKTBAUM

SHORTCUT	DESCRIPTION
Enter	Open object or edit object if it cannot be opened
Del	Delete object
Alt + 0	Add category

SHORTCUT	DESCRIPTION
Alt + 1	Add instance
Alt + 2	Add variable
Alt + 3	Add script
Alt + 4	Add event
Alt + 5	Add media
Alt + 6	Add link
Ctrl + C	Copy object via Ctrl + V, copy ObjectID to clipboard
Ctrl + L	Create link to object via Ctrl + V
Ctrl + X	Move object via Ctrl + V
Ctrl + V	Copy, insert, or move object
Ctrl + E	Execute object
Ctrl + F	Search ID
Ctrl + R, F2	Rename object
Ctrl + Enter	Edit object

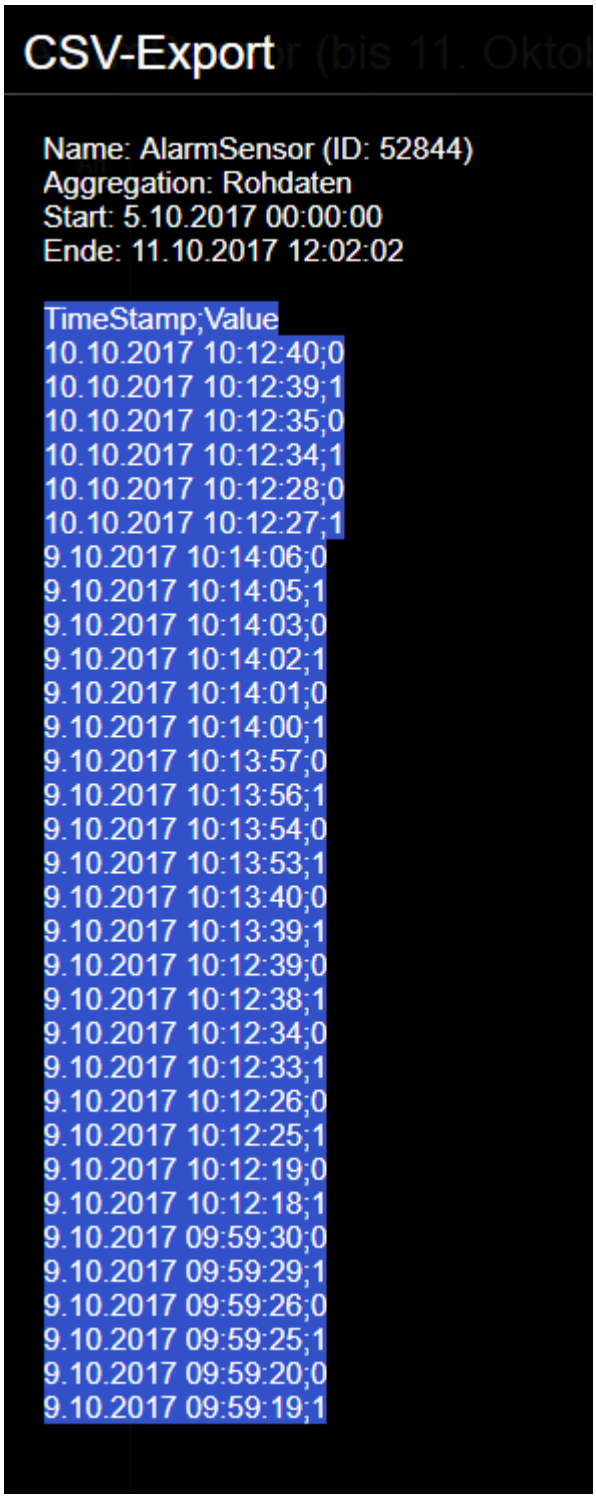
Save CSV data from WebFront to Excel

Graphs in the WebFront are displayed using CSV data sets. In order to make these available in a neat and legible form for further use in Excel, there is the "Text-To-Columns" functionality. This works for any CSV data set. The CSV data broken up in this way can, for example, be used for Excel graphics or the like of.

EXAMPLE

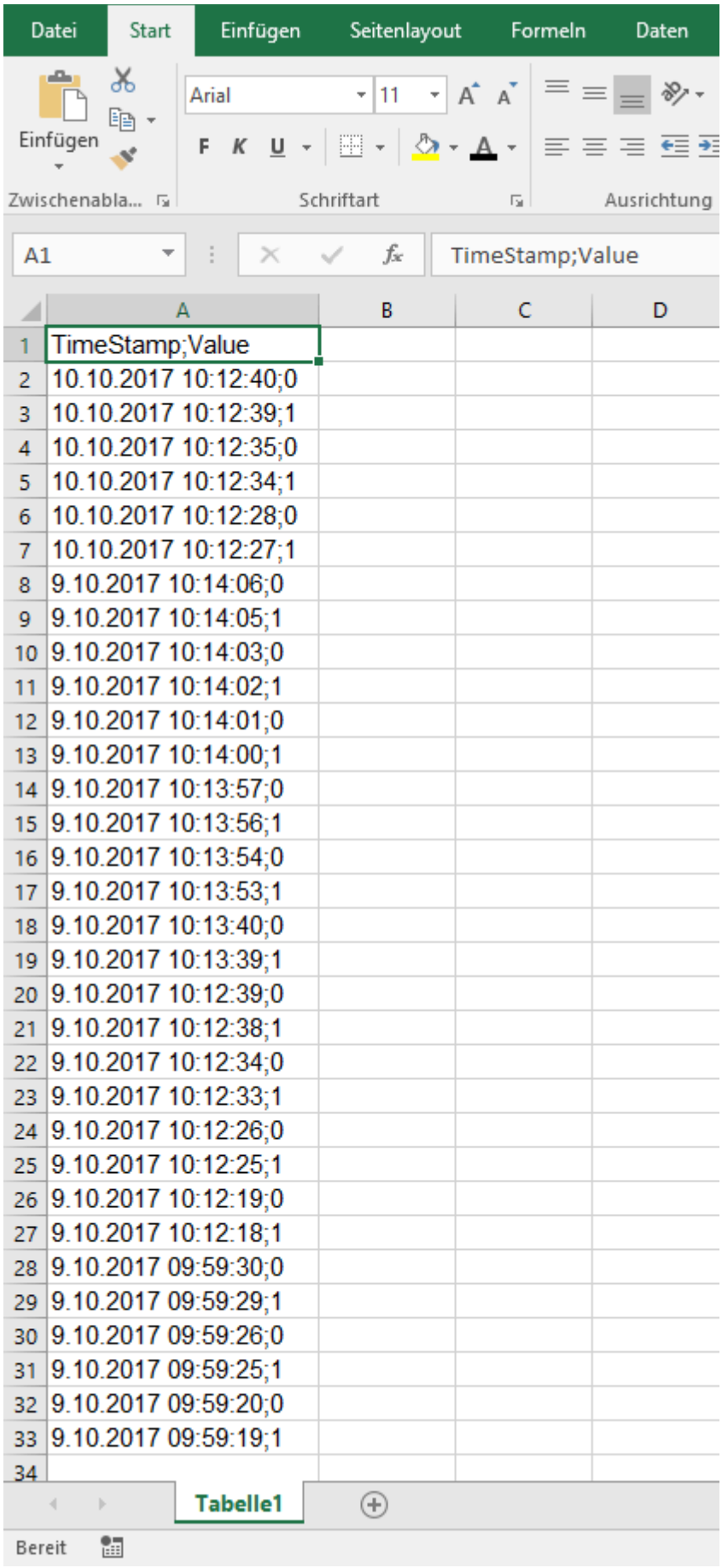
STEP 1 - CSV EXPORT COPY RECORD

Within the WebFront the graph symbol and then the menu item "CSV" have to be clicked on. In the popup (see picture) the data records simply have to be marked and copied with "Ctrl + C".



STEP 2 - PASTE

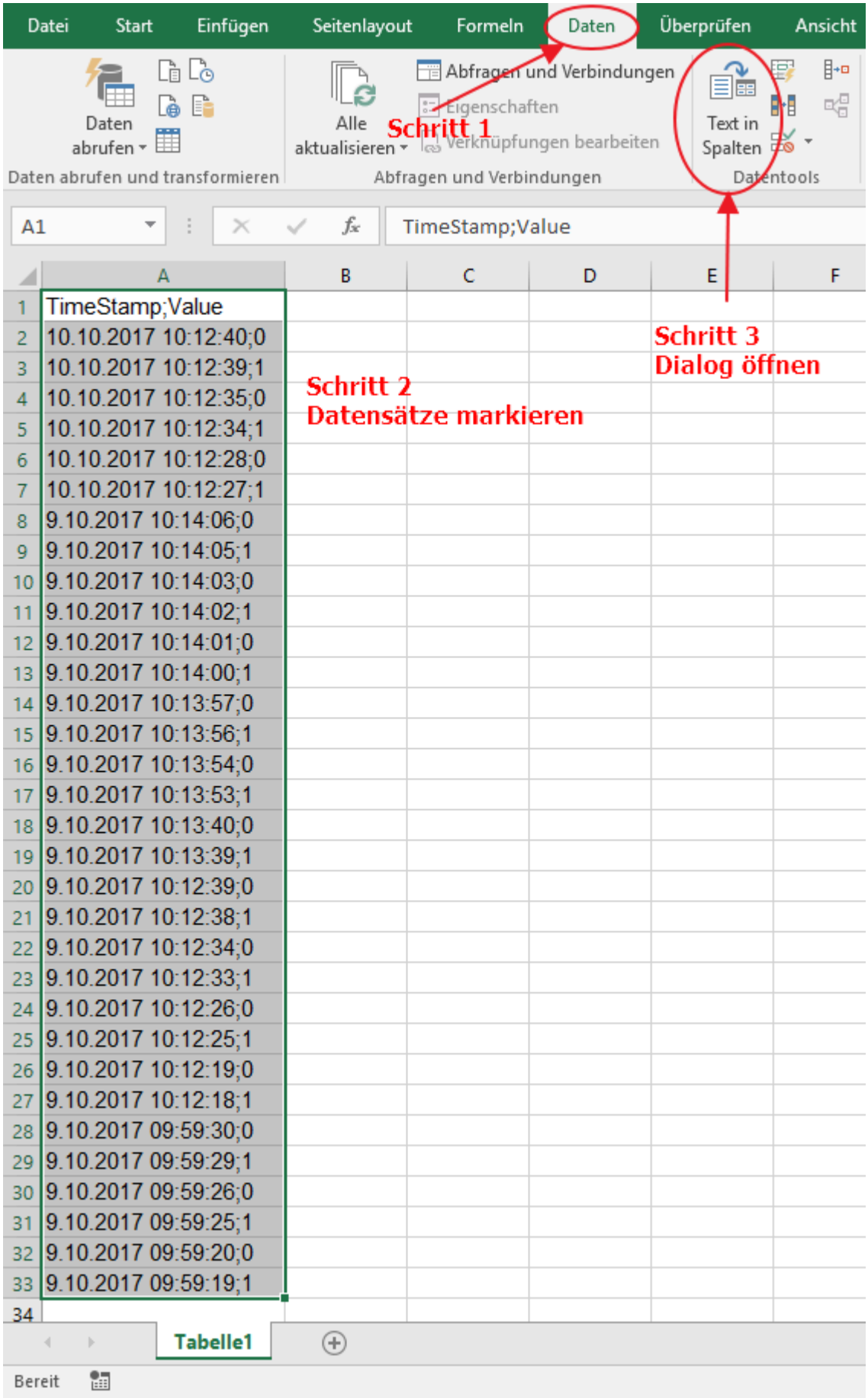
The data needs to be pasted into an empty table with "Ctrl + V". If necessary, the font color must be changed to black, as the white font color may have been adopted.



STEP 3 - TEXT TO COLUMNS

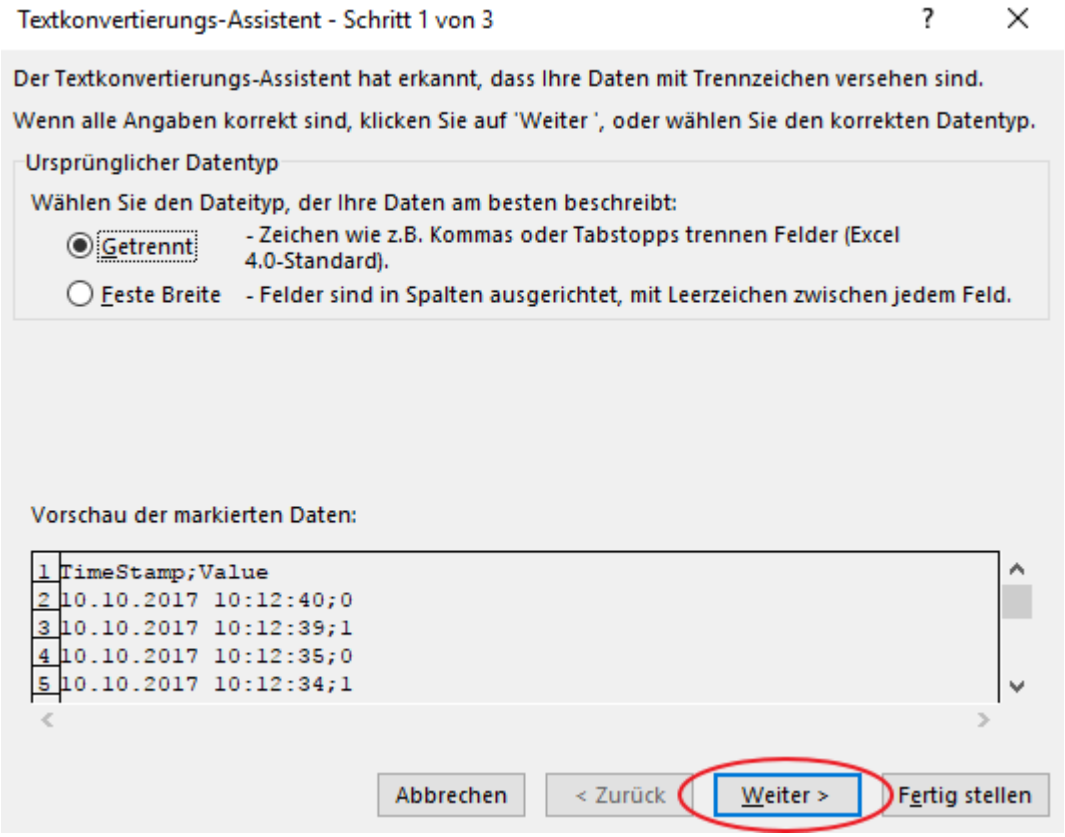
The "Text to Columns" function is used to split the data records.
To do this, the "Data" tab must be selected and all data records highlighted.

Then the "Text to columns" button must be clicked on and a dialog opens.

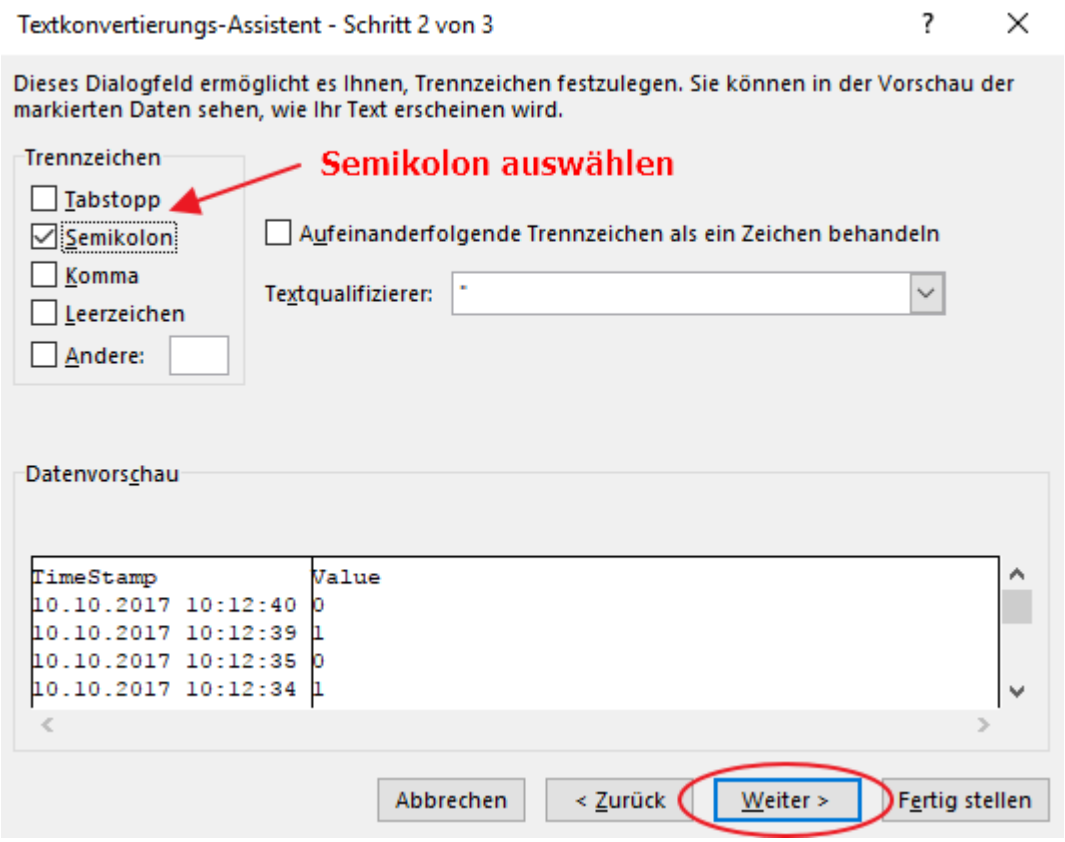


STEP 4 - DIALOG

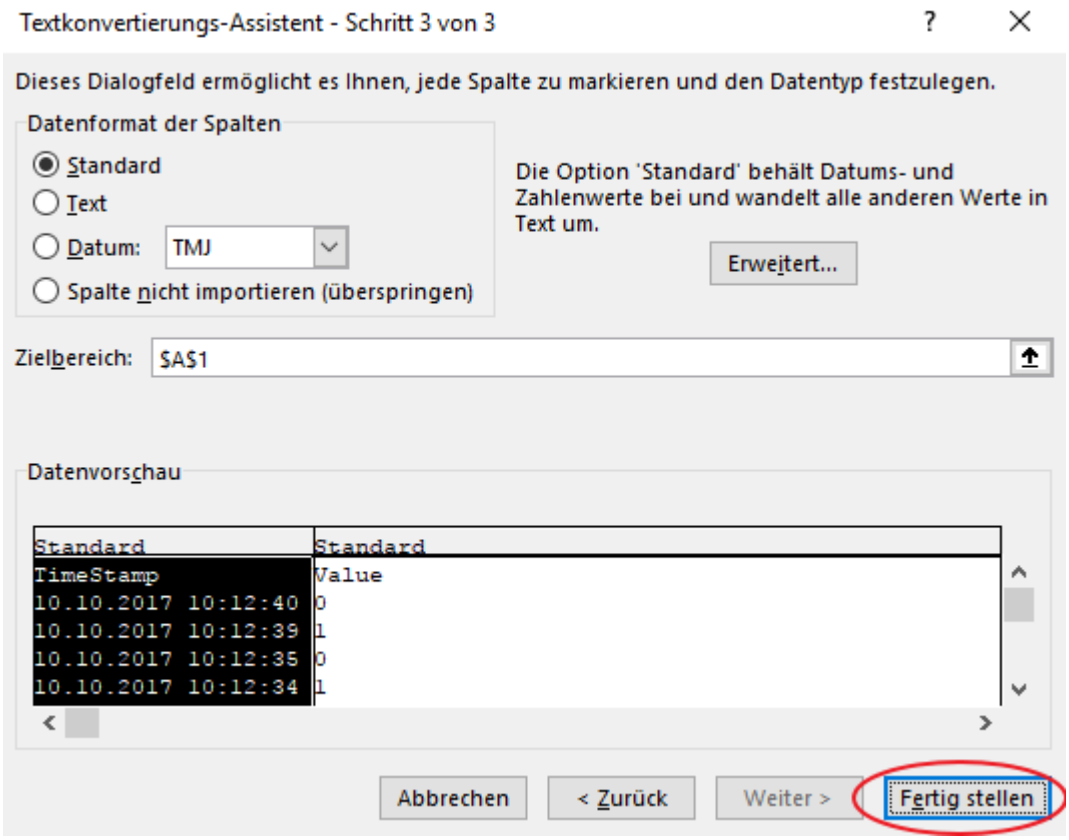
The dialog proceeds in 3 steps.



Click "Next".



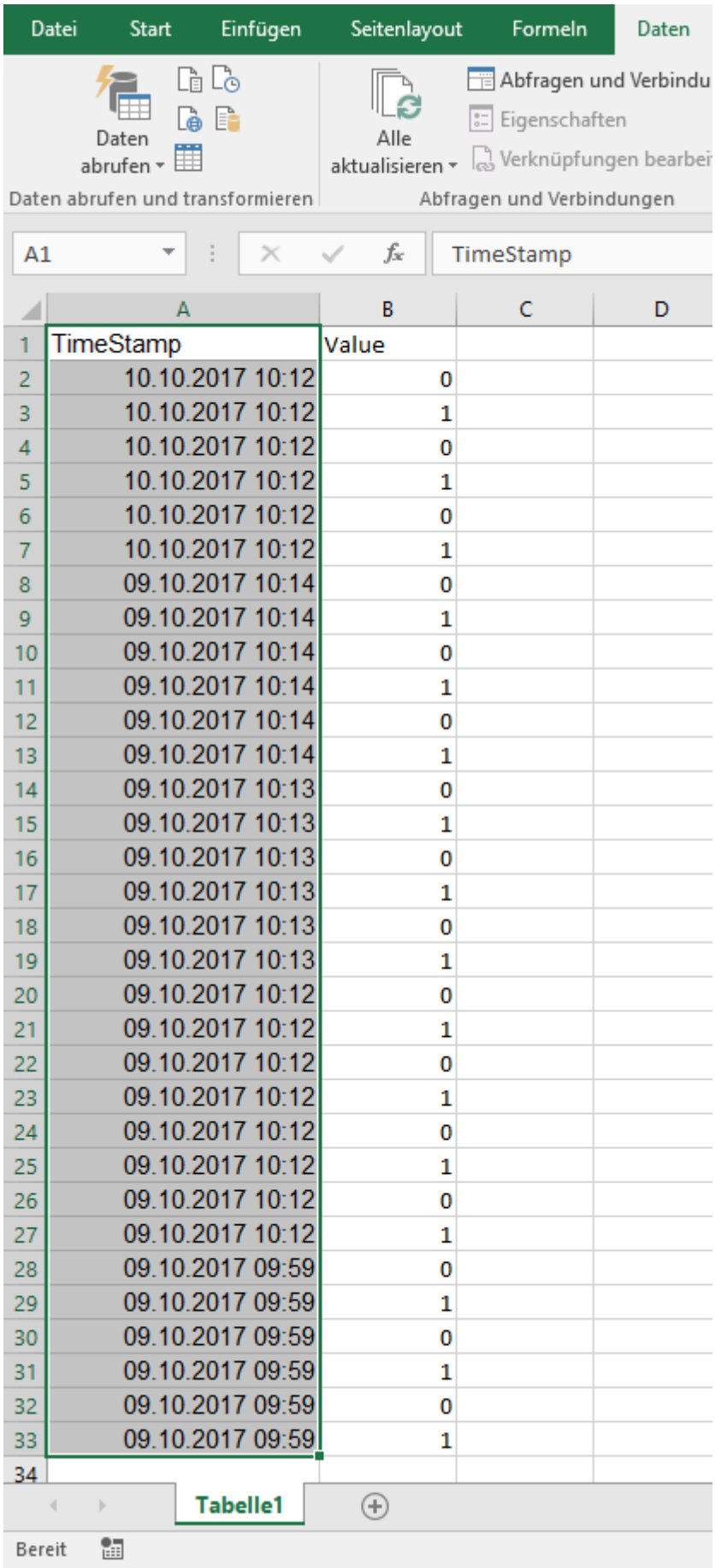
Select a semicolon as it separates the records. Click "Next".



Click "Finish".

RESULT

The table should then look like this.



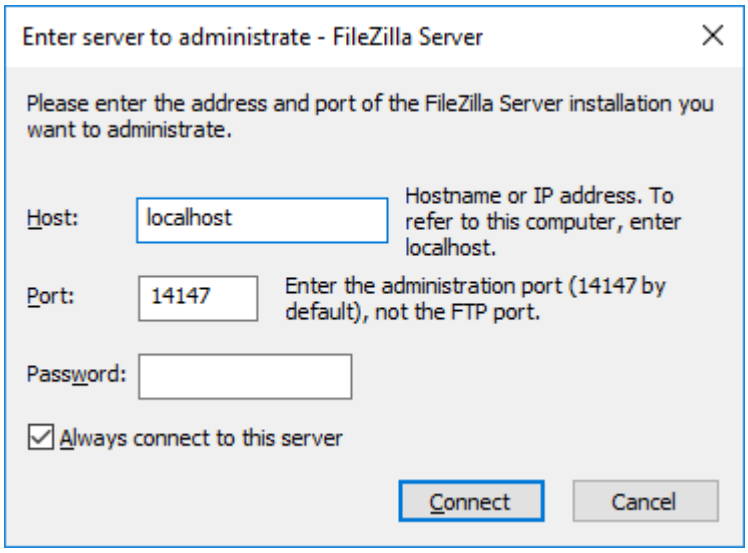
	A	B	C	D
1	TimeStamp	Value		
2	10.10.2017 10:12	0		
3	10.10.2017 10:12	1		
4	10.10.2017 10:12	0		
5	10.10.2017 10:12	1		
6	10.10.2017 10:12	0		
7	10.10.2017 10:12	1		
8	09.10.2017 10:14	0		
9	09.10.2017 10:14	1		
10	09.10.2017 10:14	0		
11	09.10.2017 10:14	1		
12	09.10.2017 10:14	0		
13	09.10.2017 10:14	1		
14	09.10.2017 10:13	0		
15	09.10.2017 10:13	1		
16	09.10.2017 10:13	0		
17	09.10.2017 10:13	1		
18	09.10.2017 10:13	0		
19	09.10.2017 10:13	1		
20	09.10.2017 10:12	0		
21	09.10.2017 10:12	1		
22	09.10.2017 10:12	0		
23	09.10.2017 10:12	1		
24	09.10.2017 10:12	0		
25	09.10.2017 10:12	1		
26	09.10.2017 10:12	0		
27	09.10.2017 10:12	1		
28	09.10.2017 09:59	0		
29	09.10.2017 09:59	1		
30	09.10.2017 09:59	0		
31	09.10.2017 09:59	1		
32	09.10.2017 09:59	0		
33	09.10.2017 09:59	1		
34				

Using FTP

It is possible to access files that are stored on an FTP server via script.

PREPARE FTP SERVER

If this was not done already, it is required to set up and configure an FTP server. The explanation will be given for the tool FileZilla Server . However, the process is similar for other tools.
During the installation, FileZilla Server already initializes an FTP server on the local host, if needed.



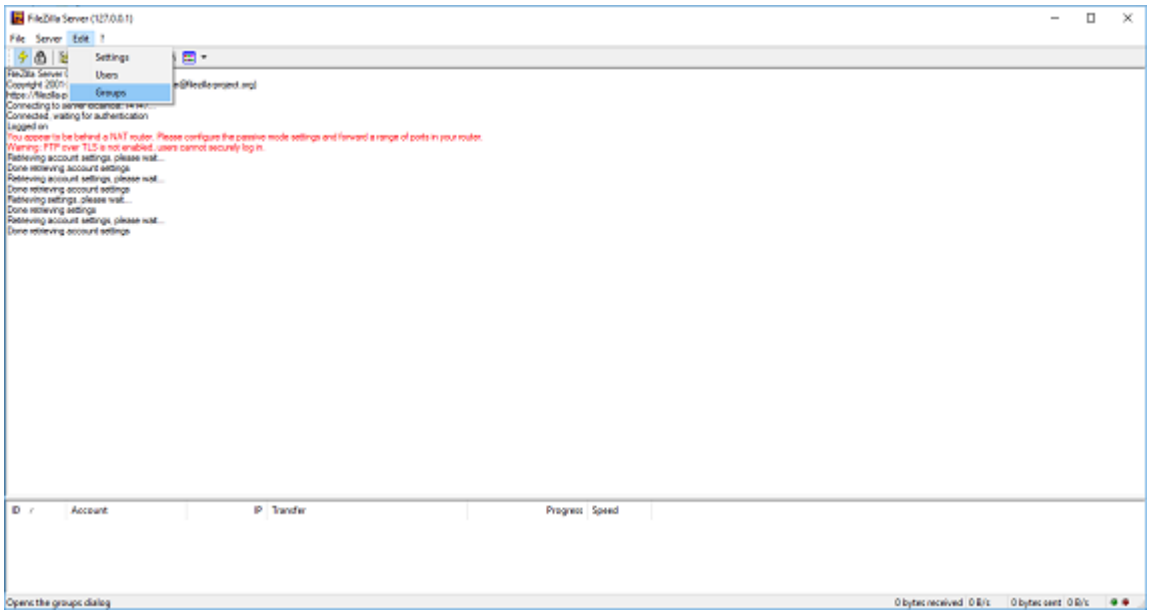
When launching the tool, the shown dialog is opened. It is used to connect to the created FTP server. If the default settings from the installation were used, the data can be used as shown in the screenshot, i.e., "Host": "localhost", "Port": 14147, and no password.

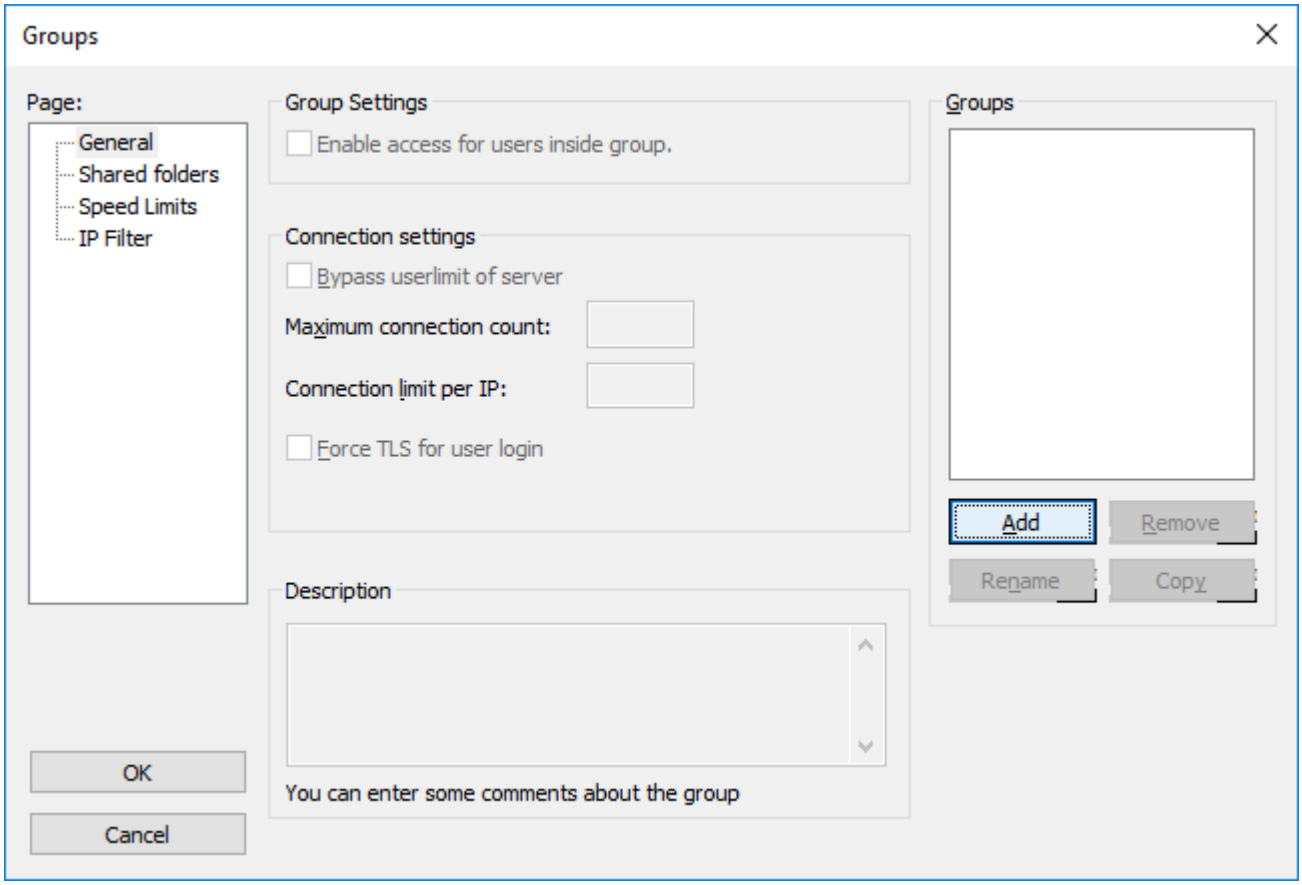


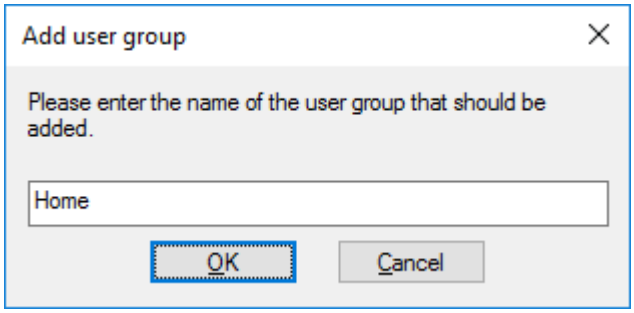
In most cases, the firewall needs to be configured to allow access to the FTP server. The process is explained for Filezilla here

CREATING A GROUP

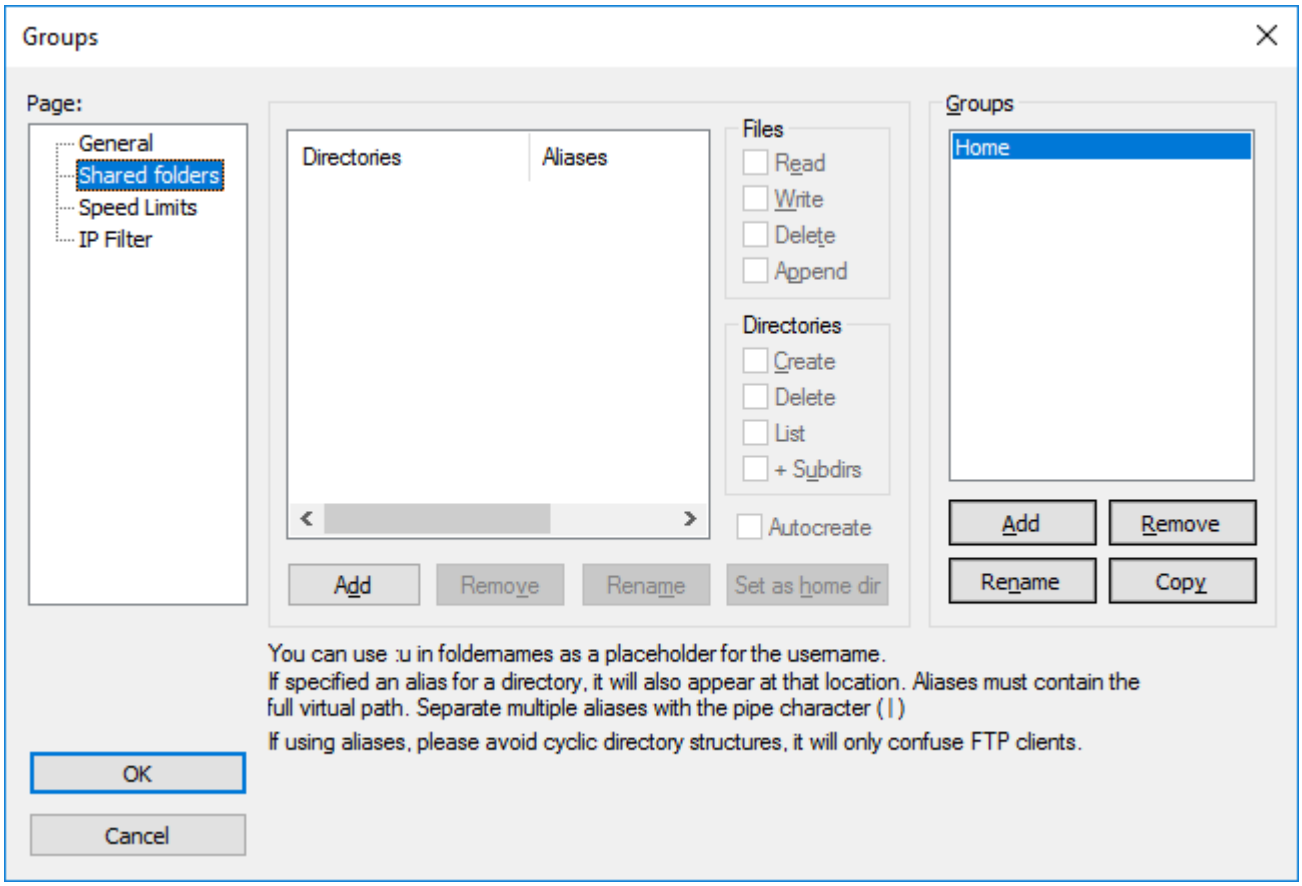
The next step is setting up a group for users. Groups can be used to categorize users and provide different rights or accessible folders. However, it is also possible to simply create a single group that is used for all users, thus providing the same rights to every user.

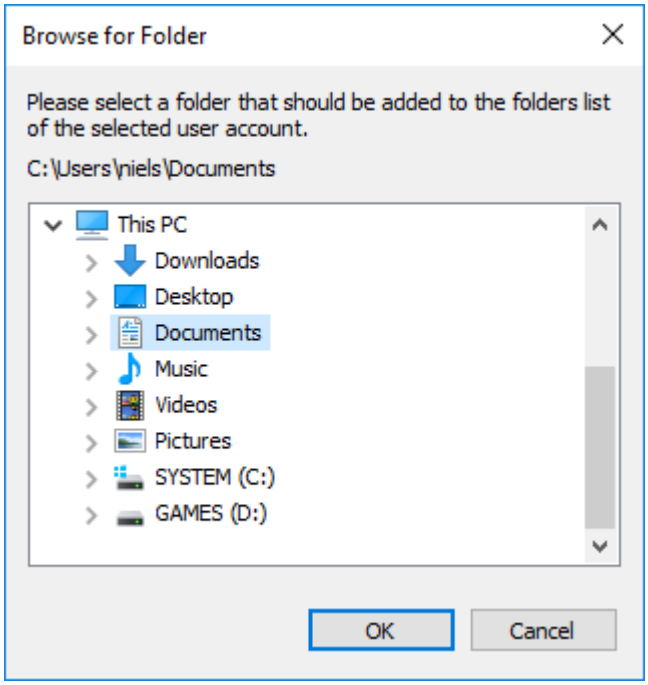




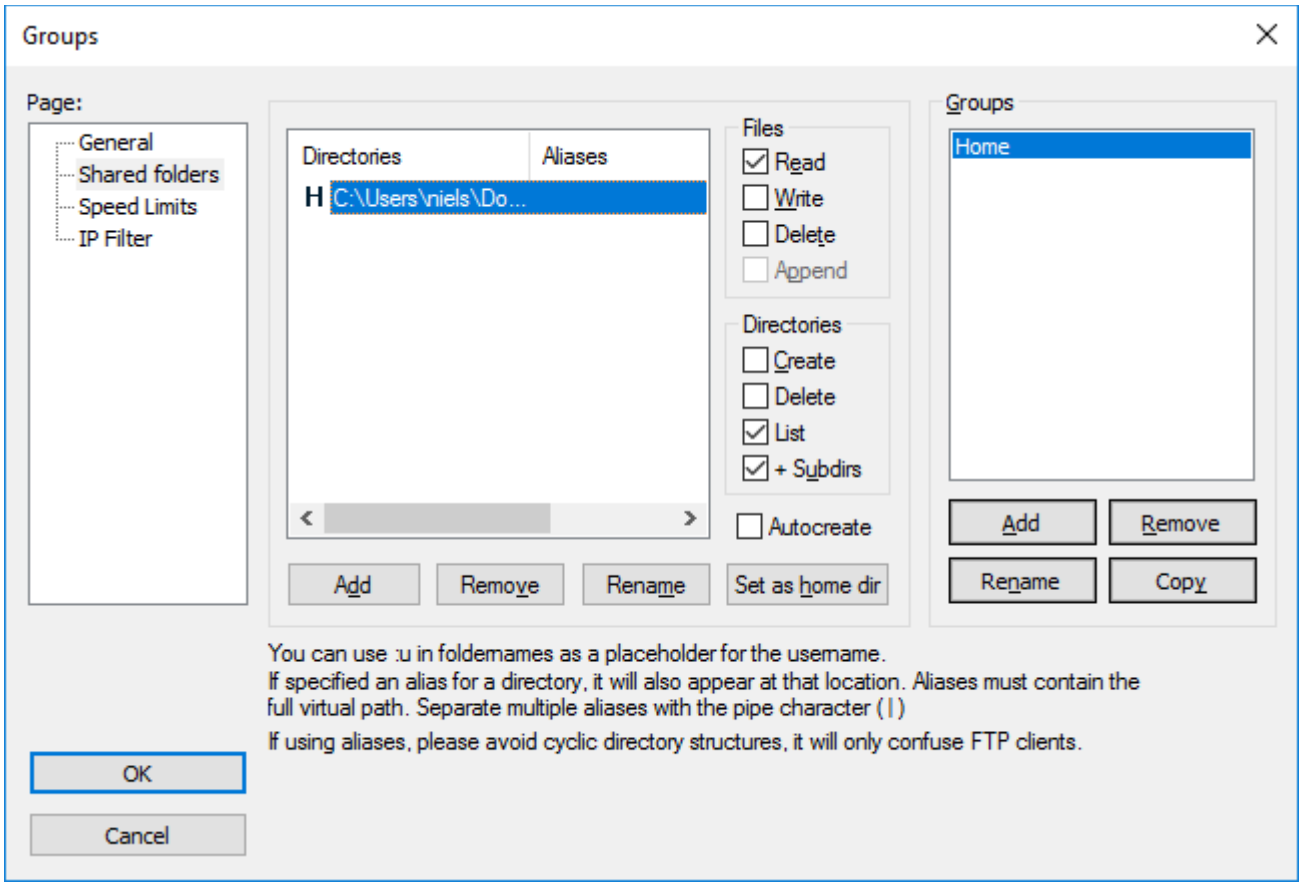


The Groups settings can be accessed via the menu "Edit->Groups". In the shown dialog, a new group can be added via "Add" below the initially empty list of groups. After entering a name and confirming the choice, a new group is created.



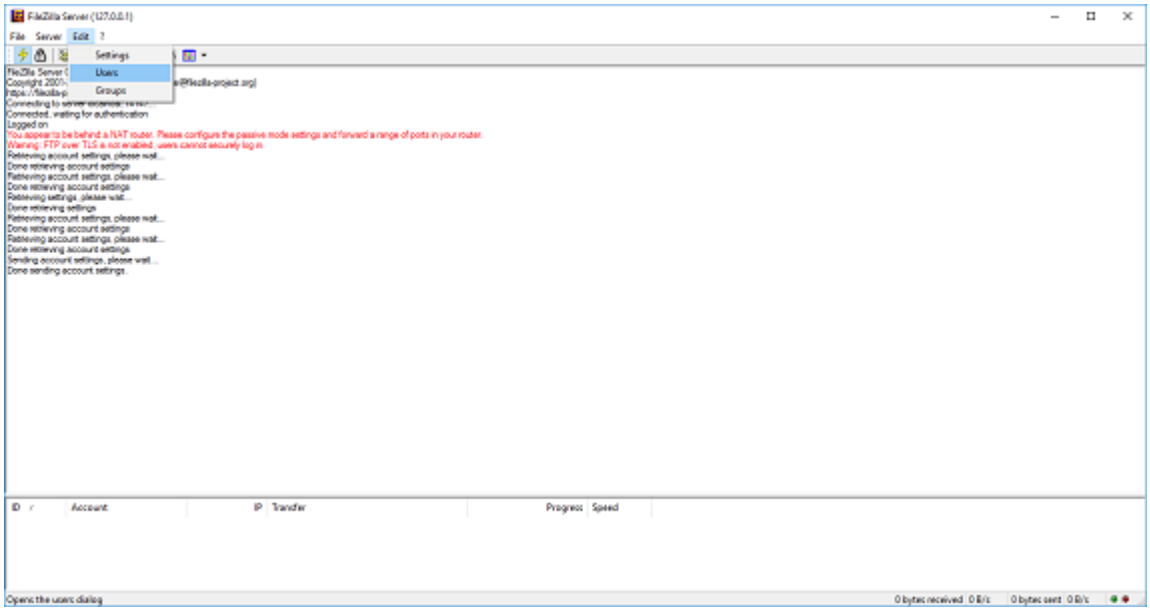


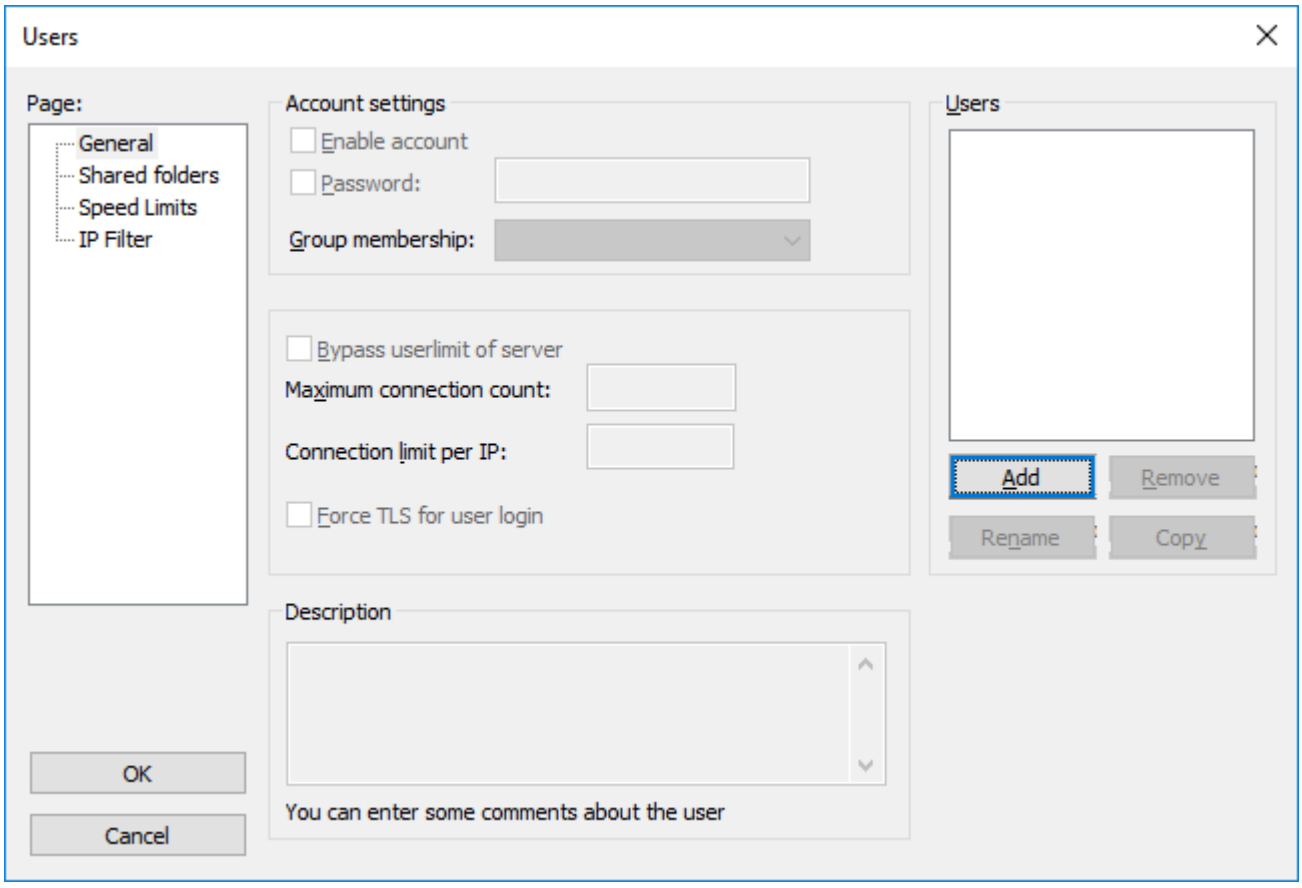
In the category "Shared folders", the folder that is meant to be accessed via FTP is chosen by clicking "Add" below the initially empty list of directories. This will open a dialog to select a folder.

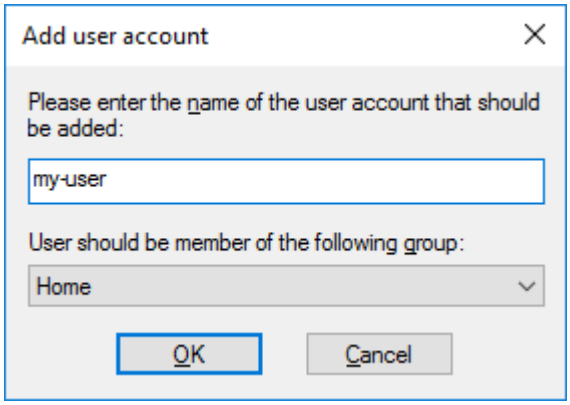


After adding a shared folder, it is possible to configure the rights of accessing the folder. By default, the files in the folder and its subdirectories can be read but not modified.

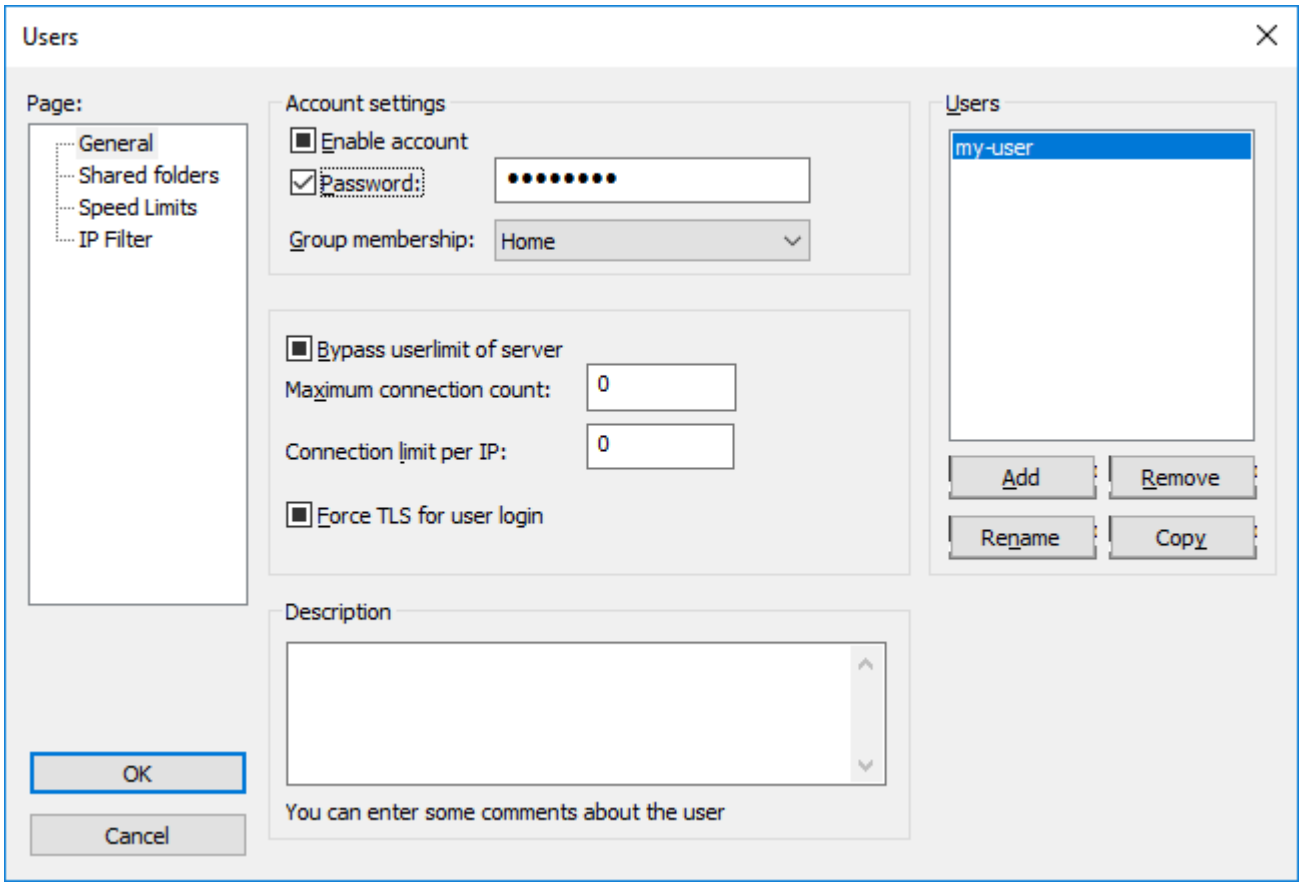
CREATING A USER







New users are added in the Users settings that are accessed via "Edit->Users". A new user is added by clicking "Add" below the initially empty list of users. In the shown dialog, a user name and a group is chosen. After confirming the choice, the user is added.



A password can be set for the newly created user by activating the checkbox "Password" and entering a chosen password.



If only one user exists, the creation of a group can be skipped. Instead, the user is assigned a shared folder directly.

ACCESS A FILE FROM THE FTP SERVER VIA SCRIPT

```
// Read content
$address = "ftp://my-user:password@localhost/test.txt";
$content = file_get_contents($address);
```

```
echo $content;
// Write content
$targetAddress = "ftp://my-user:password@localhost/test.txt";
$newContent = "Hello World! This is new content.";
file_put_contents($zielAdresse, $neuerInhalt);
```

A file on an FTP server can be accessed like it was a local file. Merely the address of the file changes. The code example reads and outputs the content of the file "test.txt" on the local FTP server. The address is structered as follows: "ftp://<user>(:<password>)@<servername>/<path-to-file>".



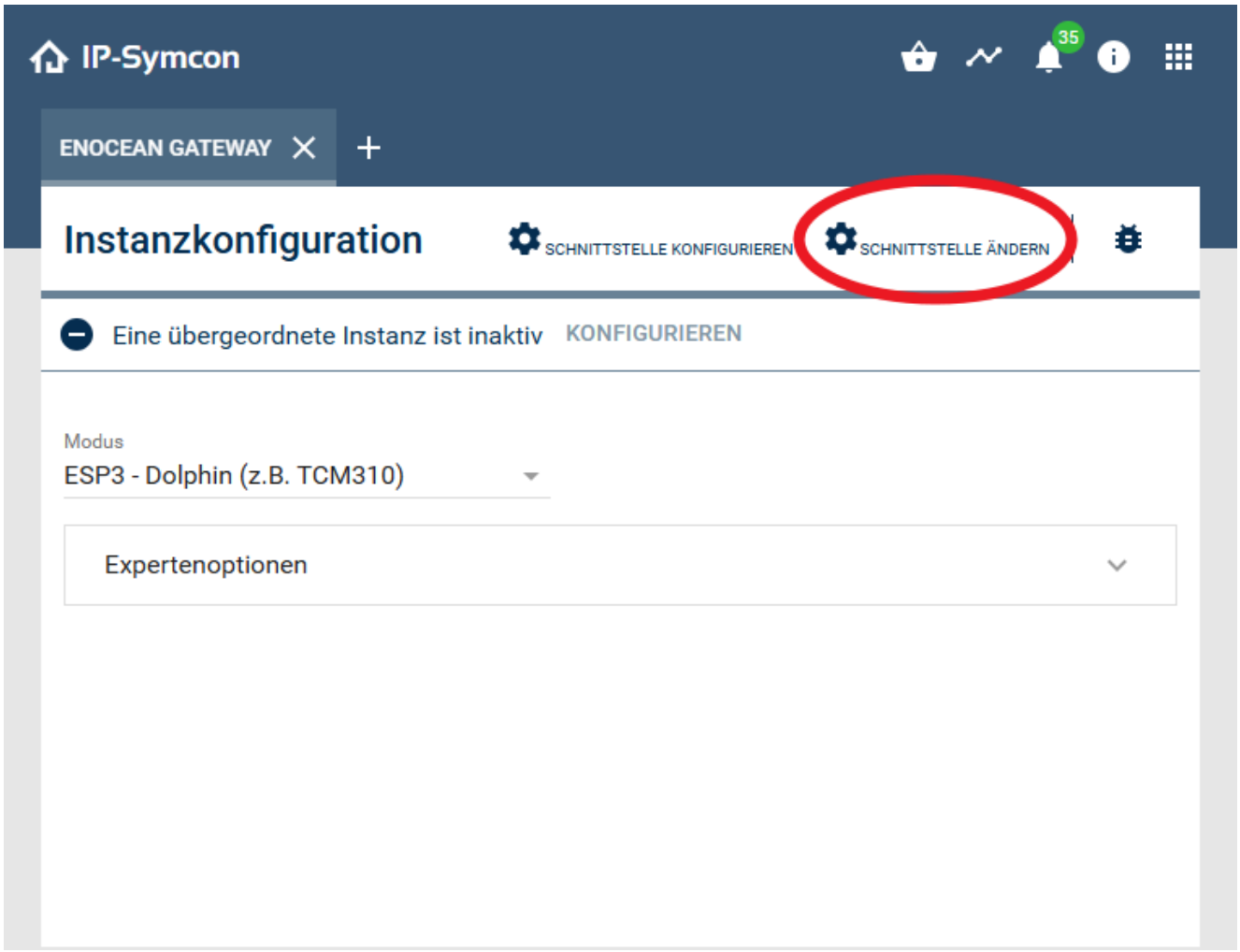
It is also possible to use special FTP functions when accessing an FTP server as shown here

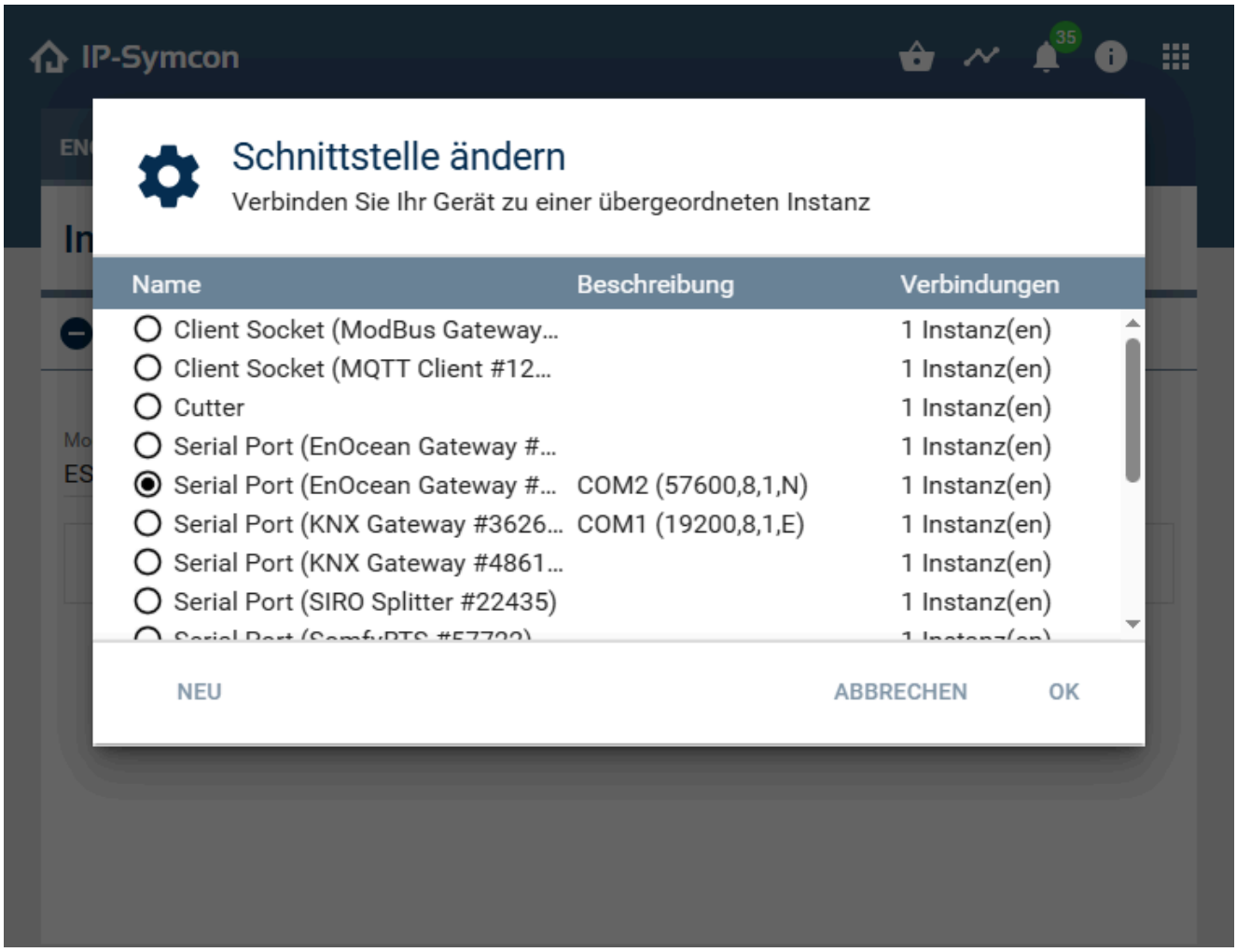
Replace I/O of a gateway

Sometimes it is necessary to replace the I/O instance of a gateway.
This would be the case if, for example, TCP-based communication is to be used instead of serial. Or another/new gateway is used.

> A list of the different I/O's can be found under Instances -> Connection components .

The easiest way to access the gateway for the respective device instance is via 'Configure gateway' in the upper area of the instance configuration.
In the instance configuration of the gateway, the parent instance, in this case the I/O instance, can now be adjusted via 'Change interface' in the upper area.
In the dialogue box that appears, you can select an existing compatible I/O or alternatively create a new I/O by clicking on 'New'.



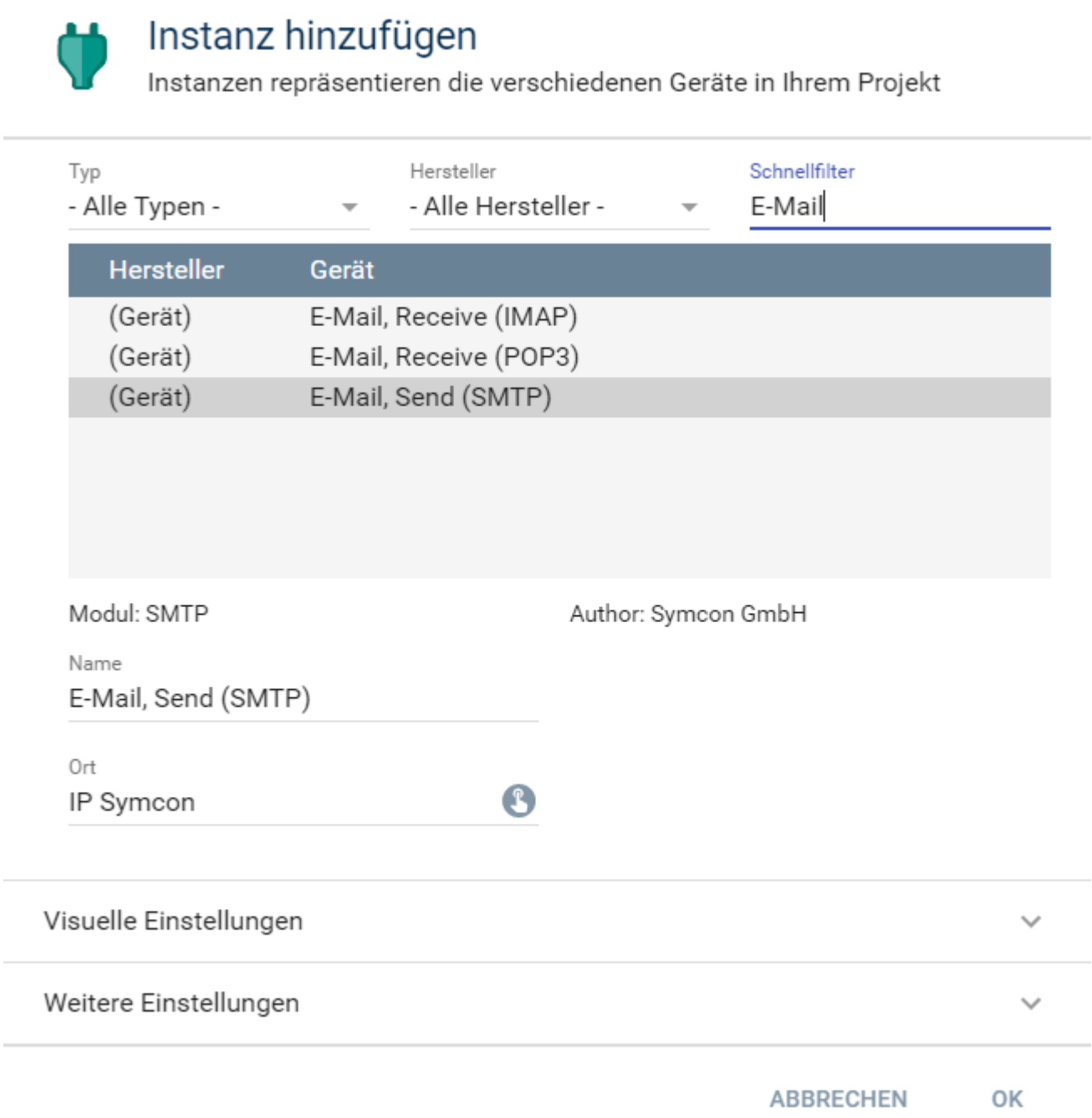

IP-Symcon
35
Schnittstelle ändern
Verbinden Sie Ihr Gerät zu einer übergeordneten Instanz
Name
Beschreibung
Verbindungen
Client Socket (ModBus Gateway...
1 Instanz(en)
Client Socket (MQTT Client #12...
1 Instanz(en)
Cutter
1 Instanz(en)
Serial Port (EnOcean Gateway #...
1 Instanz(en)
Serial Port (EnOcean Gateway #...
COM2 (57600,8,1,N)
1 Instanz(en)
Serial Port (KNX Gateway #3626...
COM1 (19200,8,1,E)
1 Instanz(en)
Serial Port (KNX Gateway #4861...
1 Instanz(en)
Serial Port (SIRO Splitter #22435)
1 Instanz(en)
NEU
ABBRECHEN
OK

E-Mail notification

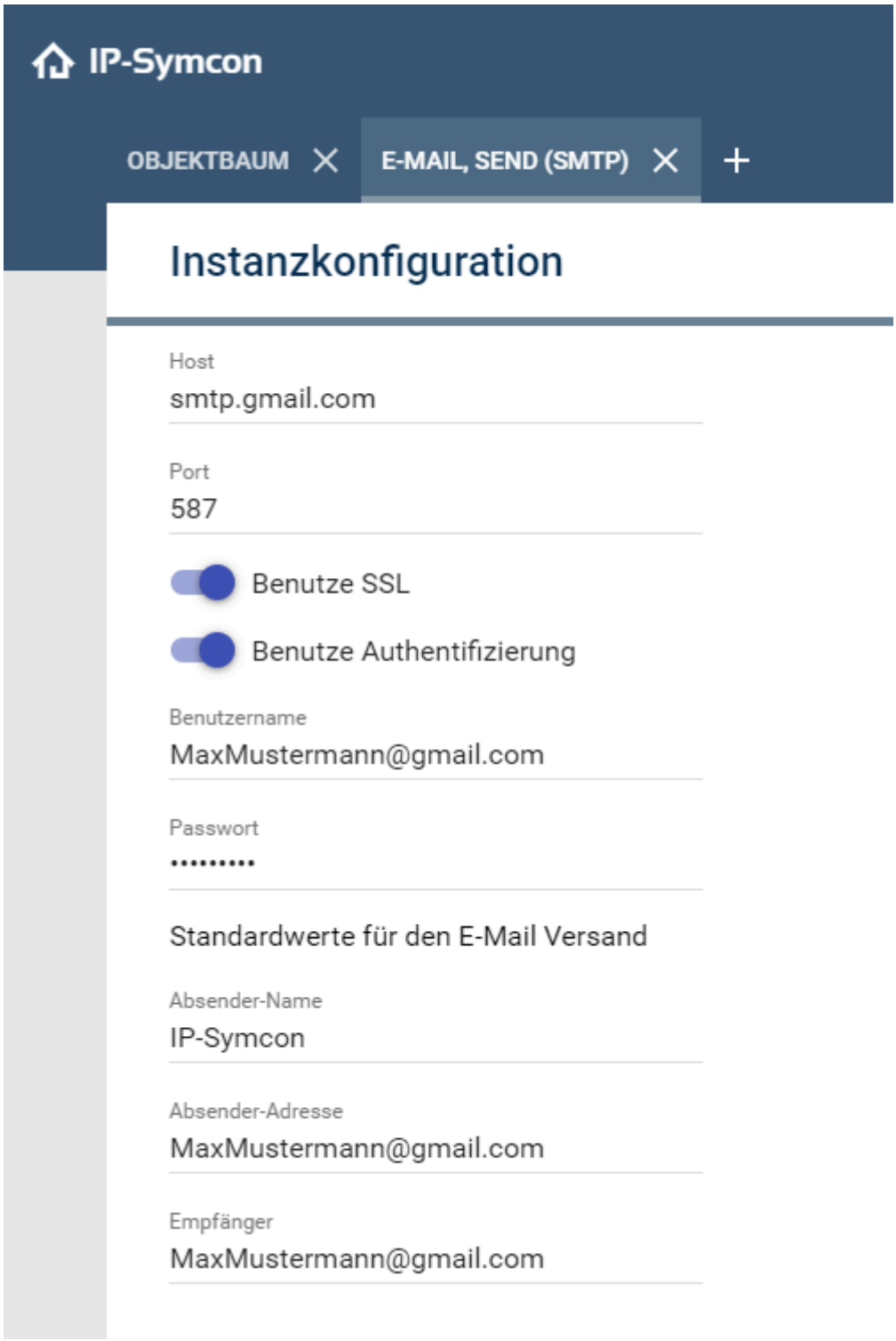
IP-Symcon offers the possibility to send e-mails via SMTP-Instance. For example, this offers the possibility to inform whether the front door or a window (requiring sensors) was opened in the SmartHome, whether a certain person is present (requiring the Presence Control Module) or the general status of various data and consumption. On the one hand, this can provide increased security (e.g. if a burglar tampers with a window or door) or, on the other hand, it can also increase comfort (e.g. general SmartHome information). Notifications at a specific point in time (e.g. as a reminder, monthly report) can also be implemented.

CREATE INSTANCE

In order to send an email at a specific time, an Instance must first be created. Instances represent, for example, devices that are connected to IP-Symcon. These can be both configured and receive functions.
A new SMTP instance can be created by clicking on "+" in the Object tree of the Management Console or by selecting the item "Add object" -> "Instance" from the context menu. The second option has the advantage that the instance is created directly below the selected object in the object tree and does not have to be sorted afterwards.
In the "Add Instance" menu, "Email" can be searched for via the quick filter and "Email, Send (SMTP)" can be added.



The personal access data must be entered on the "E-Mail, Send (SMTP)" configuration page.



"Host", "Port" and "Use SSL" (encryption) must be configured. With the respective provider the correct settings (keyword: SMTP) can be searched for.
The username and password of the e-mail account must be entered under "Use authentication". The "Sender Name" can be chosen freely. This will later be displayed as the sender in the recipient's mailbox.
E-mail addresses must be entered for "Sender address" and for "Recipient".

The "Sender address" is the address that is later displayed as the sender address when an e-mail arrives, while "Recipient" is the address to which IP-Symcon sends a message.

After everything has been filled out correctly, the configuration must be saved with "Apply". A test message can now be sent to the "recipient" email address within the test environment in the lower area of the configuration page.

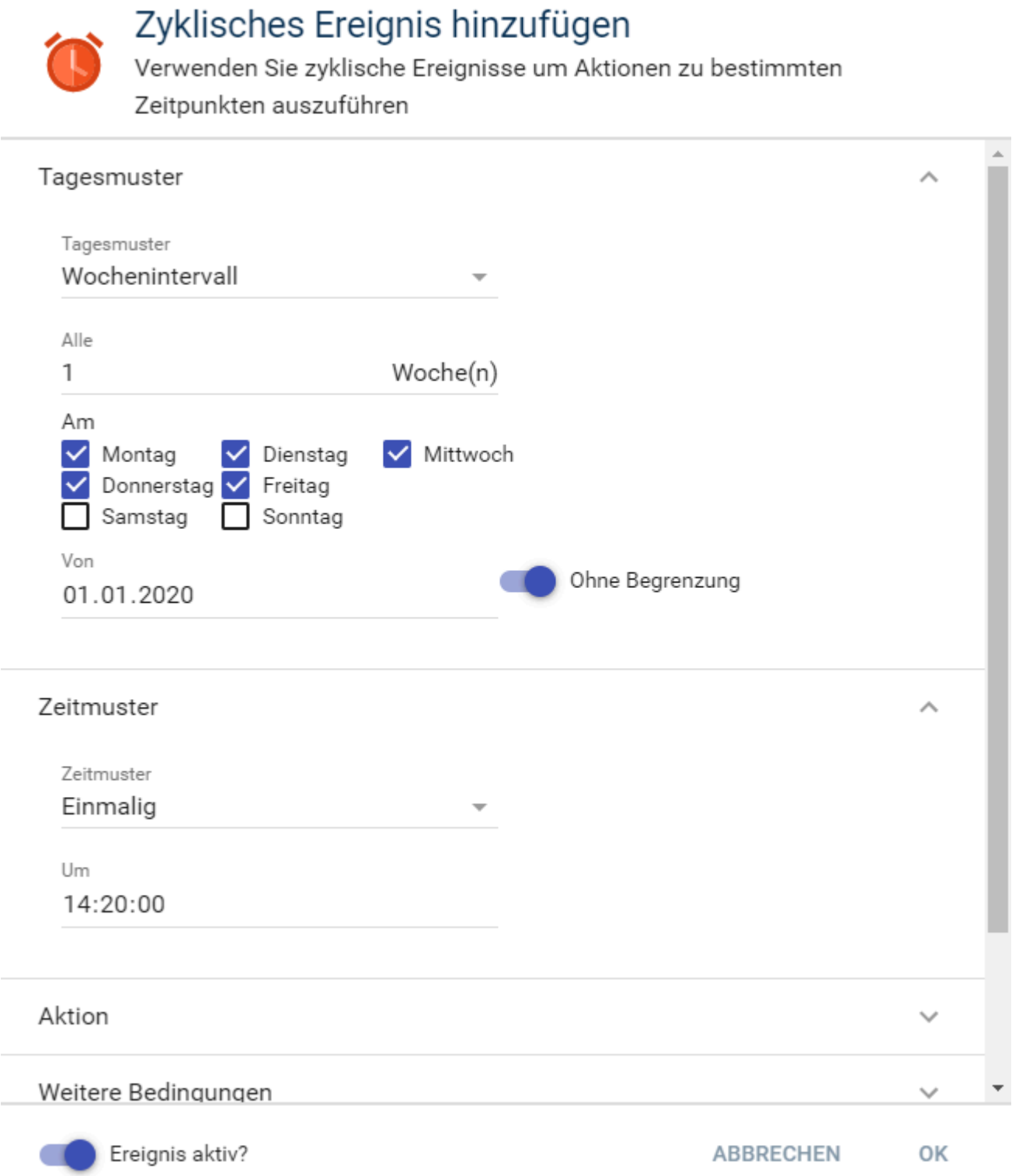
EXAMPLES

1. EMAIL AT A SPECIFIC TIME

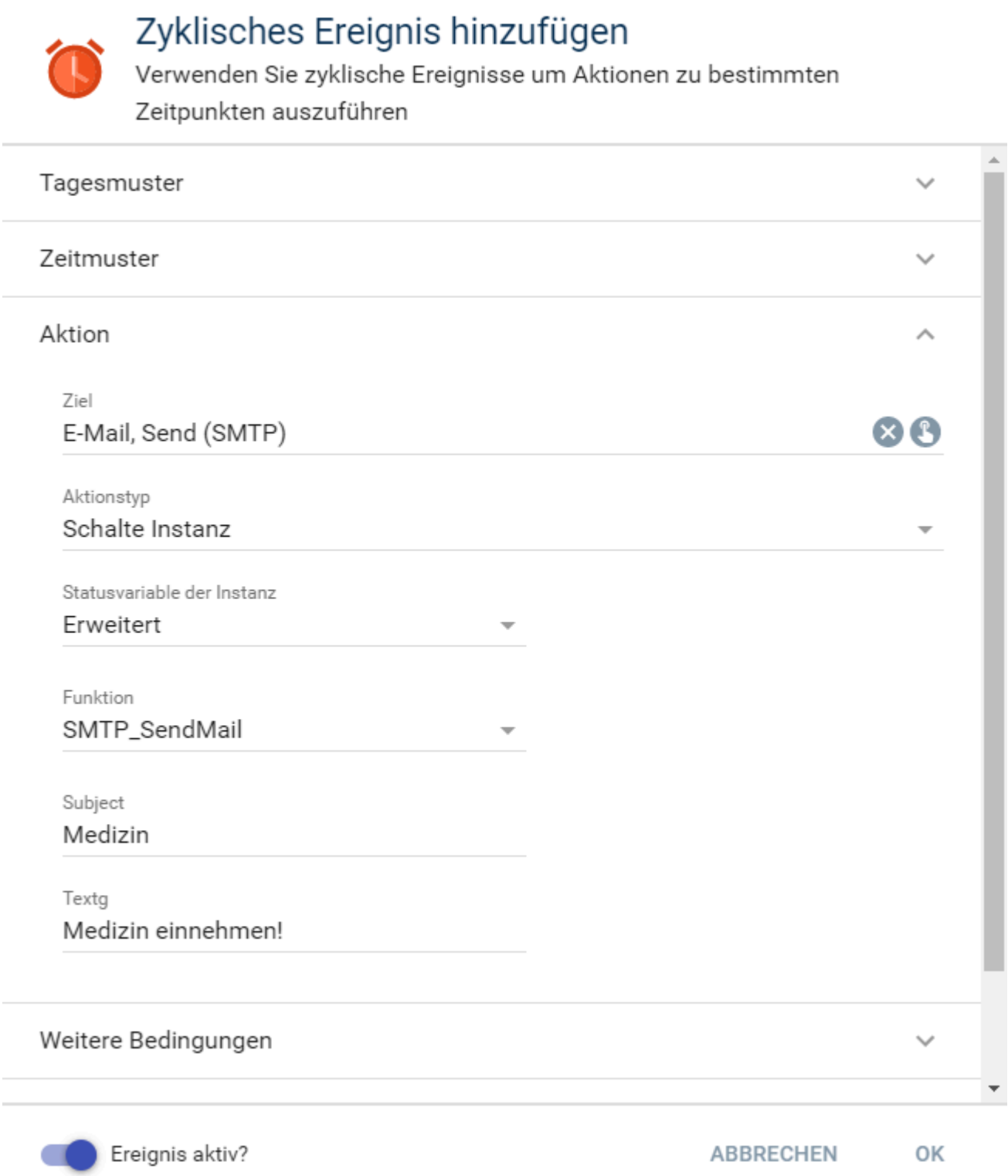
In this example, a reminder email is supposed to be sent at a specific time.

CREATE EVENT

A Cyclic Event must be created. Events can start specific operations on specific conditions or times.
A new event can be created by clicking on "+" in the Object tree of the Management Console or by selecting the item "Add object" -> "Event" -> "Cyclic" via the context menu. The second option has the advantage that the instance is created directly below the selected object in the object tree and does not have to be sorted in afterwards.



Now it has to be entered exactly when the reminder is to be sent by e-mail. This is useful if, for example, one wants to be reminded by e-mail during work in the office (Monday to Friday) that a certain medication needs to be taken after the lunch break (at 2:20 p.m.).



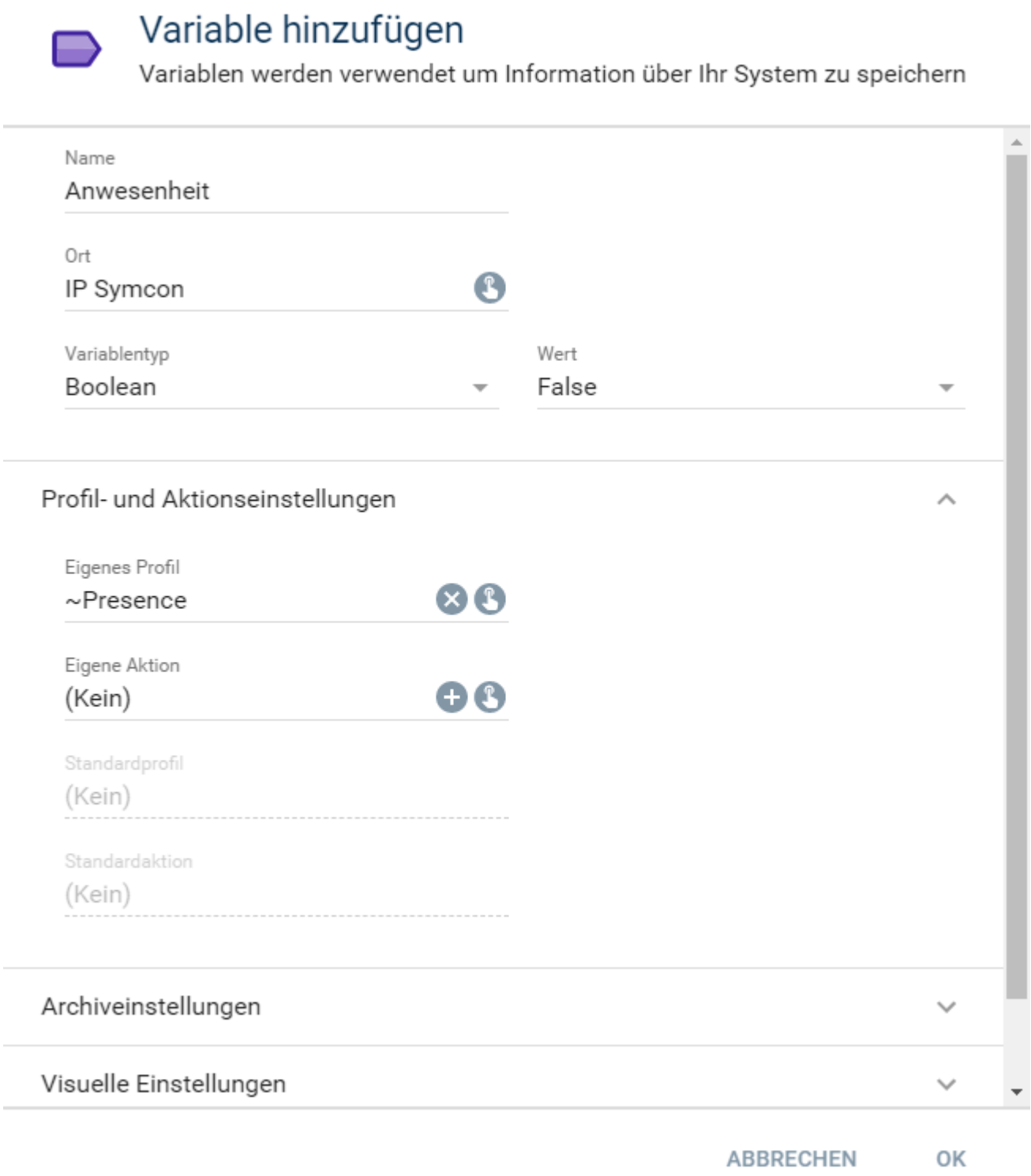
The previously created SMTP instance must be selected for "Action" "Switch instance" and for "Target". "SMTP_SendMail" must be selected for "Function", otherwise no e-mail will be sent. The message that is to be sent by e-mail at the selected time can now be entered under Parameter. An e-mail will now be sent every day from Monday to Friday, reminding about medication intake.

2. EMAIL FOR SPECIFIC EVENT

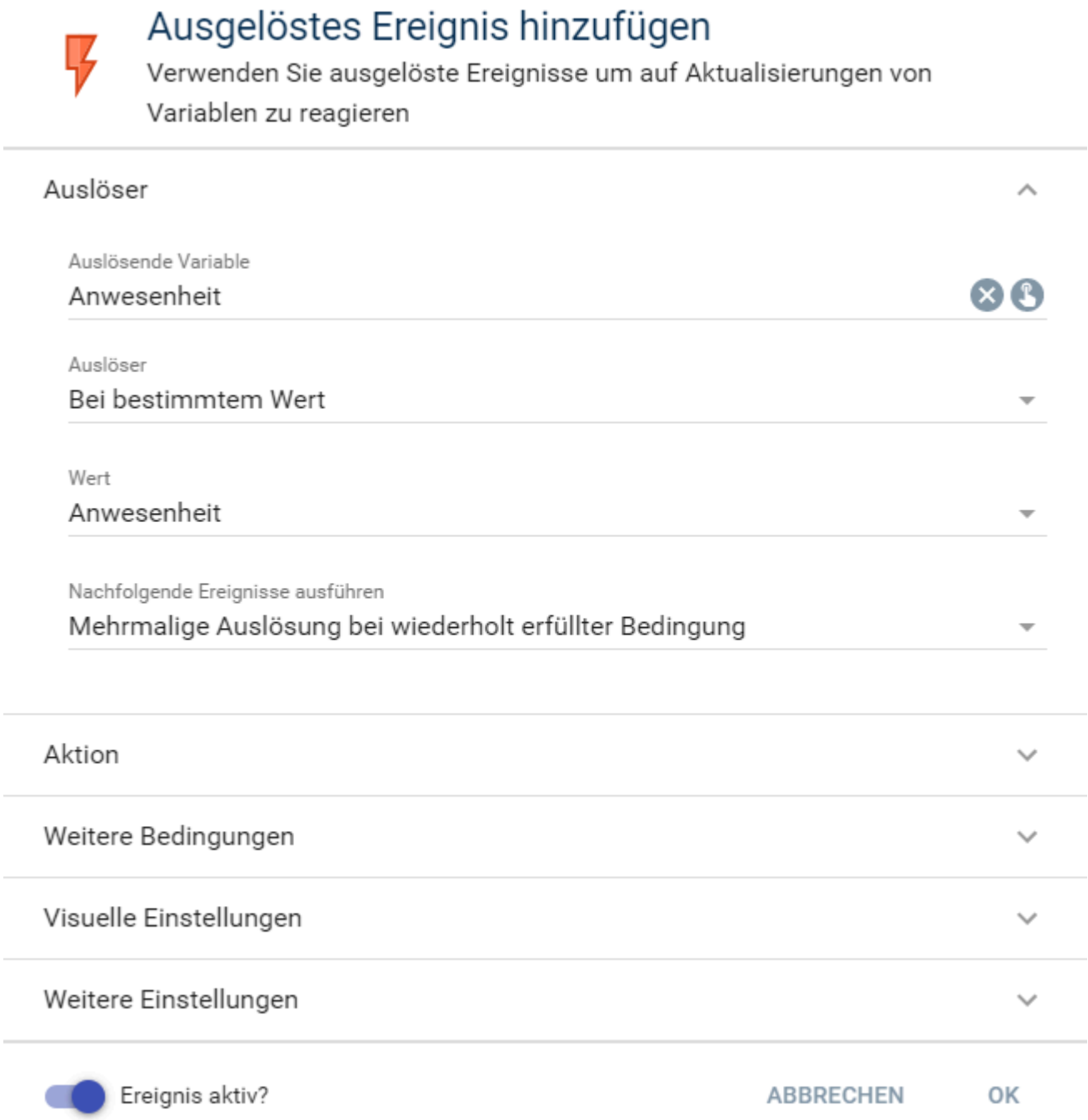
In this example, an e-mail is to be sent which is linked to a specific event.
For example, an e-mail should be sent on the go, if a window or door was opened without permission and a burglar might be at work.

CREATE VARIABLE AND EVENT

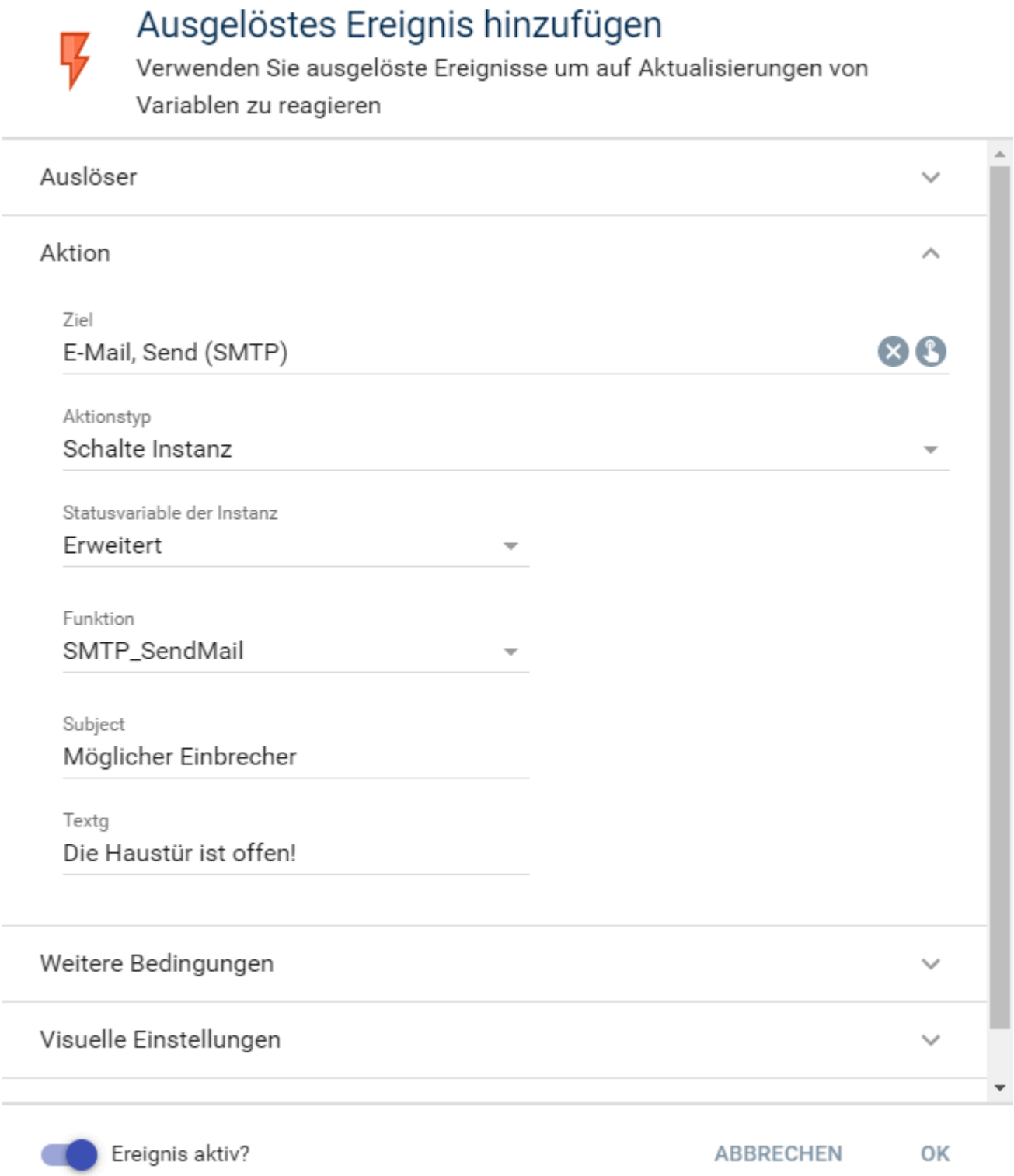
To implement the example, an event can be linked to an existing status variable or a self-created variable. Variables are data holders that enable switching on and off (e.g. the door sensor) to work and be displayed in the WebFront.
In this example, the self-created variable represents an imaginary door sensor.
A new variable can be created by clicking on "+" in the Object tree of the Management Console or by selecting the item "Add object" -> "Variable" -> "Add variable" via the context menu. The second option has the advantage that the variable is created directly below the object selected by in the object tree and does not have to be sorted in afterwards.
"Boolean" must now be selected for "Type" in the "Add variable" menu, because an e-mail should be sent when the door sensor is activated.



Then "~Presence" can be selected as "Own Profile", since the presence of a person or an open door in the house should be displayed.
Then a name for the variable must be chosen. It is recommended to choose descriptive names (e.g.: "Presence").
Now, instead of a "Cyclic" a "Triggered event" must be created. In the configuration of the event, the created presence variable must be selected under "Variable".



“Trigger” in this example is “At specific value”. The "Value" must be set to "True" (True=Present; False=Away) and "Run subsequent events" must be checked.



In "Action" "Switch instance" and in "Target" the previously created SMTP instance must be selected. "SMTP_SendMail" must be selected for "Function", otherwise no e-mail will be sent. The message that is to be sent by e-mail at the selected time must now be entered in the parameter. Now an email is sent when the variable is set to true (imaginary door opened).

Module Reference



The Product Page describes what IP-Symcon is.
A table with all supported systems is listed at Interfaces.

In the Module Reference, natively supported modules are shown. In the corresponding module references, tutorials, examples, and Tips & Tricks for the respective system. Furthermore, the module specific commands are listed.



For a list of IP Symcon specific commands, see the Command Reference.

DEVICES

Device-specific modules

1-Wire	Functions for accessing 1-Wire components
DS2405	1-channel switch
DS2406	2-channel switch
DS2408	8-channel switch
DS2413	2-channel switch
DS2438	Smart Battery Monitor
DS2450	A/D converter 16 bit
DS2890	Digital potentiometer 8 bit
Device List	Device list 1-Wire
OW_RequestStatus	Executes a new read for the device with the ID __InstanceID__.
ABL	Setup of ABL devices
Device-List	
Alfen	Alfen device setup
Device List	
ALLNET	Functions for controlling ALLNET components

ALL_SetAnalog	sets the analog value of a channel
ALL_SwitchActor	switches an ALLNET actuator on/off
ALL_SwitchMode	Switches the device with the ID __InstanceID__ to value __Status___
ALL_UpdateValues	Executes a update of the state variables for the device with the ID __InstanceID__.
Device list	Device list ALLNET

BACnet	Functions for controlling BACnet components

BAC_RelinquishPresetValue	Resetting the preset value

Catan

CATAN_CancelOverride	Cancels the overwriting of the channel
CATAN_ResetCounter	Resets the counter reading
Device List	

digitalSTROM	Functions for controlling digitalSTROM components

DS_CallScene	calls up a scene
DS_DimSet	dims a digitalSTROM terminal to a specific value
DS_MakeRequest	sends a digitalSTROM command directly to the dSS
DS_RequestStatus	retrieves the status
DS_ShutterMove	moves the roller shutter to a desired position/stop
DS_ShutterMoveDown	moves the roller shutter down to the end position/stop
DS_ShutterMoveUp	moves the roller shutter up to the end position/stop
DS_ShutterStop	stops a motion
DS_SwitchMode	switches a digitalSTROM terminal on/off
Device list	DigitalSTROM device list

DMX / ArtNet	Functions for controlling DMX components

DMX_FadeChannel	Dims the __Channel__ of the device with ID __InstanceID to __Value__ with a certain __FadeTime__
DMX_FadeChannelDelayed	Dims the __Channel__ of the device with ID __InstanceID to __Value__ with a certain __FadeTime__. You can specify a __DelayTime__ after which the dimming starts.
DMX_FadeRGB	dims the RGB channel with a fade time
DMX_FadeRGBDelayed	dims the RGB channel with a fade time
DMX_RequestInfo	Queries the information of a DMX instance
DMX_ResetInterface	Sets the DMX interface back with the ID __InstanceID__. It is recommended that in case of problems, create a script with this command and declare it as startup script. -
DMX_SetBlackout	Sets all channels of DMX interfaces with ID __InstanceID__ to value __ Status__.
DMX_SetChannel	Switches the channel from the device with the ID __InstanceID__ to value __Wert__
DMX_SetRGB	sets the RGB channel to a certain value
Device list	Device list DMX

EgiGeoZone	Receives data from the EgiGeoZone app for Google Android.
ekey	Functions for controlling ekey components

Device list	Device list eKey

ekey bionyx

EnOcean	Functions for accessing EnOcean components

ENO_DimSet	Dims the device with ID __InstanceID__ to value __Intensity__.
ENO_SetActiveMessage	activates a certain message
ENO_SetFanStage	sets the FanStage to a certain value
ENO_SetIntensity	Dims the device with ID __InstanceID__ to value __Intensity__. The value __Status__ indicates whether the device is to be switched on/ off.
ENO_SetLockFanStage	locks the FanStage
ENO_SetLockRoomOccupancy	locks the occupancy
ENO_SetMode	sets the actuator in a certain mode
ENO_SetPosition	sets the position of the actuator
ENO_SetRoomOccupancy	sets the occupancy to true/false
ENO_SetTemperature	sets the actuator to a certain target temperature
ENO_SetTemperature1	sets the device to a certain temperature
ENO_ShutterMoveDown	moves the roller shutter down to the end position/stop
ENO_ShutterMoveDownEx	moves the roller shutter for a certain time
ENO_ShutterMoveUp	moves the roller shutter up to the end position/stop
ENO_ShutterMoveUpEx	moves the roller shutter for a certain time
ENO_ShutterStop	stops a motion
ENO_SwitchMode	Switches the device with ID __InstanceID__ to value __Status__
ENO_SwitchModeEx	Switches the device with ID __InstanceID__ to value __Status__
Device list	Device list EnOcean/EEP

FHZ1X00PC	Functions for controlling FS20, HMS, FHT components

FHT	Funk-Heizungssystem
FS20	Wireless system for digital devices
Device list	Device list FHZ
HMS	HouseMessageSystem for analogue devices
KS300	Combination sensor weather station

FS10 Weather	Wireless weather station on 433 MHz
GARDENA smart system	Module for adding GARDENA smart system devices

smart Irrigation Control	A GARDENA irrigation control with 6 valves
smart SILENO Mower	A smart SILENO Mower from GARDENA
smart Power Socket	A power socket from GARDENA
smart Pressure Pump	A GARDENA smart Pump with one valve
smart Sensor	A GARDENA smart sensor with various measured values
smart Water Control	A GARDENA Irrigation Control with one valve

Geofency	Receives data from the Geofency app for Apple iOS.
Heating Control	Functions for controlling the heating module for regulating room temperatures
HC_TargetValue	Sends via the instance with the ID __ InstanceID__ a target value to the heating module.
Home Connect	Module for adding Home Connect devices
Device list	Device list Home Connect
Home Connect Device	The device instance created by the Home Connect configurator
HomeMatic	
Device list	HomeMatic Device List
HomeMatic CCU	Functions for controlling HomeMatic components
HomeMatic HCU (Beta)	
Image Grabber	Module for querying webcam images
IG_UpdateImage	updates the image of the instance
IPS-868	Functions for controlling IPS868 components

AKM-868	Presence control module including tracker
ANA-868	Analog-to-digital converter
EKM-868	An energy control module with a 4-channel counter module with 32 bits each
FD-868	A wireless display with a 2x16 character LC display
Device list	Device list IPS 868
JKM-868 LevelJET	A reader for LevelJET level indicator
JKM-868 ThermoJET	A reader for individual values of the ThermoJET temperature controller
LGS-868	Air quality sensor for room air analysis
RGBW-868	Control unit for LED RGBW strips
SERVO-868	A 4-way servo control module
WDT-868	A watchdog timer for monitoring the system

IR Trans	Functions for controlling infrared components

Device list	Device list IR-Trans
IRT_SendOnce	sends an IR command

KEBA

Device List

KNX	Functions for controlling KNX components

EIB_Char	Sends a single letter
EIB_Counter8Bit	sends an 8-bit counter value
EIB_Counter16Bit	sends a 16bit counter value
EIB_Counter32bit	sends a 32bit counter value
EIB_Date	sends the date value
EIB_DimControl	Sends a dim value date __Value__ to the device with the ID __InstanceID__. Depending on the configuration 0..15 (Standard) or -7..7 (Enhanced) can be sent.
EIB_DimValue	Sends the dim value __Value__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_DriveBladeValue	Sends the position value __Value__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_DriveMove	Moving the device with ID __InstanceID__ in a particular __Direction__. Depending on configuration the movement direction can be inverted. -
EIB_DriveShutterValue	Sends the position value __Vert__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_DriveStep	Moving the device __Stepwise__ with ID __InstanceID__ into a determined __Direction__. Depending on the configuration, the movement direction can be inverted.
EIB_FloatValue	sends a float value
EIB_Move	moves a roller shutter
EIB_Position	moves a roller shutter to a position
EIB_PriorityControl	sets a device state
EIB_PriorityPosition	Sets the device with ID __InstanceID__ to a certain __Direction__. Depending on configuration the direction can be inverted.
EIB_RequestStatus	sends a read request for EIB instances to the bus
EIB_Scale	Sends the Value __Value__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_SetRGB	sets an RGB stripe to a specific color
EIB_SetRGBW	sets an RGBW stripe to a specific color
EIB_Str	sends a text
EIB_Switch	Switches the device with the ID __InstanceID__ to value __Status__
EIB_Time	sends a time value

EIB_Value	Sends the value __Value__ to the device with ID __InstanceID__.
Device list	Device list EIB/KNX
KNX_RequestStatus	sends a read request for DPT instances on the bus

LCN	Functions for controlling LCN components

Device list	Device list LCN
LCN_AddGroup	adds a device to a group
LCN_AddIntensity	Dims the device with ID __InstanceID__ brighter by the value __Intensity__.
LCN_AddThresholdCurrent	adds threshold to current value
LCN_AddThresholdDefined	adds threshold to predefined value
LCN_Beep	Lets the instance with the ID __InstanceID__ beep
LCN_DeductIntensity	Dims the device with ID __InstanceID__ darker by the value __Intensity__.
LCN_DeductThresholdCurrent	deducts threshold from current value
LCN_DeductThresholdDefined	deducts threshold from predefined value
LCN_FadeOut	Dims the devive with ID __InstanceID__ to value __Intensity__ and dims then with the __Ramp__ seconds down to 0%.
LCN_FlipRelay	Switches over the relay with the ID __InstanceID__
LCN_LimitOutput	limits an output for a certain period of time
LCN_LoadScene	calls up a scene for an output
LCN_LockTargetValue	locks a controller of the device
LCN_RampStop	Stops the dimming process of the device with the ID __InstanceID__
LCN_ReleaseTargetValue	unlocks a controller of the device
LCN_RemoveGroup	removes a device from a group
LCN_RequestLights	queries the LEDs of the panel
LCN_RequestRead	queries the values
LCN_RequestStatus	queries all statuses of the module
LCN_RequestThresholds	queries thresholds and hysteresis
LCN_SaveScene	saves a scene for an output
LCN_SelectSceneRegister	selects a register
LCN_SendCommand	Sends a custom command to the instance with the ID __InstanceID__. For possible commands, refer to the PCHK documentation. If you do not have this, you can possibly refer to your dealer. We do not provide this option.
LCN_SetDisplayText	Shows a text on a line of the display
LCN_SetDisplayTime	sets the display time of a line in the display
LCN_SetIntensity	Dims the device with the ID __InstanceID__ to value __Intensity__ in __Ramp__ seconds.
LCN_SetLamp	switching an LED
LCN_SetRelay	sets all relays in a module (8 bit)
LCN_SetRGBW	sets an instance to an RGBW color value

LCN_SetTargetValue	Sets the regulator __Target__ of the instance with the ID __InstanceID__ to the value __Value__
LCN_ShiftTargetValue	Shifts the current target value
LCN_ShutterMove	starts shutter movement to percentage position
LCN_ShutterMoveDown	starts a shutter movement downwards
LCN_ShutterMoveUp	starts a shutter movement upwards
LCN_ShutterStop	stops a shutter movement
LCN_StartFlicker	Let the device with the ID __InstanceID__ flicker
LCN_StopFlicker	Stop flickering of the device with the ID __InstanceID__ immediately
LCN_SwitchDurationMin	after how many minutes it should be dimmed down
LCN_SwitchDurationSec	after how many seconds it should be dimmed down
LCN_SwitchMemory	Switches the device with the ID __InstanceID__to a stored value
LCN_SwitchMode	switches a device on/off
LCN_SwitchRelay	Switches the relay with the ID __InstanceID__ to value __Status__
LCN_SwitchRelayTimer	switches a relay on a time factor

LJQuick	The KNX quick (Lingg&Janke) module supports the setup of KNX instances.
LJ_SendDateTime	Sends the current date/time to the KNX bus

Matter

Note on Using Matter with Docker

Device list

M-Bus	Functions for controlling M-Bus components

Device list	Device list M-Bus
MBUS_UpdateValues	updates all device-specific values

Mennekes	Mennekes device setup

Device List

Modbus RTU/TCP	Functions for controlling ModBus RTU/TCP components

Device list	Device list ModBus RTU/TCP
ModBus_RequestRead	Executes a new read for the device with the ID __InstanceID__.
ModBus_WriteCoil	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegister	writes a value to the write address
ModBus_WriteRegisterByte	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterChar	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterDWord	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterInt64	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterInteger	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterReal	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterReal64	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterShort	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterString	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterWord	Sets address with ID __InstanceID__ to __Value__
Templates (file format)	Structure and file format of ModBus Device templates

Möhlenhoff Alpha 2	Receiving and switching Möhlenhoff Alpha2 data

MA2_RequestStatus	Gets the values stored in the Alpha2 with the InstanceID and sets the associated variables.
MA2_WriteValue	Writes the Value into the variable with the Ident in the Möhlenhoff Alpha 2 with the InstanceID.

MQTT	Functions for controlling MQTT components

Device list	Device list MQTT
MQTT Client	Functions for controlling MQTT Client
MQTT-Server	Functions for controlling MQTT Server

NEA Smart	Receiving and switching NEA Smart data

NEAS_RequestStatus	Gets the values stored in the NEA Smart with the InstanceID and sets the associated variables.
NEAS_WriteValue	Writes the Value to the variable with the ident in the NEA Smart with the InstanceID.

OCPP	The OCPP protocol allows wallboxes to be conveniently monitored and switched via Symcon.

OCPP_RemoteStartTransaction	If the status is 'Preparing' or 'Finishing', loading can be enabled by this command.
OCPP_RemoteStopCurrentTransaction	If the state is 'Charging', this command can be used to stop charging.
OCPP_RemoteStopTransaction	If the state is 'Charging', this command can be used to stop charging.

OPC UA	Functions for controlling OPC UA components
SageGlass (BACnet)	Integrates the SageGlass SIM II (BACnet) with IP-Symcon
Shutter Control	Functions for controlling shutter (roller shutter) components

SC_Move	moves the shutter to a specific position
SC_MoveDown	moves the shutter down to the end position
SC_MoveUp	moves the shutter up to the end position
SC_Stop	stops a movement

Siemens OZW	Functions for controlling Siemens OZW components

Device list	Device list Siemens OZW
OZW_RequestStatus	requests the status of a device
OZW_WriteDataPoint	writes a specific value to a data point

SNMP	Show values for any OIDs of a walk. Optionally, these can be created and described as variables.

SNMP_UpdateValues	Updates the variable values which are below the instance.

Snom	Integration of Snom IP phones
PLC: Siemens, Vipa, Logo	Functions for controlling Siemens/Vipa/Logo PLC components

Device list	Device list SPS Siemens/Vipa
Logo-VM-Addresses	VM addresses of Logo7/8
S7_RequestRead	performs a read operation on a device
S7_Write	writes a value to the configured address
S7_WriteBit	Set the address with the ID __InstanceID__ to value status
S7_WriteByte	Set the address with the ID __InstanceID__ to __Value__
S7_WriteChar	Set the address with the ID __InstanceID__ to __Value__
S7_WriteDWord	Set the address with the ID __InstanceID__ to __Value__
S7_WriteInteger	Set the address with the ID __InstanceID__ to __Value__
S7_WriteReal	Set the address with the ID __InstanceID__ to __Value__
S7_WriteShort	Set the address with the ID __InstanceID__ to __Value__
S7_WriteWord	Set the address with the ID __InstanceID__ to __Value__

PLC: Wago, Beckhoff, ABB	Functions for controlling Wago/Beckhoff PLC components
Sync Remote	Synchronization between two Symcon systems
Technische Alternative	Functions for controlling Technische Alternative components

Device list	Device list Technische Alternative

Door Intercom

Device list	
2N	
DoorBird	
FANVIL	
FRITZ!Box	
DoorIP_Accept	accepts the current call.
DoorIP_Decline	rejects the current call.
DoorIP_SendDTMF	sends DTMF characters on the current connection.

Voice over IP	Functions for controlling VoIP

Device list	Device list Voice over IP
VoIP_AcceptCall	accepts a connection
VoIP_Connect	establishes a connection
VoIP_Disconnect	terminates a connection
VoIP_GetConnection	returns information about a connection
VoIP_GetData	returns data of a connection
VoIP_PlayWave	plays a wave file
VoIP_RejectCall	rejects a connection
VoIP_SendDTMF	sends a sequence of characters as DTMF
VoIP_SetData	sets the data of a connection

Weishaupt	Installation of Weishaupt devices
Device List	

WinLIRC	Functions for controlling WinLIRC components
WinLIRC_SendOnce	Sends a remote command on the instance with ID __InstanceID__. The values remote control and button specify, which command should be sent. These must be set up accordingly in the client software (WinLIRC/ IRTrans Tray).

Wireless M-Bus	Functions for controlling wireless M-Bus components
Device list	Wireless M-Bus device list

WMRS200	Functions for controlling WMRS200 components
Device list	Device list WMRS200

W&T	Functions for controlling W&T components
WUT_SwitchMode	Switches the channel with ID __InstanceID__ to value __Status__.

XBee	Functions for controlling XBee
XBee_SendBuffer	sends a data string to a specific device
XBee_SendCommand	sends a command to a specific XBee-Splitter

Eaton xComfort	Functions for controlling xComfort components

Device list	Device list Eaton xComfort
MXC_DimBrighter	starts dimming to a brighter level
MXC_DimDarker	starts dimming to a darker level
MXC_DimSet	dims an xComfort device to a specific level
MXC_DimStop	stops a dimming process
MXC_RequestStatus	sends a status request to a device
MXC_SendBoolean	Sends a boolean value to the instance with ID __InstanceID__, which is connected to a room manager.
MXC_SendFloat	Sends a float to the instance with ID __InstanceID__, which is connected to a room manager.
MXC_SendInteger	Sends a boolean value to the instance with ID __InstanceID__, which is connected to a room manager.
MXC_SetTemperature	Sets the new target temperature of the control valve with the ID __InstanceID__ to the new __Temperature__.
MXC_ShutterMoveDown	moves the shutter down to the end position/stop
MXC_ShutterMoveUp	moves the shutter up to the end position/stop
MXC_ShutterStepDown	moves the shutter down a step
MXC_ShutterStepUp	moves the shutter up a step
MXC_ShutterStop	stops a movement
MXC_SwitchMode	switches an xComfort device on/off

Z-Wave	Functions for controlling Z-Wave components

Device list	Device list Z-Wave
ZW_Basic	Turns the basic function of the specified device with InstanceID __InstanceID__
ZW_ColorCW	sets the cold-white channel of a RGBWW Z-Wave device
ZW_ColorRGB	sets the color of a Z-Wave device
ZW_ColorRGBWW	sets the colors and intensity of an RGBWW Z-Wave device
ZW_ColorWW	sets the WarmWhite channel of a Z-Wave device
ZW_DimDown	starts dimming down
ZW_DimDownEx	starts dimming down with a runtime
ZW_DimSet	dims a Z-Wave device to a specific level
ZW_DimSetEx	dims a Z-Wave device to a specific level within a runtime
ZW_DimStop	stops dimming a Z-Wave device
ZW_DimUp	starts dimming up
ZW_DimUpEx	starts dimming up within a runtime
ZW_DoorLockOperation	Sets the door lock operation mode of a Z-Wave device
ZW_LockMode	sets the LockMode
ZW_MeterReset	resets the meter
ZW_Optimize	Starts the wireless network optimization process for a Z-Wave device
ZW_ProtectionSet	sets the Protection and its Mode
ZW_RequestStatus	Queries the status from the device with the ID __InstanceID__. Only power-operated devices can be queried directly by command, because battery-operated devices do not permanently listen to the queries.
ZW_ShutterMoveDown	Moves the shutter with the ID __InstanceID__ downwards until (link: page://f1hDSVmdoBuVvjEW text: MXC_ShutterStop) is running or the end position is reached.
ZW_ShutterMoveUp	Moves the shutter with the ID __InstanceID__ upwards until (link: page://f1hDSVmdoBuVvjEW text: MXC_ShutterStop) is running or the end position is reached.
ZW_ShutterStop	Stops a current movement of the shutter.
ZW_SwitchAllMode	Sets the Switch All mode of a Z-Wave device
ZW_SwitchMode	switches a Z-Wave device on/off
ZW_Test	Tests if a connection to a Z-Wave device is working
ZW_ThermostatFanModeSet	Sets the FanMode for the thermostat
ZW_ThermostatModeSet	Sets the thermostat mode

ZW_ThermostatSetPointSet	Sets the PointSet Value
Zevvy	Sends data to a Zevvy account
ZY_SendMeasurements	Sends the data of the set up variables to a Zevvy account

LOGIC

More complicated logic

Active List	Overview of all active variables in the Visualization
AL_SwitchOff	Switches all active variables in the list to inactive.
Presence Simulation	Simulate presence via logged values
AS_SetSimulation	De-/Activate the presence simulation
Image Archive	The module copies an image when triggered by a selected variable.
BA_AddImage	Copies the current source image to the "Images" category of the "ImageArchive" module with the InstanceID.
Countdown	
CSV ZIP Export	Exports aggregated values of a variable as a CSV file in a ZIP archive
CSV_DeleteZip	Removes the generated file.
CSV_Export	Creates a zip archive based on the given parameters.
CSV_SendMail	Sends a mail through an SMTP instance with a generated zip file
Dummy Module	Placeholder module for display as an instance in the visualization
Egg Timer	An egg timer that runs for a certain amount of time and can be rewound at any time. Ideal for use in events and scripts.
Group Control	With the help of the 'Group Control', variables can be switched together in groups.
JSON Decoder	Decodes JSON and makes the content available in variables
JSON Exporter	Exports any variable in a user-defined structure via an HTTP endpoint.
Logic Gate	Links variables by logical operations
Computation Module	Modules, which enables various calculations within IP-Symcon

Computation Module	Calculation of different auxiliary values based on a group of variables, e.g. buzzer or average.
Converter	Converts a variable using a self-defined formula.
ConvertMultiBoundaries	Converts a variable within set limits using a self-defined formula.
ValueRangeScale	Scaling of a value within a value range

RGBMultiplexer	Connects individual R, G, B channels to a variable, which can be controlled with the color wheel.
RGBM_RequestStatus	Determines the value of the individual R, G, B channels and sets the color variable.
RGBM_SetRGB	Sends the R, G, B values to the individual channels and updates the color variable

DragPointer	A variable that remains at the highest or lowest value until reset.
SZ_Reset	Resets the drag pointer

Game Collection	A collection of simple games which can be played via the Visualization.
Rock Paper Scissors	A way to play rock, paper, scissors on the Visualization.
Number Guessing	A little game where a random number has to be guessed.

Scene Control	Stores the state of any number of actuators and makes them available via Visualization/PHP function.
SZS_CallScene	Calls the scene and sets it associated variables
SZS_GetActiveScene	Returns the number of the scene that is currently active.
SZS_SaveScene	Saves the values of the variables present in the list in the corresponding scene.

Dew Point Temperature Calculation	Calculation of the dew point temperature
Staircase Light Controls	When activated, the light in the staircase is switched on and is switched off again shortly after leaving the staircase.
THL_SetActive	Activates or deactivates the staircase lighting control.
THL_Start	Activates the light in the staircase and starts the timer, which deactivates the light again.
THL_Stop	Deactivates the light in the staircase and the timer.

Renamer	Enables renaming of selected objects via Visualization
Rain Central	Fetches the current rain radar image from Wetteronline and evaluates the color pixels of the rain amount.

UWZ_RequestInfo	Calculates the variable rain value of the rain central module
Variable Comparison	The VariablenVergleich module allows you to create a point cloud from two variables and compare them using linear regression.
LR_Download	Regenerates the graph and outputs an address
LR_GenerateChart	Regenerates the chart and returns it as SVG and PNG.
LR_UpdateChart	Regenerates the diagram
Virtuelle Devices	Modules for simulating various devices
E-Car (Simulation)	Simulates the charging process of an electric car
Heater (Simulation)	Simulates a heater
Light (Simulation)	Simulates a light
Mediaplayer (Simulation)	Simulates a MediaPlayer
PV-System (Simulation)	Simulation of a PV system
Shutter (Simulation)	Simulates a roller shutter
Thermostat (Simulation)	Simulates a thermostat
Consumption/Costs	Calculates the costs according to consumption
Counter (Simulation)	Simulates an adjustable number of counters.
Random Lighting	The module enables one to switch the color value of lamps randomly between different colors.
ZB_ChangeLight	Selects random colors and sets the selected variables to them

ENERGY

Something with energy

Work Efficiency	Efficiency calculation
ARZ_Calculation	Calculates the efficiency
Operating Hours Counter	Determines and displays the operating time of a device
BSZ_Calculate	The operating hours variable is set to the calculated value
Energy Dashboard	An overview with several diagrams
Energy Manager	Module, which switches the consumers on and off when energy is available.
Energy Distribution	

Energy Counter	Calculates the consumption
Energy Counter Pulse	Converts meter values to instantaneous consumption.
Energy Counter Power	Calculates instantaneous and cumulative power consumption from current/power.
Power Billing Module	The module provides a cost statement similar to the annual statement from the energy provider.
SAM_UpdateCalculations	Calculates all values
Power price	Reads out the current/predicted electricity prices from aWATTar, Tibber or Epex Spot (via ENTSO-E).
SPX_Update	Updates the data
Consumption per Category	Shows the consumption in percent in category.
VIK_CalculationConsumption	Recalculates the categories
Consumption within Timespan	Calculates the consumption between a start and end date
VIZ_Calculate	Recalculates consumption between start and end dates
Consumption Behaviour	The consumption behavior module displays the expected consumption.
VBV_UpdateCalculation	The function updates the predicted values
Calculated Counter	Adds positive variable changes according to adjustable rules with changes of a main variable and adds this to the value of a variable.
VM_Update	Berechnet die Steigung der sekundären Messstellen und addiert dieses mit dem Ergebnis
Virtual Counter	Input mask for manually read meter readings.
VZ_WriteNewCounterValue	After checking, sets the current counter reading
Reading (Day)	The meter reading can be displayed for a selected date
ZST_Calculate	Recalculates the value for the selected date
Meter Overflow	Meters which can overflow can be displayed as meters.

VISUALIZATIONS

Visualizations for Symcon

Tile Visualization	Functions to control the tile visualization on the client devices

VISU_OpenObject	opens a specific object in the Tile Visualization.
VISU_PostNotification	sends a push notification to the tile visualization
VISU_PostNotificationEx	sends a push notification to the tile visualization

WebFront Visualization	Functions for controlling the WebFront on the client computers

WFC_AudioNotification	sends an audio message to the WebFront
WFC_OpenCategory	Opens a category as a popup in the WebFront
WFC_PushNotification	sends a push message to the mobile apps
WFC_Reload	reloads the WebFront on the client computer
WFC_SendNotification	sends a small message to the WebFront
WFC_SendPopup	sends a large message to the WebFront
WFC_SwitchPage	switches the tab in the WebFront

VOICE ASSISTENTS

Voice assistants for integration

Amazon Alexa	Module to connect Amazon Alexa

Expert Options	Expert Options of Amazon Alexa
Television	Television in Amazon Alexa
Generic Switch	Generic Switches in Amazon Alexa
Generic Slider	Generic Sliders in Amazon Alexa
Speaker	Speaker in Amazon Alexa
Speaker (Muteable)	Speaker in Amazon Alexa
Light (Dimmer)	Light in Amazon Alexa, that can be dimmed
Light (Expert)	Expert light in Amazon Alexa
Light (Color)	Colored Light in Amazon Alexa
Light (Switch)	Light in Amazon Alexa, that can be switched on or off
Mediaplayer	Mediaplayer in Amazon Alexa
Shutters	Shutters in Amazon Alexa
Lock	Locks in Amazon Alexa
Scenes	Scenes in Amazon Alexa
Scenes (deactivatable)	Deactivatable Scenes in Amazon Alexa
Temperature Sensor	Temperature Sensors in Amazon Alexa
Thermostat	Thermostats in Amazon Alexa

Google Assistant	Module to connect Google Assistant

Expert Options	Expert Options of Google Assistant
Generic Switch	Generic Switches in Google Assistant
Light (Dimmer)	Light in Google Assistant, that can be dimmed
Light (Expert)	Expert light in Google Assistant
Light (Color)	Colored Light in Google Assistant
Light (Switch)	Light in Google Assistant, that can be switched on or off
Shutter	Shutters in Google Assistant
Scenes	Scenes in Google Assistant
Scenes (deactivatable)	Deactivatable Scenes in Google Assistant
Thermostat	Thermostats in Google Assistant

NOTIFICATIONS

Module, damit sich Symcon beim Nutzer melden kann

Alerting	Triggers an alarm when one of the sensor variables becomes active.

ARM_GetLastAlertID	Returns the ID of the variable that last triggered an alarm.
ARM_SetActive	Switches the alarming module with the InstanceID __InstanceID__ to the value __Value__ (true = On; false = Off).
ARM_SetAlert	Switches the alarm with the InstanceID __InstanceID__ to the value __Value__ (true = On; false = Off).

Notification	Enables multi-level notification, with the level increasing after a defined time and resetting when acknowledged.

BN_IncreaseLevel	Increases the level and performs the action
BN_Reset	Resets the notification level and disables the chain
BN_SetNotifyLevel	Sets the notification to a specified level

Announcement	The 'Announcement' Module provides the ability to play audio generated by AWS Polly through Sonos or Media Player.

DS_Play	Plays the text as an announcement.

Dynamic Mail	Sends an e-mail with a dynamic text.

DM_SendMail	Use the subject and text to send an e-mail.

Done Notifier	Reports after a dead time, after a device has fallen below a certain consumption.
IMAP	Functions of the IMAP module for retrieving e-mails

IMAP_DeleteMail	deletes a mail with a specific e-mail(UID)
IMAP_GetCachedMails	The command returns an array containing information about the cached emails of an IMAP instance with ID __InstanceID__. Should the instance not exist, an alert is generated.
IMAP_GetMailEx	This command loads an email with the UID __UID__ from the IMAP instance with ID __InstanceID__ and returns an array containing the data of the e-mail. Should the instance or an e-mail with the given UID not exist, an alert is generated.

MediaPlayer	Functions for controlling Media Player instances

WAC_AddFile	adds a music file to the playlist
WAC_ClearPlaylist	clears the playlist
WAC_GetPlaylistLength	returns the length of the playlist
WAC_GetPlaylistPosition	returns the current position in the playlist
WAC_Next	plays the next title of the playlist
WAC_Pause	pauses the playback
WAC_Play	plays the playlist
WAC_PlayFile	plays a music file directly
WAC_Prev	plays the previous title of the playlist
WAC_SetPlaylistPosition	sets the position in the playlist
WAC_SetPosition	Sets the position marker in the media player with the ID __InstanceID__ to the specified __Seconds__. The title must be played at execution time already.
WAC_SetRepeat	switches the repetition of the playlist on/off
WAC_SetShuffle	switches the shuffle playback on/off
WAC_SetVolume	sets the volume
WAC_Stop	stops the playback

POP3	Functions of the POP3 module for retrieving e-mails

POP3_DeleteMail	deletes a mail with a specific e-mail(UID)
POP3_GetCachedMails	The command returns an array containing information about the cached e-mails of a POP3 instance with ID __InstanceID__. Should the instance not exist, an alert is generated.
POP3_GetMailEx	returns an array with information for a specific e-mail (UID)

Popup Module	Stylistic module for display in Visualization
SMS	Functions for sending SMS via SMS module

SMS_Send	sends an SMS to a number

SMTP	Functions for the SMTP module for sending e-mails

SMTP_SendMail	sends an e-mail to the default address
SMTP_SendMailAttachment	sends an e-mail with attachment to the default address
SMTP_SendMailAttachmentEx	sends an email with an attachment to any address
SMTP_SendMailEx	sends an e-mail to any address
SMTP_SendMailMedia	sends an email with an attachment of a media object of type "image/sound" to the default address
SMTP_SendMailMediaEx	sends an e-mail with an attachment of a media object of the "image/sound" type to any address

Spotify	Controls the playback of Spotify

SPO_MakeAPIRequest	Makes a request to the Spotify web API with the specified parameters
SPO_NextTrack	If a current playback exists, the next song will be played.
SPO_Pause	If a current playback exists, it will be paused.
SPO_Play	If a current, possibly paused, playback exists, it will be continued.
SPO_PlayURI	Plays the Spotify resource with the URI __URI__ on the currently selected device.
SPO_PreviousTrack	If a current playback exists, the previous song will be played.
SPO_ResetToken	Resets the OAuth-token and thus the link to the Spotify account.
SPO_SetRepeat	Sets the repeat to the value Repeat.
SPO_SetShuffle	If Shuffle is true, random playback is enabled, otherwise disabled.

Fault Manager	Indicates when selected variables are active and can acknowledge them depending on the setting.
SymconReport	Creates a report as CSV or PDF depending on the modules

MailReport	Creates a CSV report of archive values and sends it by e-mail(SMTP)
PDFReport (Energy)	Creates a report of archive values of two variables of a time span as PDF
PDFReport (Multi)	Creates a report of archive values of several variables of a time span as PDF
PDFReport (Multi Energy)	
PDFReport (Single)	Creates a report of archive values of a variable with min/max/avg of a time span as PDF

Phone Announcement	The Telephone Announcement Module enables the convenient linking of a VoIP instance and a text-to-speech instance (AWS Polly) to call a phone number and output a text when the call is answered. In addition, the module can respond to DTMF tones and output other texts.

TA_StartCall	Starts a call based on the current setting.
TA_StartCallEx	Starts a call to a phone number and outputs the text

Phone Chain	The 'Phone Chain' module allows to call a list of telephone numbers one after another.
TelegramBot	Allows to send messages and perform actions
Text to Speech	Functions for controlling a TTS module

TTS_GenerateFile	generates a WAV file with the desired text
TTS_Speak	speaks any text on the selected sound card

TTSAWSPolly	Can generate sound files in various formats

TTSAWSPOLLY_GenerateData	Queries the text via AWS and returns the speech data in the return.
TTSAWSPOLLY_GenerateFile	Queries the text via AWS and returns the file name to the vocie data.

Consumption Alert	Monitors a counter variable for unusual consumption.

VBA_CheckAlert	Checks whether a limit value is exceeded

Water Alert	Monitors a meter variable for unnatural water consumption (leak/break).

WAA_CheckAlert	Checks whether the limit values have been exceeded

Watchdog	Monitors variables and triggers an alarm if they have not been updated or changed for too long.

WD_GetAlertTargets	returns an array with the overdue variables
WD_SetActive	de-/activate the instance

CORE INSTANCES

Module, welche im Kern von Symcon verankert sind

Archive Control	Functions for managing the data logging

AC_AddLoggedValues	adds further data records to a logged variable
AC_ChangeVariableID	migrates the data of a variable into a variable that has not yet been logged
AC_DeleteVariableData	deletes all data records of a logged variable in a certain period of time
AC_GetAggregatedValues	gets aggregated data from the archive
AC_GetAggregationType	indicates the aggregation type of a variable
AC_GetAggregationVariables	returns an array of all logged variables
AC_GetCompaction	returns the compaction entries of a variable
AC_GetCounterIgnoreZeros	gets the status whether zeros and negative values are ignored
AC_GetGraphStatus	asks whether a variable is being visualized
AC_GetLoggedValues	gets raw data from the Archive Control
AC_GetLoggingStatus	asks whether a variable is logged
AC_ReAggregateVariable	starts the re-aggregation of a variable
AC_SetAggregationType	sets the aggregation type of a variable
AC_SetCompaction	Configures the compression of the variable
AC_SetCounterIgnoreZeros	sets the status whether zeros and negative values are ignored
AC_SetGraphStatus	sets the property for the visualization of a variable
AC_SetLoggingStatus	sets the property for logging a variable
Data Format	

Permission Control	User management and synchronization with LDAP server
Calendar Control	Calendar module
Connect Control	Connection module used to enable a secure connection from the outside

Cutter

Cutter_ClearBuffer	clears the buffer of the cutter instance

DNS-SD Control	manages established Bonjour services

ZC_QueryService	requests information from a single service
ZC_QueryServiceType	requests information from services of a certain type
ZC_QueryServiceTypes	Queries all the services that have been set up

Event Control	Module to start scripts on set events
Location Control	calculates twilight times based on latitude and longitude
Module Control	manages external and added PHP modules
Notification Control	a module for sending push notifications

Presence Control	Functions for detecting presence within an area
PC_Enter	sets presence to true
PC_Leave	sets presence to false
RegisterVariable	RegisterVariables serve as an interface for forwarding and processing data
RegVar_GetBuffer	reads the internal buffer
RegVar_SendEvent	sends an event to the parent HID instance
RegVar_SendPacket	sends a data packet to the specified IP address
RegVar_SendText	sends data to the parent instance
RegVar_SetBuffer	sets the internal buffer
Skin Control	manages external and uploaded skins
SSDP Control	enables the search for UPnP devices
YC_SearchDevices	sends a request to the local network
System Information	Functions for querying system information and hardware
Sys_GetBattery	displays information about the battery
Sys_GetCPUInfo	shows the CPU usage
Sys_GetHardDiskInfo	shows the hard disk space/usage
Sys_GetMemoryInfo	shows the memory usage
Sys_GetNetworkInfo	shows information about the network adapters
Sys_GetProcessInfo	lists all processes
Sys_GetSpooler	displays information about the printer queues
Sys_GetURLContent	reads the content from an URL as String
Sys_GetURLContentEx	Reads the contents of the __URL__ and returns it as a return value. Here, some __Parameter__ for timeouts, authentication and proxy are made. -
Sys_Ping	sends a ping to a network device
Tailscale VPN	Use of Tailscale VPN on the SymBox
TextParser	provides text snippets from websites/files
Translation Control	
Util Control	serves as an interface with extended functions between the IP-Symcon service and the management console

UC_FindReferences	searches all places where an object is used
UC_GetEventStatistics	returns the number of updates of events
UC_GetIconList	returns the names of all available icons
UC_GetKernelStatistics	returns key figures on the message and script queue of the kernel
UC_GetLastLogMessages	returns the last log messages of a message type
UC_GetLogMessageStatistics	returns the number of log messages per message type
UC_GetMessageSenderIDList	returns the number of messages per sender
UC_GetMessageSenderIDSizeList	returns the amount of message data per sender
UC_GetMessageTypeList	returns the number of messages per message type
UC_GetScriptSenderList	returns the number of script executions per trigger
UC_ResetLastLogMessages	clears the stored last log messages
UC_ResetLogMessageStatistics	resets the number of log messages per message type

WebHook Control	Module for calling scripts via browser

WC_PushMessage	sends a WebSocket message to all connected clients
WC_PushMessageEx	sends a WebSocket message to a specific client

WebServer	Module which makes the Visualization accessible via port

I/O INSTANCES

Functions for managing I/O Instances

Client Socket	TCP interface

CSCK_SendText	sends a string to the I/O

HID	HID interface

HID_SendEvent	sends a string to the I/O

HTTP Client	Interface for reading websites out

WWW_UpdatePage	lets you query the configured URL

Multicast Socket	Multicast interface

MSCK_SendText	sends a string to the I/O

Serial Port	Serial interface

SPRT_SendText	sends a string to the I/O
SPRT_SetBreak	switches the break of a serial port instance
SPRT_SetDTR	switches the DTR of a serial port instance
SPRT_SetRTS	switches the RTS of a serial port instance

Server Sent Event Client	TCP interface
Server Socket	opens a port for the network

SSCK_SendPacket	sends a string to the I/O
SSCK_SendText	sends a string to the I/O

UDP Socket	UDP interface

USCK_SendPacket	sends a string to the I/O
USCK_SendText	sends a string to the I/O

Virtual I/O	Virtual I/O interface

VIO_Connect	sends a Connect Data-Packet
VIO_Disconnect	sends a Disconnect Data-Packet
VIO_GetPacketList	supplies an array of extended Data-Packets
VIO_GetTextList	returns an array of simple Data-Packets
VIO_PushPacket	sends an extended Data-Packet
VIO_PushPacketHEX	sends an extended Data-Packet
VIO_PushText	sends a simple Data-Packet
VIO_PushTextHEX	sends a simple Data-Packet
VIO_SendPacket	sends an extended Data-Packet to itself
VIO_SendText	sends a simple Data-Packet to itself

WebSocket Client	TCP interface

WSC_SendMessage	sends a message using the WebSocket

BACKUPS

Data should be backed up

Backup	Creates a backup via SFTP, FTP or FTPS
SB_CreateBackup	Creates or updates a backup

LEGACY

Modules that should no longer be actively maintained or used

RRDTool	Direct integration of RRDTool
RRD_Exexute	Directs a command to the RRD Tool Library and executes it. Once the command has been executed, the execution will be proceed.
Shutter Control (legacy)	Functions for controlling shutter components (legacy)
SC_Move	moves the shutter to a specific position
SC_MoveDown	moves the shutter down to the end position
SC_MoveUp	moves the shutter up to the end position
SC_Stop	stops a motion
USBMapper	Automatically sets the correct USB port to the entered serial ports.
USBM_FixPorts	Checks whether the USB ports are still configured correctly
Web Graph	Any graphs can be displayed independently via the WebHook
Wunderground Weather	Gets weather data (current/hourly/12-hourly/weather warning) via API from www.wunderground.com
WGW_UpdateStormWarningData	Updates the severe weather warnings of the Weatherground module
WGW_UpdateWeatherData	Updates the weather data (current/hourly/daily) of the Weatherground module

Devices

1-WIRE

Functions for accessing 1-Wire components

DS2405	1-channel switch
OW_SwitchMode	Sets the state of the device with the ID __InstanceID__ to value __Status__.
OW_ToggleMode	Changes the state of the device with the ID __InstanceID__. The current value can be queried in the state variable after the operation. -
DS2406	2-channel switch
OW_SetPin	Sets the state of the device with the ID __InstanceID__ to value __Status__
DS2408	8-channel switch
OW_SetPin	sets a pin from the DS2408 to on/ off
OW_SetPort	Sets the state of all eight pins on the value of __Bitmask__ of the device with ID __InstanceID__
OW_SetStrobe	Makes it possible to put the RSTZ-pin of DS2408 of the device with ID __InstanzID__ to value __Status__
OW_WriteBytes	Sets a sequence of bytes (__Data__) from all pins of the device with the ID __InstanceID__.
OW_WriteBytesMasked	Sets a sequence of bytes (__Data__) taking into account a bitmask __Mask__ from all pins of the device with the ID __InstanceID__.
DS2413	2-channel switch
OW_SetPin	Sets the state of the pin __Pin__ of the device with the ID __InstanceID__ to value __Status__
DS2438	Smart Battery Monitor
DS2450	A/D converter 16 bit
OW_SetPin	Sets the state of the pin __Pin__ of the device with the ID __InstanceID__ to value __Status__
DS2890	Digital potentiometer 8 bit

OW_SetPosition	Sets the device with the ID __InstanceID__ to value __Value__

Device List	Device list 1-Wire
OW_RequestStatus	Executes a new read for the device with the ID __InstanceID__.

ABL

Setup of ABL devices

Device-List

ALFEN

Alfen device setup

Device List

ALLNET

Functions for controlling ALLNET components

ALL_SetAnalog	sets the analog value of a channel
ALL_SwitchActor	switches an ALLNET actuator on/off
ALL_SwitchMode	Switches the device with the ID __InstanceID__ to value __Status___
ALL_UpdateValues	Executes a update of the state variables for the device with the ID __InstanceID__.
Device list	Device list ALLNET

BACNET

Functions for controlling BACnet components

BAC_RelinquishPresetValue	Resetting the preset value

CATAN

CATAN_CancelOverride	Cancels the overwriting of the channel
CATAN_ResetCounter	Resets the counter reading

Device List

DIGITALSTROM

Functions for controlling digitalSTROM components

DS_CallScene	calls up a scene
DS_DimSet	dims a digitalSTROM terminal to a specific value
DS_MakeRequest	sends a digitalSTROM command directly to the dSS
DS_RequestStatus	retrieves the status
DS_ShutterMove	moves the roller shutter to a desired position/stop
DS_ShutterMoveDown	moves the roller shutter down to the end position/stop
DS_ShutterMoveUp	moves the roller shutter up to the end position/stop
DS_ShutterStop	stops a motion
DS_SwitchMode	switches a digitalSTROM terminal on/off
Device list	DigitalSTROM device list

DMX / ARTNET

Functions for controlling DMX components

DMX_FadeChannel	Dims the __Channel__ of the device with ID __InstanceID to __Value__ with a certain __FadeTime__
DMX_FadeChannelDelayed	Dims the __Channel__ of the device with ID __InstanceID to __Value__ with a certain __FadeTime__. You can specify a __DelayTime__ after which the dimming starts.
DMX_FadeRGB	dims the RGB channel with a fade time
DMX_FadeRGBDelayed	dims the RGB channel with a fade time
DMX_RequestInfo	Queries the information of a DMX instance
DMX_ResetInterface	Sets the DMX interface back with the ID __InstanceID__. It is recommended that in case of problems, create a script with this command and declare it as startup script. -
DMX_SetBlackout	Sets all channels of DMX interfaces with ID __InstanceID__ to value __ Status__.
DMX_SetChannel	Switches the channel from the device with the ID __InstanceID__ to value __Wert__
DMX_SetRGB	sets the RGB channel to a certain value
Device list	Device list DMX

EGIGEOZONE

Receives data from the EgiGeoZone app for Google Android.

EKEY

Functions for controlling ekey components

Device list	Device list eKey

EKEY BIONYX

ENOCEAN

Functions for accessing EnOcean components

ENO_DimSet	Dims the device with ID __InstanceID__ to value __Intensity__.
ENO_SetActiveMessage	activates a certain message
ENO_SetFanStage	sets the FanStage to a certain value
ENO_SetIntensity	Dims the device with ID __InstanceID__ to value __Intensity__. The value __Status__ indicates whether the device is to be switched on/ off.
ENO_SetLockFanStage	locks the FanStage
ENO_SetLockRoomOccupancy	locks the occupancy
ENO_SetMode	sets the actuator in a certain mode
ENO_SetPosition	sets the position of the actuator
ENO_SetRoomOccupancy	sets the occupancy to true/false
ENO_SetTemperature	sets the actuator to a certain target temperature
ENO_SetTemperature1	sets the device to a certain temperature
ENO_ShutterMoveDown	moves the roller shutter down to the end position/stop
ENO_ShutterMoveDownEx	moves the roller shutter for a certain time
ENO_ShutterMoveUp	moves the roller shutter up to the end position/stop
ENO_ShutterMoveUpEx	moves the roller shutter for a certain time
ENO_ShutterStop	stops a motion
ENO_SwitchMode	Switches the device with ID __InstanceID__ to value __Status__
ENO_SwitchModeEx	Switches the device with ID __InstanceID__ to value __Status__
Device list	Device list EnOcean/EEP

FHZ1X00PC

Functions for controlling FS20, HMS, FHT components

FHT	Funk-Heizungssystem	
	FHT_SetMode	Sets the new desired mode for the FHT with ID __InstanzID__ to value __Mode__.
	FHT_SetTemperature	Sets the new desired temperature for the FHT with ID __InstanzID__ to value __Temperature__.
FS20	Wireless system for digital devices	

FS20_DimDown	Dims the device with ID __InstanceID__ down a level.
FS20_DimUp	Dims the device with ID __InstanceID__ up a level.
FS20_SetIntensity	Dims the device with ID __InstanceID__ to value __Intensity__ in __Duration__ seconds.
FS20_SwitchDuration	Switches the device with ID __InstanceID__ to value __Status__ in __Duration__ seconds.
FS20_SwitchMode	Switches the device with ID __InstanceID__ to value __Status__.

Device list	Device list FHZ
HMS	HouseMessageSystem for analogue devices

HMS_ReleaseFI	Triggers the FI over the HMS device with ID __InstanceID__. The __TriggerDelay__ parameter is specified in seconds.

KS300	Combination sensor weather station

FS10 WEATHER

Wireless weather station on 433 MHz

GARDENA SMART SYSTEM

Module for adding GARDENA smart system devices

smart Irrigation Control	A GARDENA irrigation control with 6 valves
smart SILENO Mower	A smart SILENO Mower from GARDENA
smart Power Socket	A power socket from GARDENA
smart Pressure Pump	A GARDENA smart Pump with one valve
smart Sensor	A GARDENA smart sensor with various measured values
smart Water Control	A GARDENA Irrigation Control with one valve

GEOFENCY

Receives data from the Geofency app for Apple iOS.

HEATING CONTROL

Functions for controlling the heating module for regulating room temperatures

HC_TargetValue	Sends via the instance with the ID __ InstanceID__ a target value to the heating module.

HOME CONNECT

Module for adding Home Connect devices

Device list	Device list Home Connect
Home Connect Device	The device instance created by the Home Connect configurator

HOMEMATIC

Device list	HomeMatic Device List
HomeMatic CCU	Functions for controlling HomeMatic components

BidCos Service	BidCos Service Setup
HM_ReadServiceMessages	Reads the service news of the CCU/ LAN adapter with the ID __InstanceID__.
HM_RequestStatus	R
HM_WriteValueBoolean	Sets the __Parameter__ of the device with ID __InstanceID__ to value __Value__.
HM_WriteValueFloat	S
HM_WriteValueInteger	Sets the __Parameter__ of the device with ID __InstanceID__ to value __Value__.
HM_WriteValueString	S

HomeMatic HCU (Beta)

IMAGE GRABBER

Module for querying webcam images

IG_UpdateImage	updates the image of the instance

IPS-868

Functions for controlling IPS868 components

AKM-868	Presence control module including tracker
ANA-868	Analog-to-digital converter

PJ_RequestStatus	Query the status of variables and set them
PJ_SetVoltage	Sets the output to a specific voltage

EKM-868	An energy control module with a 4-channel counter module with 32 bits each

PJ_RequestStatus	Queries the current values for the device with the ID __InstanceID__ and writes it to the appropriate state variables. [
FD-868	A wireless display with a 2x16 character LC display
PJ_Backlight	turns the display backlight on or off
PJ_Beep	outputs an audio signal
PJ_LCDText	sends a text to the display
PJ_SwitchLED	switches an LED of the display
PJ_SwitchMode	switches an FS20 device on/off
Device list	Device list IPS 868
JKM-868 LevelJET	A reader for LevelJET level indicator
PJ_RequestStatus	Query the status of variables and set them
JKM-868 ThermoJET	A reader for individual values of the ThermoJET temperature controller
PJ_RequestStatus	Query the status of variables and set them
LGS-868	Air quality sensor for room air analysis
PJ_RequestStatus	Query the status of variables and set them
PJ_SetLEDs	switches LEDs on the AirQualitySensor
RGBW-868	Control unit for LED RGBW strips
PJ_DimRGBW	Sends a dimming command to the RGBW-868 controller of the ID __InstanceID__.
PJ_RunProgram	Sends a program command to the RGBW-868 controller of the ID __InstanceID__.
PJ_SetRGBW	sets a color value
SERVO-868	A 4-way servo control module
PJ_DimServo	incrementally sets a servo value
PJ_SetServo	sets a servo channel
WDT-868	A watchdog timer for monitoring the system

PJ_SwitchDuration	Switches the relay from the WDT with ID __InstanceID__ to value __Status__ for __Duration__ seconds.
PJ_SwitchMode	Sends a switching command to the WDT-868 module of the ID __InstanceID__.

IR TRANS

Functions for controlling infrared components

Device list	Device list IR-Trans
IRT_SendOnce	sends an IR command

KEBA

Device List

KNX

Functions for controlling KNX components

EIB_Char	Sends a single letter
EIB_Counter8Bit	sends an 8-bit counter value
EIB_Counter16Bit	sends a 16bit counter value
EIB_Counter32bit	sends a 32bit counter value
EIB_Date	sends the date value
EIB_DimControl	Sends a dim value date __Value__ to the device with the ID __InstanceID__. Depending on the configuration 0..15 (Standard) or -7..7 (Enhanced) can be sent.
EIB_DimValue	Sends the dim value __Value__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_DriveBladeValue	Sends the position value __Value__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_DriveMove	Moving the device with ID __InstanceID__ in a particular __Direction__. Depending on configuration the movement direction can be inverted. -
EIB_DriveShutterValue	Sends the position value __Vert__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_DriveStep	Moving the device __Stepwise__ with ID __InstanceID__ into a determined __Direction__. Depending on the configuration, the movement direction can be inverted.
EIB_FloatValue	sends a float value

EIB_Move	moves a roller shutter
EIB_Position	moves a roller shutter to a position
EIB_PriorityControl	sets a device state
EIB_PriorityPosition	Sets the device with ID __InstanceID__ to a certain __Direction__. Depending on configuration the direction can be inverted.
EIB_RequestStatus	sends a read request for EIB instances to the bus
EIB_Scale	Sends the Value __Value__ to the device with the ID __InstanceID__. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.
EIB_SetRGB	sets an RGB stripe to a specific color
EIB_SetRGBW	sets an RGBW stripe to a specific color
EIB_Str	sends a text
EIB_Switch	Switches the device with the ID __InstanceID__ to value __Status__
EIB_Time	sends a time value
EIB_Value	Sends the value __Value__ to the device with ID __InstanceID__.
Device list	Device list EIB/KNX
KNX_RequestStatus	sends a read request for DPT instances on the bus

LCN

Functions for controlling LCN components

Device list	Device list LCN
LCN_AddGroup	adds a device to a group
LCN_AddIntensity	Dims the device with ID __InstanceID__ brighter by the value __Intensity__.
LCN_AddThresholdCurrent	adds threshold to current value
LCN_AddThresholdDefined	adds threshold to predefined value
LCN_Beep	Lets the instance with the ID __InstanceID__ beep
LCN_DeductIntensity	Dims the device with ID __InstanceID__ darker by the value __Intensity__.
LCN_DeductThresholdCurrent	deducts threshold from current value
LCN_DeductThresholdDefined	deducts threshold from predefined value
LCN_FadeOut	Dims the devive with ID __InstanceID__ to value __Intensity__ and dims then with the __Ramp__ seconds down to 0%.
LCN_FlipRelay	Switches over the relay with the ID __InstanceID__
LCN_LimitOutput	limits an output for a certain period of time
LCN_LoadScene	calls up a scene for an output
LCN_LockTargetValue	locks a controller of the device
LCN_RampStop	Stops the dimming process of the device with the ID __InstanceID__
LCN_ReleaseTargetValue	unlocks a controller of the device

LCN_RemoveGroup	removes a device from a group
LCN_RequestLights	queries the LEDs of the panel
LCN_RequestRead	queries the values
LCN_RequestStatus	queries all statuses of the module
LCN_RequestThresholds	queries thresholds and hysteresis
LCN_SaveScene	saves a scene for an output
LCN_SelectSceneRegister	selects a register
LCN_SendCommand	Sends a custom command to the instance with the ID __InstanceID__. For possible commands, refer to the PCHK documentation. If you do not have this, you can possibly refer to your dealer. We do not provide this option.
LCN_SetDisplayText	Shows a text on a line of the display
LCN_SetDisplayTime	sets the display time of a line in the display
LCN_SetIntensity	Dims the device with the ID __InstanceID__ to value __Intensity__ in __Ramp__ seconds.
LCN_SetLamp	switching an LED
LCN_SetRelay	sets all relays in a module (8 bit)
LCN_SetRGBW	sets an instance to an RGBW color value
LCN_SetTargetValue	Sets the regulator __Target__ of the instance with the ID __InstanceID__ to the value __Value__
LCN_ShiftTargetValue	Shifts the current target value
LCN_ShutterMove	starts shutter movement to percentage position
LCN_ShutterMoveDown	starts a shutter movement downwards
LCN_ShutterMoveUp	starts a shutter movement upwards
LCN_ShutterStop	stops a shutter movement
LCN_StartFlicker	Let the device with the ID __InstanceID__ flicker
LCN_StopFlicker	Stop flickering of the device with the ID __InstanceID__ immediately
LCN_SwitchDurationMin	after how many minutes it should be dimmed down
LCN_SwitchDurationSec	after how many seconds it should be dimmed down
LCN_SwitchMemory	Switches the device with the ID __InstanceID__to a stored value
LCN_SwitchMode	switches a device on/off
LCN_SwitchRelay	Switches the relay with the ID __InstanceID__ to value __Status__
LCN_SwitchRelayTimer	switches a relay on a time factor

LJQUICK

The KNX quick (Lingg&Janke) module supports the setup of KNX instances.

LJ_SendDateTime	Sends the current date/time to the KNX bus

MATTER

Note on Using Matter with Docker

Device list

M-BUS

Functions for controlling M-Bus components

Device list	Device list M-Bus
MBUS_UpdateValues	updates all device-specific values

MENNEKES

Mennekes device setup

Device List

MODBUS RTU/TCP

Functions for controlling ModBus RTU/TCP components

Device list	Device list ModBus RTU/TCP
ModBus_RequestRead	Executes a new read for the device with the ID __InstanceID__.
ModBus_WriteCoil	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegister	writes a value to the write address
ModBus_WriteRegisterByte	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterChar	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterDWord	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterInt64	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterInteger	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterReal	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterReal64	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterShort	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterString	Sets address with ID __InstanceID__ to __Value__
ModBus_WriteRegisterWord	Sets address with ID __InstanceID__ to __Value__
Templates (file format)	Structure and file format of ModBus Device templates

MÖHLENHOFF ALPHA 2

Receiving and switching Möhlenhoff Alpha2 data

MA2_RequestStatus	Gets the values stored in the Alpha2 with the InstanceID and sets the associated variables.
MA2_WriteValue	Writes the Value into the variable with the Ident in the Möhlenhoff Alpha 2 with the InstanceID.

MQTT

Functions for controlling MQTT components

Device list	Device list MQTT
MQTT Client	Functions for controlling MQTT Client
MQTT-Server	Functions for controlling MQTT Server

NEA SMART

Receiving and switching NEA Smart data

NEAS_RequestStatus	Gets the values stored in the NEA Smart with the InstanceID and sets the associated variables.
NEAS_WriteValue	Writes the Value to the variable with the ident in the NEA Smart with the InstanceID.

OCPP

The OCPP protocol allows wallboxes to be conveniently monitored and switched via Symcon.

OCPP_RemoteStartTransaction	If the status is 'Preparing' or 'Finishing', loading can be enabled by this command.
OCPP_RemoteStopCurrentTransaction	If the state is 'Charging', this command can be used to stop charging.
OCPP_RemoteStopTransaction	If the state is 'Charging', this command can be used to stop charging.

OPC UA

Functions for controlling OPC UA components

SAGEGLASS (BACNET)

Integrates the SageGlass SIM II (BACnet) with IP-Symcon

SHUTTER CONTROL

Functions for controlling shutter (roller shutter) components

SC_Move	moves the shutter to a specific position
SC_MoveDown	moves the shutter down to the end position
SC_MoveUp	moves the shutter up to the end position
SC_Stop	stops a movement

SIEMENS OZW

Functions for controlling Siemens OZW components

Device list	Device list Siemens OZW
OZW_RequestStatus	requests the status of a device
OZW_WriteDataPoint	writes a specific value to a data point

SNMP

Show values for any OIDs of a walk. Optionally, these can be created and described as variables.

SNMP_UpdateValues	Updates the variable values which are below the instance.

SNOM

Integration of Snom IP phones

PLC: SIEMENS, VIPA, LOGO

Functions for controlling Siemens/Vipa/Logo PLC components

Device list	Device list SPS Siemens/Vipa
Logo-VM-Addresses	VM addresses of Logo7/8
S7_RequestRead	performs a read operation on a device
S7_Write	writes a value to the configured address
S7_WriteBit	Set the address with the ID __InstanceID__ to value status
S7_WriteByte	Set the address with the ID __InstanceID__ to __Value__
S7_WriteChar	Set the address with the ID __InstanceID__ to __Value__
S7_WriteDWord	Set the address with the ID __InstanceID__ to __Value__
S7_WriteInteger	Set the address with the ID __InstanceID__ to __Value__
S7_WriteReal	Set the address with the ID __InstanceID__ to __Value__
S7_WriteShort	Set the address with the ID __InstanceID__ to __Value__
S7_WriteWord	Set the address with the ID __InstanceID__ to __Value__

PLC: WAGO, BECKHOFF, ABB

Functions for controlling Wago/Beckhoff PLC components

SYNC REMOTE

Synchronization between two Symcon systems

TECHNISCHE ALTERNATIVE

Functions for controlling Technische Alternative components

Device list	Device list Technische Alternative

DOOR INTERCOM

Device list	
2N	
DoorBird	
FANVIL	
FRITZ!Box	
DoorIP_Accept	accepts the current call.
DoorIP_Decline	rejects the current call.
DoorIP_SendDTMF	sends DTMF characters on the current connection.

VOICE OVER IP

Functions for controlling VoIP

Device list	Device list Voice over IP
VoIP_AcceptCall	accepts a connection
VoIP_Connect	establishes a connection
VoIP_Disconnect	terminates a connection
VoIP_GetConnection	returns information about a connection
VoIP_GetData	returns data of a connection
VoIP_PlayWave	plays a wave file
VoIP_RejectCall	rejects a connection
VoIP_SendDTMF	sends a sequence of characters as DTMF
VoIP_SetData	sets the data of a connection

WEISHAUPT

Installation of Weishaupt devices

Device List	

WINLIRC

Functions for controlling WinLIRC components

WinLIRC_SendOnce	Sends a remote command on the instance with ID __InstanceID__. The values remote control and button specify, which command should be sent. These must be set up accordingly in the client software (WinLIRC/ IRTrans Tray).

WIRELESS M-BUS

Functions for controlling wireless M-Bus components

Device list	Wireless M-Bus device list

WMRS200

Functions for controlling WMRS200 components

Device list	Device list WMRS200

W&T

Functions for controlling W&T components

WUT_SwitchMode	Switches the channel with ID __InstanceID__ to value __Status__.

XBEE

Functions for controlling XBee

XBee_SendBuffer	sends a data string to a specific device
XBee_SendCommand	sends a command to a specific XBee-Splitter

EATON XCOMFORT

Functions for controlling xComfort components

Device list	Device list Eaton xComfort
MXC_DimBrighter	starts dimming to a brighter level
MXC_DimDarker	starts dimming to a darker level
MXC_DimSet	dims an xComfort device to a specific level
MXC_DimStop	stops a dimming process
MXC_RequestStatus	sends a status request to a device
MXC_SendBoolean	Sends a boolean value to the instance with ID __InstanceID__, which is connected to a room manager.

MXC_SendFloat	Sends a float to the instance with ID __InstanceID__, which is connected to a room manager.
MXC_SendInteger	Sends a boolean value to the instance with ID __InstanceID__, which is connected to a room manager.
MXC_SetTemperature	Sets the new target temperature of the control valve with the ID __InstanceID__ to the new __Temperature__.
MXC_ShutterMoveDown	moves the shutter down to the end position/stop
MXC_ShutterMoveUp	moves the shutter up to the end position/stop
MXC_ShutterStepDown	moves the shutter down a step
MXC_ShutterStepUp	moves the shutter up a step
MXC_ShutterStop	stops a movement
MXC_SwitchMode	switches an xComfort device on/off

Z-WAVE

Functions for controlling Z-Wave components

Device list	Device list Z-Wave
ZW_Basic	Turns the basic function of the specified device with InstanceID __InstanceID__
ZW_ColorCW	sets the cold-white channel of a RGBWW Z-Wave device
ZW_ColorRGB	sets the color of a Z-Wave device
ZW_ColorRGBWW	sets the colors and intensity of an RGBWW Z-Wave device
ZW_ColorWW	sets the WarmWhite channel of a Z-Wave device
ZW_DimDown	starts dimming down
ZW_DimDownEx	starts dimming down with a runtime
ZW_DimSet	dims a Z-Wave device to a specific level
ZW_DimSetEx	dims a Z-Wave device to a specific level within a runtime
ZW_DimStop	stops dimming a Z-Wave device
ZW_DimUp	starts dimming up
ZW_DimUpEx	starts dimming up within a runtime
ZW_DoorLockOperation	Sets the door lock operation mode of a Z-Wave device
ZW_LockMode	sets the LockMode
ZW_MeterReset	resets the meter
ZW_Optimize	Starts the wireless network optimization process for a Z-Wave device
ZW_ProtectionSet	sets the Protection and its Mode
ZW_RequestStatus	Queries the status from the device with the ID __InstanceID__. Only power-operated devices can be queried directly by command, because battery-operated devices do not permanently listen to the queries.
ZW_ShutterMoveDown	Moves the shutter with the ID __InstanceID__ downwards until (link: page://f1hDSVmdoBuVvjEW text: MXC_ShutterStop) is running or the end position is reached.

ZW_ShutterMoveUp	Moves the shutter with the ID __InstanceID__ upwards until (link: page://f1hDSVmdoBuVvjEW text: MXC_ShutterStop) is running or the end position is reached.
ZW_ShutterStop	Stops a current movement of the shutter.
ZW_SwitchAllMode	Sets the Switch All mode of a Z-Wave device
ZW_SwitchMode	switches a Z-Wave device on/off
ZW_Test	Tests if a connection to a Z-Wave device is working
ZW_ThermostatFanModeSet	Sets the FanMode for the thermostat
ZW_ThermostatModeSet	Sets the thermostat mode
ZW_ThermostatSetPointSet	Sets the PointSet Value

ZEVVY

Sends data to a Zevvy account

ZY_SendMeasurements	Sends the data of the set up variables to a Zevvy account

1-Wire

The 1-Wire bus system is similar to I2C. In a simple way different sensors can be connected to a normal 3-wire cable (ground, data, and +5 V). For connection to a PC, a USB adapter (DS9490R) or serial interface (DS9097U-S09) are available.

The following devices are supported by IP-Symcon:
Supported Devices

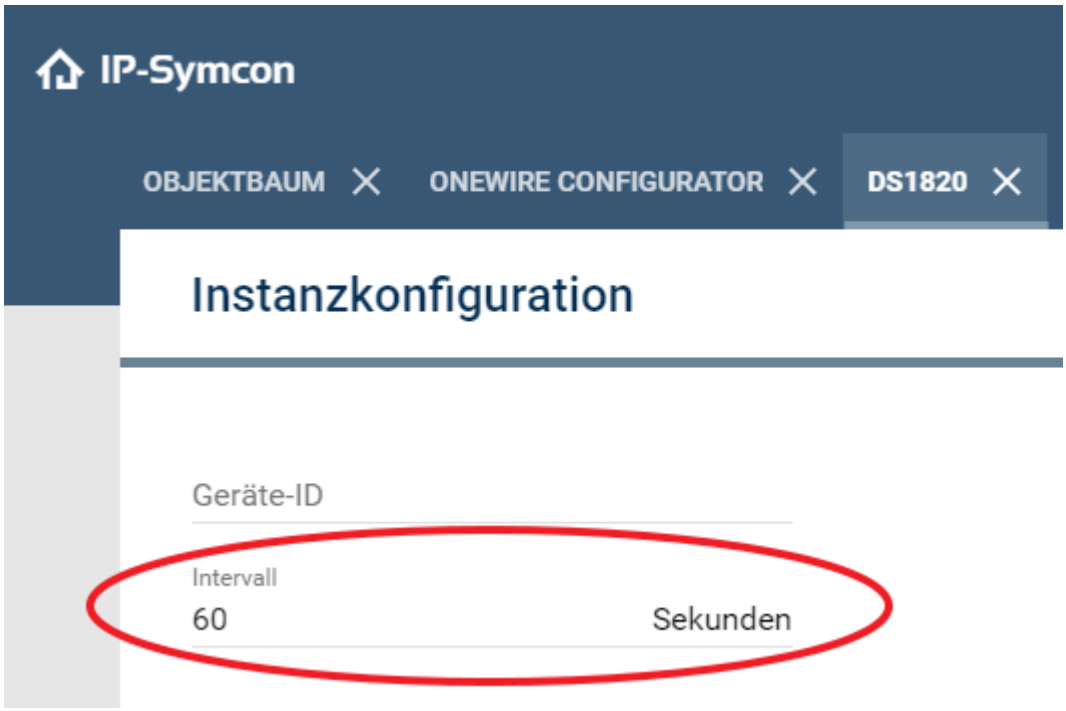
Since version 4.0 **no** TMEX driver is required for usage, except for DS2490/DS9490R

IP-Symcon does not support the **parasitic** mode.
A **+5V** power supply is essential!

VIDEO TUTORIAL FOR CONFIGURATION (GERMAN)

INTEGRATION INTO IP-SYMCON

When using the LAN gateway, it can be integrated via the Device Search. As a system, "OneWire Discovery" must be selected. The Discovery instance offers the creation of a OneWire Configurator. After the configurator is created, it can be used to create the individual devices as instances via "Create".
The building blocks that are connected to the 1-Wire bus do not send their values automatically - they must be requested. The **Interval** in the respective instance configuration defines how often the request is done.



Every request takes time and occupies the bus. For this reason, the "Interval" should be used sparingly. In practice, for example, a request cycle of 60 seconds to read a room temperature is more than sufficient. Smaller values under one second cannot be configured for that reason.

Furthermore, it should be considered that the bus workload rises with more devices connected to an adapter. In our model home, 30 devices are working without issue on one adapter. Wie recommend an individual power supply and multiple adapters for bigger installations.

TIPS & TRICKS

The serial adapter has a different pinout than the USB adapter and does not provide +5 V bus voltage.
For power supply a regulated 5V power supply with current limitation is recommended.
For smaller systems, we recommend using a modular cable with RJ12 or RJ45 plug for creating cable itself and distributing a 3 Series modular socket. The system is then wired 1:1 (star-shaped junctions).

Further links:
Wikipedia: en.wikipedia.org/wiki/1-Wire
Manufacturer: [https://www.maximintegrated.com/en/pl_list.cfm/filter/21]
Read the COM Port of a USB Device in Windows (German)

DS2405	1-channel switch

OW_SwitchMode	Sets the state of the device with the ID __InstanceID__ to value __Status__.
OW_ToggleMode	Changes the state of the device with the ID __InstanceID__. The current value can be queried in the state variable after the operation. -

DS2406 2-channel switch

OW_SetPin	Sets the state of the device with the ID __InstanceID__ to value __Status__

DS2408 8-channel switch

OW_SetPin	sets a pin from the DS2408 to on/ off
OW_SetPort	Sets the state of all eight pins on the value of __Bitmask__ of the device with ID __InstanceID__
OW_SetStrobe	Makes it possible to put the RSTZ-pin of DS2408 of the device with ID __InstanzID__ to value __Status__
OW_WriteBytes	Sets a sequence of bytes (__Data__) from all pins of the device with the ID __InstanceID__.
OW_WriteBytesMasked	Sets a sequence of bytes (__Data__) taking into account a bitmask __Mask__ from all pins of the device with the ID __InstanceID__.

DS2413 2-channel switch

OW_SetPin	Sets the state of the pin __Pin__ of the device with the ID __InstanceID__ to value __Status__

DS2438 Smart Battery Monitor

DS2450 A/D converter 16 bit

OW_SetPin	Sets the state of the pin __Pin__ of the device with the ID __InstanceID__ to value __Status__

DS2890 Digital potentiometer 8 bit

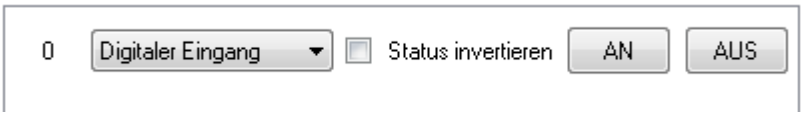
OW_SetPosition	Sets the device with the ID __InstanceID__ to value __Value__

Device List Device list 1-Wire

OW_RequestStatus Executes a new read for the device with the ID __InstanceID__.

DS2405

The DS2413 is a 1-port switch (4mA) & input (PIO) module.
Unfortunately it is no longer manufactured. As a replacement, we recommend the type DS2413.
Alternatively, you can set each of the eight channels as "Digital Output" or "Digital Input".
Additionally you can "invert the status" (see DS2413).
For testing purposes, you have the ability to switch the output "ON" and "OFF".



OW_SwitchMode

```
bool OW_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the state of the device with the ID **InstanceID** to value **Status**.



Since the module does not support defined settings, the command operates on the principle of reading the input and for the case the state <> input state, to toggle the output of the module. When designing external circuits that should not be overlooked.

EXAMPLE

```
OW_SwitchMode(12345, true); //Turn on device
```

OW_ToggleMode

```
bool OW_ToggleMode (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Changes the state of the device with the ID **InstanceID**. The current value can be queried in the state variable after the operation.

EXAMPLE

```
OW_ToggleMode(12345); //Toggle device
```

DS2406

The DS2406 is a 2-way switch (50 / 8mA), input (PIO) module & 1Kb memory module.

OW_SetPin

```
bool OW_SetPin (int $InstanceID, int $Pin, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Pin	Pin of the module (0..1)
Status	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the state of the device with the ID **InstanceID** to value **Status**

EXAMPLE

```
OW_SetPin(12345, 0, true); //Turn on device
```

DS2408

The DS2408 and DS2408+ is a 8-port switch & input (PIO) module.
Alternatively, you can set each of the eight channels as "Digital Output" or "Digital Input".
Additionally you can "invert the status" (see DS2413).
For testing purposes, you have the ability to switch the output "ON" and "OFF".



OW_SetPin

```
bool OW_SetPin (int $InstanceID, int $Pin, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Pin	0-7
Status	TRUE for on, FALSE for off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sets the status of the pin Pin of the device with the ID InstanceID to the value Status

EXAMPLE

```
OW_SetPin(12345, 2, true); //Turn on pin 2 of the device
```

OW_SetPort

```
bool OW_SetPort (int $InstanceID, int $Bitmaske)
```

PARAMETERS

InstanceID	ID of the device to be switched
Bitmaske	0-255 (Bit0 = Pin0 … Bit7 = Pin7)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the state of all eight pins on the value of **Bitmask** of the device with ID **InstanceID**

EXAMPLE

```
OW_SetPort(12345, 0); //Turn everything off
```

OW_SetStrobe

```
bool OW_SetStrobe (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Makes it possible to put the RSTZ-pin of DS2408 of the device with ID **InstanzID** to value **Status**

EXAMPLE

```
OW_SetStrobe(12345, true); //Turn on strobe
```

OW_WriteBytes

```
bool OW_WriteBytes (int $InstanceID, string $Data)
```

PARAMETERS

InstanceID	ID of the device to be switched
Data	Sequence of bytes to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets a sequence of bytes (**Data**) from all pins of the device with the ID **InstanceID**.



Only pins defined as the output are described!

EXAMPLE

```
OW_WriteBytes(12345, Chr(0).Chr(1)); //Send sequence
```

OW_WriteBytesMasked

```
bool OW_WriteBytesMasked (int $InstanceID, string $Data, int $Mask)
```

PARAMETERS

InstanceID	ID of the device to be switched
Data	Sequence of bytes to be sent
Mask	Bitmask of the bit to be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets a sequence of bytes (**Data**) taking into account a bitmask **Mask** from all pins of the device with the ID **InstanceID**.



Only pins defined as the output are described!

EXAMPLE

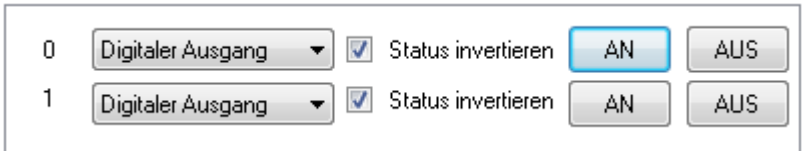
```
OW_WriteBytesMasked(12345, Chr(255), Chr(1)); //Set only Bit 0
```

DS2413

The DS2413 is a 2-port switch & input (PIO) module.
Its outputs can deliver 20mA. For further details check out the data sheet.

Definition of “Status pins”:
A logic low level at the output is rated as "Off"/ "False" (the internal transistor is active).

If e.g. an LED is connected with a common anode, it lights at this "Status pin = Off".
In this case "Invert status" should be activated.
2



If a pin is used as a "digital input", make sure that the DS205 has no latch. Thus, no switching pulses are stored.

To e.g. evaluate a PIRI (PIR), the timer should be set for ten seconds. Additionally provide at the input a 1M resistor and a capacitor 10uF. Both are commonly connected to 0V. A switching pulse pulls the input short to +5 V. The capacitor buffers the voltage, until the next polling cycle recognizes the pin as HI.

OW_SetPin

```
bool OW_SetPin (int $InstanceID, int $Pin, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Pin	0-1
Status	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

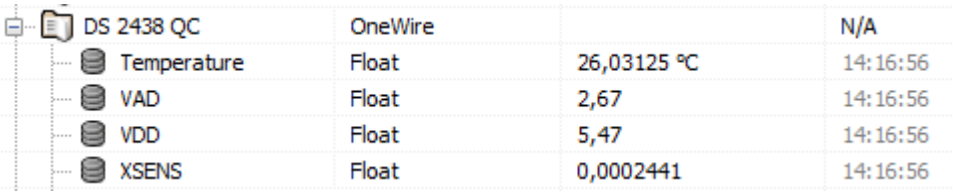
Sets the state of the pin **Pin** of the device with the ID **InstanceID** to value **Status**

EXAMPLE

```
OW_SetPin(12345, 0, true); //Turn on Pin 0
```

DS2438

The DS2438 is originally a "Smart Battery Monitor".
For use in the home automation, he has four interesting features:
A voltmeter, this provides the bus voltage - ideal for monitoring (VDD)
A thermometer - can always be useful (Temperature)
An AD converter (VAD) - can be used for measuring the moisture or brightness
A current input (Xsens) - at leisure



DS 2438 QC	OneWire		N/A
Temperature	Float	26,03125 °C	14:16:56
VAD	Float	2,67	14:16:56
VDD	Float	5,47	14:16:56
XSENS	Float	0,0002441	14:16:56

Practical example: moisture measurement
The humidity sensor **HIH-4000** from HONEYWELL has an accuracy of 3.5%, ranging from 0 to 100% RH.
Because of its linear output the DS2438 can be read out directly. A temperature compensation can be taken into account because the DS2438 has an internal temperature sensor.
As it goes, illustrates the following script:

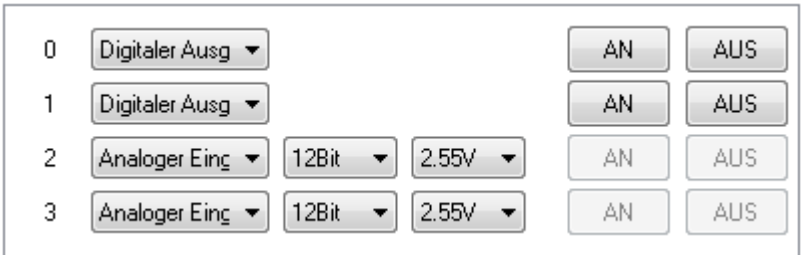
```
$Vad = GetValue(44045);
$Vdd = GetValue(14570);
$temp = GetValue(18691);
$Srh = ($Vad - 0.958062) * 30.680;
$Srh = $Srh / ((1.0305 + (0.000044 * $temp) - (0.0000011 *
pow($temp,2))));
echo "Humidity: $Srh %rh Temp.Comp.\n";
SetValue(49935 /*[Humidity %rh]*/, $Srh);
```

DS2450

The DS2450 is an A/D converter module with a resolution of up to 16-bit.
Additionally, the four channels can also be used as an output (4 mA).
Further details check out on the data sheet.

CONFIGURATION

Alternatively, you can select any of the four channels as a "Digital Output" and "Analog Input".
In addition, two voltage ranges are available: 2.55 V and 5.1 V.
Furthermore, the resolution between 1-bit and 16-bit is adjustable:



OW_SetPin

```
bool OW_SetPin (int $InstanceID, int $Pin, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Pin	0-3 (A-D)
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sets the state of the pin Pin of the device with the ID InstanceID to value Status

EXAMPLE

```
OW_SetPin(12345, 2, true); //Turn on Pin C of the device
```

DS2890

The DS2890 is a digital potentiometer with 8-bit resolution.
Unfortunately he is no longer manufactured. No replacement type exists.



"CHARGE PUMP CONSIDERATIONS" should be used on the 1-Wire Center 0 to 10V output. To do this, enable "Use CPC."
For a test the slider can be changed and the "SET" button can be pushed.

OW_SetPosition

```
bool OW_SetPosition (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	0-255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the device with the ID **InstanceID** to value **Value**



Please note the CPC parameter in the configuration.

EXAMPLE

```
OW_SetPosition(12345, 123); //Set the device on value 123
```

Device List

SUPPORTED GATEWAYS

GATEWAY	DESCRIPTION
Symcon 1-Wire LAN Gateway Product information | Order now	IP Gateway
DS2480B	serial interface
DS2490 (Windows only)	USB adapter
DS9097U	serial interface
DS9490R (Windows only)	USB adapter
Link45	serial interface
LinkUSB	USB adapter

SUPPORTED COMPONENTS

PRODUCT	DESCRIPTION
DS1820	Thermometer -55°C to +125°C, 0.5 exact at -10°C to +85°C
DS18B20	Thermometer -55°C to +125°C, 0.5 exact at -10°C to +85°C
DS18S20	Thermometer -55°C to +125°C, 0.5 exact at -10°C to +85°C
DS1920	Thermometer -55°C to +100°C, 0.5
DS2405	1-way: Switch & Input (PIO)
DS2406	2-way: Switch & Input (PIO)
DS2407	2-way: Switch & Input (PIO)
DS2408	8-way: Switch & Input (PIO)
DS2408+	8-way: Switch & Input (PIO)
DS2413	2-way: Switch & Input (PIO)
DS2423P	4KB counter
DS2438	Current and voltage measurement
DS2450	A/D converter
DS2890	Digital potentiometer

OW_RequestStatus

```
bool OW_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Executes a new read for the device with the ID InstanceID.
Once the function has been performed, the new values are available in the state variables.

EXAMPLE

```
OW_RequestStatus(12345); //Read out the device
```

ABL

ABL offers various wallboxes/charging stations. These can be read out via ModBus TCP. A connection with IP-Symcon is possible via a LAN-IP gateway.

The following devices are supported by IP-Symcon:
Supported components

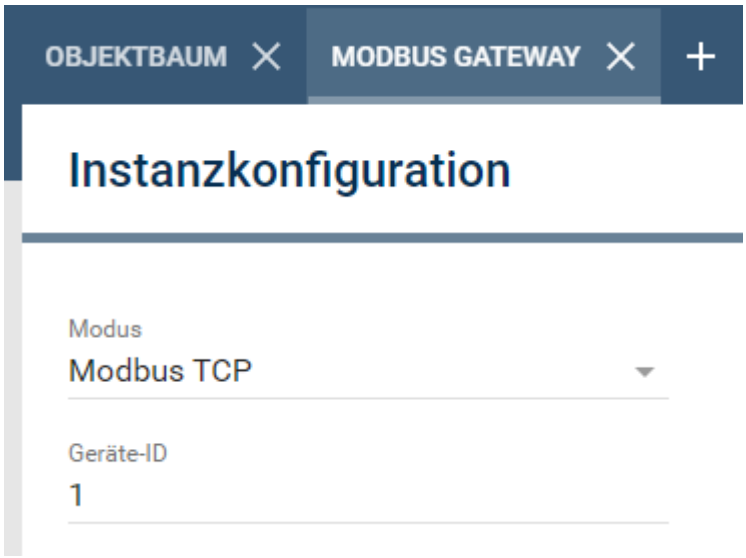
INSTALLATION

To use ABL wallboxes/charging stations in IP-Symcon, a connection via Ethernet must be available. Inside the wallbox/charging station only the Ethernet RJ45 plug has to be plugged in according to the instructions.

INTEGRATION IP-SYMCON

First a "ModBus Device" instance must be added within the object tree of IP-Symcon. In the following dialog the IP address of the wallbox has to be entered. The port is 502 by default. Both information can be viewed on the web interface of the wallbox/charging station.

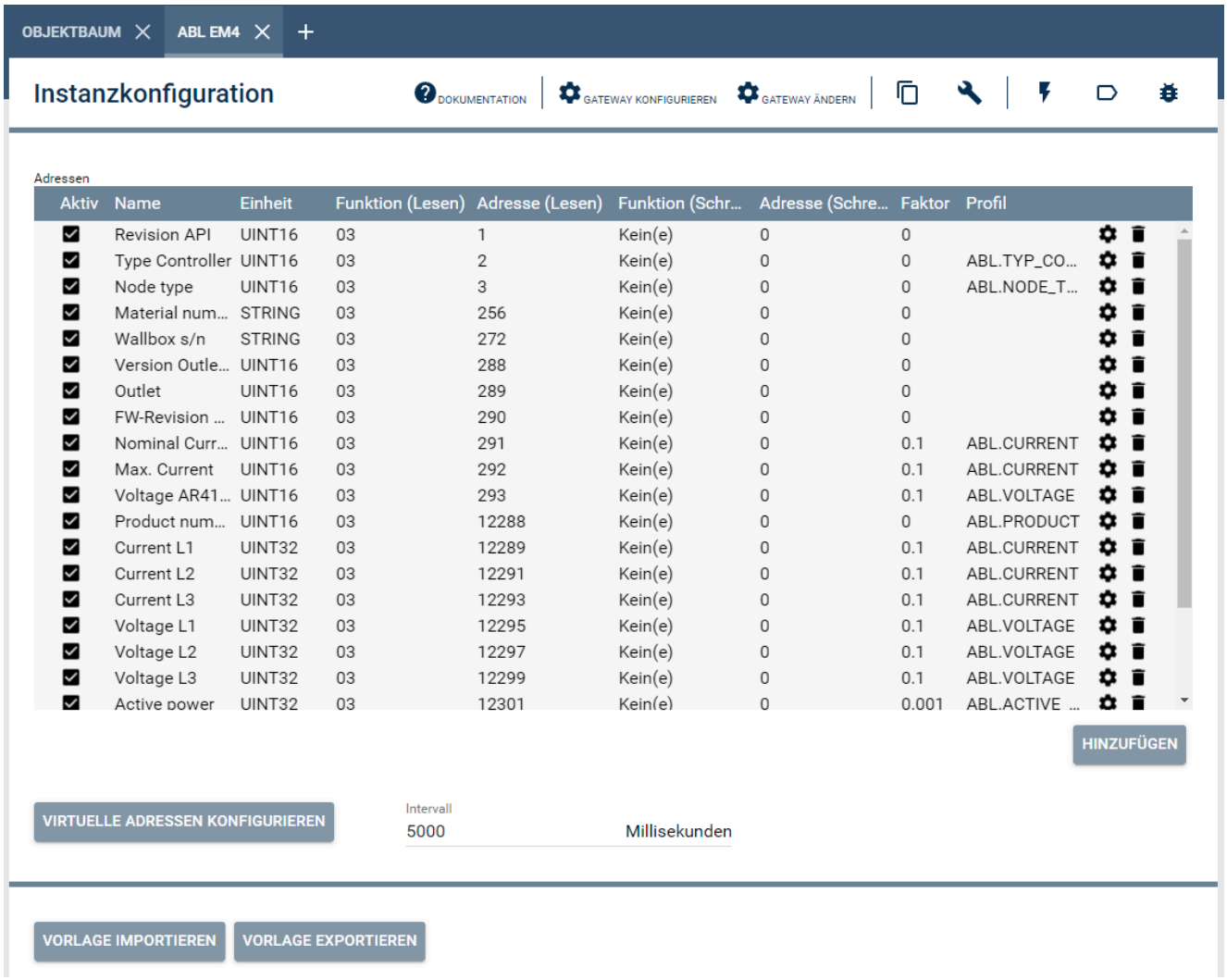




Afterwards the ModBus template for ABL wallboxes/charging stations can be downloaded. This contains the whole configuration of the ModBus device. After downloading, the "ABLmodbusX_vx.json" can be loaded via "Import". Afterwards all ModBus addresses of the common ABL wallboxes/charging stations are set up.

ModBus template for ABL wallboxes/charging columns:
Download eMH1
Download eM4
This includes the outlet/counter 1.
If more are needed the Support can be contacted

eMH2 and eMH3 can be used via OCPP.



ADD ADDRESSES

If further addresses are to be added, this can be realized via "Add".
Depending on the design of the wallbox, individual addresses may have to be de-/activated. This can be controlled via the Active column.

Device-List

Require: Symcon >= 7.0

SUPPORTED COMPONENTS

SYSTEM	DESCRIPTION
Wallbox eMH1	Supported in all versions
Wallbox eMH2	Supported in all versions
Wallbox eMH3	Supported in all versions
Wallbox eM4	Supported in all versions
Wallbox eMC	Supported in all versions

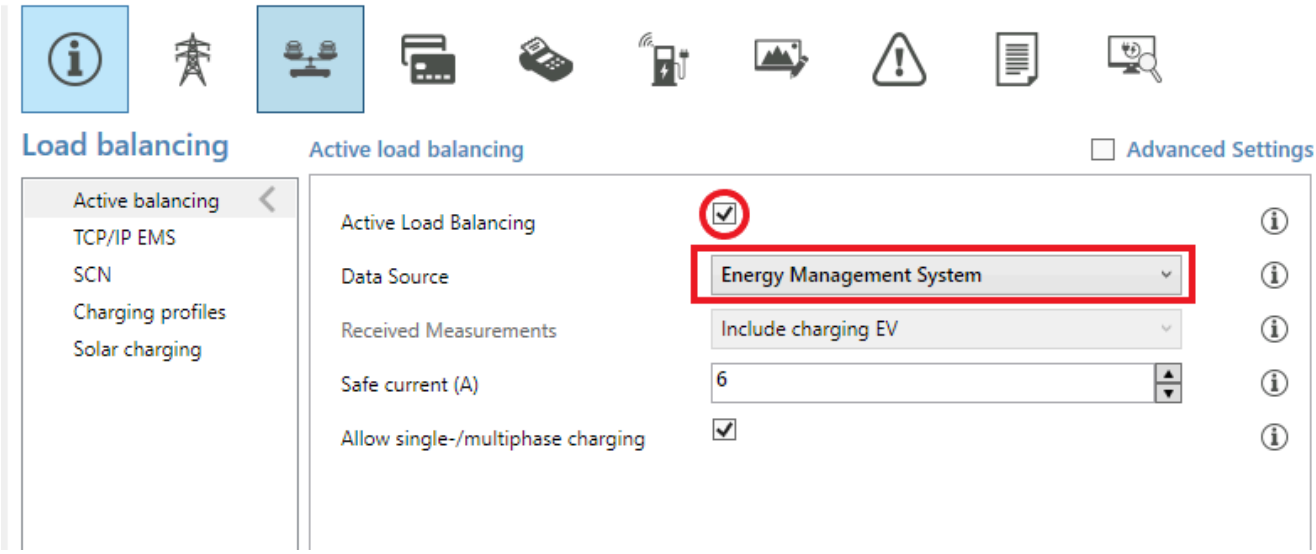
Alfen

Require: Symcon >= 7.0
Alfen offers various wallboxes/charging stations. These can be read out via ModBus TCP. A connection with IP-Symcon is possible via a LAN-IP gateway.

The following devices are supported by IP-Symcon:
Supported components

INSTALLATION

To use Alfen wallboxes/charging columns in IP-Symcon, a connection via Ethernet to the wallbox must be available. A LAN connection is integrated within the wallbox/charging station.

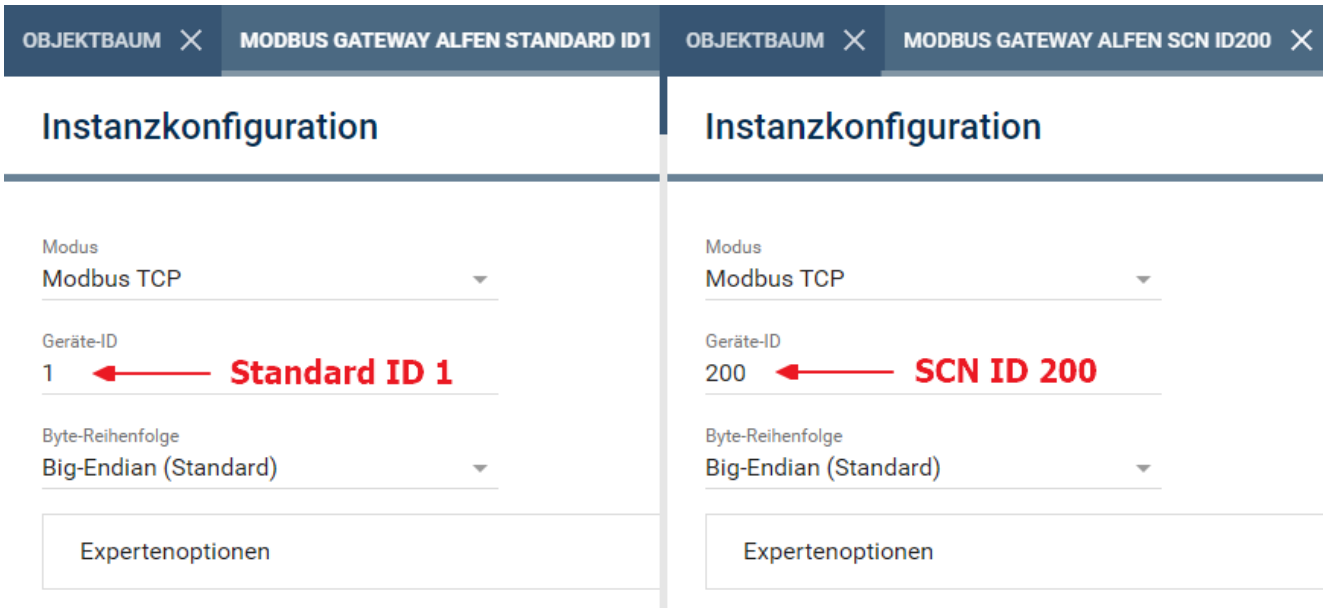


Within the ACE Service Software, as visible in the image, Active Load Balancing must be activated and "Energy Managment System" must be selected. This will also activate the ModBus TCP.

INTEGRATION IP-SYMCON

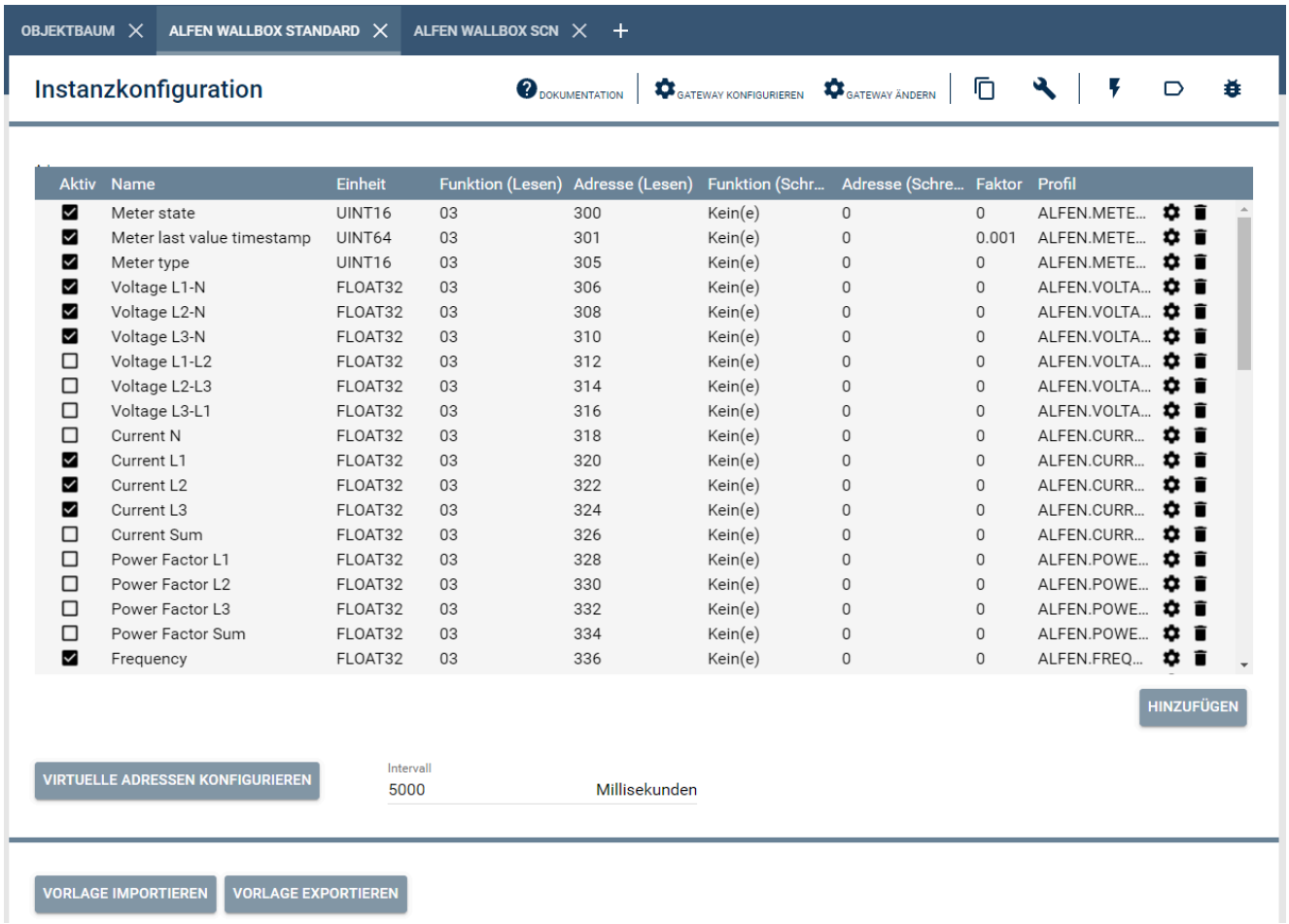
First a "ModBus Device" instance must be added within the object tree of IP-Symcon. In the following dialog the IP address of the wallbox has to be entered. The port is 502 by default. Both information can be seen in the app or ACE service software of the wallbox/charging station.

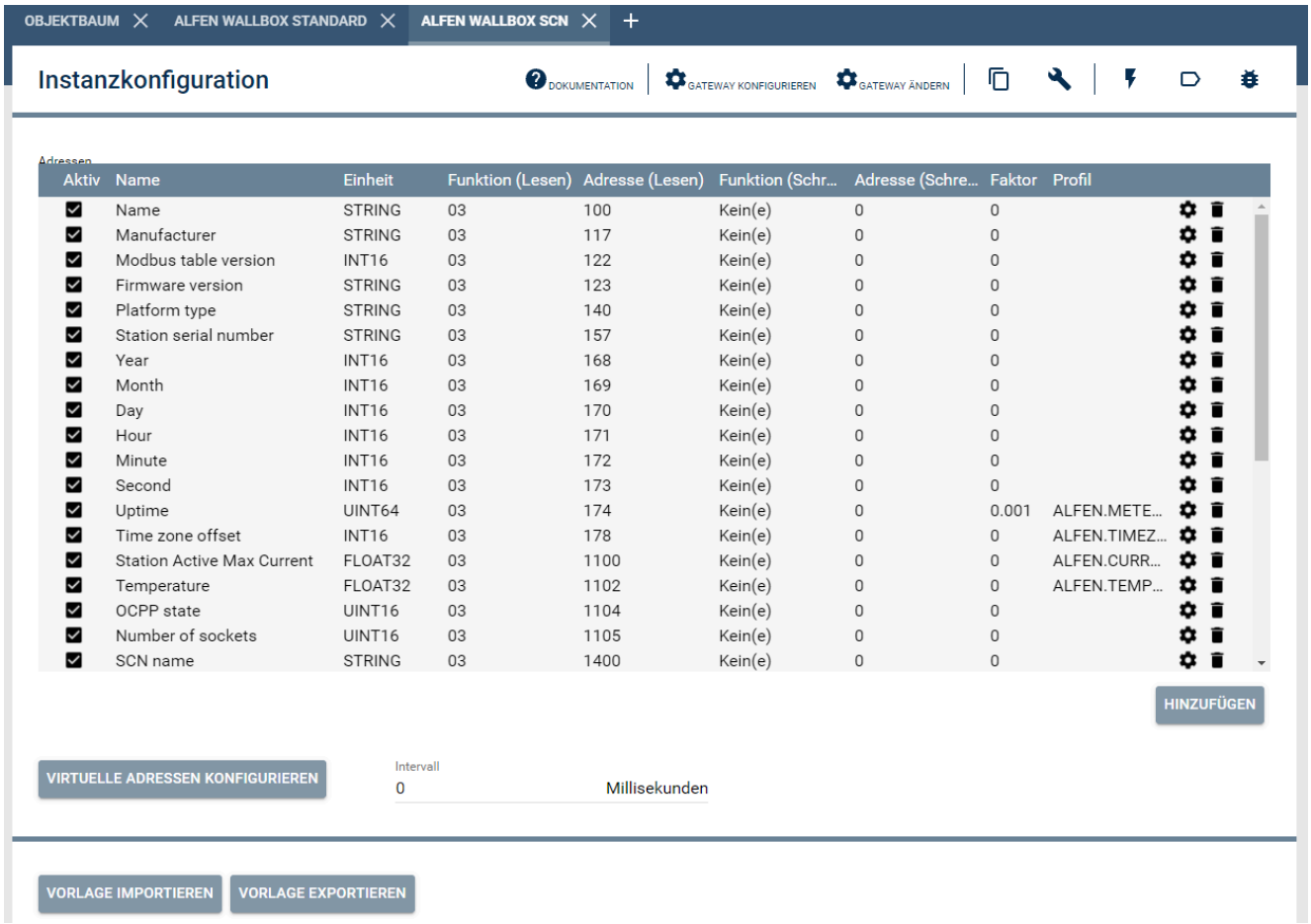




Afterwards you can download the ModBus template ("AlfenModBusX_vx.json") for Alfen wallboxes/charging stations. This download contains the standard or the SCN configuration of the ModBus device. Via "Import" the desired template can be loaded. Afterwards, all ModBus addresses of the common Alfen wallboxes/charging columns are set up.

ModBus template for Alfen wallboxes/charging columns:
Download Standard
Download SCN



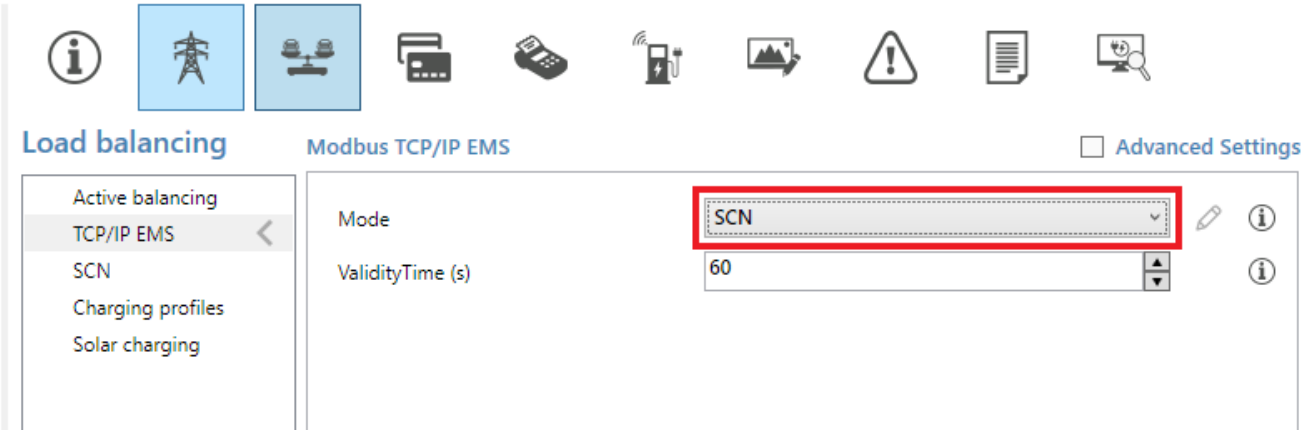


ADD ADDRESSES

If more addresses are to be added, this can be done via "Add".
Depending on the design of the wallbox, individual addresses may have to be de-/activated. This can be controlled via the Active column.

USE STANDARD OR SCN

In the ACE Service Software you can choose between a standard load management (socket) or SCN operation, as shown in the picture. SCN maps several wallboxes, whereas Socket supports only one wallbox.



PROTOCOL DESCRIPTION

protocol description for Alfen wallboxes/charging stations:
download

Device List

Require: Symcon >= 7.0

SUPPORTED COMPONENTS

SYSTEM	DESCRIPTION
Eve Single S-line	Supported in all versions
Eve Single Pro-line	Supported in all versions
Eve Single Pro-line DE	Supported in all versions
Eve Double Pro-line	Supported in all versions
Eve Double Pro-line DE	Supported in all versions
Eve Double PG-line DE	Supported in all versions
Eve Twin 4/5XL	Supported in all versions

ALLNET

IP-Symcon supports the "tax and measuring computer ALL4000" - for it, numerous sensors and actuators are available. The ALL4000 is connected to the network via its 100Mbit Ethernet interface, and provides a web server on which are displayed the measurements graphically and numerically.
All products have their own web server for configuration.
Manufacturers and other product information as PDF: allnet.de
List of supported devices:

ALL5000
ALL4500
ALL4000
ALL4001
ALL3000
ALL3075 > Network socket for switching over LAN
ALL3076 > Network socket for dimming over LAN
ALL3270 > Narrowband Powerline Master Ethernet Bridge/ requirement for:
ALL3275 > Narrowband Powerline slave socket for switching over Powerline
ALL3090 / All 3096 > 8 Port Reset Switch
ALL4100 > 8 Port Power Switch 19″ with 8 Heating appliance
ALL3000RF MK2 > Control unit for Intertechno radio sockets
ALL3690 > PowerMeter (3 Phases)
ALL3691 > PowerMeter (2*3 Phases, 2*S0, 2*D0)

ALL_SetAnalog

Require: Symcon >= 5.0

```
bool ALL_SetAnalog (int $InstanceID, int $ChannelID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
ChannelID	ID of the channel to be switched
Value	Value to which the channel should be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the value of the channel with the ChannelID **Channel** of the device with the ID **InstanceID** to the value **Value**



The ChannelID can be seen from the status variables of the instance. See find out ChannelID

EXAMPLE

```
// Sets channel 3 of device 12345 to 20.4
ALL_SetAnalog(12345, 3, 20.4); //Turn device on
```

ALL_SwitchActor

```
bool ALL_SwitchActor (int $InstanceID, int $ChannelID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
ChannelID	ID of the channel to be switched
Status	TRUE for on, FALSE for off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the actuator with the DeviceID **InstanceID** on the channel **ChannelID** to the value **Status**



The ChannelID can be seen from the status variables of the instance. See find out ChannelID

EXAMPLE

```
ALL_SwitchActor(12345, 1, true); //Switch on the actuator of device
12345 on channel 1
```

ALL_SwitchMode

```
bool ALL_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with the ID **InstanceID** to value __Status___

EXAMPLE

```
ALL_SwitchMode(12345, true); //Turn on device
```

ALL_UpdateValues

```
bool ALL_UpdateValues (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be updated

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Executes a update of the state variables for the device with the ID **InstanceID**.
Once the function has been performed, the new values are available in the state variables.

EXAMPLE

```
//Update Device with the ID "12345"
ALL_UpdateValues(12345);
```

Device list

SUPPORTED COMPONENTS

An excerpt of supported components.
On request we will be happy to check other devices that are not listed for their compatibility.

ALLNET DEVICES WITH FIRMWARE 3.X

PRODUCT NAME	DESCRIPTION
ALL3040	WiFi Basic Sensor Central
ALL3072	Network socket for switching via LAN/W-LAN
ALL3073WLAN	Network socket for switching via LAN/W-LAN
ALL3075 V2	Network socket for switching via LAN (including consumption measurement)
ALL3075 V3	Network socket for switching via LAN (including consumption measurement)
ALL3418 V2	IP Triometer
ALL3419	IP Thermometer/sensor center
ALL3500	IP Homeautomation Appliance
ALL3505	IP Homeautomation Appliance
ALL3653	Remote Control
ALL3690	PM1 PowerMeter (3 phases)
ALL3691	PM2 PowerMeter (2x3 phases, 2xS0, 2xD0)
ALL3692	PM3 PowerMeter (4xD0, 2xI2C)
ALL3696	IP PowerMeter
ALL3697-32A	PM4 PowerMeter (HUT Powermeter, 3 phases)
ALL4075	4-way Network relay (LAN/WLAN)
ALL4076	6-way IP socket strip
ALL4175	4-way Network relay
ALL4176	6-way IP socket strip
ALL4500	Sensoric Appliance
ALL4550	POE information display
ALL5000	IP Facility Control Server

ALLNET DEVICES WITH FIRMWARE 2.X

PRODUCT NAME	DESCRIPTION
ALL3000	Internet Thermometer
ALL3000RF MK2	Control unit for Intertechno radio-controlled sockets
ALL3075	Network socket for switching via LAN
ALL3076	Network socket for dimming via LAN
ALL3090 / ALL3096	8 Port Reset Switch
ALL3100	Ethernet Power Switch
ALL3270	Narrowband Powerline Master Ethernet Bridge
ALL3275	Narrowband Powerline slave socket for switching via Powerline
ALL3418	IP Triometer
ALL3421	IP Triometer
ALL4000	Ethernet sensor meter
ALL4001	HUT Ethernet sensor meter for top hat rail
ALL4100	8 Port Power Switch 19 " with 8 IEC sockets

BACnet

Require: Symcon >= 6.3



The BACnet module is a paid extension that can be purchased for any existing Symcon license in the Shop. Up to 3 BACnet objects can be created free of charge for each Symcon license.

The BACnet (client) connection allows the comfortable setup of BACnet devices within Symcon. Thereby all BACnet devices available in the local network are searched via a discovery instance and created accordingly. In the next step each created device is queried and the data points available on the device are displayed. Any data points can be created or in the simplest case all data points can be imported into Symcon. The data points show the "Present Value", which is updated automatically via COV, provided the device supports this. Alternatively, the value can also be queried cyclically. The value can also be changed via visualization, script and flowchart.
The BACnet (server) allows to bring any variables from Symcon to BACnet via a configurator. The available data points can be read out automatically by the client and COV is also supported by the server accordingly. Symcon can also be automatically recognized as BACnet devices in the network.

VIDEO TUTORIAL FOR SETUP

SUPPORTED PROTOCOL

BACnet/IP

SUPPORTED BIBBS (CLIENT)

DM-DDB-A (Dynamic Device Binding)
DS-RP-A (Read Property)
DS-RPM-A (Read Property Multiple)
DS-WP-A (Write Property)
DS-COV-A (Change of Value Notification)
DS-COVP-A (Change Of Value Property)

SUPPORTED BIBBS (SERVER)

DM-DDB-B (Dynamic Device Binding)
DS-RP-B (Read Property)
DS-RPM-B (Read Property Multiple)
DS-WP-B (Write Property)
DS-COV-B (Change of Value Notification)
DS-COVP-B (Change Of Value Property)

UNTERSTÜTZTE BIBBS (OTHER)

NM-FDR-A (Foreign Device Registration)

SUPPORTED STANDARD OBJECT TYPES

Analog Input
Analog Output
Analog Value
Binary Input
Binary Output
Binary Value
MultiState Input
MultiState Output
MultiState Value
Accumulator

SUPPORTED SEGMENTATION

Receive: Yes (>64 segments, Max 1476 bytes per segment).
Transmit: No (not required so far)

EXAMPLE OF A WRAPPER MODULE FOR NON-STANDARD DEVICES

Integrates the SageGlass SIM II (BACnet) with Symcon

BAC_RelinquishPresetValue

```
bool BAC_RelinquishPresetValue (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Reset the value so that lower priority values can also be set.

EXAMPLE

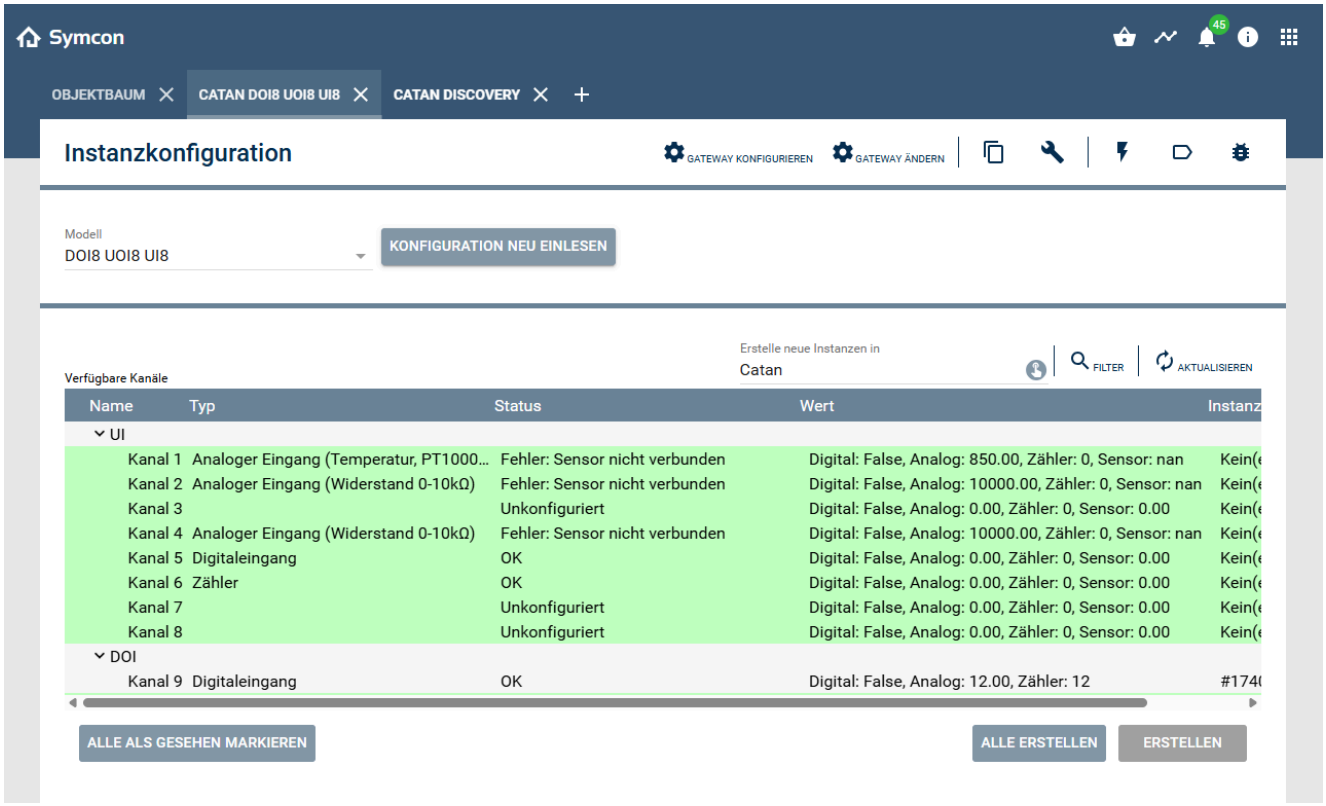
```
BAC_RelinquishPresetValue(12345);
```

Catan

Require: Symcon >= 9.0

INTEGRATION IN SYMCON

The Catan interfaces can be integrated via the Device search. To do this, "Catan Discovery" must be selected as the system. The Discovery instance then offers the creation of a Catan Configurator Configurator for each interface.



All channels of a device are displayed in the configurator and can be created and configured. Different channel types are available depending on the device: DOI UOI UI and DOR. A type can be assigned to each of these channels in the instance configuration. A variable is created below each channel instance that corresponds to the value of the channel. The name as well as the display and action are specified by the channel type.

UI

- Unconfigured
- Digital output
- Counter
- Analog input (voltage 0-10V)
- Analog input (current 0-20mA)
- Analog input (current 4-20mA)
- Analog input (resistance 0-10kΩ)
- Analog input (temperature, PT1000, DIN)
- Analog input (temperature, NI1000, DIN)
- Analog input (temperature, NI1000, LG)
- Analog input (temperature, NTC 10k)
- Analog input (temperature, NTC 10k, PRE)

Analog input (temperature, NTC 20k)

UOI

Unconfigured
Digital output
Digital input
Counter
Analog output (voltage 0-10V)

DOI

Unconfigured
Digital output
Digital input

CONFIGURATION

Which of the following setting options are available depends on the channel type.

CONFIGURATION OF THE DISPLAY

NAME	DESCRIPTION
Allow override on display	If active, the value can be changed via the control panel
Override on display can be canceled	If active, the override can be canceled via the "Cancel override" button in the instance configuration. Alternatively, the PHP function CATAN_CancelOverride can also be used. If the override is canceled, the value is reset to the previous value.
Name source	The name of the channel that is displayed in the Control Panel. From instance name: Name of the instance. From text field: Freely selectable.
Unit	The text that is displayed after the value on the control panel.

CONFIGURATION OF THE VALUE CHANGE

NAME	DESCRIPTION
Minimum send interval	The interval in milliseconds at which value changes are sent
Value change increment	The threshold value at which the change to a value is sent

CONFIGURATION OF THE DEFAULT VALUE

NAME	DESCRIPTION
Default value	The default value
Use default value at start	If active, the default value is set when Catan is started.
Use default value on failure	If active, the default value is set in the event of an error or if the supply voltage fails.

CONFIGURATION OF THE COUNTER

NAME	DESCRIPTION
Scaling factor	The factor by which the counter value is multiplied before it is written to the counter variable
Show raw value of counter	If active, an additional variable is created with the raw value of the counter.

ADDITIONAL CONFIGURATION

NAME	DESCRIPTION
Display channel status	If active, a variable is created that displays the channel status. (States: OK, Error, Overwritten, Deactivated)
Display error status	If active, a variable is created that displays the current error code.

ERROR CODES

VALUE	DESCRIPTION
0	No error
5	Peripheral error
10	Resistance too high
11	Temperature too high
12	Input voltage too high
13	Input current or shunt resistance too high
14	Output voltage too high
15	Upper warning limit exceeded
20	Temperature too low
21	Negative input voltage
22	Input current too low
23	Negative input current
24	Negative input current or sensor not connected
25	Output voltage too low
26	Lower warning limit not reached
30	Sensor not connected
31	Load resistance too high for valid voltage
40	Output overload
50	Virtual zero
51	Virtual alarm

VALUE	DESCRIPTION
52	Virtual obsolete
53	Virtual crash
54	Virtual error
55	Virtual Disabled

Catan also supports other systems

KNX
ModBus
DMX

CATAN_CancelOverride

Require: Symcon >= 9.0

```
bool CATAN_CancelOverride (int $InstanzID)
```

PARAMETERS

InstanzID	The ID of the Catan channel instance whose overwriting is to be canceled.

RETURNS

If the command could be executed successfully, it returns TRUE, otherwise FALSE.

DESCRIPTION

Cancels the overwriting of the channel with the ID InstanzID.

EXAMPLE

```
CATAN_CancelOverride(12345); // Bricht die Überschreibung des Kanals ab
```

CATAN_ResetCounter

Require: Symcon >= 9.0

```
bool CATAN_ResetCounter (int $InstanzID)
```

PARAMETERS

InstanzID	The ID of the Catan channel instance whose counter is to be reset.

RETURNS

If the command could be executed successfully, it returns TRUE as the result, otherwise FALSE.

DESCRIPTION

Sets the counter of the channel with the ID InstanzID.

EXAMPLE

```
CATAN_ResetCounter(12345); // Setzt den Zähler des Kanals ab
```

Device List

SUPPORTED DEVICES

DEVICE	DESCRIPTION
Catan C1	Control
Catan DOR6 UI8	Relay module
Catan DOI8 UOI8 UI8	input/output module

digitalSTROM

DigitalSTROM is a system that uses power grid wiring. A connection with IP-Symcon is established via LAN(IP) with the digitalSTROM server.
Furthermore, a digitalSTROM terminal block is installed in the device to be operated, or the conventional luster terminal is replaced.
All electrical devices that are compatible with the digitalSTROM server via digitalSTROM terminal block are also compatible with IP-Symcon and can be integrated.

SETUP VIDEO-TUTORIAL

CONNECTION

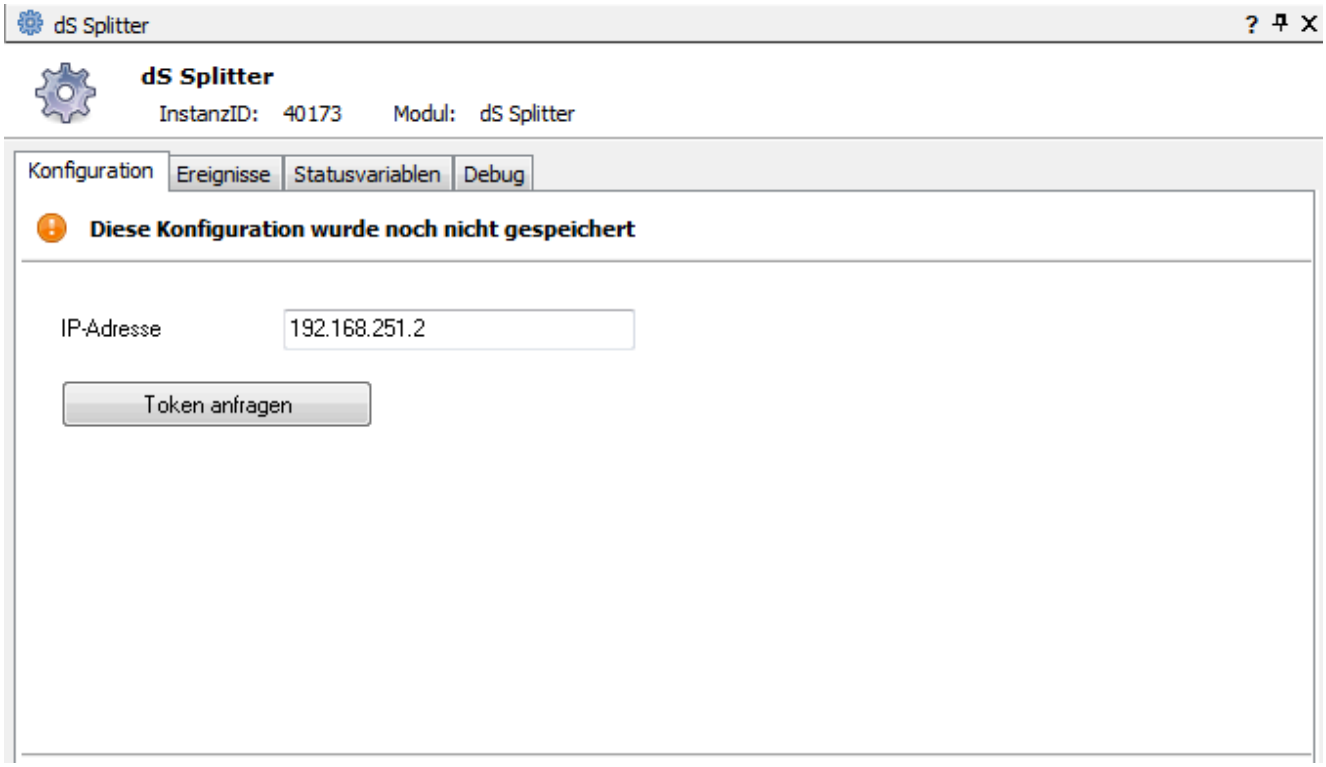
Before the digitalSTROM Server is connected to the PC, all electrical devices to be controlled, which are to be operated with the PC, should be correctly connected to the PC via digitalSTROM terminal block. Then the digitalSTROM Server must be connected to the network with a network cable.

INSTALLATION

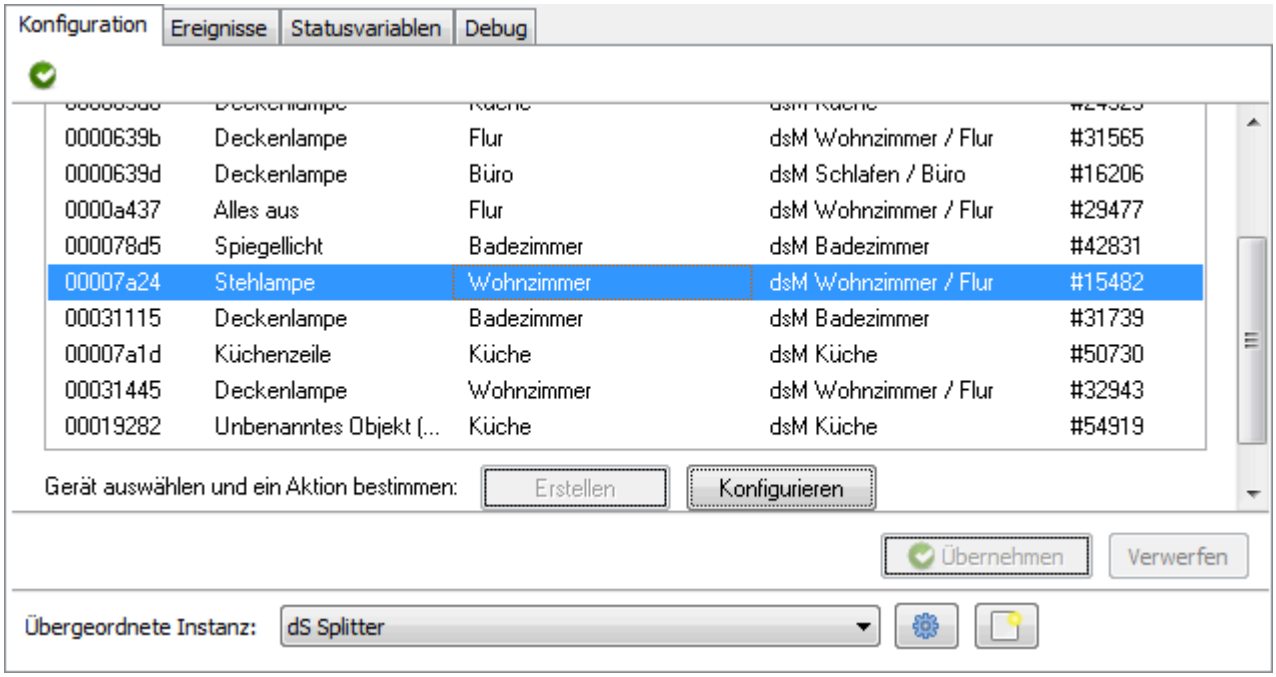
If the digitalSTROM Server is correctly connected and a DHCP server is used, the digitalSTROM Server is now connected to the local network.
(The DHCP server automatically assigns addresses for the network participants.)
The digitalSTROM Server appears in the network environment under “Other devices” as “dSS”.
By "double-clicking" on "dSS", a window appears in the lower area of which the IP address of the digitalSTROM Server is located. After it has been copied and pasted into any browser, the digitalSTROM configuration webpage opens. A security warning may appear, which must be confirmed with "I know the risk".
The following applies to the first login: "Name" = "dssadmin" and "Password" = "dssadmin".
The password should be changed after the first login. Note that it is case-sensitive.
Some pre-settings can be made on the configuration webpage.
It is also possible to assign the IP address manually.

INTEGRATION IN IP-SYMCON

A "dS configurator" can be created within the Management Console via "Welcome page -> Manage configurators". The "dS Splitter" can be configured using the gear wheel in the lower window area.

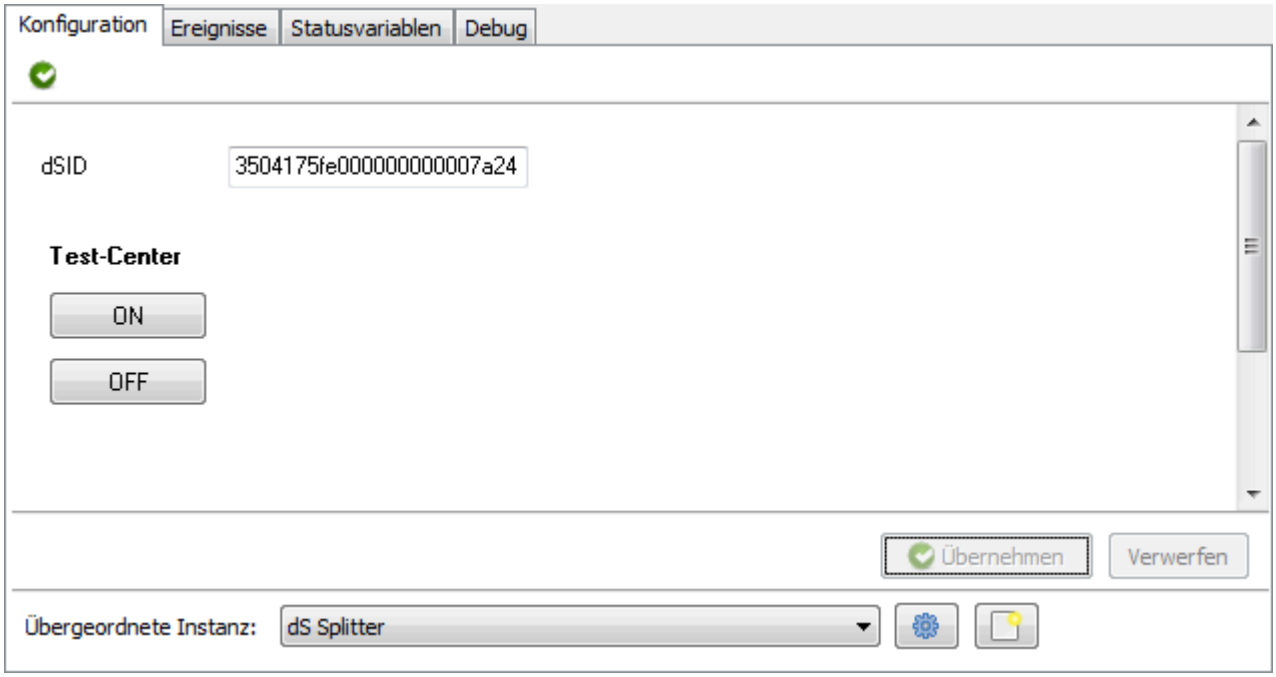


The configuration of the "dS Splitter" opens. The IP address of the digitalSTROM server must be entered for "IP address". The setting is saved with "Apply". After clicking on “Request Token”, the name (username) and password (password) of the configuration webpage must be entered. If the entry is correct, a lot of numbers and small letters will appear next to tokens. The "dS Splitter" configuration can now be closed. The search can now be started in the digitalSTROM Configurator via “Search”. Then all devices connected to the digitalSTROM server via digitalSTROM terminals appear.



The ID on the left is the identification number that can be found on the back of the respective digitalSTROM terminal block. In the case of a large number of installed devices, this ensures problem-free assignment and configuration. The ID on the right is the InstanceID assigned by IP-Symcon. This is unique and unchangeable.
Each device can be created individually as an instance. To do this, it must be selected and the "Create" button clicked. The created devices can now be operated.

After selecting a digitalSTROM device and clicking on “Configure”, the function of each individual device can be tested.



The configuration page can also be called up via a "double click" within the object tree. (These devices can be found in the object tree under splitter instances.)

DS_CallScene

```
bool DS_CallScene (int $InstanceID, int $Scene)
```

PARAMETERS

InstanceID	ID of the device to be switched
Scene	Scenes from 0 to 127

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with the ID **InstanceID** to the value **Scene**.
More information about the server scenes

EXAMPLE

```
DS_CallScene(12345, 42);
```

DS_DimSet

```
bool DS_DimSet (int $InstanceID, int $Intensity)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	Value of 0-255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with the ID **InstanceID** to the value **Intensity**

EXAMPLE

```
DS_DimSet(12345, 128); //Dim to 50%
```

DS_MakeRequest

```
string DS_MakeRequest (int $InstanceID, string $Request, string $Data)
```

PARAMETERS

InstanceID	ID of the device to be switched
Request	Request which should be called
Data	Parameters for request

RETURNS

JSON encoded string

DESCRIPTION

Sends the request **Request** with the parameter **Data** to the device with the ID **InstanceID**. Further information on the possible queries and the respective required parameters can be found under dSS Json .

EXAMPLE

```
// Request for system version
// Sends the "system/version" request to the instance with ID 12345
DS_MakeRequest(12345, "system/version", "");

// Recalls scene 0
// Sends the request "apartment/callScene" with the data "sceneNumber =
0" to the instance with the ID 12345
DS_MakeRequest(12345, "apartment/callScene", "sceneNumber=0");
```

DS_RequestStatus

Require: Symcon >= 4.0

```
bool DS_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the status for the device with the ID **InstanceID**.



This command queries the terminal via the dS bus, which can take some time. During this time the dS bus is busy and cannot carry out any further actions. If the command is called often and for many terminals, this can lead to noticeable delays in the entire digitalStrom system.

EXAMPLE

```
DS_RequestStatus(12345);
```

DS_ShutterMove

```
bool DS_ShutterMove (int $InstanceID, int $Position)
```

PARAMETERS

InstanceID	ID of the device to be switched
Position	Position from 0-255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the roller shutter/awning/ blind with the ID **InstanceID** to the position **Position** until DS_ShutterStop is executed or the target position is reached.

EXAMPLE

```
DS_ShutterMove(12345, 128); //Set to 50%
```

DS_ShutterMoveDown

```
bool DS_ShutterMoveDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the roller shutter/awning/blind with the ID **InstanceID** down until DS_ShutterStop is executed or the end position is reached.

EXAMPLE

```
DS_ShutterMoveDown(12345); //Move down
```

DS_ShutterMoveUp

```
bool DS_ShutterMoveUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the roller shutter/awning/blind with the ID **InstanceID** upwards until DS_ShutterStop is executed or the end position is reached.

EXAMPLE

```
DS_ShutterMoveUp(12345); //Move up
```

DS_ShutterStop

```
bool DS_ShutterStop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Stops a current motion of the roller shutter/awning/blind.

EXAMPLE

```
DS_ShutterStop(12345); //Stop
```

DS_SwitchMode

```
bool DS_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for on, FALSE for off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the device with the ID InstanceID to the value Status.

EXAMPLE

```
DS_SwitchMode(12345, true); //Turn device on
```

Device list

SUPPORTED GATEWAYS

GATEWAY	DESCRIPTION
dSS11	digitalSTROM Server
dSS20	digitalSTROM Server

SUPPORTED COMPONENTS

LIGHT (YELLOW)

MANUFACTURER DESIGNATION	CONNECTION TYPE	DESCRIPTION
GE-KL200	Relay terminal	Switching 1400 W
GE-KM200	Luster terminal	Switching/dimming 150 W
GE-KM220	Luster terminal	Save 150 W
GE-SDM200	Cord dimmer	Dimming 150 W
GE-SDS200	Cord dimmer	Dimming 150 W
GE-SDS210	Cord dimmer	Save 150 W
GE-TKM210	Push button terminal	with output control + dimming 150 W
GE-TKM220	Push button terminal	Operate 1-way
GE-TKM230	Push button terminal	Operate 2-way
GE-TUP200	Button	Operate 1-way
GE-TUP210	Button	Operate 2-way
GE-TUP220	Button+	Operate 1-way plus
GE-UMV200	Screw terminals	1-10V universal module
GE-ZWS210	Adapter plug	Switching 2300 W
GE-ZWS200	Adapter plug	Switching 2300 W

SHADOW (GRAY)

MANUFACTURER DESIGNATION	CONNECTION TYPE	DESCRIPTION
GR-HKL230	Relay terminal	Blind switch actuator
GR-KL200	Relay terminal	700 VA roller shutter

MANUFACTURER DESIGNATION	CONNECTION TYPE	DESCRIPTION
GR-KL210	Relay terminal	Awning 700 VA
GR-KL220	Relay terminal	Blind 700 VA
GR-TKM200	Push button terminal	Operate 1-way
GR-TKM210	Push button terminal	Operate 2-way
GR-TUP200	Button	Operate 1-way
GR-TUP210	Button	Operate 2-way
GR-TUP220	Button	Operate 1-way plus

HEATING/AIR CONDITIONING (BLUE)

MANUFACTURER DESIGNATION	CONNECTION TYPE	DESCRIPTION
BL-KL200	Relay terminal	Switching 700 W
BL-KM200	Luster terminal	Switch 150 W
BL-SDS2001	Sensor with output	Switch 150 W + sensor
BL-TUP2001	Sensor	Climate sensor

JOKER (BLACK)

MANUFACTURER DESIGNATION	CONNECTION TYPE	DESCRIPTION
dS-iSens200	Luster terminal	Room sensor for temperature and humidity
dS-Alarm400	Push button terminal	4 sensors can be connected
SW-AKM200	Push button terminal	4 sensors can be connected
SW-AKM210	Push button terminal	2 sensors can be connected
SW-SSL200	Cord dimmer	1-way switch
SW-TKM200	Push button terminal	4-way universal operation
SW-TKM210	Push button terminal	2-way universal operation
SW-TUP200	Button	1-way universal operation
SW-UMR200	Screw terminals	Relay universal module
SW-ZWS200	Adapter plug	Switching 2300 W.
SW-ZWS210	Adapter plug	Switching 2300 W.

ACCESS (GREEN)

No separate control required.

SECURITY (RED)

No separate control required.

AUDIO (LIGHT BLUE)

No separate control required.

DMX / ArtNet

DMX or DMX512 is a protocol for controlling lighting technology components. The connection to IP-Symcon is established, for example, via a DMX USB interface (serial) or DMX LAN interface. The IP protocol ArtNet is also supported via the LAN interface.

The following devices are supported by IP-Symcon:
Supported components

DESCRIPTION

DMX (lighting technology) or also called DMX512 is a digital control protocol that is used in stage and event technology. It is used to control dimmers, spotlights and effect devices on stages or in discos. The abbreviation DMX stands for Digital Multiplex, it is a standard that was standardized by DIN 56930-2. The connection to IP-Symcon takes place via a DMX USB interface/DMX LAN interface. ArtNet has also been supported since IP-Symcon 3.0. It is an IP based protocol for transferring large amounts of DMX information over LAN. The specification was created by the company "Artistic License" and is freely available. IP-Symcon supports all ArtNet-compatible DMX interfaces.

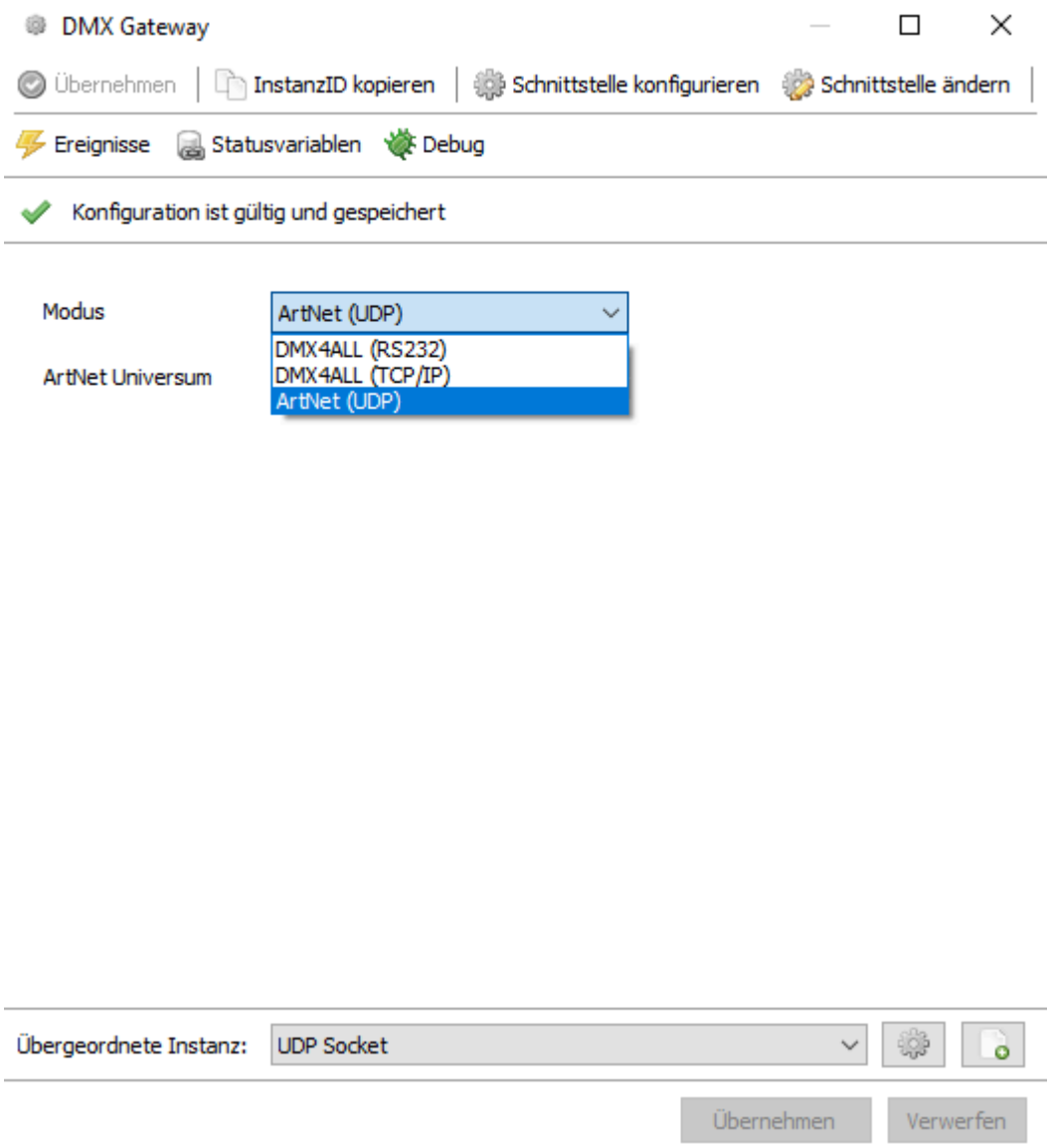
SETUP VIDEO-TUTORIAL



INTEGRATION IN IP-SYMCON

A prerequisite for operation in connection with IP-Symcon is properly installed hardware and knowledge of the COM port interface used or the IP address.
A DMX instance must be added in IP-Symcon and the DMX or ArtNet mode for the

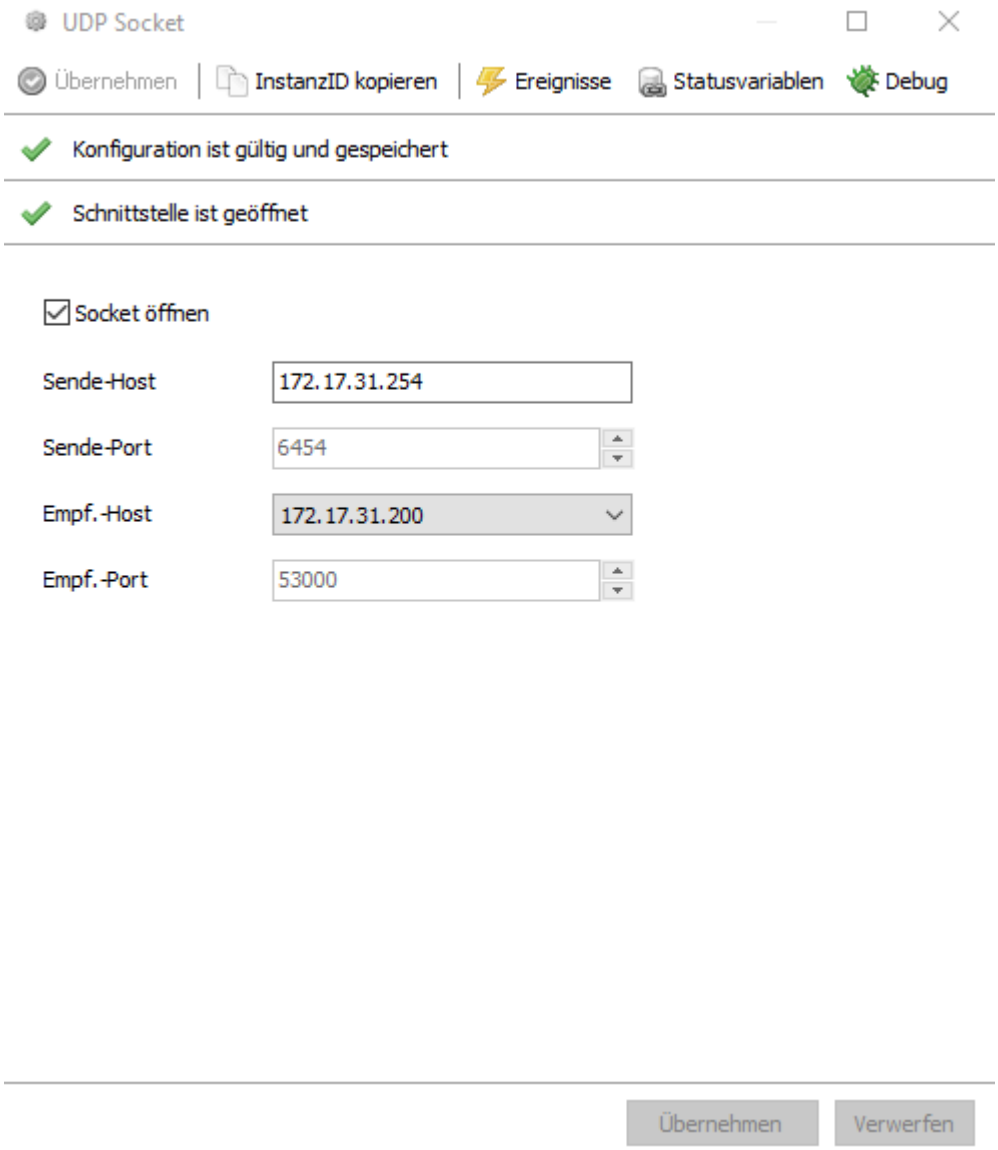
connected hardware must be selected in the parent instance. With ArtNet the "Universe" is usually 0 and with DMX the entry is irrelevant.



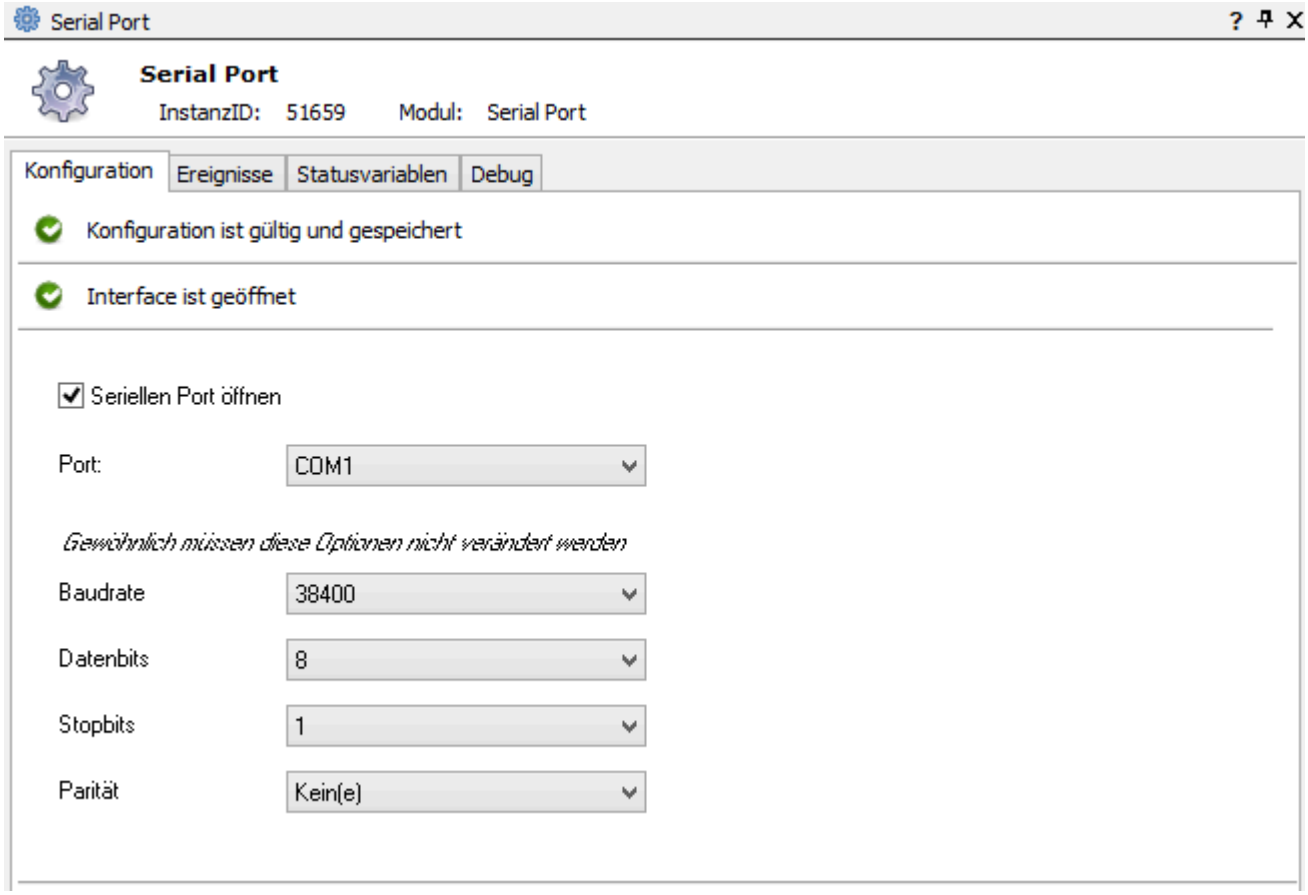
When using ArtNet, the following settings must be made in the parent instance:
'Send Host' is the IP address of the ArtNet interface and the 'Send Port': 6454 - the default port of the ArtNet protocol.
'Receive Host' is the IP address of the network card to which the interface is connected. It is usually the address of the PC on which the IP-Symcon server is running.



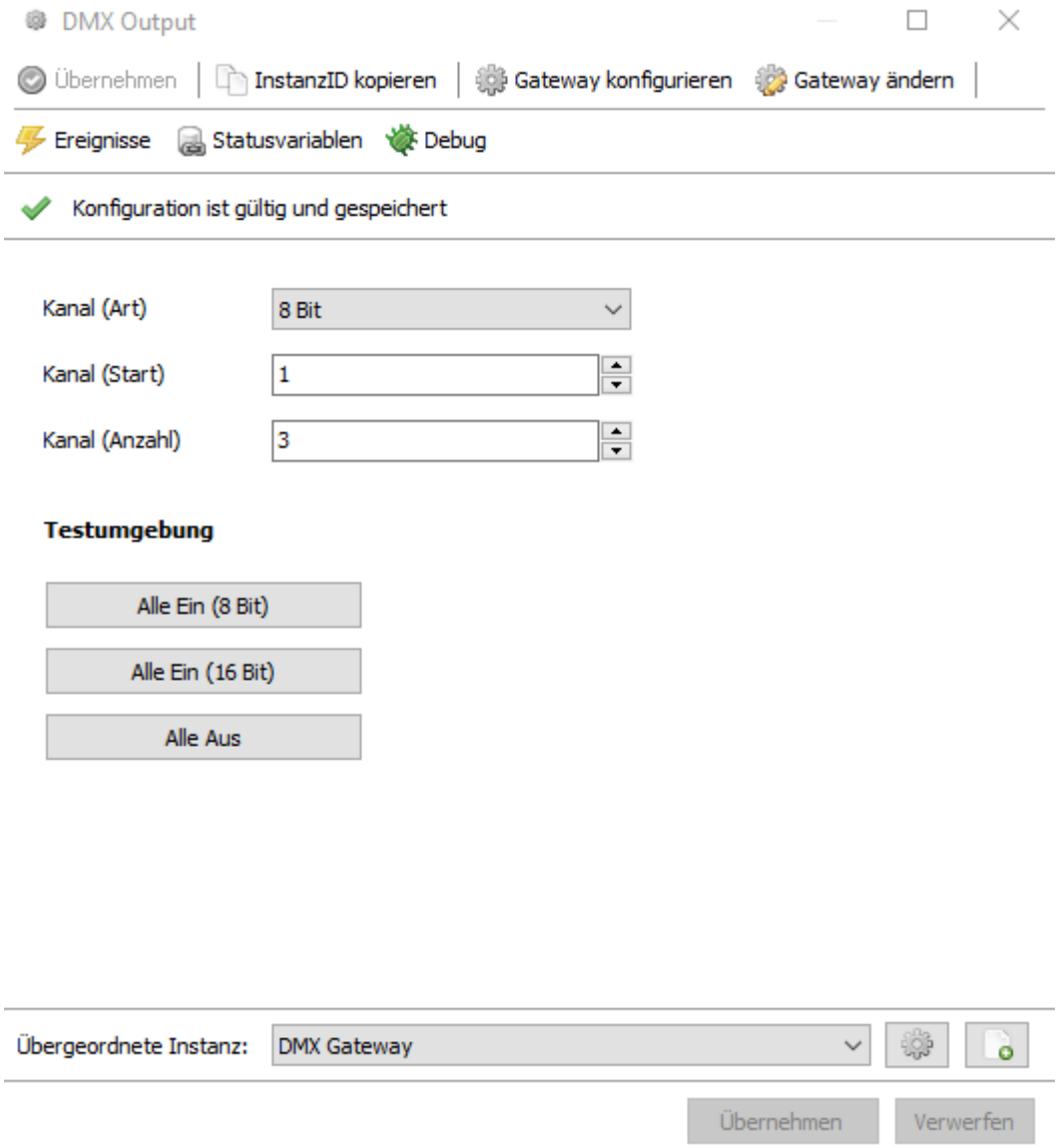
If several ArtNet interfaces are created, the receiving port (default: 53000) must be increased by 1 for each each UDP Socket. (i.e. the second interface has e.g. port 53001)



With a DMX interface that is connected via USB, a virtual COM port is installed in the Windows device manager. This is to be entered here. The other baud rate settings: 38400,8,1,N must not be changed.
Once all the settings have been made, the port or socket must be opened and "Apply" pressed.



In the 'DMX output' instance the functionality can be checked in the test environment.



Several channels of the DMX device can be addressed with one instance. This is useful, for example, to be able to display complete DMX-RGB spotlights as a coherent device in the visualization. All channels assigned to the instance can be switched on or off using the "All On" and "All Off" buttons.

In the case of a DMX RGB instance with channel(type) = 16bit, it should be noted that not only the set channel is used, but also the next higher one. DMX channels are 8 bit. To control 16 bit, 2 channels are used. Example: Channel(Red) = 40 means that channels 40 and 41 are used.

DMX_FadeChannel

```
bool DMX_FadeChannel (int $InstanceID, int $Channel, int $Value, float
$FadeTime)
```

PARAMETERS

InstanceID	ID of the device to be switched
Channel	0 = all module channels, 1 – 512 depending on configuration
Value	0-255
FadeTime	Time in seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the **Channel** of the device with ID **InstanceID to** Value **with a certain** FadeTime__

EXAMPLE

```
DMX_FadeChannel(12345, 0, 255, 2.5); //Dim all channels of the module
to 255 in 2,5 sec
```

DMX_FadeChannelDelayed

```
bool DMX_FadeChannelDelayed (int $InstanceID, int $Channel, int $Value,
float $FadeTime, float $DelyTime)
```

PARAMETERS

InstanceID	ID of the device to be switched
Channel	0 = all module channels, 1 – 512 depending on configuration
Value	0-255
FadeTime	Time in seconds
DelyTime	Time in seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the **Channel** of the device with ID **InstanceID to** Value **with a certain** FadeTime**. You can specify a** DelayTime__ after which the dimming starts.



When no delay time is set this command will abort all pending fading which were queued by DMX_FadeChannelDelayed. This allows creating a new queue of fadings. This hint is only relevant when using fadings with delay time

EXAMPLE

```
//Dims a module completely down and back up.
DMX_FadeChannel(12345, 0, 255, 5.0); //Dim all channels of the module
to 255 in 2,5 sec
DMX_FadeChannelDelayed(12345, 0, 0, 2.5, 5); //Dim all channels of the
module to 0 in 2,5sec., after a delay of 5sec.
```

DMX_FadeRGB

```
bool DMX_FadeRGB (int $InstanceID, int $R, int $G, int $B, float
$FadeTime)
```

PARAMETERS

InstanceID	ID of the device to be switched
R	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
G	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
B	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
FadeTime	Time in seconds

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Dims the red, green and blue channels of the ID InstanceID with a certain FadeTime.

EXAMPLE

```
// Fade color to yellow in 5 seconds
// 8bit instance
DMX_FadeRGB(12345, 255, 255, 0, 5.0);

// 16bit instance
DMX_FadeRGB(12345, 65535, 65535, 0, 5.0);
```

DMX_FadeRGBDelayed

```
bool DMX_FadeRGBDelayed (int $InstanceID, int $R, int $G, int $B, float
$FadeTime, float $DelayTime)
```

PARAMETERS

InstanceID	ID of the device to be switched
R	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
G	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
B	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
FadeTime	Time in seconds
DelayTime	Time in seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the red, green, and blue channels of ID **InstanceID** with a specific **FadeTime**. A delay time can be specified after which the dimming process is started.



If no delay time is specified, this command cancels all fadings set for the affected channels via DMX_fadergbDelayed. This allows a new chain of delayed fadings to be set cleanly. If delayed fading is not used, this note is not relevant

EXAMPLE

```
//Dims a module completely up and down to yellow.

//8bit
//Dim the instance to yellow within 5 seconds
DMX_FadeRGB(12345, 255, 255, 0, 5.0);
//Dim the instance to yellow within 2.5 seconds, after a delay of 5
```

```
seconds
DMX_FadeRGBDelayed(12345, 255, 255, 0, 2.5, 5);

//16 bit
//Dim the instance to yellow within 5 seconds
DMX_FadeRGB(12345, 65535, 65535, 0, 5.0);
//Dim the instance to yellow within 2.5 seconds, after a delay of 5
seconds
DMX_FadeRGBDelayed(12345, 65535, 65535, 0, 2.5, 5);
```

DMX_RequestInfo

```
bool DMX_RequestInfo (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the DMX splitter instance

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the information of the DMX instance with the ID **InstanceID**.

EXAMPLE

```
// Request information from instance 12345.
DMX_RequestInfo(12345);
```

DMX_ResetInterface

```
bool DMX_ResetInterface (int $InstanceID)
```

PARAMETERS

InstanceID	ID of DMX I/O Instance

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the DMX interface back with the ID **InstanceID**. It is recommended that in case of problems, create a script with this command and declare it as startup script.

EXAMPLE

```
//Reset interface
DMXI_ResetInterface(12345);
```

DMX_SetBlackout

```
bool DMX_SetBlackOut (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for old value, FALSE for 0

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sets all channels of DMX interfaces with ID InstanceID to value Status.
This command does not affect any state variables of the modules.

EXAMPLE

```
//Deactivate all channels
DMXI_SetBlackout(12345, true);
//Reactivate all channels
DMXI_SetBlackout(12345, false);
```

DMX_SetChannel

```
bool DMX_SetChannel (int $InstanceID, int $Channel, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Channel	0 = all module channels, 1 – 512 depending on configuration
Value	0-255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the channel from the device with the ID **InstanceID** to value **Wert**

EXAMPLE

```
DMX_SetChannel(12345, 0, 255); //Switch channel 0 to 255
```

DMX_SetRGB

```
bool DMX_SetRGB (int $InstanceID, int $R, int $G, int $B)
```

PARAMETERS

InstanceID	ID of the device to be switched
R	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
G	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)
B	Brightness value 0 (off) to 255/65535 (8/16 bit) (on, maximum brightness)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the red, green and blue channels of the ID **InstanceID**.

EXAMPLE

```
// set color to yellow

// 8bit instance
DMX_SetRGB(12345, 255, 255, 0);

// 16bit instance
DMX_SetRGB(12345, 65535, 65535, 0);
```

Device list

SUPPORTED INTERFACES

DMX4ALL Mini-USB-DMX-Interface
DMX4ALL NanoDMX USB Interface
DMX4ALL USB-DMX STAGE-PROFI MK2
DMX4ALL USB-DMX STAGE-PROFI MK3
LAN-DMX STAGE-PROFI
AVR Art Net Node
Art Net Box (Quad Art Net Box)
All ArtNet interfaces are supported

SUPPORTED DEVICES

UP TO VERSION 4.0

All devices that use the interfaces listed above and a maximum of 8 bit (256 steps) resolution.

VERSION 4.1 AND ABOVE

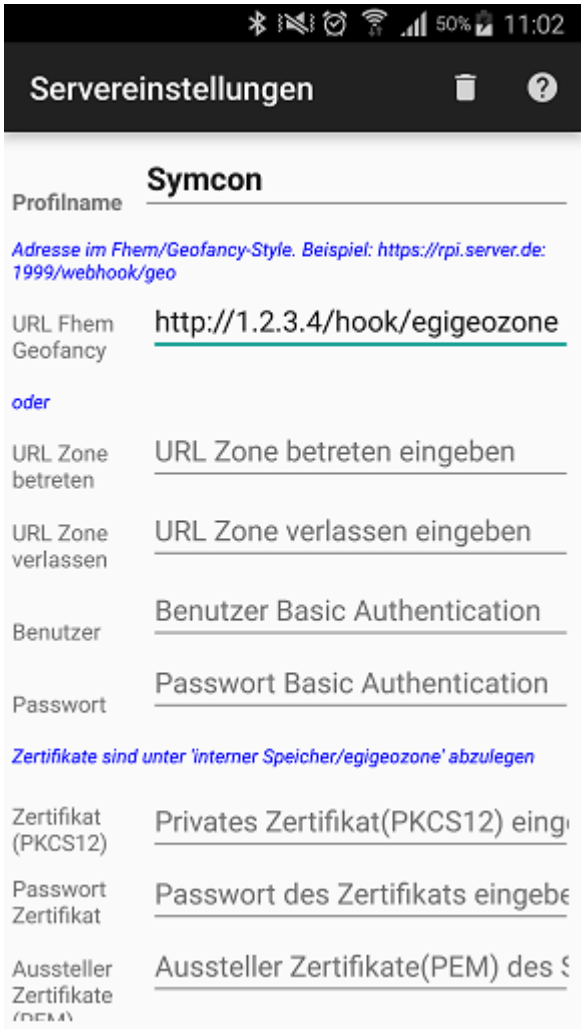
All devices that use the interfaces listed above.
The 16bit (65535 steps) resolution is also supported.

EgiGeoZone

Require: Symcon >= 4.2
The module is used to receive EgiGeoZone data.

FUNCTION SCOPE

Separate location list for each device
Username and password identification within IP-Symcon.
Automatically sets up the webhook "/hook/egigeozone".
It is recommended to use this in combination with the Connect module.



PREREQUISITES

EgiGeoZone App for Google Android

SOFTWARE INSTALLATION

Via the Module Store install the module EgiGeoZone.

INSTANCE SETUP IN IP-SYMCON

Under "Add Instance" the 'EgiGeoZone' module can be found using the quick search.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Username	Username which must be specified in the EgiGeoZone App to send IP-Symcon data.
Password	Password which must be specified in the EgiGeoZone App.

*If this data is left empty, anyone can send data to IP-Symcon via the hook

STATUS VARIABLES

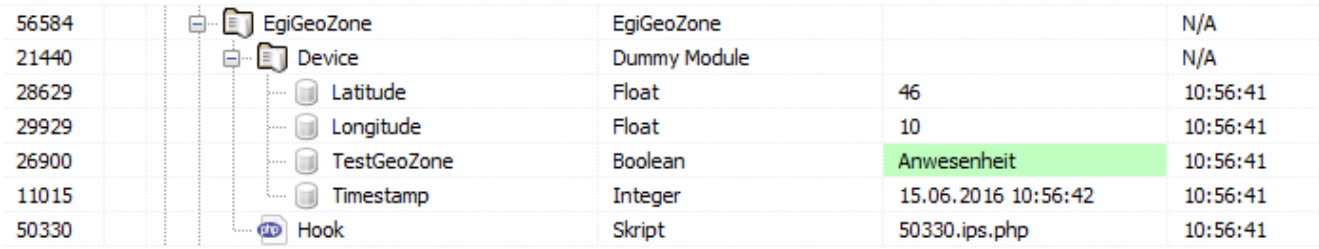
The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

The variables are created automatically based on the device ID and when sending within the EgiGeoZone module for the first time. Multiple devices can run through one hook. Each device is set up under its own "category".

NAME	TYPE	DESCRIPTION
Device Name	Instance (Dummy)	Serves as a "category" in which all monitored locations, as well as the timestamp and longitude/latitude are located. Created per device.
Latitude	Float	Latitude of the last activity.
Longitude	Float	Longitude of the last activity.
Timestamp	Integer	UnixTimestamp of the last activity.
Sample Location (GeoZoneTest)	Boolean	Present or Absent. Information supplied by Gefency.

Example:



VISUALIZATION

There is no native display via Visualization or in the mobile apps. Devices and variables that should be displayed can be displayed via link.

ekey

The company ekey manufactures various finger scanners. These can be integrated into IP-Symcon using a gateway. IP-Symcon receives the activities of the finger scanner and makes them available via variable.



The following devices are supported by IP-Symcon:
Supported devices
To integrate ekey bionyx, please go to the following page ekey bionyx

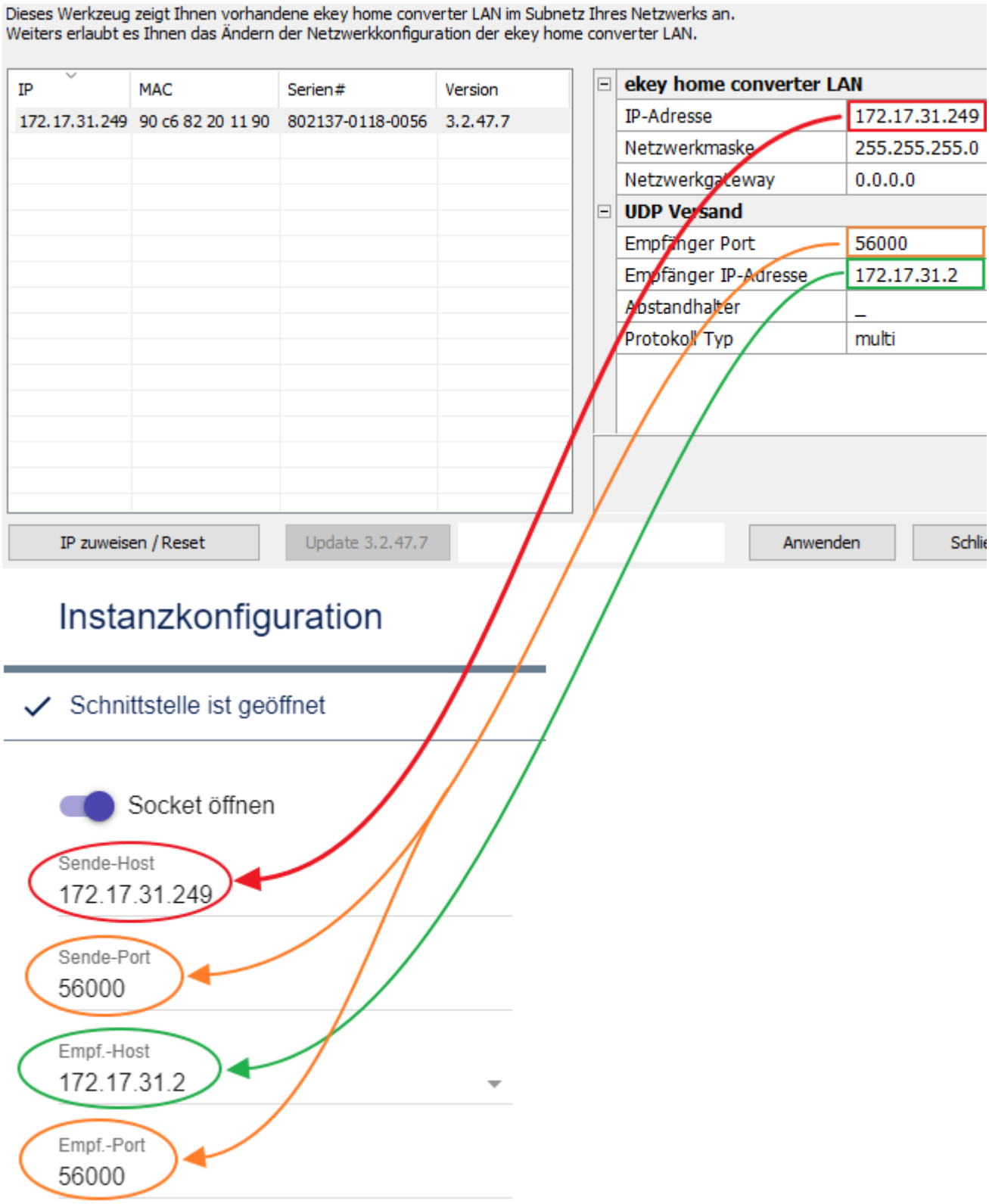
INSTALLATION

In order to grant IP-Symcon access to the Home/Multi/Net system, the appropriate CV LAN module (gateway) must be integrated into the network.
The IP address for the UDP socket can be set using the eKey home CV LAN tool.



The CV LAN Tool is available in different versions for Home , Multi and Net.

After setting up the CV LAN, the specified IP addresses must be configured in IP-Symcon.
The protocol and separator must be set in the eKey instance.
The sending and receiving host and port must be entered via "Configure gateway".
The IP from the IP-Symcon server must be entered as the recipient IP.


Dieses Werkzeug zeigt Ihnen vorhandene ekey home converter LAN im Subnetz Ihres Netzwerks an.
Weiters erlaubt es Ihnen das Ändern der Netzwerkkonfiguration der ekey home converter LAN.
IP
MAC
Serien#
Version
172.17.31.249
90 c6 82 20 11 90
802137-0118-0056
3.2.47.7
ekey home converter LAN
IP-Adresse
172.17.31.249
Netzwerkmaske
255.255.255.0
Netzwerkgateway
0.0.0.0
UDP Versand
Empfänger Port
56000
Empfänger IP-Adresse
172.17.31.2
Abstandhalter
_
Protokoll Typ
multi
IP zuweisen / Reset
Update 3.2.47.7
Anwenden
Instanzkonfiguration
Schnittstelle ist geöffnet
Socket öffnen
Sende-Host
172.17.31.249
Sende-Port
56000
Empf.-Host
172.17.31.2
Empf.-Port
56000

Device list

SUPPORTED GATEWAYS

GATEWAY	DESCRIPTION
ekey Home Converter LAN RS-485	Home/Multi System
ekey Net CV LAN RS-485	Net System
ekey bionyx	Cloud System

SUPPORTED COMPONENTS

IP-Symcon supports all devices that can be connected to ekey Home/Multi/Net.

ekey bionyx

Require: Symcon >= 7.0
The company ekey manufactures various finger scanners and these can be integrated into IP-Symcon using a gateway. IP-Symcon receives the activities of the finger scanner via webhooks.

RANGE OF FUNCTIONS

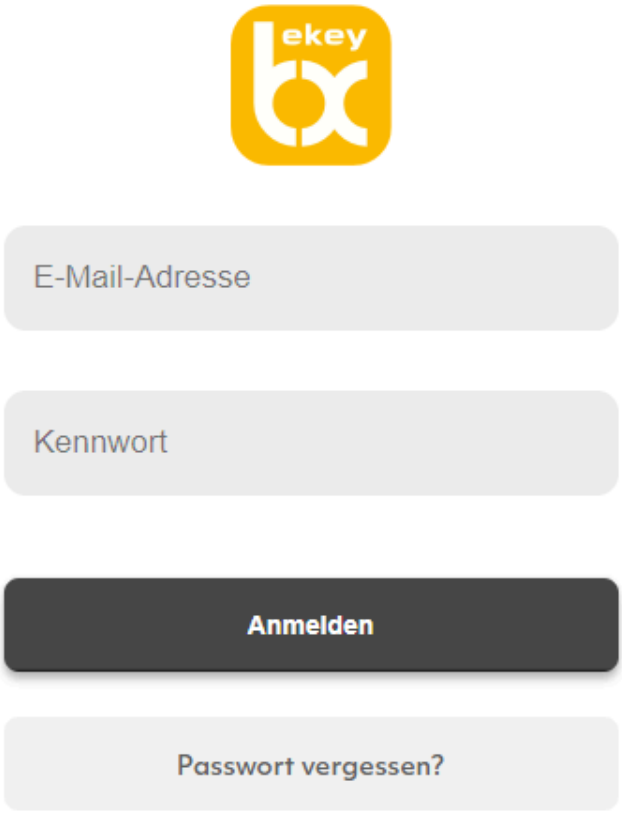
Set up one or more ekey bionyx systems
Storing actions for webhooks

SETTING UP THE INSTANCES IN IP-SYMCON

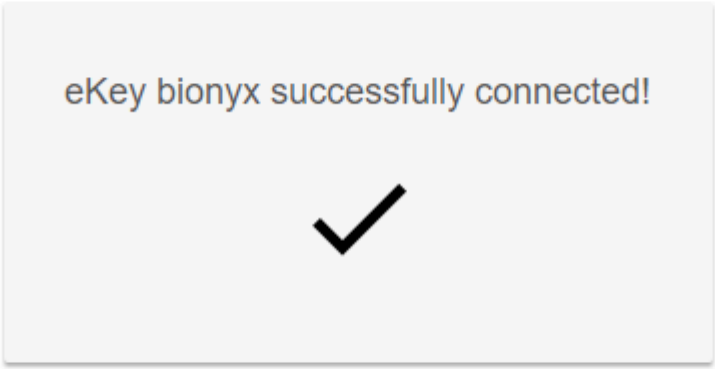
Under 'Add instance', the 'ekey bionyx configurator' module can be found using the quick filter.
Further information on adding instances can be found in the Instance documentation
The interface can be configured when creating the instance.
Configuration page of the cloud:

NAME	DESCRIPTION
Connection	**Local**: If the ekey controller and IP-Symcon are in the same network, this option is used. **Connect**: If the ekey controller and IP-Symcon are not in the same network, communication takes place via the Connect service
IP address	The IP address of the IP-Symcon server

The instance is then created. The message "A higher-level instance is inactive" appears in the configurator. The ekey Cloud instance is called up. To complete the setup of the ekey Cloud instance, click on the Register button. A new tab opens in the browser with the ekey login screen.



After entering the login data, the following dialog is displayed if successful.



You can now return to IP-Symcon. A token should now be available in the ekey bionyx Cloud instance.
In the ekey bionyx configurator, you can now click on Update to display the existing systems that are linked to the account. These can now be created and configured.
Configuration page of the configurator:

NAME	DESCRIPTION
iD	ID
Name	Name of the ekey Account Admin
Information	Information on how many webhooks are available and present for this system

5 webhooks are allowed per account.
configuration page of the system__:
Functions Webhooks:

NAME	DESCRIPTION
ID	ID of the webhook
Function	Function name that is displayed in the ekey bionyx app
Location	Location name which is displayed in the ekey bionyx app
Action	Action that is executed when the webhook is triggered

EnOcean

EnOcean is a wireless system that is supported by multiple manufacturers. For a connection to IP-Symcon a USB or LAN interface is required.



The following devices are supported by IP-Symcon:
Supported Devices (incl. gateways) and EEP List

CONTENT

Installation
Integration into IP-Symcon
Configuration
- Actor
- Actor (Eltako)
- Actor (OPUS)
- Actor (Actuator)
- Sensor

Tips & Tricks

INSTALLATION

The **USB Gateway** is connected to the server and is accessible via Serial Port.
The **Symcon LAN Gateway** is connected to the server via LAN. For the configuration, the IP-Symcon "Network Configuration Tool" is required. It is available for Download . A simple Description for Configuration of the Gateway (German) is available. Afterwards, the gateway can be accessed at the corresponding IP address and Port (default: 5000).



If no IP-Symcon house is printed on the front of the LAN gateway, it is an older model of the gateway. The installation manual for that model can be downloaded here (German) .

The **Eltako FGW14** is connected to a SymBox with over RS232. The SymBox must have the RS232 module fitted. The FGW14 is in turn connected to the **Eltako FAM14** via the "Eltako Bus14" pluggable jumper bridges. The following connectivity configuration is to be observed:
FAM14:
Rotary Mode Selector (BA):
Set According to manual. Select mode 2 for bi-directional communication.
Select mode 1 for address assignment.
Terminal Pins "N" and "L":
Connect with 230V mains power. Either by attaching a suitable mains plug or connect directly to live and neutral rail inside a control cabinet or fuse box.
Terminal Pin "Hold":
Connect with terminal pin "Hold" on FGW14.

Terminal Pin "GND":
Connect with ground pin (⊖) of the RS232 interface of the SymBox.

FGW14:

Rotary Mode Selector (BA):
Select mode 6: "Bus14 ↔ RS232 58k Baud".
Terminal Pin "Hold":
Connect with terminal pin "Hold" on FAM14.
Terminal Pin "Tx":
Connect with terminal pin ↓ of the RS232 interface of the SymBox.
Terminal Pin "Tx":
Connect with terminal pin ↑ of the RS232 interface of the SymBox.

INTEGRATION INTO IP-SYMCON

When using the LAN gateway it can be integrated via Device Search. This is realized by selecting the system "EnOcean Discovery". The Discovery instance offers the creation of an EnOcean Configurator. After the configurator was created, individual devices can be integrated as described below.
The USB gateway is directly created as configurator as it cannot be found via Device Search. This is done by creating an "EnOcean Configurator" in the Object Tree. During the configuration of the interface, the according ESP2 or ESP3 mode must be selected and finally the according port selected and opened.

How the I/O instance (parent instance of the gateway) is selected and eventually replaced, can be found here: Replace I/O Instance

CONFIGURATION

Depending on the device type, the configuration differs. There are four different device types. Each has its one different approach during configuration. (see below in the corresponding section)

To use an EnOcean device, it must be taught in explicitely into IP-Symcon.

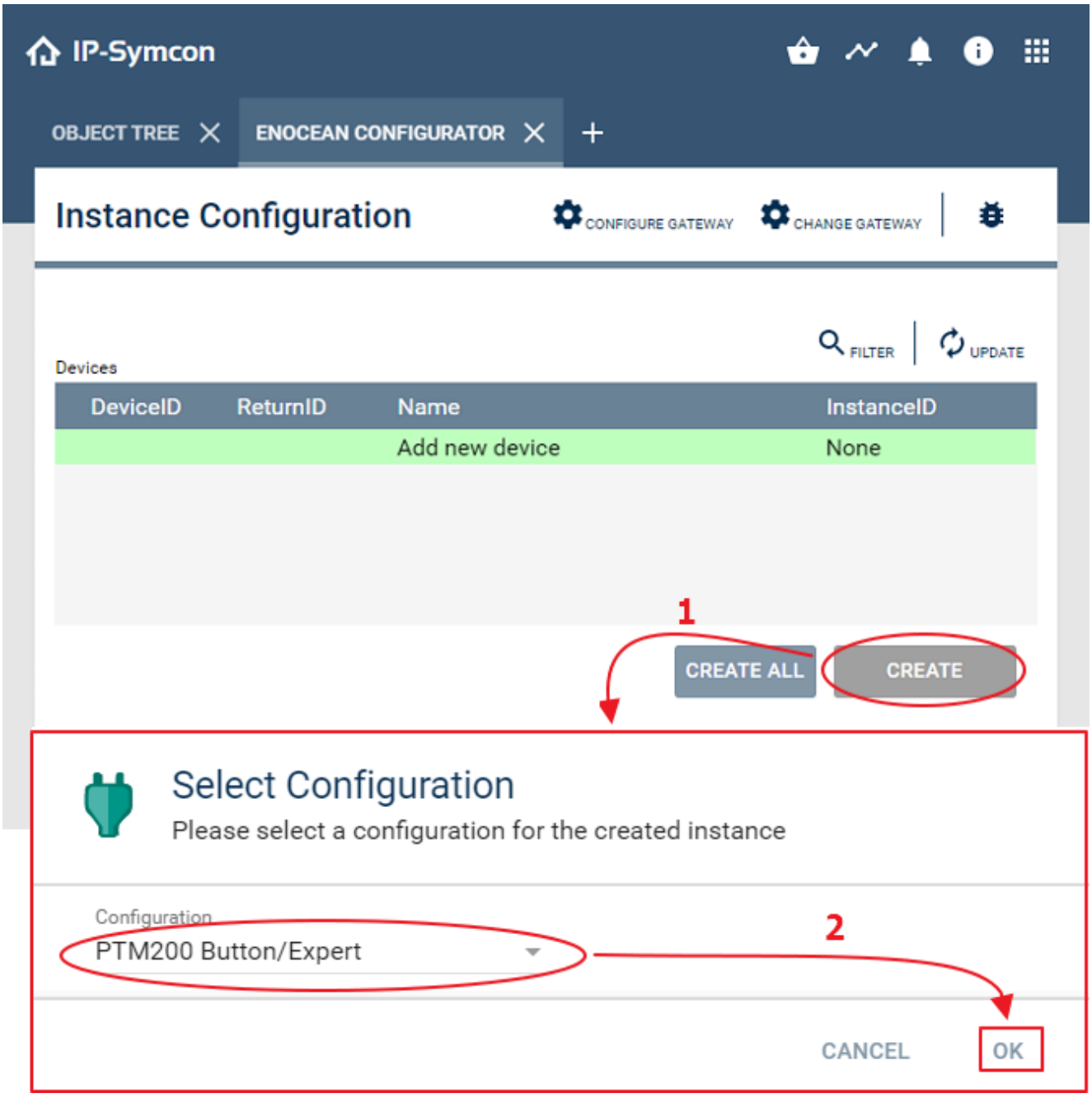
Every Device ID (0-127) may only be given once.

Before starting a search, the parent instance needs to be configured. The correct serial ports need to be entered and the configuration applied. The setting of Baud rate etc. is not required and is done automatically by the system.

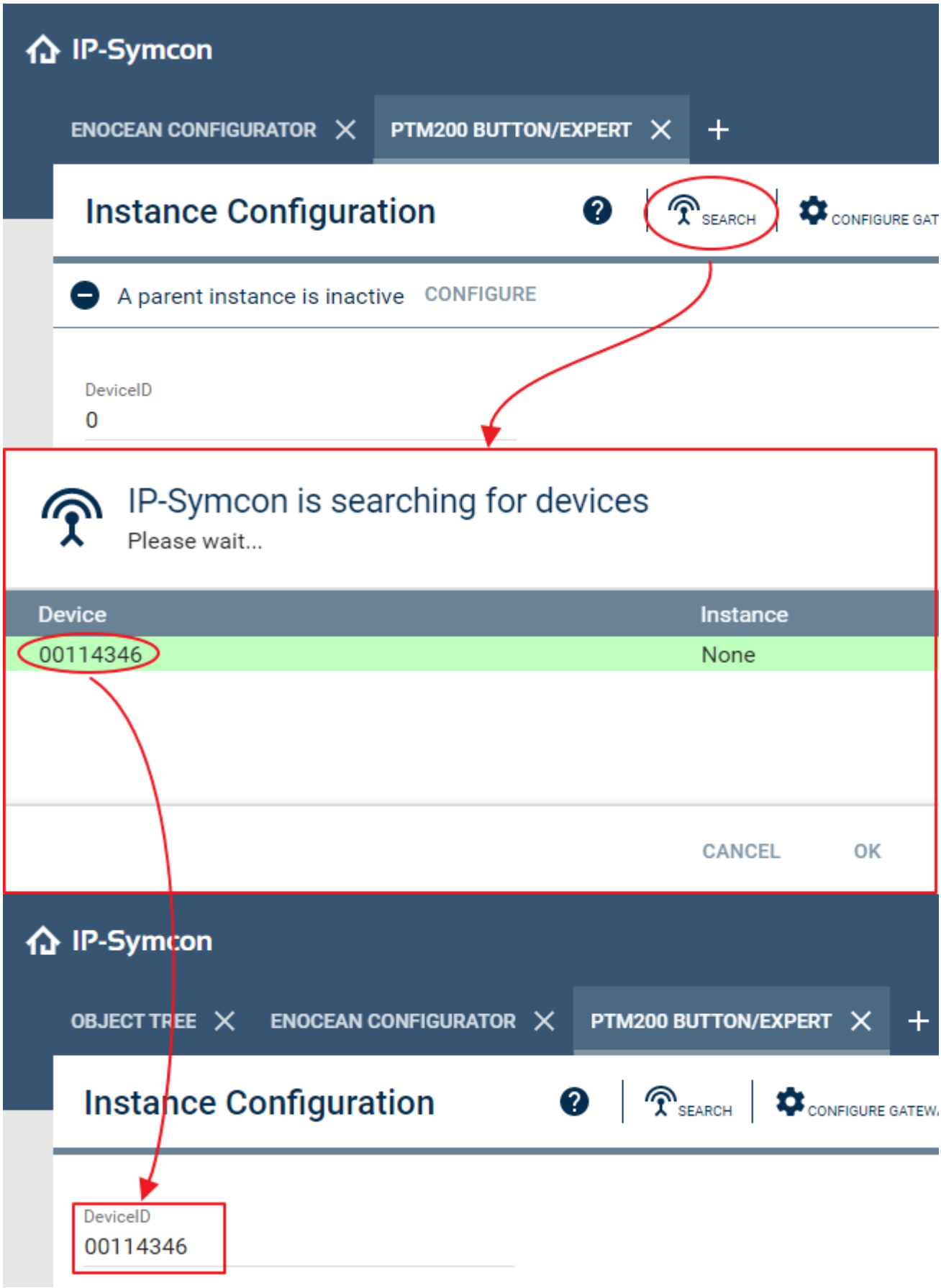
A general tutorial for creating devices/instances in IP-Symcon can be found here: Integrate Devices

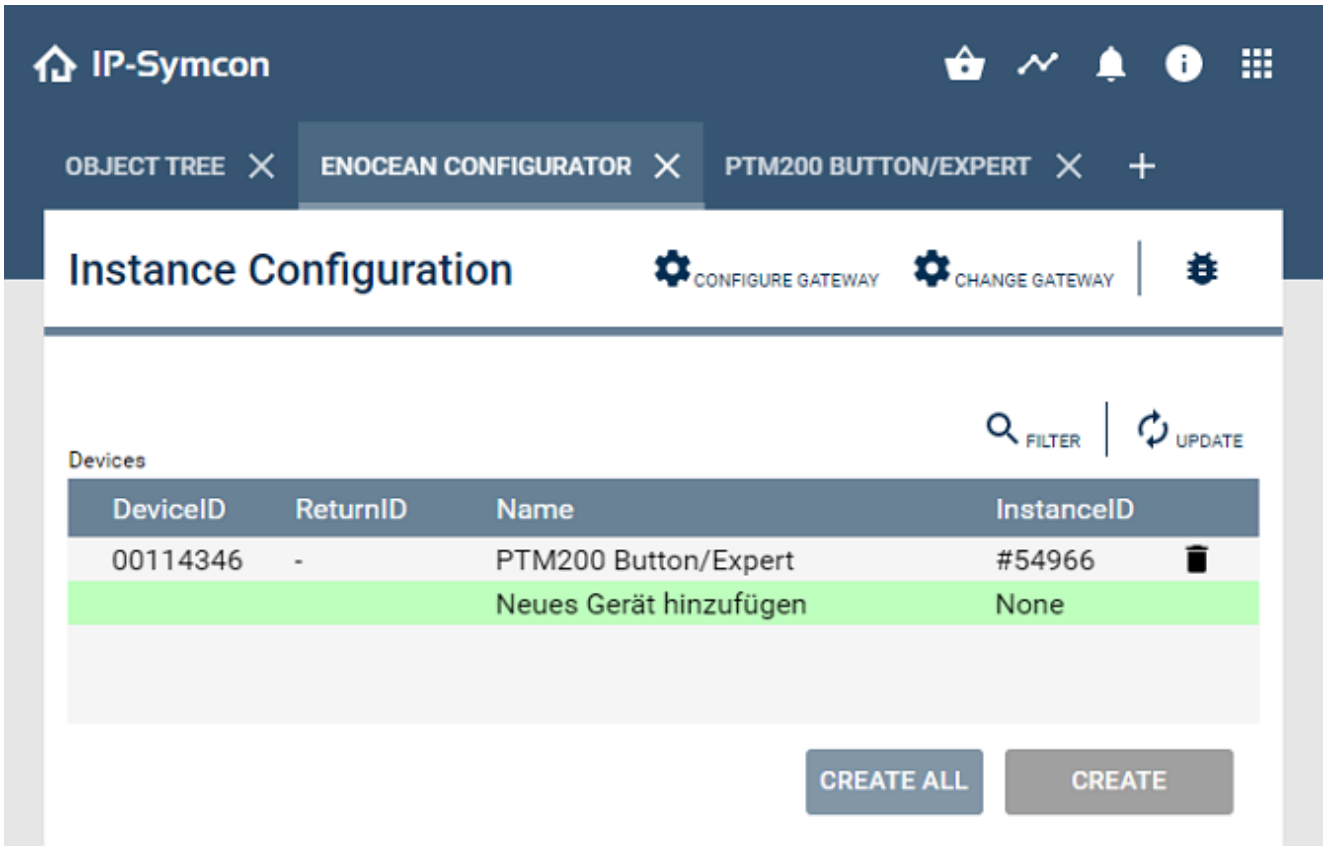
CONFIGURATOR

The configurator can create any EnOcean device as instance. This is done by selecting "New device", clicking "Create", and selecting the according device from the list.



After the configurator has created the instance and the configuration button was pressed, the configuration for the device instance opens. Here, the Device ID can be searched and selected by pressing the "Search" button and multiple confirmations on the EnOcean device.


IP-Symcon
ENOCEAN CONFIGURATOR
PTM200 BUTTON/EXPERT
Instance Configuration
SEARCH
CONFIGURE GAT
A parent instance is inactive
CONFIGURE
DeviceID
0
IP-Symcon is searching for devices
Please wait...
Device
Instance
00114346
None
CANCEL
OK
IP-Symcon
OBJECT TREE
ENOCEAN CONFIGURATOR
PTM200 BUTTON/EXPERT
Instance Configuration
SEARCH
CONFIGURE GATEW
DeviceID
00114346



New instances are created in the main category in the Object Tree. These created instances can be renamed and positioned at another location. It is also possible to open the corresponding instance configurations via "Configure" in the configurator.

ACTOR

Enter Device ID and confirm with "Apply"
Put device into teach in mode (see manufacturer manual)
Click "Teach In"
Eventually test functionality with On/Off in the test environment

ID	DESCRIPTION
Device ID	Provided by the user. 0-127 in whole numbers possible

ACTOR (ELTAKO)

Assign an ID to the devices via login at the FAM14 (see manufacturer manual)
Enter Device ID and confirm with "Apply"
It can be seen easiest via the PCT14
Put device into teach in mode (see manufacturer manual)
Click "Teach In" in the IP-Symcon instance
Eventually test functionality with On/Off in the test environment
Click "Search". The device should switch On/Off five times
Select found ID and confirm with "OK"

Special handling for certain devices

FUD14: In PCT14, in the configuration area, the options "Confirmation Telegram with Dim Value" and "Confirmation Telegram with Button Telegram" must be set to "On". If not inserted

by the teach in, the function 32 "Dim Value from GFVS / Rotary Switch" must be configured in function group 3 of the tab "ID - Allocation Area" of the device.
FSR14-xR: Must be teached in as directional button or universal button, depending on whether the button should switch On/Off or switch with two buttons.
FSB14: If not inserted by the teach in, the function 32 "Drive Time with predefined time value FVS" must be configured in the function group of the tab "ID - Allocation Area" of the PCT14.



While the connection between FAM14 and PCT14 is active via USB, wireless telegrams cannot be sent.



For Eltako devices, the switch actor must be taught in in directional button mode. The dim actor must be taught in in the mode P,L,C (also PCT). This is realized by the especially available instance and the Teach In button. (since IP-Symcon 2.5)

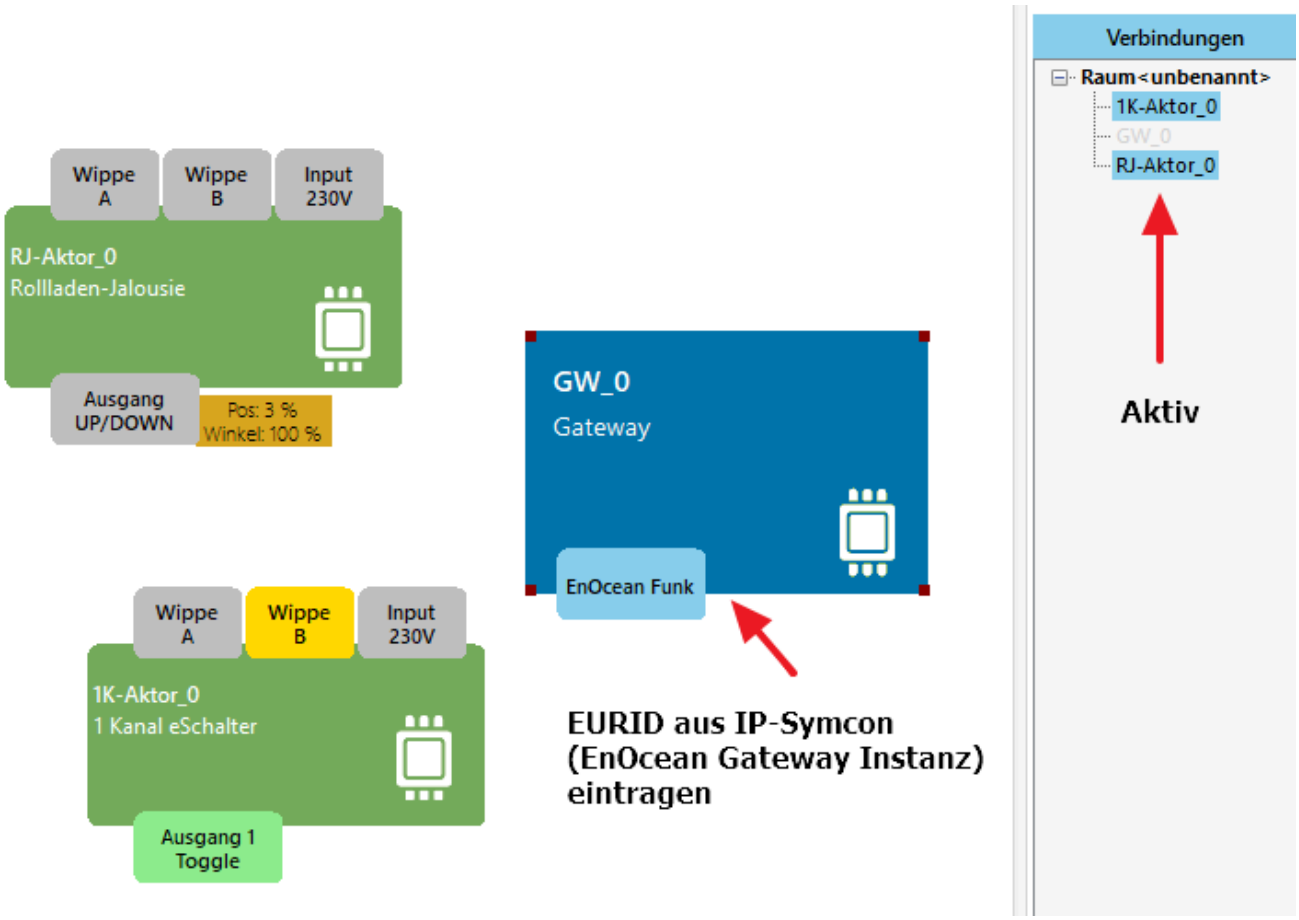


The device Eltako FSB14 can only be taught in in RV 180/200 (Motor 1/2).

ID	DESCRIPTION
Device ID	Provided by the user. 0-127 in whole numbers possible
Report ID	Sent by the device. Used for device allocation

ACTOR (OPUS)

The OPUS Config Tool is required in a current version, so a generic "Gateway" can be added.
In the gateway, the EURID of the used EnOcean gateway needs to be entered. (It is found below the EnOcean Gateway Splitter instance) [For older TCM310 it is eventually required to enter the BaseID]
"Connect" device in gateway [It is highlighted in blue then] and let the device program
Add devices manually via "Add Instance" in IP-Symcon and enter the known Device ID in the instance
Switching and receiving should be possible effortlessly



ID	DESCRIPTION
Device ID	8 digit EnOcean ID. It is found on the device

ACTOR (ACTUATOR)

Enter Device ID and confirm with "Apply"
Click "Search"
Put device into teach in mode (see manufacturer manual)
Device confirms teach in (see manufacturer manual)
Select found ID, confirm with "OK" and "Apply"

ID	DESCRIPTION
Device ID	Provided by the user. 0-127 in whole numbers possible
Report ID	Sent by the device. Used for device allocation

SENSOR

Click "Search"
Wait for data receival (evntually press button)
Select found ID and confirm with "OK"

The search dialog display a device only after it has received data. For a button, it can be pressed to call a sending process. For temperature sensors, waiting for the next interval or entering the ID manually is required.

ID	DESCRIPTION
Device ID	Sent by the device. Used for device allocation

TIPS & TRICKS

To eliminate reception problems or to bridge large distances, maximum two repeater of type "TCM 120" can be used.
The manual for changing the Base ID (not to be confused with the Device ID) can be found here (German).

Additional information about the used technology and sources of supply are available on online presence of EnOcean.
Link: enocean.com/en

ENO_DimSet

```
bool ENO_DimSet (int $InstanceID, int $Intensity)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	1..100

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** to value **Intensity**.



This function can only be used with Eltako dimmer instances.

EXAMPLE

```
ENO_DimSet(12345, 10); //Device dimmed to 10%
```

ENO_SetActiveMessage

```
bool ENO_SetActiveMessage (int $InstanceID, int $Message)
```

PARAMETERS

InstanceID	ID of the instance of the device to be switched
Message	0..8

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Activates a message with the value **Message** on the instance with the ID **InstanceID**.



This function can only be used with EnOcean Thanos instances.

EXAMPLE

```
ENO_SetActiveMessage(12345, 2);
```

ENO_SetFanStage

```
bool ENO_SetFanStage (int $InstanceID, int $FanStage)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
FanStage	0..4

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the FanStage of the device with the ID **InstanceID** to the value **Fanstage**.

FANSTAGE VALUE	DESCRIPTION
0	off
1	low
2	medium
3	high
4	automatic



This function can only be used with EnOcean Thanos instances.

EXAMPLE

```
ENO_SetFanStage(12345, 4); //Device is set to automatic
```

ENO_SetIntensity

```
bool ENO_SetIntensity (int $InstanceID, bool $Status, int $Intensity)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off
Intensity	1..20, 1 = 5%, 20 = 100%

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** to value **Intensity**. The value **Status** indicates whether the device is to be switched on/ off.



Since the EnOcean system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the interface is not connected).



The EnOcean protocol does not provide dimming. The Opus dimmer is controlled by a “Trick” over a time function, so that the result of the brightness can vary a little depending on the operation. The percentages are only approximate and no linear increase in brightness will be accepted.



This function can only be used with the Opus dimmer instance.

EXAMPLE

```
ENO_SetIntensity(12345, true, 10); //Device dimmed to about 50%
```

ENO_SetLockFanStage

```
bool ENO_SetLockFanStage (int $InstanceID, bool $Locked)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Locked	TRUE for on, FALSE for off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Locks the FanStage of the device with the ID InstanceID.



This function can only be used with EnOcean Thanos instances.

EXAMPLE

```
ENO_SetLockFanStage(12345, true);
```

ENO_SetLockRoomOccupancy

```
bool ENO_SetLockRoomOccupancy (int $InstanceID, bool $Locked)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Locked	TRUE for on, FALSE for off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Locks the occupancy of the device with the ID InstanceID.



This function can only be used with EnOcean Thanos instances.

EXAMPLE

```
ENO_SetLockRoomOccupancy(12345, true);
```

ENO_SetMode

```
bool ENO_SetMode (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Value	0..5

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the mode of the actuator with the ID **InstanceID** to the value **Value**.

VALUE-VALUE	DESCRIPTION
0	Default
1	Summer Mode
2	TestClose
3	TestOpen
4	Liftset
5	Init

EXAMPLE

```
ENO_SetMode(12345, 1); //Actuator is set to summer mode
```

ENO_SetPosition

```
bool ENO_SetPosition (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Value	1..100

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the position of the actuator with the ID **InstanceID** to the value **Value**



The actuator mode must be set to "Valve position" so that this function can be executed.

EXAMPLE

```
ENO_SetPosition(12345, 10);
```

ENO_SetRoomOccupancy

```
bool ENO_SetRoomOccupancy (int $InstanceID, bool $Occupied)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Occupied	TRUE for on, FALSE for off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sets the occupancy of the device with the ID InstanceID to the value Occupied.



This function can only be used with EnOcean Thanos instances.

EXAMPLE

```
ENO_SetRoomOccupancy(12345, true);
```

ENO_SetTemperature

```
bool ENO_SetTemperature (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Value	1...100

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the target temperature value of the actuator with the ID **InstanceID** to the value **Value**.



The actuator mode must be set to "Temperature Setpoint" so that this function can be called.



If an EnOcean Thanos device is to be set to a target temperature, the command ENO_SetTemperature1 must be used.

EXAMPLE

```
ENO_SetTemperature(12345, 22.7);
```

ENO_SetTemperature1

```
bool ENO_SetTemperature1 (int $InstanceID, float $Temperature)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Temperature	1..100

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the temperature value of the device with the ID **InstanceID** to the value **Temperature**.



This function can only be used with EnOcean Thanos instances. If an actuator is to be set to a target temperature, the command ENO_SetTemperature must be used.

EXAMPLE

```
ENO_SetTemperature1(12345, 10);
```

ENO_ShutterMoveDown

```
bool ENO_ShutterMoveDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Moves the roller shutter with the ID InstanceID down until ENO_ShutterStop is executed or the end position is reached.

EXAMPLE

```
ENO_ShutterMoveDown(12345);
```

ENO_ShutterMoveDownEx

Require: Symcon >= 4.2

```
bool ENO_ShutterMoveDownEx (int $InstanceID, float $Runtime)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Runtime	How long the roller shutter should run for in seconds. (0 - 6535.5)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the roller shutter with the ID InstanceID down until ENO_ShutterStop is executed or the runtime **Runtime** has expired.



This function is only supported by Eltako shutters

EXAMPLE

```
//Move the roller shutter down for 10.5 seconds
ENO_ShutterMoveDownEx(12345, 10.5);
```

ENO_ShutterMoveUp

```
bool ENO_ShutterMoveUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Moves the roller shutter with the ID InstanceID up until ENO_ShutterStop is executed or the end position is reached.

EXAMPLE

```
ENO_ShutterMoveUp(12345);
```

ENO_ShutterMoveUpEx

Require: Symcon >= 4.2

```
bool ENO_ShutterMoveUpEx (int $InstanceID, float $Runtime)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched
Runtime	How long the roller shutter should run for in seconds.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the roller shutter with the ID InstanceID down until ENO_ShutterStop is executed or the runtime **Runtime** has expired.



This function is only supported by Eltako shutters

EXAMPLE

```
//Move the roller shutter up for 10.5 seconds
ENO_ShutterMoveUpEx(12345, 10.5);
```

ENO_ShutterStop

```
bool ENO_ShutterStop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Stops a current movement of the roller shutter.

EXAMPLE

```
ENO_ShutterStop(12345);
```

ENO_SwitchMode

```
bool ENO_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with ID **InstanceID** to value **Status**



Since the EnOcean system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the interface is not connected).

EXAMPLE

```
ENO_SwitchMode(12345, true); //Turn on device
```

ENO_SwitchModeEx

```
bool ENO_SwitchModeEx (int $InstanceID, bool $Status, int $SendMode)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for On, FALSE for Off
SendMode	0 = NMessage, 1 = UMessage, 2 = Both (Like ENO_SwitchMode)

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the device with ID **InstanceID** to value **Status**



Since the EnOcean system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the interface is not connected).



For explanation of NMessage and UMessage you should check out the EnOcean documentation.

EXAMPLE

```
ENO_SwitchModeEx(12345, true, 0); //Send NMessage
IPS_Sleep(500); //Warten
ENO_SwitchModeEx(12345, true, 1); //Send UMessage
```

Device list

SUPPORTED GATEWAYS

PRODUCT	DESCRIPTION
EnOcean LAN Gateway (TCM310/TCM515 LAN Gateway) Product information	IP Gateway
BSC-BOR	USB Gateway
EnOcean USB300	USB Gateway
EnOcean Pi	Raspberry Pi plug-in board
EnOcean FGW14 (with FAM14)	USB/serial Gateway (operating mode 6) and Radio extension FAM14 (operating mode 2)
FAM USB	USB Gateway (must be entered as TCM 130)
TCM130	All devices that support the ESP2 protocol
TCM310	All devices that support the ESP3 protocol (Dolphin)
TCM515	All devices that support the ESP3 protocol (Dolphin)
Thermokon STC-Ethernet	IP Gateway

SUPPORTED DEVICES

The following devices have been examined for their compatibility with IP-Symcon.
On request we will be happy to check other devices that are not listed.

GENERAL

PRODUCT	DESCRIPTION
PTM 200	Button/Expert
PTM 200	Switch
RCM 1xx	Switching actuator (universal)
STM100	Sensor
STM250	Sensor

ALPHAEOS

PRODUCT	DESCRIPTION
Drive B2	Radio actuator
SENSE TF-H	Indoor climate sensor (temperature and humidity)

ELTAKO

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
DSZ14DRS	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
DSZ14WDRS	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
F2L14	RS485 bus actuator 2-stage ventilation relay	stand-alone module
F3Z14D	RS485 bus electricity meter collector	EEP: A5-12-01
F4HK14	4-channel heating-cooling relay	stand-alone module
F4T65	Wireless sensors without batteries or wires	EEP: F6-02-01
FABH65S	Outdoor motion/brightness sensor	stand-alone module
FADS60-230V	Wireless actuator for outdoor twilight sensor	stand-alone module
FAE14LPR	Individual room control heating/cooling for 2 zones	stand-alone module
FAE14SSR	Individual room control heating/cooling for 2 zones with solid-state relays	stand-alone module
FAFT60	Outdoor humidity temperature sensor	stand-alone module
FAH60	Outdoor brightness sensor	stand-alone module
FAH60B	Outdoor brightness sensor	stand-alone module, EEP: A5-06-01
FAH63	Outdoor brightness sensor	stand-alone module
FAH65S	Outdoor brightness sensor	stand-alone module
FASM60	Wireless outdoor transmitter module	EEP: F6-02-01
FBH65B	Motion/brightness sensor	stand-alone module
FBH65S	Motion/brightness sensor	stand-alone module
FBH65TFB	Motion/brightness sensor	stand-alone module
FCO2TF65	CO2 sensor	EEP: A5-09-04
FD62NP-230V	Wireless universal dimming actuator without N-connection	stand-alone module
FD62NPN-230V	Wireless universal dimming actuator	stand-alone module
FFR14	RS485 bus actuator field release switch	stand-alone module

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
FFR61-230V	Field release switch	stand-alone module
FHF	Wireless sensor window handle	EEP: F6-10-00
FHK14	Wireless actuator for heating/cooling relays	stand-alone module
FHK61SSR-230V	Wireless actuator for heating/cooling relays	stand-alone module
FHK61U-230V	Wireless actuator for heating relay	stand-alone module
FIFT65S	Indoor humidity temperature sensor	stand-alone module
FIH65S	Outdoor brightness sensor	stand-alone module
FIH65Bv	Indoor brightness sensor	EEP: A5-06-02
FIUS65	Indoor wireless flush-mounted signal generator	stand-alone module
FJ62/12-36V DC	Wireless blind and roller shutter actuator	stand-alone module
FJ62NP-230V	Wireless blind and roller shutter actuator	stand-alone module
FKC	Wireless card switch	stand-alone module
FKF	Wireless card switch	stand-alone module
FL62-230V	Wireless light actuator 10A/250V AC	stand-alone module
FL62NP-230V	Wireless light actuator 10A/250V AC	stand-alone module
FLC61-230V	Wireless light controller	stand-alone module
FMS14	RS485 bus actuator multifunction impulse switching relay	stand-alone module
FMS61NP-230V	Wireless actuator multifunction impulse switch	stand-alone module
FMZ14	RS485 bus actuator multifunction time relay	stand-alone module
FMZ61-230V	Wireless actuator multifunction time relay	stand-alone module
FR62-230V	Wireless relay actuator 10 A/250 V AC	stand-alone module
FR62NP-230V	Wireless relay actuator 10 A/250 V AC	stand-alone module
FRGBW71L	Dimming actuator	stand-alone module
FRW	Wireless sensor smoke alarm device	stand-alone module
FSB14	Actuator for shading elements and roller shutters	stand-alone module
FSB61	Wireless actuator for shading elements and roller shutters	stand-alone module
FSB71	Wireless actuator for shading elements and roller shutters	stand-alone module

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
FSDG14	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
FSG14/1-10V	Dimming actuator	stand-alone module
FSG71/1-10V	Dimming actuator	stand-alone module
FSM14	Radio transmitter module	EEP: F6-02-01
FSM60B	4-channel input	EEP: A5-30-03
FSM61	Wireless 2-way transmitter module	EEP: F6-01-01
FSR14SSR	RS485 bus actuator 2-channel impulse switching relay noiseless	stand-alone module
FSR14-2x	RS485 bus actuator 2-channel impulse switching relay	stand-alone module
FSR14-4x	RS485 bus actuator 4-channel impulse switching relay	stand-alone module
FSR61-230V	Wireless actuator for impulse switching relays	stand-alone module
FSR61/8-24V	Wireless actuator for impulse switching relays	stand-alone module
FSR61LN-230V	Wireless actuator for impulse switching relays	stand-alone module
FSR61NP-230V	Wireless actuator for impulse switching relays	stand-alone module
FSR61VA	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
FSR61VA-10A	Wireless actuator for 10A impulse switching relays	stand-alone module
FSR71	Wireless actuator for 2-channel impulse switching relays	stand-alone module
FSS12	Wireless electricity meter transmitter module	EEP: A5-12-01
FSSA-230V	Wireless actuator socket switch actuator	stand-alone module
FSU14	RS485 bus display timer	EEP: A5-13-04
FSU65D	Wireless sensor timer with display	EEP: F6-01-01
FSUD-230V	Dimming actuator	stand-alone module
FSVA-230V	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
FT-FTSB	Tip wireless humidity/temperature sensor	EEP: A5-04-02
FT4F	Wireless sensor surface wireless pushbutton	EEP: F6-02-01
FT55	Wireless sensor surface wireless pushbutton	EEP: F6-02-01
FTF65S	Temperature sensor	EEP: A5-02-05

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
FTK	Wireless window/door contact	EEP: D5-00-01
FTKB	Wireless window/door contact	EEP: D5-00-01
FTKE	Wireless window/door contact	EEP: F6-10-00
FTN14	RS485 bus actuator staircase lighting overrun switch	stand-alone module
FTN61NP-230V	Wireless actuator for staircase lighting overrun switch	stand-alone module
FTR65DS	Wireless sensor temperature controller	stand-alone module
FTR65HS	Wireless sensor temperature controller	stand-alone module
FTR78S	Wireless temperature controller with rotary knob	EEP: A5-10-03
FUD14	Dimming actuator	stand-alone module
FUD14-800W	Dimming actuator	stand-alone module
FUD61NP	Dimming actuator	stand-alone module
FUD61NPN	Dimming actuator	stand-alone module
FUD70S-230V	Dimming actuator	stand-alone module
FUD71	Dimming actuator	stand-alone module
FUTH55D	Wireless sensor clock thermo-hygrostat	stand-alone module
FUTH65D	Wireless sensor clock thermo-hygrostat	stand-alone module
FWS61	Wireless weather data transmitter module	EEP: A5-13-01, A5-13-02
FWWKW71L	PWM Dimming actuator for LED	stand-alone module
FWZ12	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
FWZ14	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
FWZ61	Wireless single-phase energy meter transmitter module	EEP: A5-12-01
FZK14	RS485 bus actuator time relay for card switches/smoke alarm devices	stand-alone module
FZK61NP-230V	Wireless actuator for time relays with card switch/smoke alarm device	stand-alone module
FZS	Wireless sensor pull switch	stand-alone module
FMMS44SB	Wireless mini multisensor	EEP: D2-14-41

HOPPE

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
FHF	Window handle	F6-10-00
SecuSignal	eWindow handle Amsterdam E0400/FR-408 F69	F6-10-00
SecuSignal	eWindow handle Atlanta 0530S/FR-408 100 Nm F9	F6-10-00
SecuSignal	eWindow handle Toulon E0400/FR-408 F69	F6-10-00
ConnectHome	eWindow handle Tokyo	F6-10-00
ConnectHome	eWindow handle New York	F6-10-00

HORA

PRODUCT	DESCRIPTION
SmartDrive MX	Actuator

KIEBACK & PETER

PRODUCT	DESCRIPTION
MD15	Small actuator

MACO

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
Smart Ready Sensor eTRONIC	Door/window sensor	EEP: A5-14-01

NODON

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
ASP-2-1-00	Smart Plug	EEP: D2-01-0A
ASP-2-1-01	Smart Plug Metering	EEP: D2-01-0B
ASP-2-1-10	Smart Plug	EEP: D2-01-0A
ASP-2-1-11	Smart Plug Metering	EEP: D2-01-0B
SIN-2-1-01	In-wall Multifunction Relay Switch 1-Channel	EEP: D2-01-0F
SIN-2-2-01	In-wall ON/OFF Lighting Relay Switch 2-Channel	EEP: D2-01-0B

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
SIN-2-RS-01	Roller Shutter Relay Switch	EEP: D2-05-00

OPUS

PRODUCT	DESCRIPTION	EEP NUMBER (SEE EEP LIST BELOW)
514.441	55 Wall transmitter module standard	EEP: F6-02-01
561.150.9052	Temperature sensor (2 buttons)	EEP: A5-10-12
561.150.9069	Temperature sensor (4 buttons)	EEP: A5-10-22
563.010-C*	Bridge 1 channel	EEP: D2-01-01
563.014-C*	Bridge 1 channel, 16A	EEP: D2-01-01
563.020-C*	Bridge 2 channel	EEP: D2-01-11
563.031-C*	Bridge roller shutter/ blind	EEP: D2-05-02
563.035-C*	Bridge dimmer	EEP: D2-01-03
563.051-C	Smart Motion Sensor (light)	EEP: F6-02-01
563.052-C	Smart motion sensor multifunction	EEP: F6-02-01
563.053	Smart Motion Sensor Ambient Assisted Living - Inability to move	EEP: A5-07-03
563.055-C	Smart Motion Sensor (presence)	EEP: A5-07-03
563.067-C	Water alarm (instant alarm)	EEP: F6-05-01
563.068-C	Water alarms (source disks)	EEP: A5-30-03
563.069-C	Smoke alarm device	EEP: F6-05-02

*Requires setup via Opus Config Tool (563.040)

PEHA

PRODUCT	DESCRIPTION
Various	All devices that support EEP

THERMOKON

PRODUCT	DESCRIPTION
SR04	Temperature/humidity
SAB02	Actuator
Thanos	Room control unit

EEP

All devices that use the following EEPs are supported by IP-Symcon.

ACTUATORS

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-20-12	4.0	07-20-12	Temperature Controller Input

BI-DIRECTIONAL

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-20-01	4.0	07-20-01	Battery Powered Actuator
D2-01-00	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-01	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-02	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-03	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-04	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-05	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-06	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-07	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-08	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-09	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-0A	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-0B	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-0C	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-0D	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-0E	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-0F	4.3		Electronic switches and dimmer with Energy Measurement and Local Content

NEW	SINCE VERSION	OLD	DESCRIPTION
D2-01-10	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-11	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-12	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-13	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-01-14	4.3		Electronic switches and dimmer with Energy Measurement and Local Content
D2-03-00	5.5		Light, Switching + Blind Control Type 0x00
D2-03-0A	5.5		Push Button – Single Button
D2-03-10	5.5		Mechanical Handle
D2-05-00	5.3		Blinds Control for Position/Angle Type 0x00
D2-05-01	5.3		Blinds Control for Position/Angle Type 0x01
D2-05-02	5.3		Blinds Control for Position/Angle Type 0x02
D2-06-01	4.3		Multisensor Window Handle - Alarm, Position Sensor, Vacation Mode, Optional Sensors
D2-07-00	5.5		Locking Systems Control - Mortise lock
D2-11-01	5.3		Room Operating Panel (BI-DIR) Type 0x01
D2-11-02	5.3		Room Operating Panel (BI-DIR) Type 0x02
D2-11-03	5.3		Room Operating Panel (BI-DIR) Type 0x03
D2-11-04	5.3		Room Operating Panel (BI-DIR) Type 0x04
D2-11-05	5.3		Room Operating Panel (BI-DIR) Type 0x05
D2-11-06	5.3		Room Operating Panel (BI-DIR) Type 0x06
D2-11-07	5.3		Room Operating Panel (BI-DIR) Type 0x07
D2-11-08	5.3		Room Operating Panel (BI-DIR) Type 0x08
D2-15-00	5.5		People Activity Counter

NEW	SINCE VERSION	OLD	DESCRIPTION
D2-A0-01	5.5		Valve Control (BI-DIR)

SENSORS

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-02-01	4.0	07-02-01	Temperature Sensor Range -40°C to 0°C
A5-02-02	4.0	07-02-02	Temperature Sensor Range -30°C to +10°C
A5-02-03	4.0	07-02-03	Temperature Sensor Range -20°C to +20°C
A5-02-04	4.0	07-02-04	Temperature Sensor Range -10°C to +30°C
A5-02-05	4.0	07-02-05	Temperature Sensor Range 0°C to +40°C
A5-02-06	4.0	07-02-06	Temperature Sensor Range +10°C to +50°C
A5-02-07	4.0	07-02-07	Temperature Sensor Range +20°C to +60°C
A5-02-08	4.0	07-02-08	Temperature Sensor Range +30°C to +70°C
A5-02-09	4.0	07-02-09	Temperature Sensor Range +40°C to +80°C
A5-02-0A	4.0	07-02-0A	Temperature Sensor Range +50°C to +90°C
A5-02-0B	4.0	07-02-0B	Temperature Sensor Range +60°C to +100°C
A5-02-10	4.0	07-02-10	Temperature Sensor Range -60°C to +20°C
A5-02-11	4.0	07-02-11	Temperature Sensor Range -50°C to +30°C
A5-02-12	4.0	07-02-12	Temperature Sensor Range -40°C to +40°C
A5-02-13	4.0	07-02-13	Temperature Sensor Range -30°C to +50°C
A5-02-14	4.0	07-02-14	Temperature Sensor Range -20°C to +60°C
A5-02-15	4.0	07-02-15	Temperature Sensor Range -10°C to +70°C
A5-02-16	4.0	07-02-16	Temperature Sensor Range 0°C to +80°C

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-02-17	4.0	07-02-17	Temperature Sensor Range +10°C to +90°C
A5-02-18	4.0	07-02-18	Temperature Sensor Range +20°C to +100°C
A5-02-19	4.0	07-02-19	Temperature Sensor Range +30°C to +110°C
A5-02-1A	4.0	07-02-1A	Temperature Sensor Range +40°C to +120°C
A5-02-1B	4.0	07-02-1B	Temperature Sensor Range +50°C to +130°C
A5-02-20	4.0	07-02-20	10 Bit Temperature Sensor Range -10°C to +41.2°C
A5-02-30	4.0	07-02-30	10 Bit Temperature Sensor Range -40°C to +62.3°C
A5-04-01	4.0	07-04-01	Range 0°C to +40°C and 0% to 100%
A5-04-02	4.1	07-04-02	Range -20°C to +60°C and 0% to 100%
A5-04-03	4.1	07-04-03	Range -20°C to +60°C 10bit-measurement and 0% to 100%
A5-04-03	5.3	07-04-03	Temperature and Humidity Sensor
A5-06-01	4.0	07-06-01	Range 300lx to 60.000lx
A5-06-02	4.0	07-06-02	Range 0lx to 1.020lx
A5-06-03	4.0	07-06-03	10-bit measurement (1-Lux resolution) with range 0lx to 1000lx
A5-07-01	4.0	07-07-01	Occupancy with Supply voltage monitor
A5-07-01	5.3	07-07-01	gN-Deckenbewegungsmelder 360°
A5-07-01	5.3	07-07-01	Desk Occupancy
A5-07-02	4.0	07-07-02	Occupancy with Supply voltage monitor
A5-07-03	4.0	07-07-03	Occupancy with Supply voltage monitor and 10-bit illumination measurement
A5-08-01	4.0	07-08-01	Range 0lx to 510lx
A5-08-01	4.0	07-08-01	0°C to +51°C and Occupancy Button

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-08-02	4.0	07-08-02	Range 0lx to 1020lx
A5-08-02	4.0	07-08-02	0°C to +51°C and Occupancy Button
A5-08-03	4.0	07-08-03	Range 0lx to 1530lx
A5-08-03	4.0	07-08-03	-30°C to +50°C and Occupancy Button
A5-09-01	4.0	07-09-01	CO Sensor (not in use)
A5-09-02	4.0	07-09-02	CO-Sensor 0 ppm to 1020 ppm
A5-09-04	4.0	07-09-04	CO2 Sensor
A5-09-04	5.3	07-09-04	CO2, Temperature and Humidity Sensor
A5-09-05	4.0	07-09-05	VOC Sensor
A5-09-06	4.0	07-09-06	Radon
A5-09-07	4.0	07-09-07	Particles
A5-09-08	4.0	07-09-08	Pure CO2 Sensor
A5-10-01	4.0	07-10-01	Temperature Sensor
A5-10-01	4.0	07-10-01	Set Point
A5-10-01	4.0	07-10-01	Fan Speed and Occupancy Control
A5-10-02	4.0	07-10-02	Temperature Sensor
A5-10-02	4.0	07-10-02	Set Point
A5-10-02	4.0	07-10-02	Fan Speed and Day/Night Control
A5-10-03	4.0	07-10-03	Temperature Sensor
A5-10-03	4.0	07-10-03	Set Point Control
A5-10-04	4.0	07-10-04	Temperature Sensor
A5-10-04	4.0	07-10-04	Set Point and Fan Speed Control

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-10-05	4.0	07-10-05	Temperature Sensor
A5-10-05	4.0	07-10-05	Set Point and Occupancy Control
A5-10-06	4.0	07-10-06	Temperature Sensor
A5-10-06	4.0	07-10-06	Set Point and Day/Night Control
A5-10-07	4.0	07-10-07	Temperature Sensor
A5-10-07	4.0	07-10-07	Fan Speed Control
A5-10-08	4.0	07-10-08	Temperature Sensor
A5-10-08	4.0	07-10-08	Fan Speed and Occupancy Control
A5-10-09	4.0	07-10-09	Temperature Sensor
A5-10-09	4.0	07-10-09	Fan Speed and Day/Night Control
A5-10-0A	4.0	07-10-0A	Temperature Sensor
A5-10-0A	4.0	07-10-0A	Set Point Adjust and Single Input Contact
A5-10-0B	4.0	07-10-0B	Temperature Sensor and Single Input Contact
A5-10-0C	4.0	07-10-0C	Temperature Sensor and Occupancy Control
A5-10-0D	4.0	07-10-0D	Temperature Sensor and Day/Night Control
A5-10-10	4.0	07-10-10	Temperature and Humidity Sensor
A5-10-10	4.0	07-10-10	Set Point and Occupancy Control
A5-10-11	4.0	07-10-11	Temperature and Humidity Sensor
A5-10-11	4.0	07-10-11	Set Point and Day/Night Control
A5-10-12	4.0	07-10-12	Temperature and Humidity Sensor and Set Point

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-10-13	4.0	07-10-13	Temperature and Humidity Sensor
A5-10-13	4.0	07-10-13	Occupancy Control
A5-10-14	4.0	07-10-14	Temperature and Humidity Sensor
A5-10-14	4.0	07-10-14	Day/Night Control
A5-10-15	4.0	07-10-15	10 Bit Temperature Sensor
A5-10-15	4.0	07-10-15	6 bit Set Point Control
A5-10-16	4.0	07-10-16	10 Bit Temperature Sensor
A5-10-16	4.0	07-10-16	6 bit Set Point Control;Occupancy Control
A5-10-17	4.0	07-10-17	10 Bit Temperature Sensor
A5-10-17	4.0	07-10-17	Occupancy Control
A5-10-18	4.0	07-10-18	Illumination
A5-10-18	4.0	07-10-18	Temperature Set Point
A5-10-18	4.0	07-10-18	Temperature Sensor
A5-10-18	4.0	07-10-18	Fan Speed and Occupancy Control
A5-10-19	4.0	07-10-19	Humidity
A5-10-19	4.0	07-10-19	Temperature Set Point
A5-10-19	4.0	07-10-19	Temperature Sensor
A5-10-19	4.0	07-10-19	Fan Speed and Occupancy Control
A5-10-1A	4.0	07-10-1A	Supply voltage monitor
A5-10-1A	4.0	07-10-1A	Temperature Set Point
A5-10-1A	4.0	07-10-1A	Temperature Sensor

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-10-1A	4.0	07-10-1A	Fan Speed and Occupancy Control
A5-10-1B	4.0	07-10-1B	Supply Voltage Monitor
A5-10-1B	4.0	07-10-1B	Illumination
A5-10-1B	4.0	07-10-1B	Temperature Sensor
A5-10-1B	4.0	07-10-1B	Fan Speed and Occupancy Control
A5-10-1C	4.0	07-10-1C	Illumination
A5-10-1C	4.0	07-10-1C	Illumination Set Point
A5-10-1C	4.0	07-10-1C	Temperature Sensor
A5-10-1C	4.0	07-10-1C	Fan Speed and Occupancy Control
A5-10-1D	4.0	07-10-1D	Humidity
A5-10-1D	4.0	07-10-1D	Humidity Set Point
A5-10-1D	4.0	07-10-1D	Temperature Sensor
A5-10-1D	4.0	07-10-1D	Fan Speed and Occupancy Control
A5-10-1E	5.5	07-10-1E	Temperature Sensor, Supply Voltage, Light, Fan Speed, and Occupancy Control
A5-10-1F	4.0	07-10-1F	Temperature Sensor
A5-10-1F	4.0	07-10-1F	Set Point
A5-10-1F	4.0	07-10-1F	Fan Speed

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-10-1F	4.0	07-10-1F	Occupancy and Unoccupancy Control
A5-10-20	4.0	07-10-20	Temperature and Set Point with Special Heating States
A5-10-21	4.0	07-10-21	Temperature
A5-10-21	4.0	07-10-21	Humidity and Set Point with Special Heating States
A5-10-22	5.5	07-10-22	Temperature Sensor, Set Point, Humidity, and Fan Speed
A5-10-23	5.5	07-10-23	Temperature Sensor, Set Point, Humidity, Fan Speed, and Occupancy Control
A5-11-01	4.0	07-11-01	Lighting Controller
A5-11-02	4.0	07-11-02	Temperature Controller Output
A5-11-03	4.0	07-11-03	Blind Status
A5-11-04	4.0	07-11-04	Extended Lighting Status
A5-12-00	4.0	07-12-00	Counter
A5-12-00	5.3	07-12-00	Operations Counting
A5-12-01	4.0	07-12-01	Electricity
A5-12-02	4.0	07-12-02	Gas
A5-12-03	4.0	07-12-03	Water
A5-13-01	4.0	07-13-01	Weather Station
A5-13-02	4.0	07-13-02	Sun Intensity
A5-13-03	4.0	07-13-03	Date Exchange
A5-13-04	4.0	07-13-04	Time and Day Exchange
A5-13-05	4.0	07-13-05	Direction Exchange
A5-14-01	4.0	07-14-01	Single Input Contact (Window/Door)
A5-14-01	4.0	07-14-01	Supply voltage monitor

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-14-02	4.0	07-14-02	Single Input Contact (Window/Door)
A5-14-02	4.0	07-14-02	Supply voltage monitor and Illumination
A5-14-03	4.0	07-14-03	Single Input Contact (Window/Door)
A5-14-03	4.0	07-14-03	Supply voltage monitor and Vibration
A5-14-04	4.0	07-14-04	Single Input Contact (Window/Door)
A5-14-04	4.0	07-14-04	Supply voltage monitor
A5-14-04	4.0	07-14-04	Vibration and Illumination
A5-14-05	4.0	07-14-05	Vibration/Tilt
A5-14-05	4.0	07-14-05	Supply voltage monitor
A5-14-05	5.3	07-14-05	Chair Occupancy Solar Sensor
A5-14-06	4.0	07-14-06	Vibration/Tilt
A5-14-06	4.0	07-14-06	Illumination and Supply voltage monitor
A5-14-07	5.0	07-14-07	Dual door contact with States (Open/Closed/Locked/Unlocked)
A5-14-07	5.0	07-14-07	Supply voltage monitor
A5-14-08	5.0	07-14-08	Dual door contact with States (Open/Closed/Locked/Unlocked)
A5-14-08	5.0	07-14-08	Supply voltage monitor and Vibration detection
A5-14-09	5.0	07-14-09	Window/Door Sensor with States (Open/Closed/Tilt)
A5-14-09	5.0	07-14-09	Supply voltage monitor
A5-14-0A	5.0	07-14-0A	Window/Door Sensor with States (Open/Closed/Tilt)
A5-14-0A	5.0	07-14-0A	Supply voltage monitor and Vibration detection
A5-20-01	5.3	07-14-01	Heating Valve Controller

NEW	SINCE VERSION	OLD	DESCRIPTION
A5-20-06	5.5	07-20-06	Heating Valve Controller
A5-30-01	4.0	07-30-01	Single Input Contact
A5-30-01	4.0	07-30-01	Battery Monitor
A5-30-02	4.0	07-30-02	Single Input Contact
A5-30-03	4.0	07-30-03	4 Digital Inputs
A5-30-03	4.0	07-30-03	Wake and Temperature
A5-30-04	4.0	07-30-04	3 Digital Inputs
A5-30-04	4.0	07-30-04	1 Digital Input 8 Bits
A5-37-01	4.0	07-37-01	Demand Response
A5-38-08	4.0	07-38-08	Gateway
A5-38-09	4.0	07-38-09	Extended Lighting-Control
D2-14-30	6.0		Sensor for Smoke, Air quality, Hygrothermal comfort, Temperature and Humidity
D2-14-41	6.4		Sensor for Humidity, Illumination, Temperature, Dissolved oxegen, Contact, Acceleration
D2-15-00	5.4		People Activity Counter
D2-32-00	4.3		A.C. Current Clamp
D2-32-00	5.3		Single Channel CT Clamp
D2-32-01	4.3		A.C. Current Clamp
D2-32-02	4.3		A.C. Current Clamp
D5-00-01	4.0	06-00-01	Single Input Contact
D5-00-01	5.3	06-00-01	Magnet Contact
F6-01-01	5.5	05-01-01	Push Button
F6-02-01	4.0	05-02-01	Light and Blind Control - Application Style 1

NEW	SINCE VERSION	OLD	DESCRIPTION
F6-02-02	4.0	05-02-02	Light and Blind Control - Application Style 2
F6-02-03	4.0	05-02-03	Light Control - Application Style 1
F6-02-04	4.0	05-02-04	Light and blind control ERP2
F6-03-01	4.0	05-03-01	Light and Blind Control - Application Style 1
F6-03-02	4.0	05-03-02	Light and Blind Control - Application Style 2
F6-04-01	4.0	05-04-01	Key Card Activated Switch
F6-04-02	4.0	05-04-02	Key Card Activated Switch ERP2
F6-05-00	5.0	05-05-00	Wind Speed Threshold Detector
F6-05-01	4.0	05-05-01	Liquid Leakage Sensor (mechanic harvester)
F6-05-02	5.0	05-05-02	Smoke Detector
F6-10-00	4.1	05-10-00	Window Contact Sensor
F6-10-01	5.5	05-10-01	Window Handle ERP2

FHZ1X00PC

The FHZ1X00PC (WLAN) is an interface developed by Contronics and sold by ELV that allows the end user to communicate with FS20, HMS and FHT components via PC. From the FHZ1300PC also the combined weather sensor KS300 is supported. The FHZ1350PC is only on Contronics available and provides the ability to access the dialer of the ELV. This feature is not supported by IP-Symcon.
The FS20 system provides an affordable entry into the building technology and automation. Parts of it are also sold under other labels (e.g. Conrad) - but are 100 percent compatible.

LIST OF SUPPORTED DEVICES

FS20 sender and receiver
All FS20 sender and receiver
HMS sensors and actuators
All HMS alarm sensors and actuators
FHT components
FHT80B/FHT80B2 (FHT8 is not compatible!)

INSTALLATION

Once you have connected the device to your PC, you must install the drivers. If you have installed this, you can download it on our website: Download Drivers

TIPS & TRICKS

The FHZ is compatible with all USB-> LAN converters. (e.g. Silex, Lantronix)
Information to the change to LAN or to extend the range, see in the forum.
The FHZ1300PC WLAN is not officially supported in IP-Symcon. In order to use it in IP-Symcon, the encryption algorithm must be disabled.
The necessary steps you can read here:
http://www.thinkwiki.org/wiki/User:Akw/FHZ

PROBLEM SOLVING

The configuration says: "A parent configuration seems to be broken"
Click on the messages and activate the correct driver.
No data is being transmitted/ received
Verify that the FHZ is connected to your system. (Hardware Manager)
Verify that by sending/ receiving the red LED flashes on the FHZ.
If not, the FHZ has no power or has not been configured in the system. Check whether the FTDI driver has been activated properly. (see above)
Devices are connected only sporadically (FS20)
Is the unit too far away?
Try to place the FHZ somewhere else or expand your system by another FHZ.
Is there jammer nearby?

FHT

The radio heating system FHT is ideally suited to save energy through intelligent control of the heating valves and to create a comfortable indoor climate at different times.

Only FHT8b/FHT80b can be connected to the FHZ1000PC Professional, FHZ1300PC or FHZ1300PC WLAN. The much cheaper FHT8R are only suitable for single-mode and can not be connected to the FHZ.

Weekly and daily programs can/ are not be transferred to the FHT with IP Symcon. This feature has been requested many a time, but it will definitely **not** given.
Link: Feature-Request

CONFIGURATION

Register FHT at FHZ:
Press the key "PROG" until "Sond" is displayed.
Turn the wheel until "Code" appears, then press the "Function" button.
Now you have the opportunity to select a code from a pair of numbers (00-99).
Confirm by pressing the "PROG" button.
Press the "PROG" button again.
Set FHT in IPSymcon:
Create FHT instance. See integrate devices
Enter the code in the property page and click the "Apply" button.

If the room controller was probably already registered to a central station, this application must be deleted.

Press the key "PROG" until "Sond" is displayed.
Turn the wheel until "Cent" appears, then press the "PROG" button.
Turn the wheel until "nA" appears und press “PROG”.
Wait about 15 minutes for safety until the logout process has been completed.
Perform the described registration process.

With the option "Emulate status" the value of the target variable is updated immediately after sending the command without waiting for the actual feedback from the FHT.

After the registration process must be at least one command (e.g. set target temperature) be sent to the FHT to start the communication.

AVAILABLE STATUS VARIABLES

STATUS	DESCRIPTION
LowBattery	Indicates whether the FHT displays a low battery warning. If the value is TRUE, you should change the battery and then manually reset the value to FALSE.
Window Open	Indicates whether the window contact which is connected to the FHT reports an open window. TRUE = window open, FALSE = window closed
Position	Value of the control valve from 0 to 100%
Temperature	Actual temperature received from the FHT
Target Mode	Received mode from the FHT
Beim Auslesen	The integer variable uses the following values: 0: Automatic, 1: Manual; 2: Vacation; 3: Party
Target Mode (Pending)	The mode sent to the FHT. Once the radio command could be discontinued, "Target Mode" and "Target Mode (Pending)" are identical.
Target Temperature	Target temperature received from the FHT
Target Temperature (Pending)	The target temperature sent to the FHT. Once the radio command could be discontinued, are "Target Temperature" and "Target Temperature (Pending)" identical.

FHT_SetMode

```
bool FHT_SetMode (int $InstanceID, float $Mode)
```

PARAMETERS

InstanceID	ID of the device to be switched
Mode	**0** = Automatic, **1** = Manual

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the new desired mode for the FHT with ID **InstanzID** to value **Mode**.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).



The FHT commands are buffered by the FHZ and sent every few minutes to the FHT. Please note that the transmit buffer is limited, and repeatedly sending of values increases the transmission time, and possibly the FHT overloaded.

EXAMPLE

```
FHT_SetMode(12345, 1); //Switch to manual mode
```

FHT_SetTemperature

```
bool FHT_SetTemperature (int $InstanceID, float $Temperature)
```

PARAMETERS

InstanceID	ID of the device to be switched
Temperature	Temperature value in 0.5 ° C increments

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the new desired temperature for the FHT with ID **InstanzID** to value **Temperature**.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).



The FHT commands are buffered by the FHZ and sent every few minutes to the FHT. Please note that the transmit buffer is limited, and repeatedly sending of values increases the transmission time, and possibly the FHT overloaded.

EXAMPLE

```
FHT_SetTemperature(12345, 22.5); //Set to 22.5 ° C
```

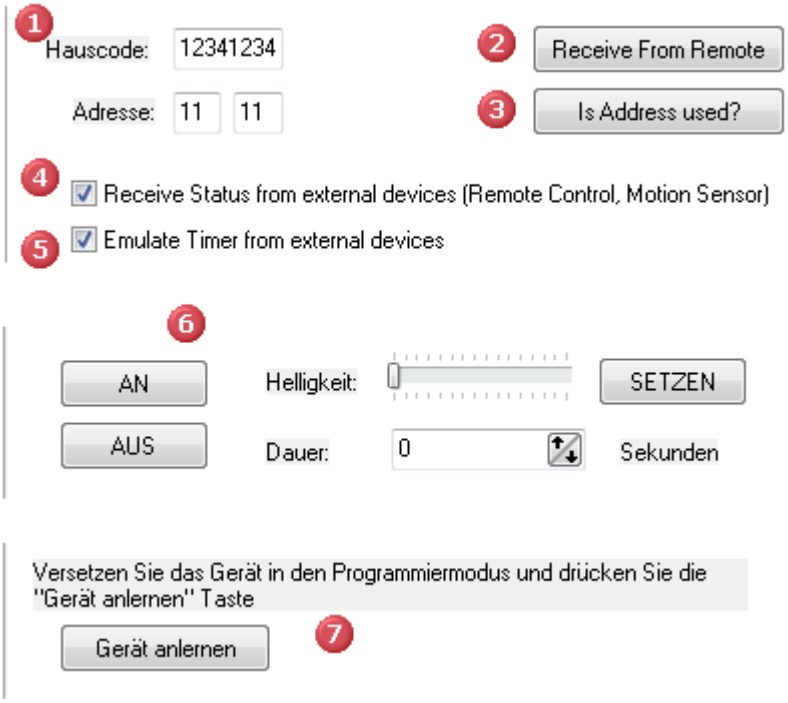
FS20

When creating the instance you have three choices for the FS20 system:

INSTANCENAME	PURPOSE
FS20 Receiver	Sockets, dimmers, canopy control, …
FS20 Sender	Motion sensor, …
FS20 Remote control	Multi-channel remote controls, push-button, transmitter

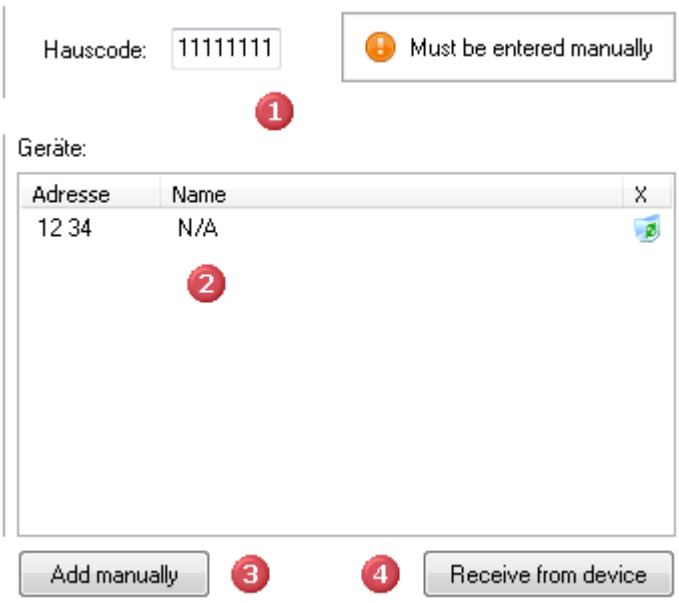
If you have not set up your device, please follow the steps on this page: embed devices

FS20 RECEIVER / CONFIGURATE SENDER



You can enter and take over the house code manually. Possible characters: 1-4
You can receive the house code from another transmitter (e.g. a remote control).
You can check if the address is already in use by another instance.
When activated, the instance receives data from FS20 transmitters and updated accordingly, the state variable.
When activated, the instance receives data from FS20 transmitters, evaluates the timer data and emulates the state of the timer (e.g. ON for 30sec).
Once the new settings have been applied, you can test it by sending on / off telegrams. When transmitting the value in the duration field is rounded to the nearest possible time that can evaluate the FS20 system.
Once the new settings have been applied, you can teach one FS20 receiver (eg switch socket). You must put this in the "learning mode" and program it within the next 15 seconds by pressing the teach button.

FS20 CONFIGURATE REMOTE CONTROL



You must enter the house code by hand before you can continue the configuration. Possible characters: 1-4
The list shows already learned buttons of the remote control.
If you know the address, you can enter it manually. Possible characters: 1-4
If you do not know the address, you can receive it from your remote control. Please note that you must have entered the house code and that it must be saved via the "Apply" button.

AVAILABLE STATUS VARIABLES

VARIABLE	FUNCTION
Status Variable	Displays the status of the device
Intensity Variable	Shows the dim value of the device in %. Note that the values 0-16 are used internal, if you load in the variable in your scripts.
Timer Variable	The time, in seconds, the timer runs. Point 5 must be enabled in the configuration, so this variable is evaluated by IP Symcon. (It appears the total received time - not the time to maturity.)
Data Variable	Displays the internal received status code.

Available status codes:

```
0-16 => 0% – 100%
17 => on, old value (corresponds to value 11 at PIRI)
18 => swap
19 => dim up (Simple channel number)
20 => dim down (Simple channel number)
21 => dim up/down (Double number of channels)
24 => PIRI; off for the ON duration, afterwards on
```

```
25 => PIRI; on, 100% for the ON duration, afterwards off
26 => PIRI; on, old value for the ON duration, afterwards off
30 => PIRI; on, 100% for the ON duration, afterwards old state
31 => PIRI; on, afterwards old state for the ON duration
```

FS20_DimDown

```
bool FS20_DimDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** down a level.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).

EXAMPLE

```
FS20_DimDown(12345); //Dimming the device down
```

FS20_DimUp

```
bool FS20_DimUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** up a level.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).

EXAMPLE

```
FS20_DimUp(12345); //Dimming the device up
```

FS20_SetIntensity

```
bool FS20_SetIntensity (int $InstanceID, int $Intensity, int $Duration)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	0-16 (0=0%, 16=100%)
Duration	Duration of the dimming

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** to value **Intensity** in **Duration** seconds.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).

EXAMPLE

```
FS20_SetIntensity(12345, 16, 5); //Dimming device to 100% in 5 seconds
```

FS20_SwitchDuration

```
bool FS20_SwitchDuration (int $InstanceID, bool $Status, int $Duration)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off
Duration	On/ Off duration in seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with ID **InstanceID** to value **Status** in **Duration** seconds.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).

EXAMPLE

```
FS20_SwitchDuration(12345, true, 60); //Turn on device for 60 seconds
```

FS20_SwitchMode

```
bool FS20_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with ID **InstanceID** to value **Status**.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).

EXAMPLE

```
FS20_SwitchMode(12345, true); //Turn on device
```

Device list

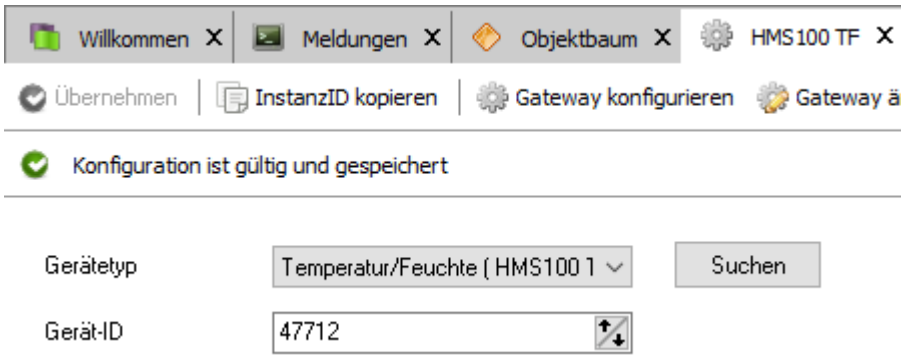
LIST OF SUPPORTED DEVICES

FS20 transmitter and receiver
All FS20 transmitters and receivers
HMS sensors and actuators
All HMS-Alaram sensors and actuators
FHT components
FHT80B/FHT80B2 (FHT8 is not compatible!)

HMS

SETUP IN IP-SYMCON

An HMS instance must be added (Integrate devices). The type can still be changed later on the configuration page.



After "Searching" the device must be selected. A device instance with the appropriate variables is automatically created.
Here using the example of an HMS100 TF.



HMS_ReleaseFI

```
bool HMS_ReleaseFI (int $InstanceID, int $TriggerDelay)
```

PARAMETERS

InstanceID	ID of the device to be switched
TriggerDelay	Delay in seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Triggers the FI over the HMS device with ID **InstanceID**. The **TriggerDelay** parameter is specified in seconds.



Since the FS20 system is unidirectional, can not be determined whether the command was successful. Error messages are only sent when an internal system error occurred (for example, if the FHZ is not connected).



If the FI can not trigger the FI, its status variable is set to **FALSE**.

EXAMPLE

```
FS20_ReleaseFI(12345, 0); //Trigger FI immediatly
```

KS300

INSTALLATION

If you have not set up your device, please follow the steps on this page: Integrate devices

TIPS & TRICKS

It is known that the FHZ1300PC from KS300 receives only irregularly.
A discussion can be traced in the forum. Link

FS10 Weather

The **FS10 radio system** is the predecessor system of the FS20 and operates in the 433 MHz range.
All devices are no longer available in the free trade.
The internal coding allows up to 392 code combinations. To evaluate the components of the FS10 weather WS2000er series with IP-Symcon, the PC weather sensor from ELV is needed. This is simply connected to the serial port and therefore can existing weather components be integrated easily in the building management system. To increase the quality of reception, the installation of Superhet receiver is recommended.

INSTALLATION

For the weather receiver no special installation is required. Only a connection to the serial port is necessary.

CONFIGURATION

Before you start searching, you still need to configure the parent instance by specifying the correct com port and apply the configuration. The Setting of the baud rate ect. is not necessary, because it is being made by the system automatically.

The search dialog shows a device only when data has been received. With a remote control, you can press the button to display a broadcast transmission. With a temperature sensor, you should wait for the next interval or specify the ID manually.

Once you specify an ID, you must apply the configuration. Thus also the corresponding state variables are created automatically.

GARDENA smart system

Require: Symcon >= 6.0

DESCRIPTION

The GARDENA module enables supported devices from GARDENA smart system to be integrated into IP-Symcon.

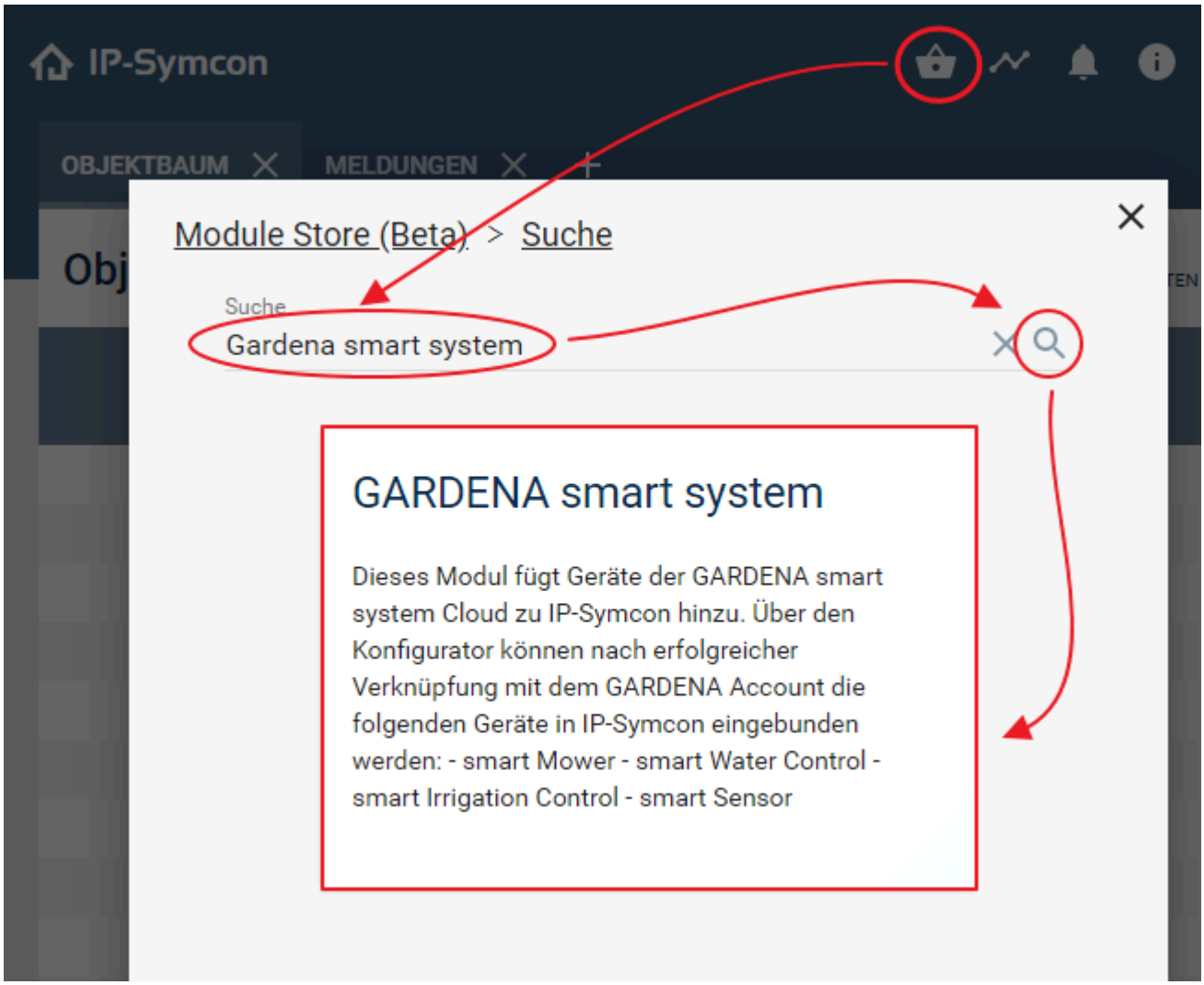
INTEGRATION IN IP-SYMCON

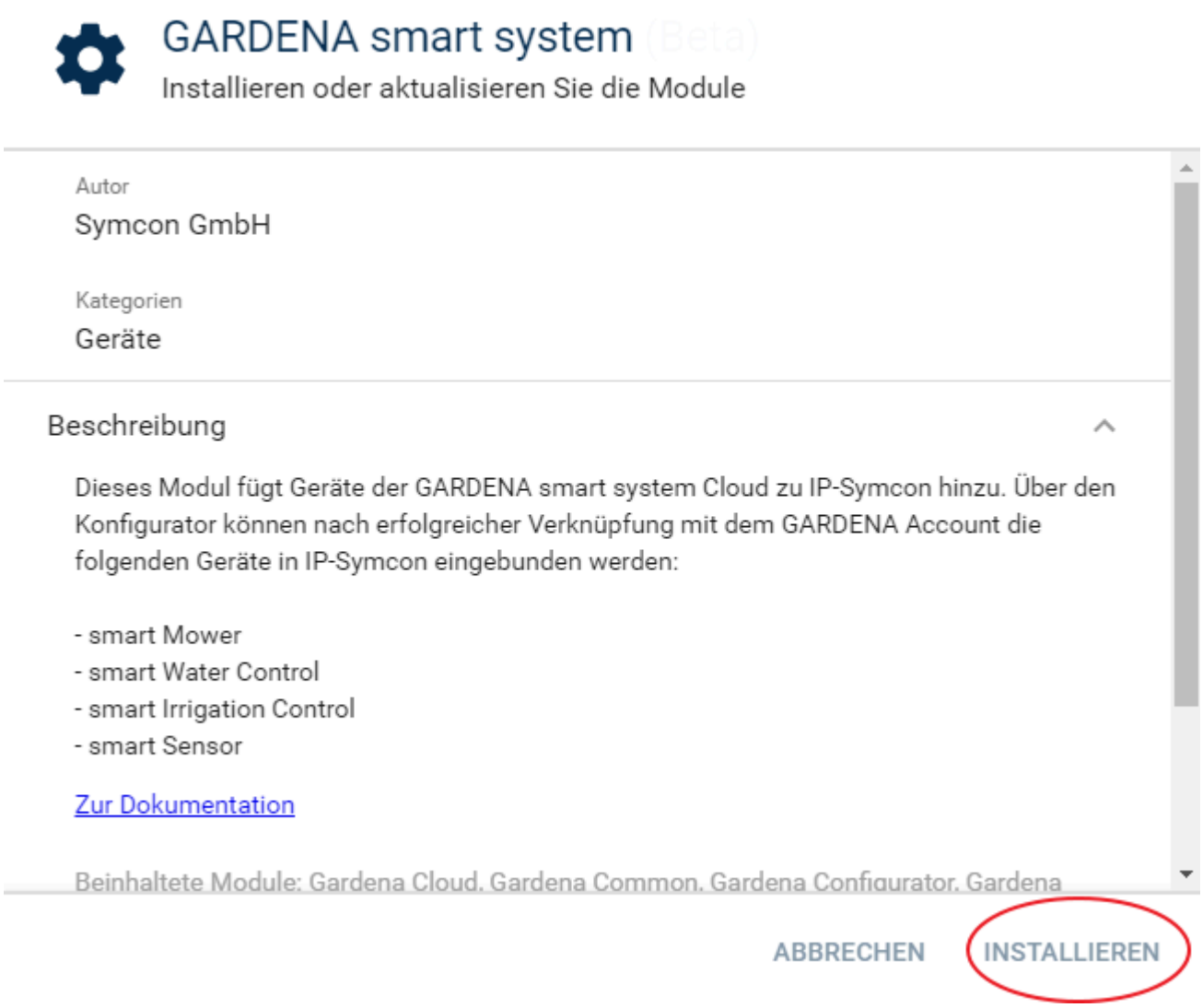


The use of the GARDENA module requires an activated connection to the Connect service which, in turn, requires an active subscription.

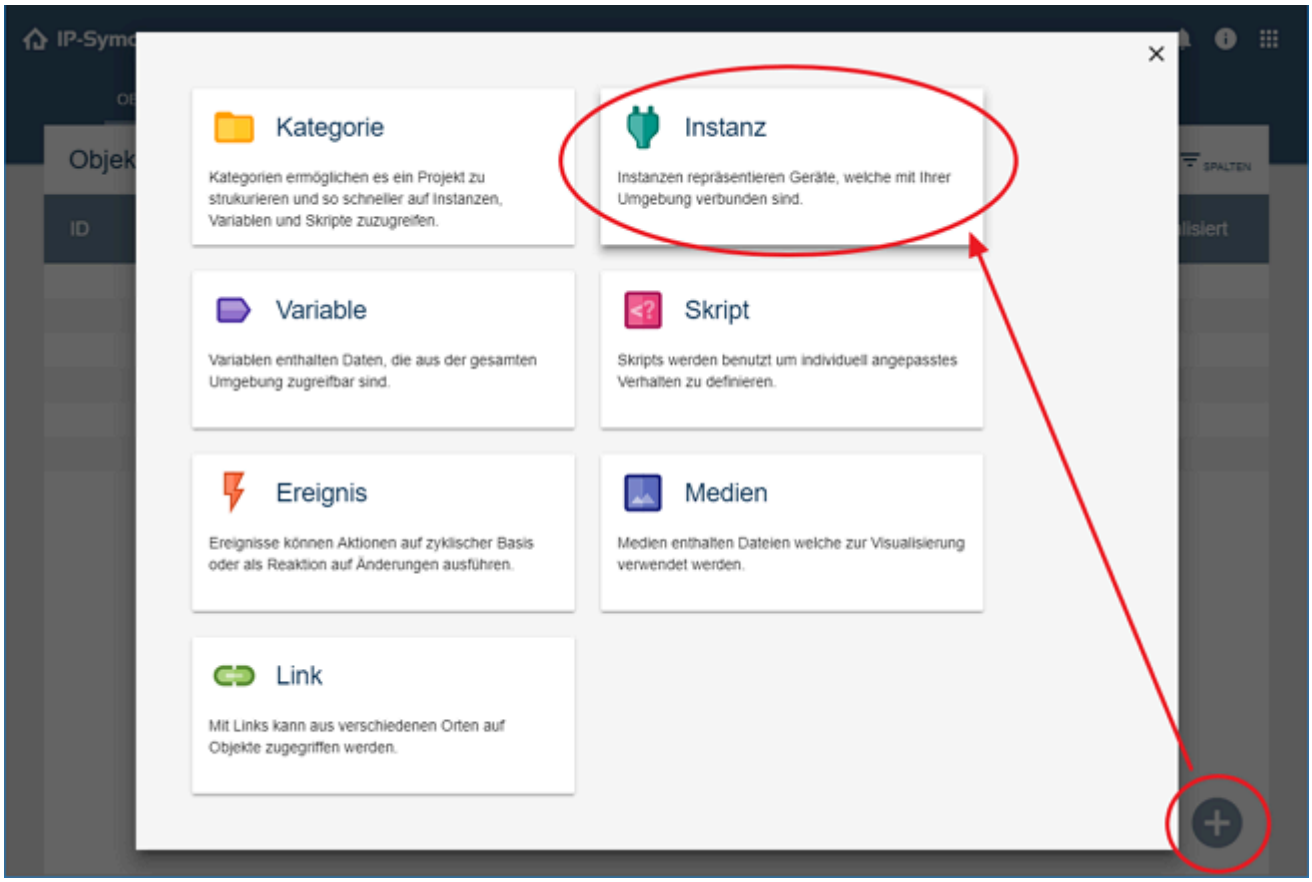
CREATE GARDENA CONFIGURATOR INSTANCE

In order for the GARDENA module to be used, it must first be installed via the Module Store. The Module Store must be opened for this. It is located in the upper right area. The Gardena smart system module can be found via the search field by entering "Gardena smart system". When opening the found module, the installation of the module can be initiated in the following dialog via the "Install" button.

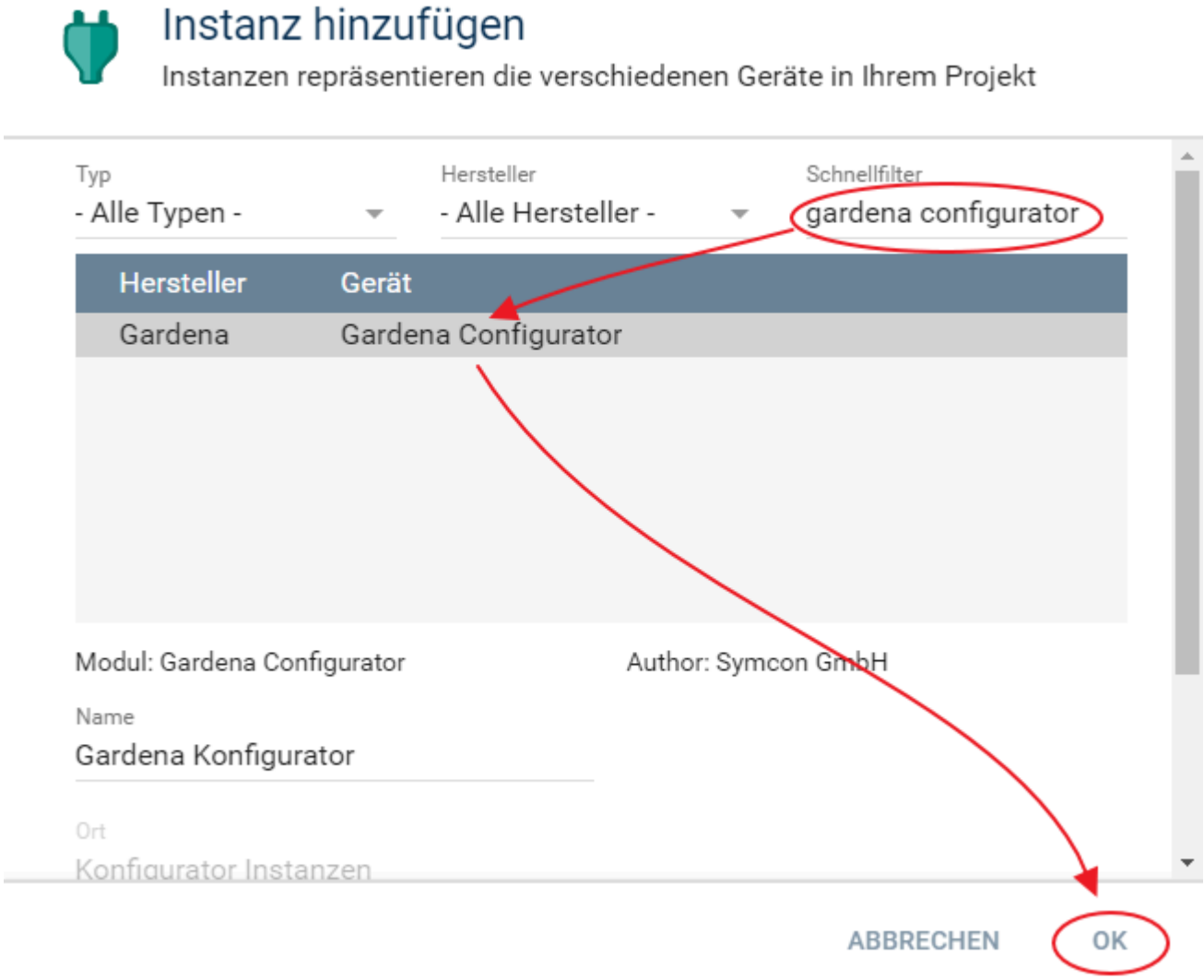

IP-Symcon
OBJEKTBAUM
MELDUNGEN
Module Store (Beta) > Suche
Suche
Gardena smart system
GARDENA smart system
Dieses Modul fügt Geräte der GARDENA smart system Cloud zu IP-Symcon hinzu. Über den Konfigurator können nach erfolgreicher Verknüpfung mit dem GARDENA Account die folgenden Geräte in IP-Symcon eingebunden werden: - smart Mower - smart Water Control - smart Irrigation Control - smart Sensor



The add dialog for creating a Gardena configurator instance then opens.
If the instance is to be added manually, an Instance must be created by the Gardena configurator. To do this, the object tree must first be opened. Here, the "+" button at the bottom right must be pressed and Instance selected.



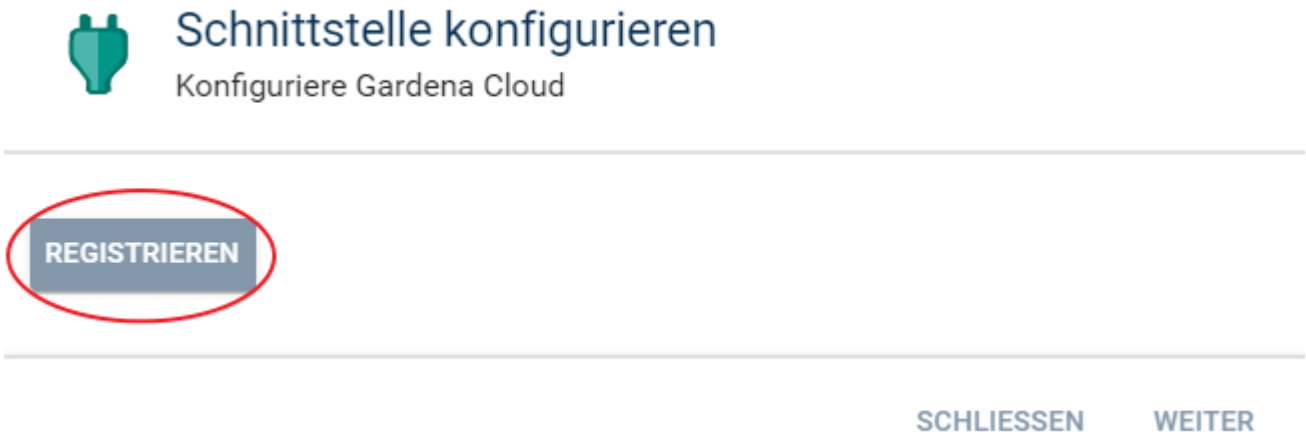
The "Gardena Configurator" from the manufacturer "Gardena" can be found using the quick search. The location should not be changed, the name can be chosen freely. Finally, confirm with "OK".



After the configurator instance has been created, it automatically opens and can be set up.

SET UP GARDENA CONFIGURATOR-INSTANCE

Before devices can be created, they must be registered by clicking on Register in order to connect IP-Symcon to the Husqvarna Group account.



One has to log in with the Husqvarna Group account in order to grant IP-Symcon the necessary authorizations.

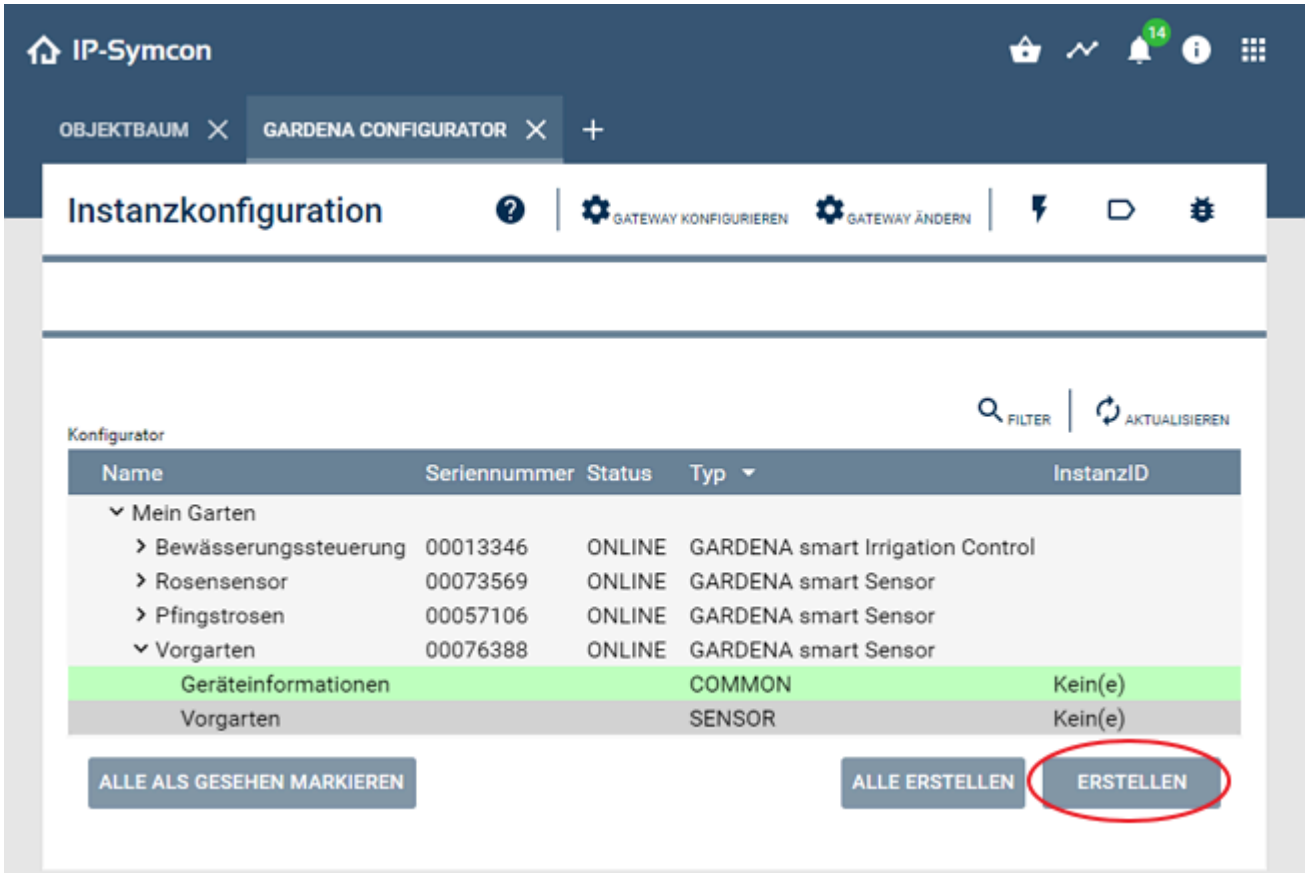




The "Configure Interface" dialog can then be confirmed with Next. Now the created WebSocket Client has to be activated.



Now the configurator opens and all recognized devices can be created with a click on "Create".



TROUBLESHOOTING

VARIABLES ARE NO LONGER UPDATED

The connection to the Husqvarna service can be re-established in the Gardena Cloud instance by clicking on 'Reset WebSocket'.

SUPPORTED DEVICES

smart Irrigation Control
smart Water Control
smart Pump
smart Sensor
smart Mower
smart Power Socket

smart Irrigation Control

Require: Symcon >= 6.0



Since the Gardena API does not transmit the schedules created in the app, irrigation plans can be implemented using the weekly schedule and the actions provided.

DESCRIPTION

The smart irrigation control device offers various services in IP-Symcon which, for example, make it possible to identify the status of the device or switch the various valves.

SERVICES

An instance can be created for each service using the configurator.
In the instance configuration, the 'Show last transmitted' switch can be activated to create a variable for each value, if available, that contains the time of the last data transfer.

SERVICE	DESCRIPTION
General	Contains general information about the device
Valve	Represents one of the 6 valves of the irrigation control
Main valve	The main valve with which all valves can be closed

GENERAL

The following state variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Battery status	String	The status of the device's battery (This device does not have a battery)
RF Link Level	Integer	The quality of the wireless connection in percent
RF Link Status	String	The status of the device's wireless connection

VALVE

The following state variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Activity	String	The current activity of the device

VARIABLE	TYPE	DESCRIPTION
Remaining	Integer	The time remaining in seconds until the valve is closed
Error code	String	The last error transmitted
Status	String	The current state of the device
Open duration	Integer	The duration for which the valve is to be opened manually
Action	String	Allows the valve to be opened or closed for a specific opening period

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Open valve	Duration (minutes)	Allows the valve to be opened for a certain period of time
Close valve	(none)	Allows the valve to close

MAIN VALVE

The following state variables are created for this instance

NAME	TYPE	DESCRIPTION
Error code	String	The last error transmitted
Status	String	The current state of the device
Action	String	Closes all valves

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Close main valve	(none)	Allows the main valve to close

smart SILENO Mower

Require: Symcon >= 6.0

DESCRIPTION

The smart SILENO Mower offers various services in IP-Symcon, which make it possible, for example, to recognize the current activity of the device or to start or stop the mowing process.

SERVICES

For each service, an instance can be created with the configurator.
In the instance configuration, the 'Show last transmitted' switch can be activated to create a variable for each value, if available, that contains the time of the last data transfer.

SERVICE	DESCRIPTION
General	Contains general information about the device
Mower	Represents the SILENO mower

GENERAL

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Battery status	String	The status of the device's battery
RF Link Level	Integer	The quality of the wireless connection in percent
RF Link Status	String	The status of the wireless connection of the device

MOWER

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Activity	String	The current activity of the device
Operating hours	Integer	The time the mower has been active for, in hours
Error code	String	The last error transmitted
Status	String	The current state of the device
Cutting duration	Integer	The time in minutes for manual mowing

VARIABLE	TYPE	DESCRIPTION
Mowing	String	Allows the mower to mow for a specific mowing time or according to a schedule
Parking	String	Allows the mower to drive to the charging station

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Automatic mowing	(none)	Makes the mower mow on schedule
Manual mowing	Duration (minutes)	Makes the mower mow for the specified time
Park until the next task	(none)	Makes the mower drive to the charging station and wait for the next task
Parking and ignore the schedule	(none)	Makes the mower drive to the charging station and ignore the schedule

smart Power Socket

Require: Symcon >= 6.0

DESCRIPTION

The smart Power Socket offers various services in IP-Symcon, which make it possible, for example, to recognize the status of the device or to switch the adapter.

SERVICES

For each service an instance can be created with the configurator.
In the instance configuration, the 'Show last transmitted' switch can be activated to create a variable for each value, if available, that contains the time of the last data transfer.

SERVICE	DESCRIPTION
General	Contains general information about the device
Power socket	Represents the power socket that can be switched

GENERAL

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Battery status	String	The status of the device's battery (This device does not have a battery)
RF Link Level	Integer	The quality of the wireless connection in percent
RF Link Status	String	The status of the wireless connection of the device

POWER SOCKET

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Activity	String	The current activity of the device
Remaining	Integer	The remaining time in seconds until the socket is switched off
Error code	String	The last error transmitted
Status	String	The current state of the device
Activity duration	Integer	The duration for which the socket is to be switched on manually
Action	String	The action which should be carried out

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Switch on the socket	Duration (minutes)	Enables the power socket to be switched on for a defined period
Switch on the power socket permanently	(none)	Allows the socket to be switched on permanently
Switch off the socket	(none)	Allows the socket to be switched off

smart Pressure Pump

Require: Symcon >= 6.0

DESCRIPTION

The smart Pump offers various services in IP-Symcon which, for example, enable the status of the device to be recognized or the valve to be switched.

SERVICES

An instance can be created for each service with the configurator.
In the instance configuration, the 'Show last transmitted' switch can be activated to create a variable for each value, if available, that contains the time of the last data transfer.

SERVICE	DESCRIPTION
General	Contains general information about the device
Valve	Represents the valve of the pump
Main valve	The main valve with which all valves can be closed

GENERAL

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Battery status	String	The status of the device's battery
RF Link Level	Integer	The quality of the wireless connection in percent
RF Link Status	String	The status of the device's wireless connection

VALVE

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Activity	String	The current activity of the device
Remaining	Integer	The time remaining in seconds until the valve is closed
Error code	String	The last error transmitted
Status	String	The current state of the device
Open duration	Integer	The duration for which the valve is to be opened manually

VARIABLE	TYPE	DESCRIPTION
Action	String	Allows the valve to be opened or closed for a specific opening period

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Open valve	Duration (minutes)	Allows the valve to be opened for a certain period of time
Close valve	(none)	Allows the valve to close

MAIN VALVE

The following status variables are created for this instance

NAME	TYPE	DESCRIPTION
Error code	String	The last error transmitted
Status	String	The current state of the device
Action	String	Closes all valves

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Close main valve	(none)	Allows the main valve to close

smart Sensor

Require: Symcon >= 6.0

DESCRIPTION

The smart sensor offers various services in IP-Symcon which, for example, make it possible to read out the different values of the sensors, such as soil moisture or light intensity.

SERVICES

An instance can be created for each service with the configurator.
In the instance configuration, the 'Show last transmitted' switch can be activated to create a variable for each value, if available, that contains the time of the last data transfer.

SERVICE	DESCRIPTION
General	Contains general information about the device
Sensor	Represents the values of the sensor

GENERAL

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Battery status	String	The status of the device's battery
RF Link Level	Integer	The quality of the wireless connection in percent
RF Link Status	String	The status of the wireless connection of the device

SENSOR

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Soil humidity	Integer	The soil moisture measured by the sensor in percent
Soil temperature	Float	The soil temperature measured by the sensor in °C
Light intensity	Float	The illuminance measured by the sensor in lux
Ambient temperature	Float	The ambient temperature measured by the sensor in °C

smart Water Control

Require: Symcon >= 6.0

DESCRIPTION

The smart Water Control offers various services in IP-Symcon, which make it possible, for example, to recognize the status of the device or to switch the valve.

SERVICES

An instance can be created for each service with the configurator.
In the instance configuration, the 'Show last transmitted' switch can be activated to create a variable for each value, if available, that contains the time of the last data transfer.

SERVICE	DESCRIPTION
General	Contains general information about the device
Valve	Represents the valve of the irrigation control
Main valve	The main valve with which all valves can be closed

GENERAL

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Battery status	String	The status of the device's battery
RF Link Level	Integer	The quality of the wireless connection in percent
RF Link Status	String	The status of the device's wireless connection

VALVE

The following status variables are created for this instance

VARIABLE	TYPE	DESCRIPTION
Activity	String	The current activity of the device
Remaining	Integer	The time remaining in seconds until the valve is closed
Error code	String	The last error transmitted
Status	String	The current state of the device
Open duration	Integer	The duration for which the valve is to be opened manually

VARIABLE	TYPE	DESCRIPTION
Action	String	Allows the valve to be opened or closed for a specific opening period

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Open valve	Duration (minutes)	Allows the valve to be opened for a certain period of time
Close valve	(none)	Allows the valve to close

MAIN VALVE

The following status variables are created for this instance

NAME	TYPE	DESCRIPTION
Error code	String	The last error transmitted
Status	String	The current state of the device
Action	String	Closes all valves

The following actions are available for this instance

ACTION	PARAMETERS	DESCRIPTION
Close main valve	(none)	Allows the main valve to close

Geofency

Require: Symcon >= 4.2
The module is used to receive Geofency data.

FUNCTIONAL SCOPE

Each device has its own location list
Username and password identification within IP-Symcon.
Automatically sets up the webhook "/hook/geofency".
It is recommended to use this in combination with the Connect module.
Optionally, the current location of Geofency can be transmitted

REQUIREMENTS

Geofency App for Apple iOS

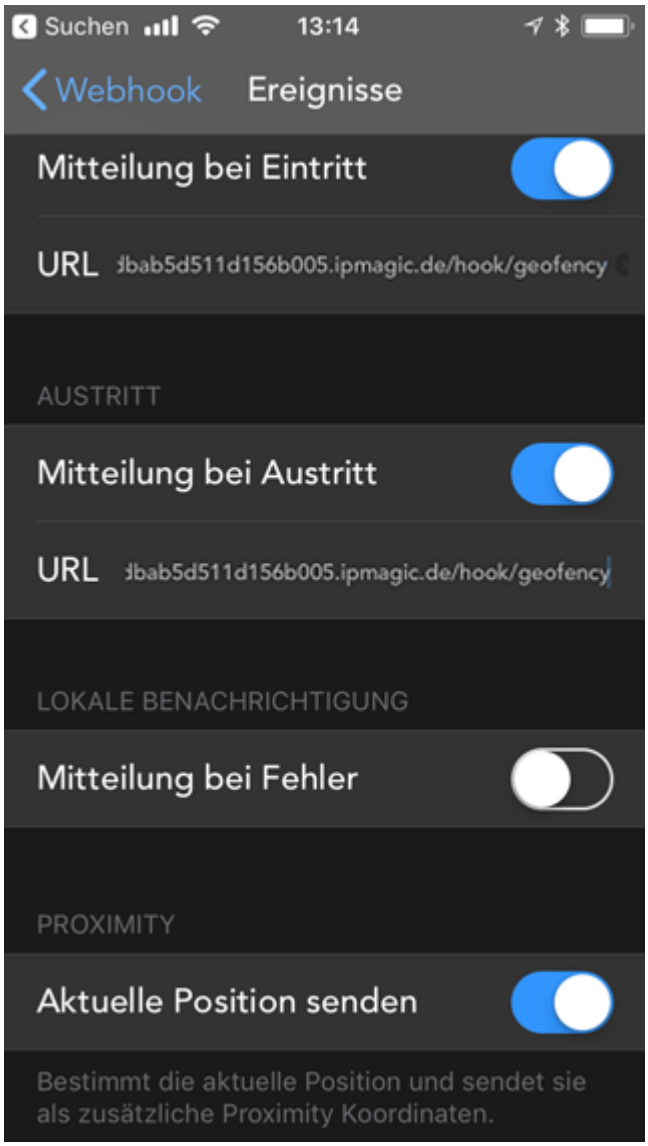
SOFTWARE INSTALLATION

Via the Module Store install the Geofency module.

GEOFENCY CONFIGURATION

In the Geofency app, use + to add a new location. Then click on the 3 dots on the right to open the menu. Here you have to select Webhook.

EVENT



The URL is the domain where IP-Symcon can be reached followed by /hook/geofency
The easiest way is to enter the IP-Symcon Connect address followed by /hook/geofency
Optionally you can activate Send current position.

POST FORMAT

Setting remains on default, JSON-encoded is disabled and not selected

AUTHENTICATION

NAME	HTTP BASIC AUTHENTICATION
Username	The Webhook username which will be stored in IP-Symcon
Password	The Webhook password which will be stored in IP-Symcon

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Geofency' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Username	Username which must be specified in the Geofency App to send data to IP-Symcon.
Password	Password, which must be specified in the Geofency App.

if this data is left empty, anyone can send data to IP-Symcon via the hook

STATUSVARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

The variables are created automatically based on the device ID and when they are sent for the first time within the Geofency module. Multiple devices can run through one hook. Each device is set up under its own "category".

NAME	TYPE	DESCRIPTION
Device Name	Instance (Dummy)	Serves as a "category" containing all monitored locations, as well as the timestamp and longitude/latitude. Created per device.
Latitude	Float	Latitude (center of geofences) of the last activity
Longitude	Float	Longitude (center of geofences) of the last activity
CurrentLatitude	Float	Proximity coordinates, i.e. the current coordinates at entry or exit (outer perimeter of the geofences), if "Send current position" was selected in the app at the webhook
CurrentLongitude	Float	Proximity coordinates, i.e. the current coordinates at entry or exit (outer perimeter of the geofence) if "Send current position" was selected in the app at the webhook
Direction	Integer	Azimuth angle (direction in degrees) of the current entry/exit point relative to the center of the geofence circle, if "Send current position" was selected in the app at the webhook
Distance	Float	Distance (in meters) of the current entry/exit point relative to the center of the geofence circle, if "Send current position" was selected in the app at the webhook
Orientation	Float	Cardinal direction (as name) of the current entry/exit point relative to the center of the geofence circle, if "Send current position" was selected in the app during the webhook
Timestamp	Integer	UnixTimestamp of the last activity.

NAME	TYPE	DESCRIPTION
Example Location (Office)	Boolean	Present or Absent. Information is supplied by Gefency.
current Longitude	Float	current Latitude
current Latitude	Float	current Longitude
Motion	Integer	current motion state like walking, driving etc.
WifiBSSD	String	BSSID of the connected WLAN
WifiSSID	String	SSID of the connected WLAN

Example:



Heating Control

DESCRIPTION

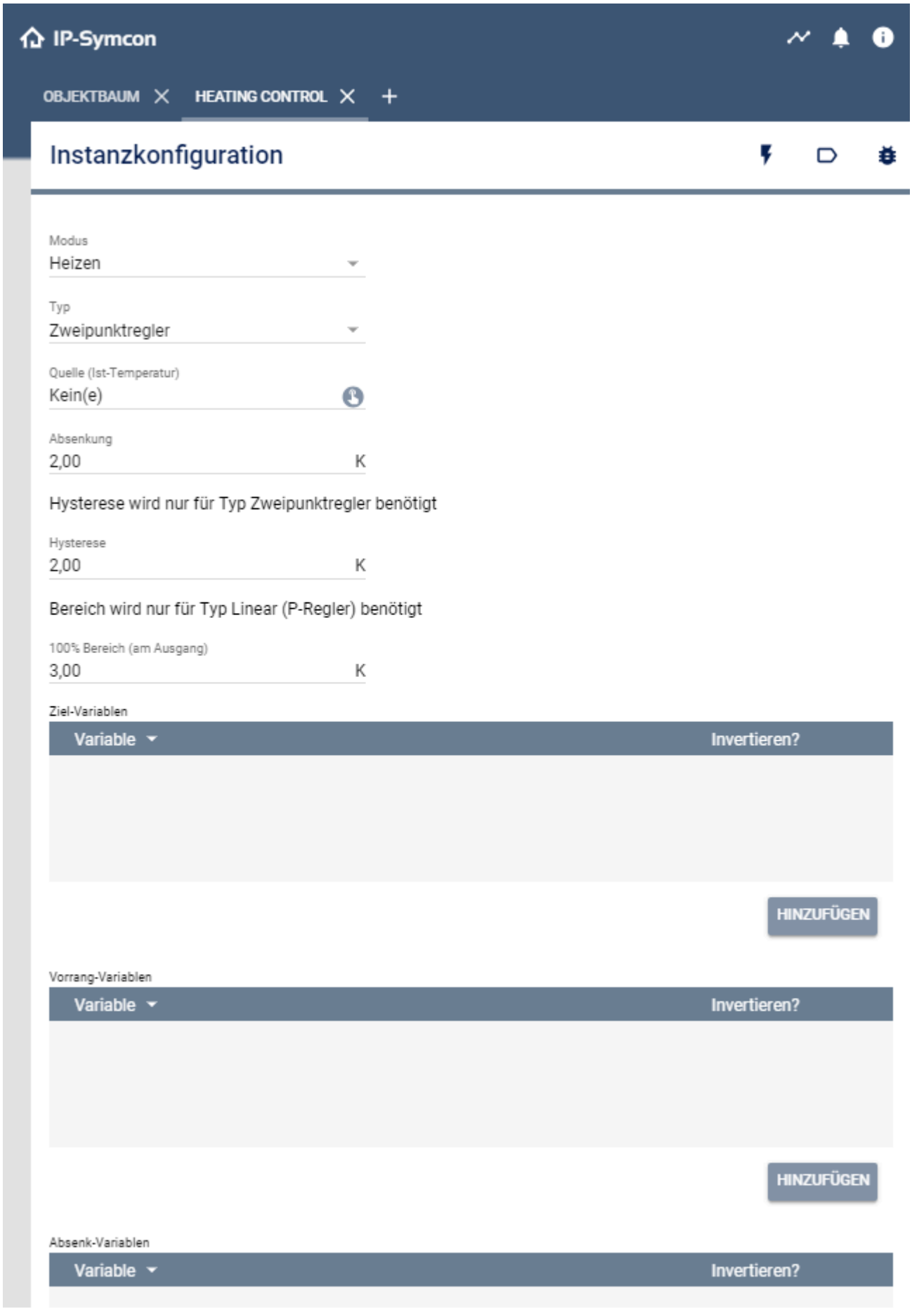
The heating module is used to control temperatures in living spaces.
It is based on a simple but effective 2-point controller.
This compares the target value with the actual value and switches an actuator on or off.
For each exceeding or falling below of the set temperature, a single switching command is sent.

REQUIREMENT

The actual value is obtained via respective temperature sensors. The value must be known of IP-Symcon as a variable in. Furthermore, the heat source must be switchable via an actuator.
Examples: Radiator with a thermal actuator or a heater and a radio socket.

CONFIGURATION

Add the "Heating Control" module (manufacturer: None) and specify optionally a name such as 'Heating module living room':



VARIABLE	DESCRIPTION
Source Variable:	Here you can select the variable that contains the actual value.

VARIABLE	DESCRIPTION
Hysteresis:	Temperature difference between the switch-on and switch-off, for example: T-Target = 20.0 degrees and hysteresis = 0.4, the actuator is switched on when dropping below 19.8 degrees and stopped when exceeding 20.2 degrees. A hysteresis of 0 causes a "toggling" of the actuator at each update of the actual variables.
Reduction:	Temperature in °C, by which is to be lowered, if one or more of the 'lowering variables' are TRUE.
Retransmission Instances:	Here the switching actuators should be entered. Thermal actuators, which are open at electroless, "Invert" must be activated.
Priority Variables:	Can be e.g. Status variables of window contacts. When true, the output of the regulator will remain off.
Retransmission Interval:	If a (radio) command is not handled correctly, it can be repeatedly cycled (no more frequently than 15 minutes).
Lowering Variables:	Can be e.g. Presence variables of movement and presence detectors. If no person is staying in a room, energy can be saved by a temperature decrease.
Expert Settings:	For special cases, a custom script can be selected. It is called for each switching operation of the controller. All necessary data is deposited in the 'System Variables'.
Test Center:	To check the function of the controller, a desired temperature can be entered. The actuator must then turn on or off as desired.
Status Variables	
Target Value:	Contains the target temperature
Override:	If True at least one priority variable is also true (OR operation), and the actuator is switched off.
Heating:	Reflects the output of the controller. Inverting the Retransmission instances has no effect on this variable.

TIPS & TRICKS

Use the lower function in order to save energy in unoccupied rooms.
Practical example:
A motion detector installed in the room provides for a few seconds in IP-Symcon a variable change to TRUE. The following script will be extended to this impulse for example for 15 minutes and writes the status in the variable 'presence' (please create it before!). If within the desired time no movement is detected in the room, the variable is set to FALSE.
It can now be added under the 'lowering variables'. However, also 'Invert' must be activated, because we want to lower the temperature in NOT-Presence. This must now be entered only under 'reduction': for example 2 °C.
An open window with his 'priority variable' also has precedence here and turns the heating off completely.

```
$id_prae = 12345 /*[Präsenz]*/; // set!

if($_IPS['SENDER'] == "Variable"){
if($_IPS['VALUE'] == True){ //Start Timer
IPS_SetScriptTimer($_IPS['SELF'], 15 * 60);
```

```
$prae = GetValue($id_prae );
if($prae == False){
SetValue($id_prae, True);
}
}
}

if($_IPS['SENDER'] == "TimerEvent"){
IPS_SetScriptTimer($_IPS['SELF'], 0);
SetValue($id_prae , False);
}
```

FURTHER LINKS

Two-point controller: de.wikipedia.org/wiki/Zweipunktregler
Hysteresis: de.wikipedia.org/wiki/Hysterese



Since at any time the PC (control) can fail, in addition a temperature limit/ safety shutdown must be installed to prevent injury or damage.

HC_TargetValue

```
bool HC_TargetValue (int $InstanceID, float $TagetValue)
```

PARAMETERS

InstanceID	ID of the device to be switched
TagetValue	New target value

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends via the instance with the ID **InstanceID** a target value to the heating module.

EXAMPLE

```
HC_TargetValue(12345, 22.5); //At 22.5°C
```

Home Connect

Require: Symcon >= 6.0

DESCRIPTION

The Home Connect module allows devices from Bosch, Siemens, Gaggenau and Neff, which support the Home Connect service, to be integrated into IP-Symcon.

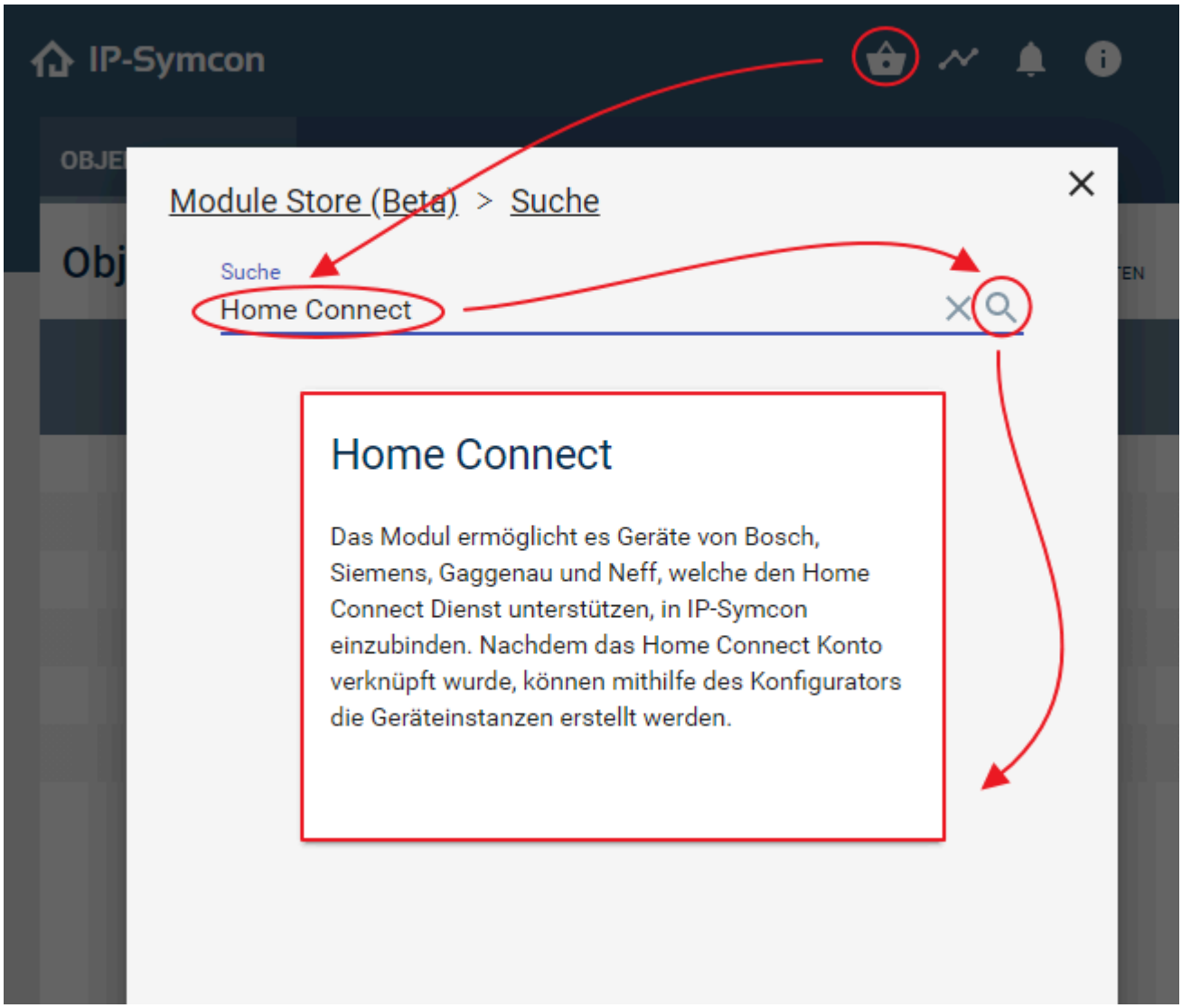
INTEGRATION IN IP-SYMCON

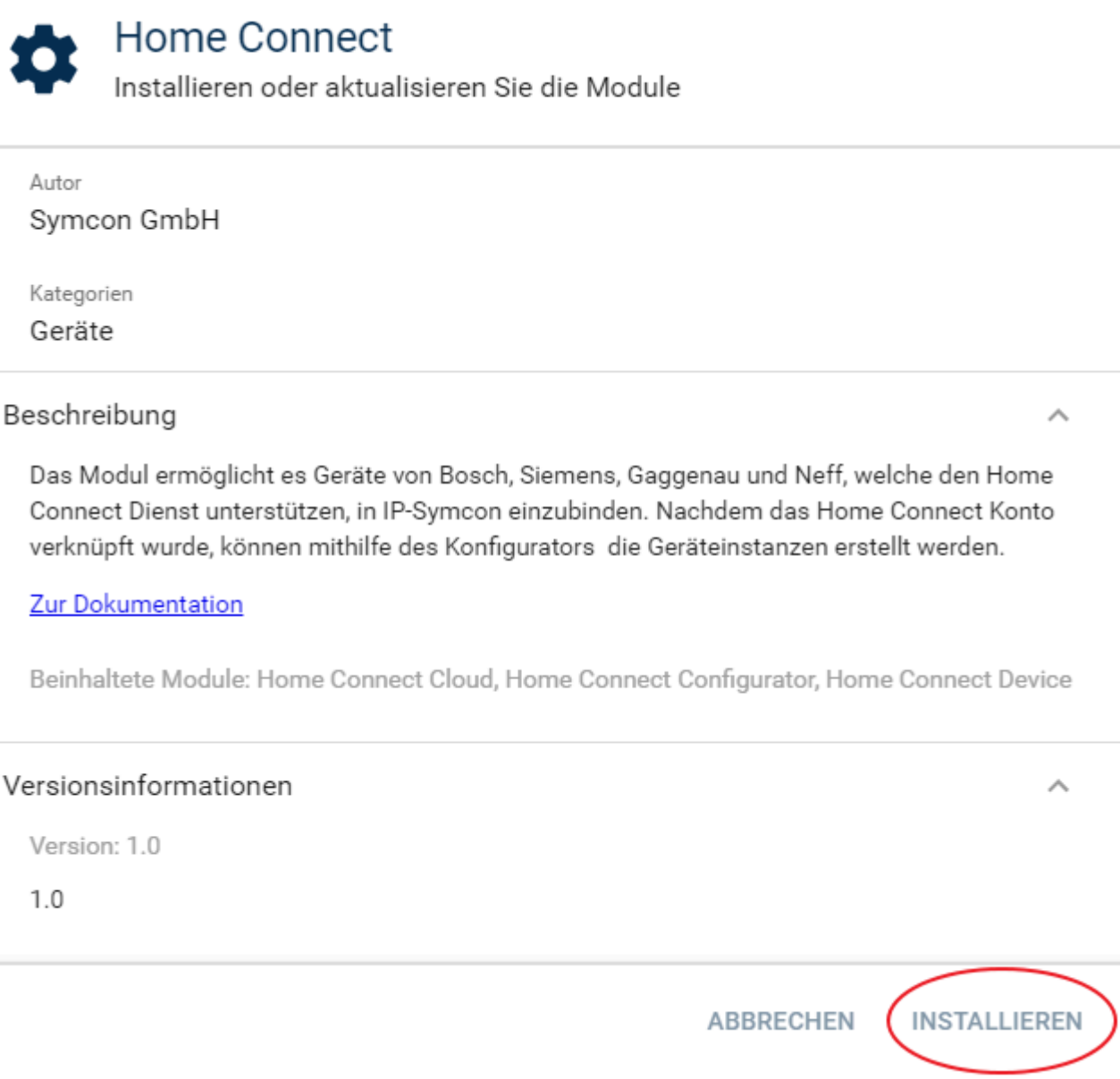


The use of the Home Connect-Module requires an activated connection to the Connect service, which in turn requires an active subscription.

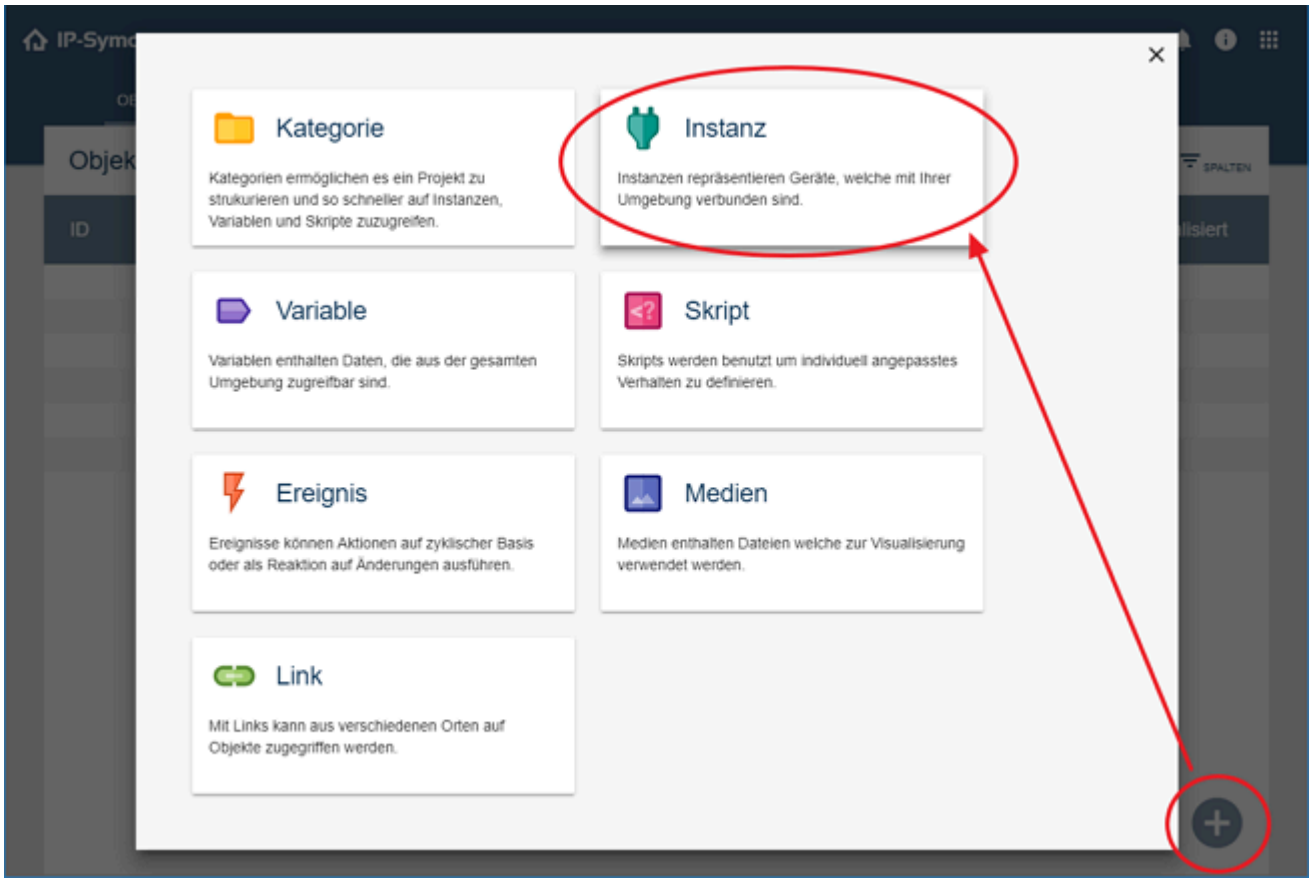
HOME CONNECT CONFIGURATOR CREATE INSTANCE

In order for the Home Connect module to be used, it must first be installed via the Module Store. To do this, the Module Store must be opened. It is located in the upper right area. The Home Connect module can be found via the search field via “Home Connect”. When opening the found module, the installation of the module can be initiated in the following dialog via the "Install" button.


IP-Symcon
Module Store (Beta) > Suche
Suche
Home Connect
Home Connect
Das Modul ermöglicht es Geräte von Bosch, Siemens, Gaggenau und Neff, welche den Home Connect Dienst unterstützen, in IP-Symcon einzubinden. Nachdem das Home Connect Konto verknüpft wurde, können mithilfe des Konfigurators die Geräteinstanzen erstellt werden.



The add dialog for creating a Home Connect configurator instance then opens.
If the instance is to be added manually, an Instance must be created by the Home Connect configurator. To do this, the object tree must first be opened. In this the "+" button at the bottom right must be pressed and Instance selected.



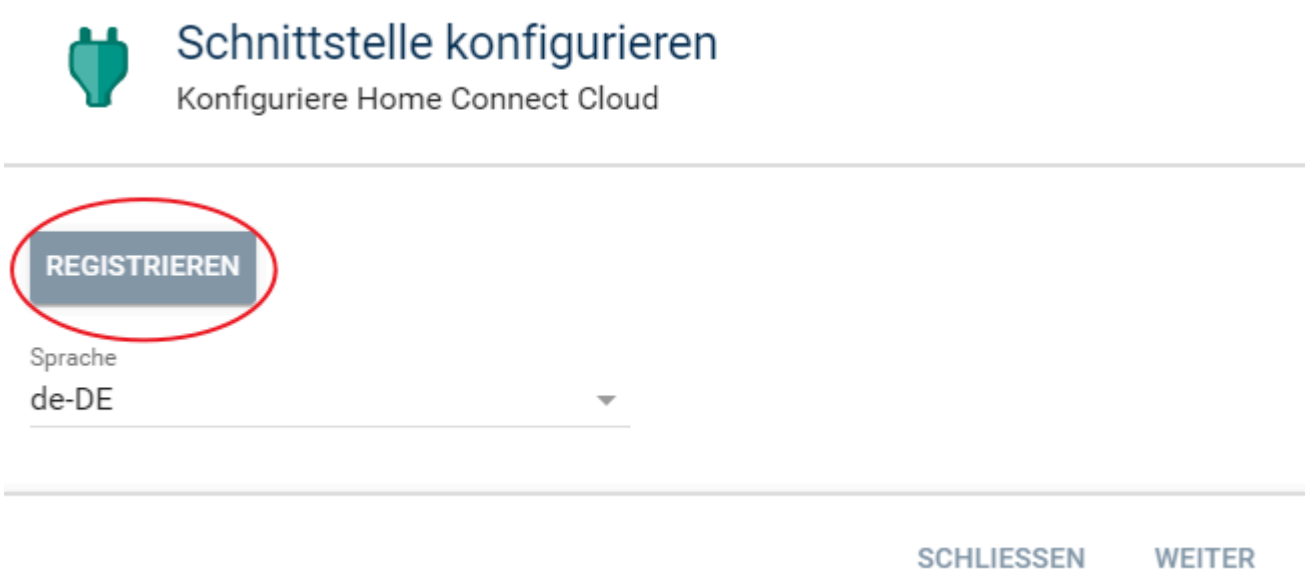
With the help of the quick search, the “Home Connect Configurator” can be found by the manufacturer “(configurator)”. The location should not be changed, the name can be chosen freely. Finally, confirm with "OK".



After the configurator instance has been created, it automatically opens and can be set up.

HOME CONNECT SETUP CONFIGURATOR INSTANCE

Before devices can be created, one must register with a click on Register in order to connect IP-Symcon to the Home Connect account.
In the 'Language' field, one can select in which language the information about the devices is processed. The language you select here is not related to the language used in IP-Symcon.



One must log in with the Home Connect account to grant IP-Symcon the necessary permissions.



Bitte melde dich hier mit deinem Home Connect Nutzerkonto an, um dieser Applikation den Zugriff auf deine Hausgeräte zu erlauben.

E-Mail

max@mustermann.com

••••••••••••••••••

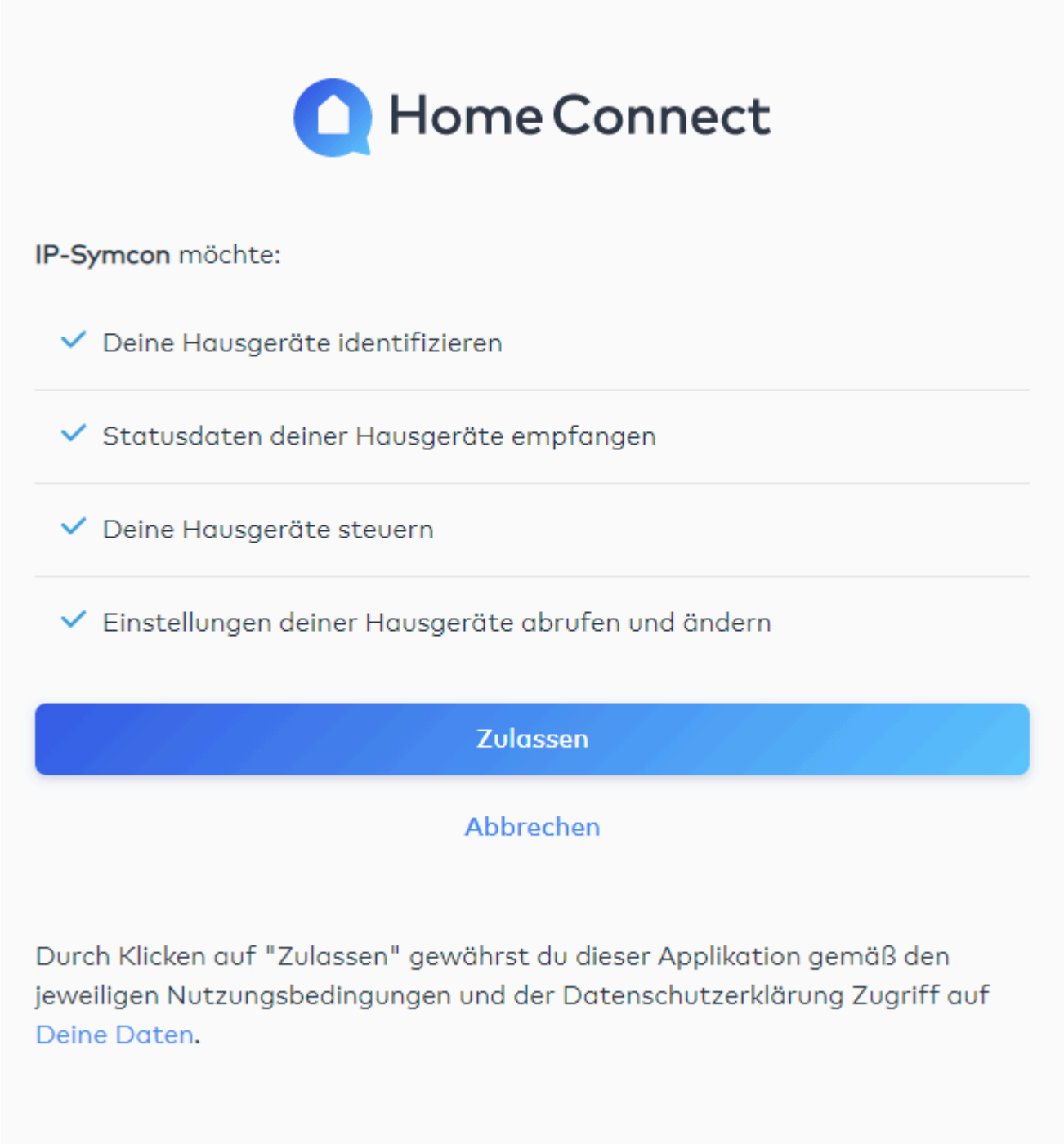
Anmelden

Abbrechen

Passwort vergessen? Klicke hier, um dein Passwort zurückzusetzen.

Falls du noch kein Benutzerkonto hast, verwende bitte die Home Connect App um dich zu registrieren.

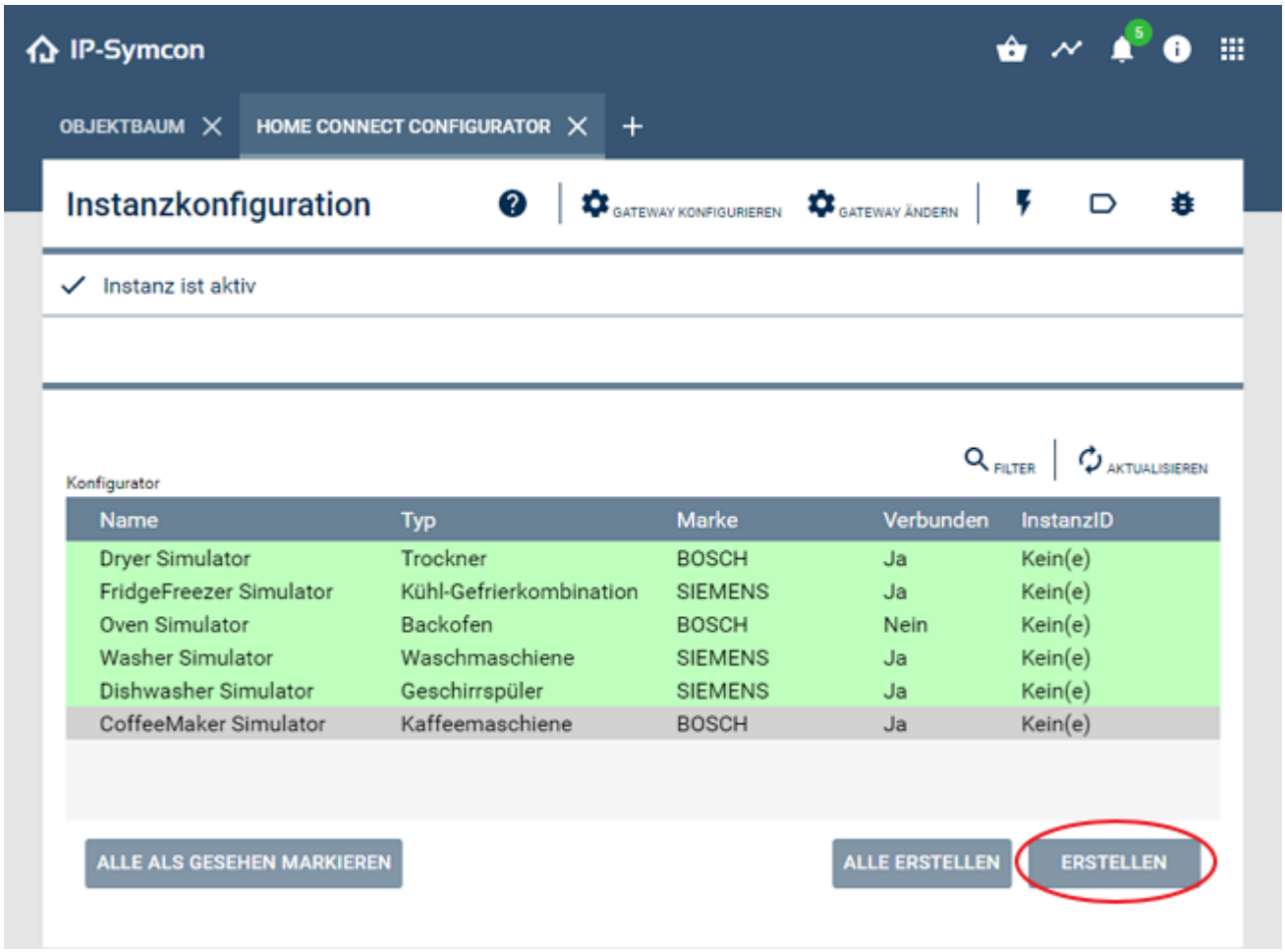
Erfahre mehr über Home Connect.



The “Configure Interface” dialog can then be confirmed with Next. The SSE client that was created must now be activated.



Now the configurator opens and all recognized devices can be created with a click on "Create".


IP-Symcon
OBJEKTBAUM
HOME CONNECT CONFIGURATOR
Instanzkonfiguration
GATEWAY KONFIGURIEREN
GATEWAY ÄNDERN
Instanz ist aktiv
FILTER
AKTUALISIEREN
Konfigurator
Name
Typ
Marke
Verbunden
InstanzID
Dryer Simulator
Trockner
BOSCH
Ja
Kein(e)
FridgeFreezer Simulator
Kühl-Gefrierkombination
SIEMENS
Ja
Kein(e)
Oven Simulator
Backofen
BOSCH
Nein
Kein(e)
Washer Simulator
Waschmaschiene
SIEMENS
Ja
Kein(e)
Dishwasher Simulator
Geschirrspüler
SIEMENS
Ja
Kein(e)
CoffeeMaker Simulator
Kaffeemaschiene
BOSCH
Ja
Kein(e)
ALLE ALS GESEHEN MARKIEREN
ALLE ERSTELLEN
ERSTELLEN

Device list

SUPPORTED DEVICES

The Home Connect service is provided by devices from Bosch, Siemens, Gaggenau and Neff.

DEVICE TYPE
Coffee machine
Oven
Dryer
Washing Machine
Dishwasher
Refrigerator
Freezer

If a device is not in the list above, it can still be added to IP-Symcon. If problems arise during setup, our support team will be happy to help.

Home Connect Device

Require: Symcon >= 6.0

DESCRIPTION

When a device is created by the Home Connect configurator, a Home Connect device instance is created which has various status variables depending on the device type.

CONTROL BY IP-SYMCON

In order to be able to use the devices to the full extent, the remote start and/or remote control must be activated directly on the device. If only the remote control is active, programs and options can be changed, but the program cannot be started. While the device is being operated locally, it cannot be controlled by IP-Symcon.

STATUS VARIABLES

Each device can have 4 different types of variables.

STATUS

Status type variables are variables that cannot be switched by IP-Symcon.
Examples: Operating status, Local operation active

SETTING

Variables of the setting type can usually be switched and are used to change various settings of the device.
Examples: Temperature (refrigerator), Energy status

PROGRAM

Each device has only one variable of the type program. This variable can be used to determine which program is to be executed by the device.



If a program, that is available in the Home Connect app, is not displayed in IP-Symcon, then it is not supported by the API. If such a program is selected manually via the app or the device, then all setting options are grayed out

OPTIONS

When a program is selected, corresponding option variables are created that change the details for the selected program.
Examples: Capacity (coffee machine), Duration (oven)
When a program has been selected and the available options have been set, the program can be started using the Control variable.
As soon as a program has been started, depending on the device, variables are created that provide information on the progress.
Examples: Program progress, Remaining runtime

PARTICULARITIES

When creating a device, it should be online and therefore switched on
The refrigerator has no programs

TROUBLESHOOTING

A DEVICE INSTANCE IS NOT CREATED CORRECTLY

All data of a device can be requested again in the device instance by clicking on the 'Initialize device' button.

VARIABLES ARE NOT UPDATED

Click on 'Register server events' to re-establish the connection to the Home Connect service.

HomeMatic

Symcon supports integration via the HCU and via the CCU.

HomeMatic CCU
HomeMatic HCU (Beta)

Device list

SUPPORTED GATEWAYS

GATEWAY	DESCRIPTION
CCU1	Central for HomeMatic radio and wired components
CCU2	Central for HomeMatic radio, wired and IP components
CCU3	Central for HomeMatic radio, wired and IP components
HM-LAN (via BidCos service)	HomeMatic LAN adapter
HM-USB (via BidCos service)	HomeMatic USB adapter
HCU	Central for HomeMatic IP components

SUPPORTED COMPONENTS

IP-Symcon supports all wired and wireless components.

HomeMatic CCU

HomeMatic is a primarily radio-based system. A connection to IP-Symcon is established via LAN through the Central Control Unit (CCU). Alternatively, this also works with a "LAN configuration adapter".



The following devices are supported by IP-Symcon:
Supported devices

INSTALLATION

The HomeMatic devices are connected to IP-Symcon via CCU (LAN) or BidCoS Service.
The instructions for setting up via BidCoS Service are located here.



Advantages of the CCU:
HomeMatic Wired RS485 components are supported.
Programs can be executed in the CCU.
Status of the actuators can be queried via script.

HOMEMATIC WIRED

HomeMatic Wired is a wired version of HomeMatic. Separate switching and dimming actuators as well as input and output modules are available for this purpose.
However, "HomeMatic Wired" is controlled by IP-Symcon in the same way as the radio-based components.
This can be enabled on the configuration page of the gateway.

HOMEMATIC IP

HomeMatic IP is the successor to HomeMatic and offers wireless and wired devices. HomeMatic IP is controlled by IP-Symcon in the same way as the original HomeMatic components. Parallel operation of HomeMatic and HomeMatic IP components is possible without any problems.

SETUP VIDEO-TUTORIAL

CONNECTION

The CCU is connected to the network by a network cable.

The antenna of the CCU is used only for radio communication with the individual actuators. However, a WIFI connection with a PC is not possible.

If no LAN network is available, a connection via USB is also possible. This requires extra USB drivers, which can be found on the included CD.

INSTALLATION

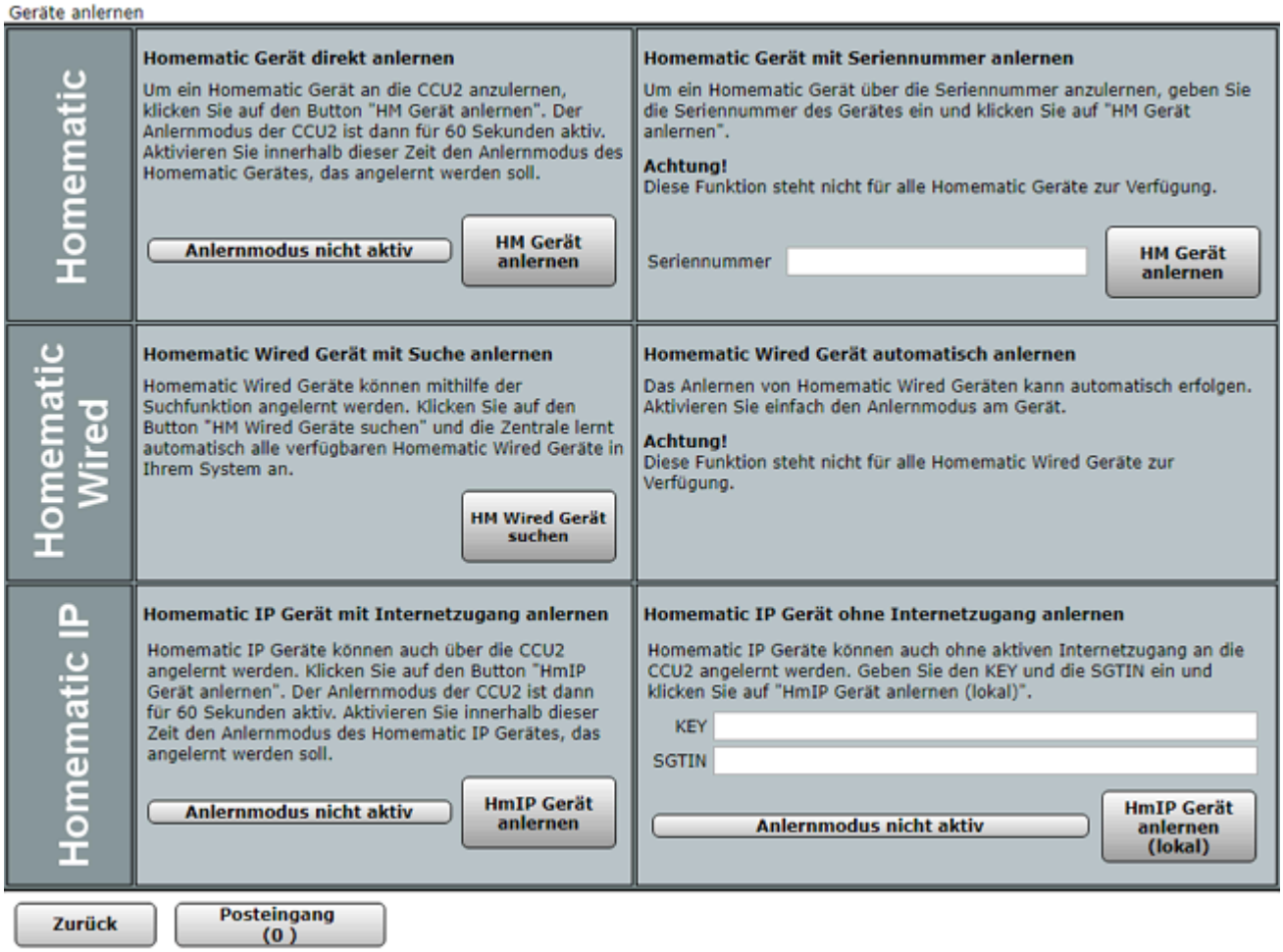
If the CCU is connected correctly and a DHCP server is used, the CCU automatically receives an IP address and is connected to the network.
If no DHCP server is available, an IP address can also be entered manually via the menu of the CCU, under the menu item "Network".
The subnet mask and the default gateway must also be assigned here. If the setup is correct, the device "HomeMatic Central" can now be found in the network overview.

If the CCU does not appear in the network, disconnecting the power supply of the CCU for 30 seconds may be helpful. After booting up the CCU and updating the network environment, it is now accessible.

A double click on "HomeMatic Central" opens a window in the lower area of which the IP address of the CCU can be found. (Optionally, this can also be found in the menu of the CCU.) By copying this address into a browser, the HomeMatic WebUI opens.

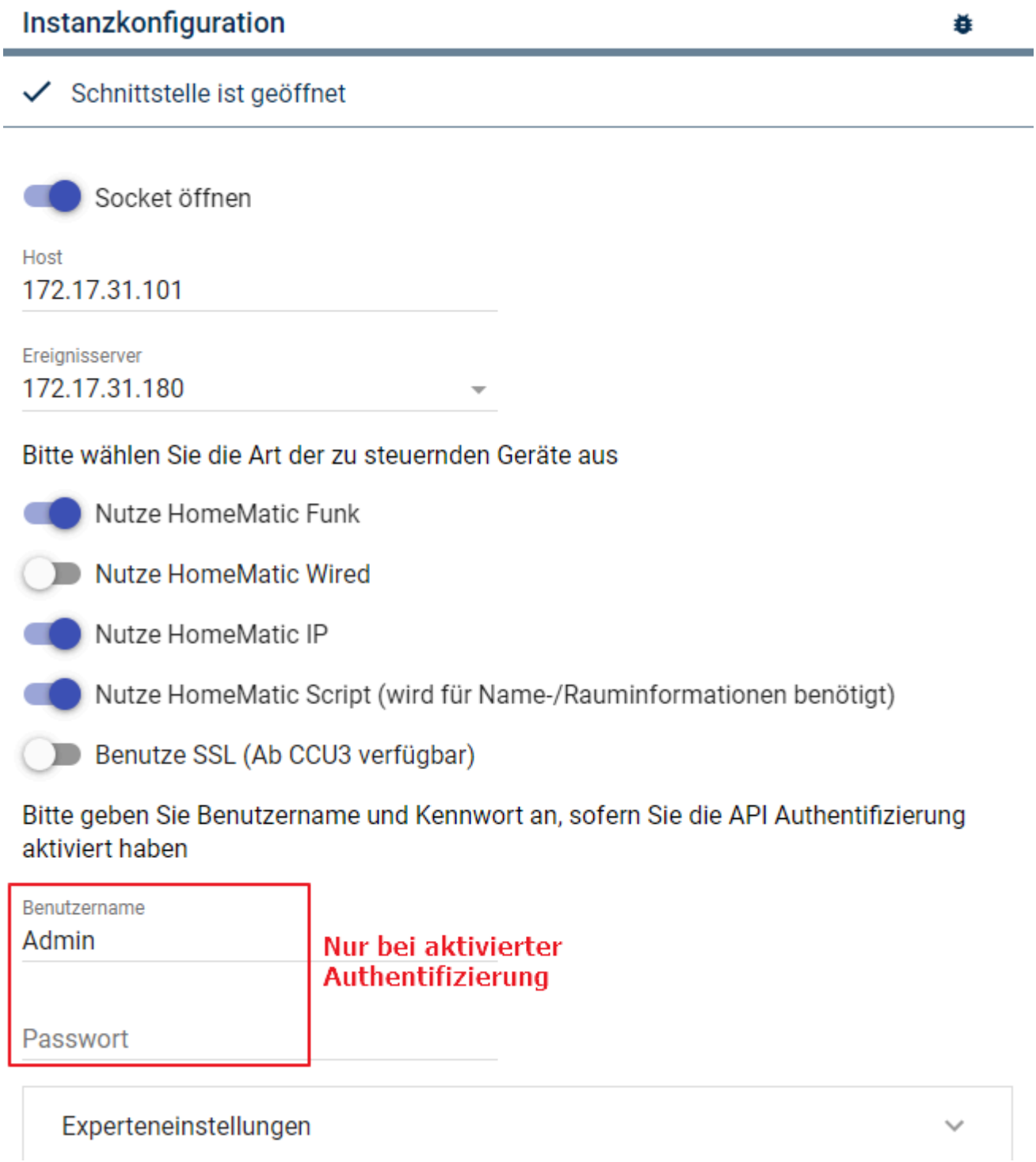
ADDING SECONDARY DEVICES

In the HomeMatic WebUI, the secondary HomeMatic devices (e.g. temperature sensors, intermediate plugs, handheld transmitters, etc.) can now be connected to the CCU. To do this, "Teach devices" (top right) must be clicked on and the further instructions followed.

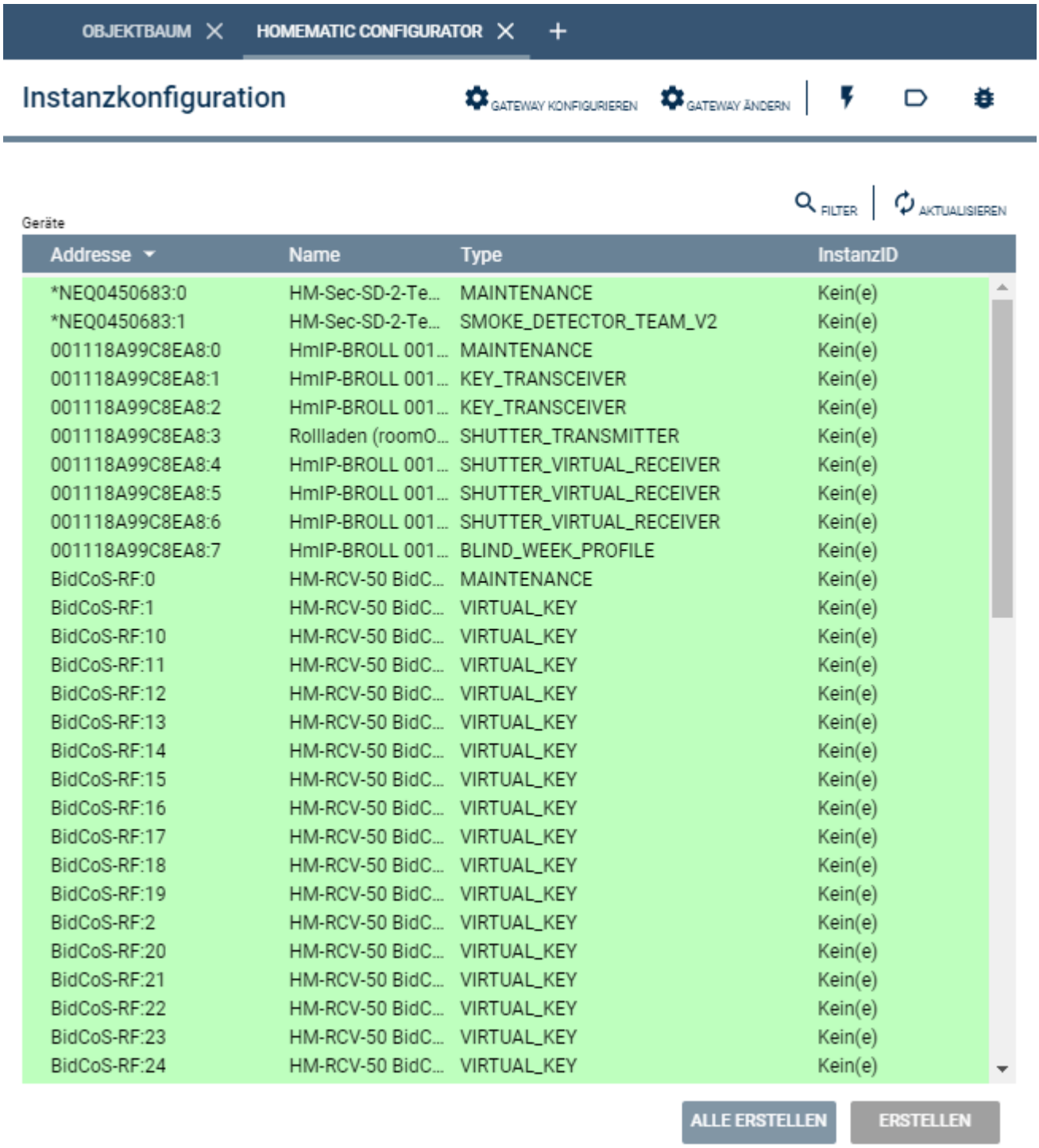


INTEGRATION IN IP-SYMCON

The CCU can be integrated via the Device Search. For this, "Homematic Discovery" must be selected as the system. The Discovery instance then offers to create a Homematic Configurator. After the configurator has been created, the individual devices can be integrated via it as described below.
Within the configurator, the configuration page of the "HomeMatic Socket" can be opened via "Configure gateway". To activate the socket, the checkbox "Open socket" must be set. The options displayed tell whether HomeMatic radio/wired/IP is to be used. If authentication is enabled, the user name and password can be entered here.



Various devices are now visible in the window that appears in the center. After selecting a device, a device instance can be created with "Create" and then the configuration page can be called up via "Configure".

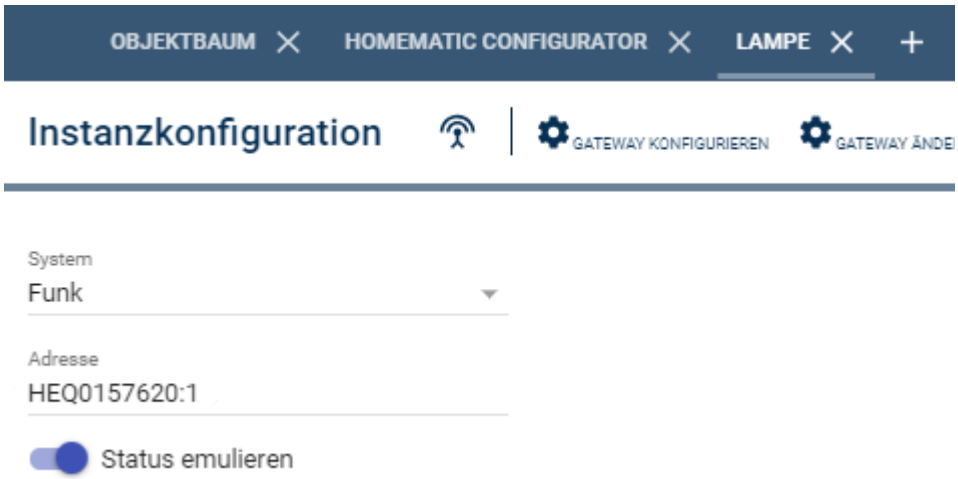


To test if e.g. a dimmer is working, the proper function can be checked, e.g. in the Visualization.

In certain circumstances, the return channel may not work. This is noticeable by the fact that IP-Symcon does not visualize if e.g. a lamp was switched via a light switch.
This problem is often due to the Windows Firewall. This can be configured in Windows under "Control Panel" -> "System and Security" -> "Windows Firewall".
In the menu "Allow a program or feature through Windows Firewall" -> "Change settings" -> "Allow other program..." there is a button "Browse".
By selecting the IP-Symcon (IP-Symcon Service) file and then "open", the return channel also works!
Another way to ensure the functionality of the return channel is to make sure that under the firewall settings in the CCU "HomeMatic XML-RPC API" and "Remote HomeMatic-Script API" full access is selected or the IP address of the IP-Symcon PC is entered.

DEVICE CONFIGURATION

It is recommended to use the HomeMatic Configurator to set up the individual devices. The following settings are possible on the configuration page of the device instance.



OPTION	DESCRIPTION
System	Select whether the device is controlled via radio, wired or IP.
Address	HomeMatic Address of the device. This can also be searched for via the "Search" dialog in the upper bar of the configuration bar.
Emulate status	If enabled, the status of the device is updated when a command is successfully sent to the CCU. If switched off, the actual response of the device to be switched is waited for. This can lead to delays. For example, with devices that wake up only every x minutes.

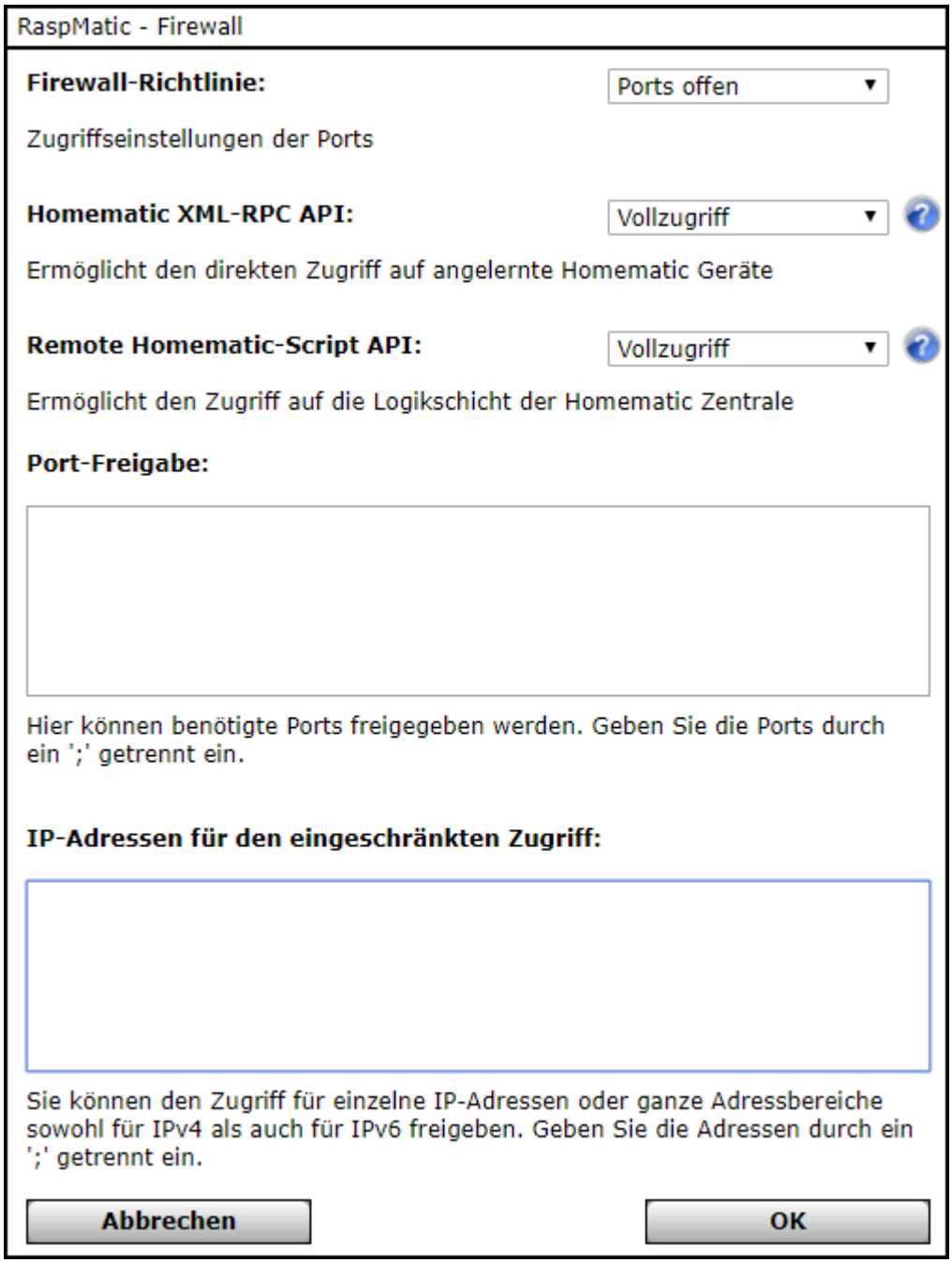
FIREWALL-, AUTHENTICATION- AND PORT SETTINGS

Authentication is possible with a CCU3 or RaspberryMatic in combination with IP-Symcon 5.1 or above.

This can be set in the web interface of the CCU. Authentication can be activated under "Settings" -> "Control Panel" -> "Security". This then uses the users that are set in the user administration.

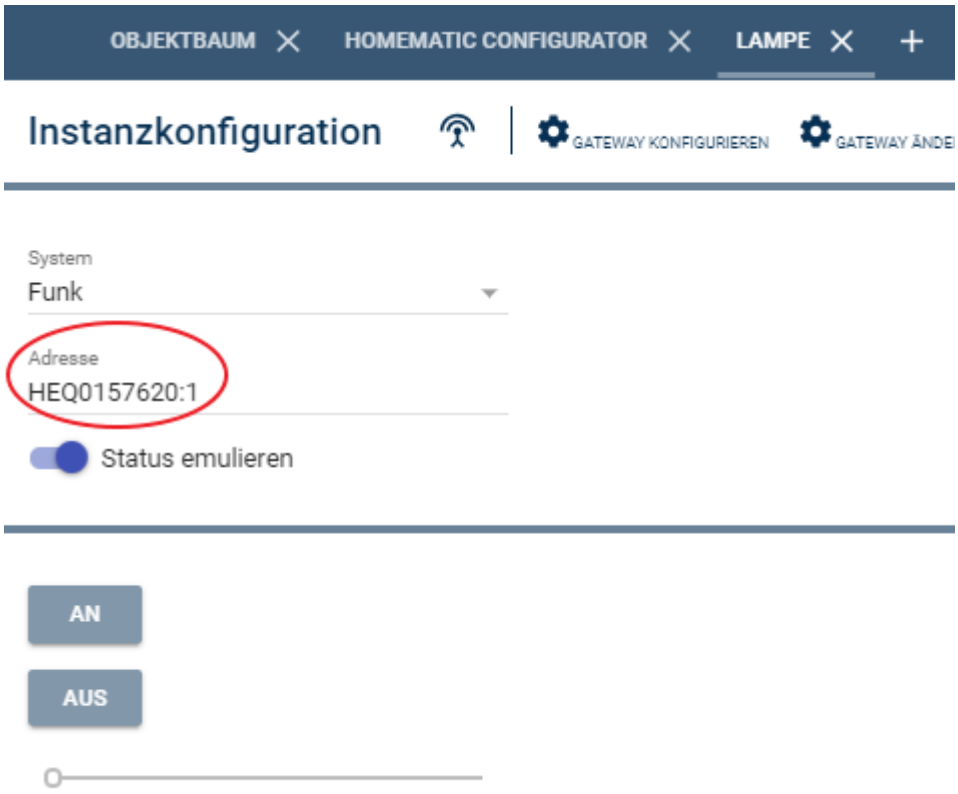


Furthermore, the firewall can be set up under "Settings" -> "Control Panel" -> "Configure Firewall". Further information can be found in the CCU manual.



REPLACEMENT OF DEFECTIVE EQUIPMENT

General information on how an exchange works in IP-Symcon can be found here
For replacement, the configuration of the defective device must be opened in IP-Symcon and the address of the old device must be replaced for that of the new one. The address is either on the device itself or can be selected via "Search".



The address in the highlighted area must be replaced and saved with "Apply".

HOMEMATIC WITH QNAP, SYNOLOGY OR DOCKER

If HomeMatic is to be used with QNAP, Synology or Docker, NAT support must be set up to receive responses from the system. This process is explained here.

TIPS & TRICKS

Documentation Homematic Data Points
Documentation HomematicIP Data Points
HomeMatic Homepage
It is possible to change the display of the 19 button remote control
To query the status of all actuators and sensors (only possible with the CCU!)
The duty cycle for an actuator can be specified with "ON_TIME" and for a dimmer additionally a ramp can be specified.

Here is an example script for three dimmers creating a light scene:

```
$id_bar = 54392 /*[EG\Table Bar]*/;
$id_ecken = 24601 /*[EG\Corner Spots]*/;
$id_tisch = 38758 /*[EG\TableLamp]*/;

$ramp = 2;
HM_WriteValueFloat($id_bar, "ON_TIME", 60*10); // x Minuten ON
HM_WriteValueFloat($id_bar, "RAMP_TIME", $ramp); // X seconds ramp
HM_WriteValueFloat($id_bar , "LEVEL" , .4); // and run on X%

HM_WriteValueFloat($id_corner , "RAMP_TIME", $ramp); // X seconds ramp
```

```
HM_WriteValueFloat($id_corner , "LEVEL" , .4); // and run on X%

HM_WriteValueFloat($id_table , "RAMP_TIME", $ramp); // X seconds ramp
HM_WriteValueFloat($id_table , "LEVEL" , .4); // and run on x%
```

Example script WinMatic 60 minutes aeration:

```
$id_actuator = 49712 /*[OG\Bedroom\HM WinMatic]*/;
HM_WriteValueFloat($id_actuator, "SPEED" , 1.0); // Maximum Speed ;
HM_WriteValueFloat($id_actuator, "RELOCK_DELAY" , 60*60); // Close
window again after XX minutes
HM_WriteValueFloat($id_actuator, "LEVEL" , 0.7); // Open window
```

Example script open KeyMatic:

```
HM_WriteValueBoolean($id, "OPEN", true);
```

Example script to get a feedback if an actuator has executed the switching command correctly:

```
$id_actuator = 12345;
$err = HM_WriteValueBoolean($id_actuator, "STATE" , False);
//echo "Err: " .(int) $err . "\n";
if ($err === False){
echo "error: Switch actuator - Command was not executed";
SetValue($id_done, False);
} else {
SetValue($id_done, True);
echo "OK: Switch actuator - Command was executed\n";
}
```

Example script to toggle (turn on and off) an actuator (lamp):

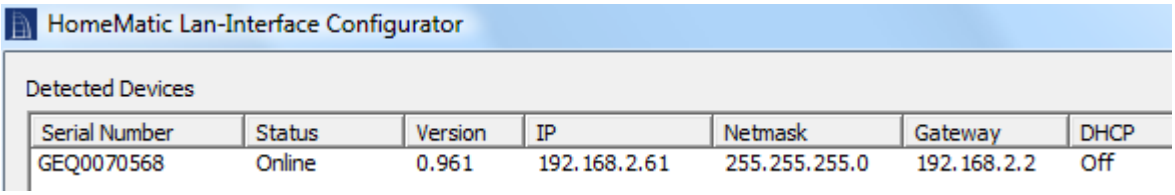
```
$id_actuator = 25404 /*[yard garden\lamp]*/;
$id_state = 55194 /*[yard garden\lamp\STATE]*/;
HM_WriteValueBoolean($id_actuator, "STATE" , !GetValue($id_state));
```

BidCos Service

The BidCos service is an alternative to the CCU-supported control of HomeMatic devices.

CONFIGURATION OF THE LAN-ADAPTER

First of all, the latest user software must be installed from the HomeMatic site.
We recommend assigning a fixed IP-address (DHCP: Off):



After successful installation a new directory was created: "x:\ProgramData\Bidcos-Service\"
The serial number (see label) and the IP address of the adapter must be adjusted in the 'bidcos.conf' file.
Example:

```
[Interface 0]
Type = Lan Interface
Serial Number = GEQ0123456

# Key is on the back of the interface.
# The function can also be deactivated via the LAN tool.
# Then the field can be left blank
Encryption Key =
Description = First Lan Interface

# IP address of the LAN interface
IP Address = 192.168.2.61
```

Further settings in bidcos.conf are not required.
It is essential that the following entry remains unchanged:

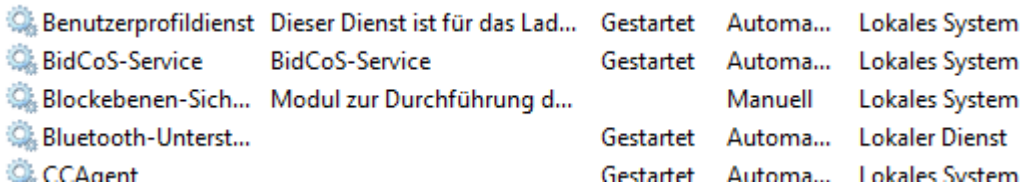
```
# TCP Port for XmlRpc connections
Listen Port = 2001
```

TESTING THE ADAPTER

A successful configuration can now be checked with the "Test-Run BidCos-Service" program.
The last line in the DOS window then reads: "… Connected to Lan Interface…".

A flashing power LED indicates that the BidCos service is not connected.

If necessary, the start options can be adjusted (Control Panel > Administration > Services) or the service can be started manually:

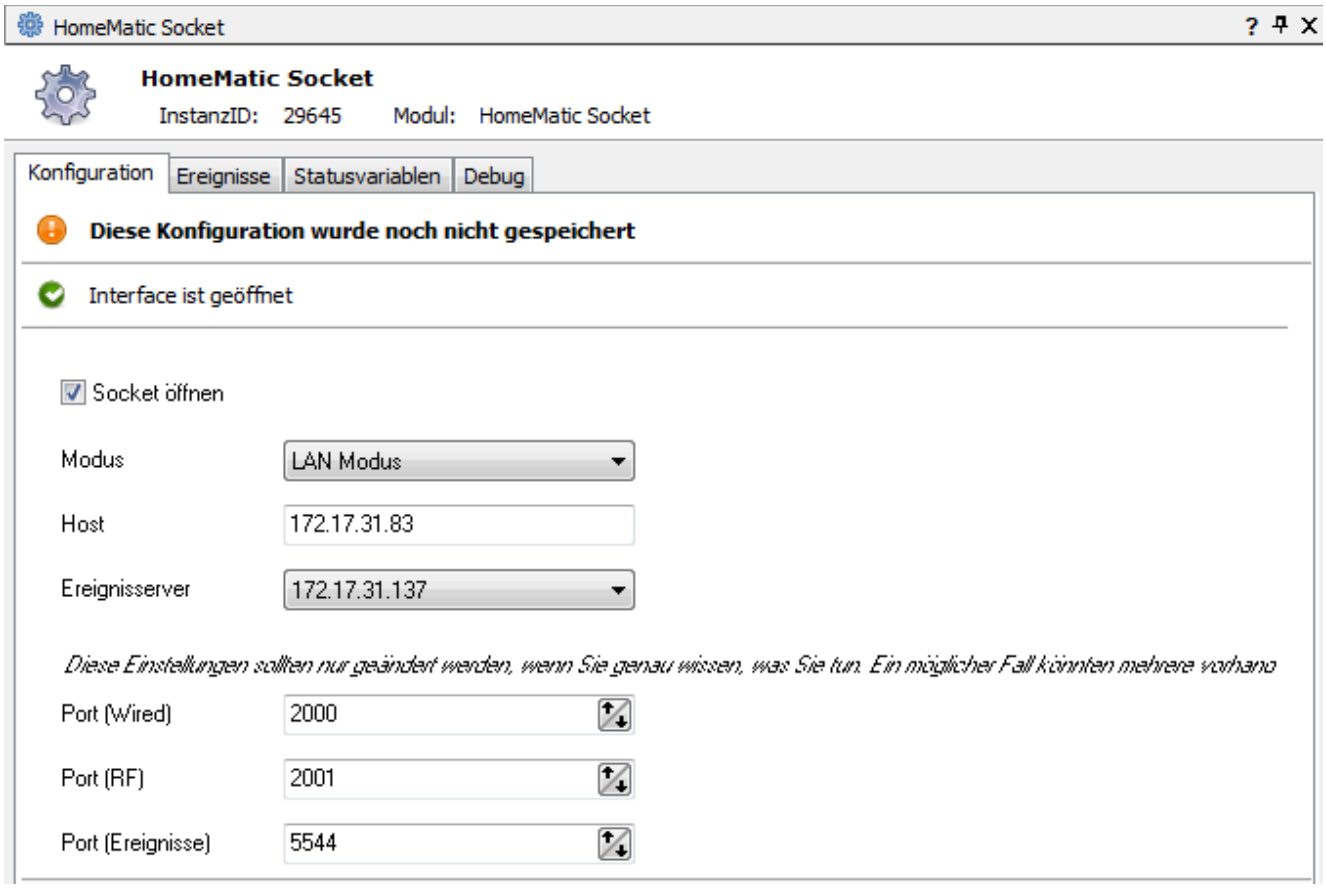


CONFIGURE HOMEMATIC-COMPONENTS

In the "HomeMatic Configurator" program, the connection "Remote BidCoS Service" must be selected under "File > Settings". Then the IP address (localhost) and the port (2001) have to be entered.
Finally, the HomeMatic components can now be added one after the other in the "Teach-in devices" menu.

CLIENT SOCKET IN IP-SYMCON

The HomeMatic Socket (in the I/O instances) must be configured. "LAN Mode" must be selected as the mode. If the BidCoS service is running on another PC, its IP address must be entered as the host. Otherwise "localhost" must remain specified as the address. If there are several network adapters, the IP address of the network-adapter, which the LAN-adapter is connected to, must be selected in the event-server. The port (5544) does not have to be changed, it only has to be enabled in an existing firewall.


HomeMatic Socket
HomeMatic Socket
InstanzID: 29645 Modul: HomeMatic Socket
Konfiguration
Ereignisse
Statusvariablen
Debug
Diese Konfiguration wurde noch nicht gespeichert
Interface ist geöffnet
Socket öffnen
Modus
LAN Modus
Host
172.17.31.83
Ereignisserver
172.17.31.137
Diese Einstellungen sollten nur geändert werden, wenn Sie genau wissen, was Sie tun. Ein möglicher Fall könnten mehrere vorhand
Port (Wired)
2000
Port (RF)
2001
Port (Ereignisse)
5544

HM_ReadServiceMessages

```
array HM_ReadServiceMessages (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the HomeMatic Socket Instance

RETURNS

The following information are available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
Address	string	Address of the relevant device
Message	string	Cryptic description of the error message. Text can be found partially in the stringtable_de.txt from the Home Automatic configuration tool.
Value	variant	Value of the error message

DESCRIPTION

Reads the service news of the CCU/ LAN adapter with the ID **InstanceID**.

EXAMPLE

```
//Creates a variable that displays all service messages in the
Visualization. Copy in a script and run.

//From here change anything
$object = IPS_GetObject($_IPS['SELF']);
$parentID = $object['ParentID'];

//Installer
if ($_IPS['SENDER'] == "Execute")
{
IPS_SetHidden($_IPS['SELF'], true);
IPS_SetName($_IPS['SELF'], "Script");
$parentObject = IPS_GetObject($parentID);
if ($parentObject['ObjectType'] !== OBJECTTYPE_INSTANCE)
{
$instanceID = IPS_CreateInstance("{485D0419-BE97-4548-AA9C-
C083EB82E61E}");
IPS_SetParent($instanceID, $parentID);
```

```
$parentID = $instanceID;
IPS_SetParent($_IPS['SELF'], $parentID);
IPS_SetName($instanceID, "Servic messages");
}
IPS_SetScriptTimer($_IPS['SELF'], 300);
}

$texte = Array(
"CONFIG_PENDING" => "Configuration data are due for transfer",
"LOWBAT" => "Low battery condition",
"STICKY_UNREACH" => "Device communication was disrupted",
"UNREACH" => "Disrupted communication devices currently"
);

$str = "<table width='90%' align='center'>"; // Adjust color or remove
style
$str .= "<tr><td><b>Gerätname</b></td><td><b>GeräteID</b></td><td>
<b>Meldung</b></td></tr>";

$ids = IPS_GetInstanceListByModuleID("{A151ECE9-D733-4FB9-AA15-
7F7DD10C58AF}");
if(sizeof($ids) == 0)
die("No HomeMatic Socket instance found!");

$msgs = HM_ReadServiceMessages($ids[0]);
if($msgs === false)
die("Connect to the CCU failed");

if(sizeof($msgs) == 0)
$str .= "<tr><td colspan=3><br/>No service messages!</td></tr>";

foreach($msgs as $msg)
{
if(array_key_exists($msg['Message'], $texte)) {
$text = $texte[$msg['Message']];
} else {
$text = $msg['Message'];
}

$id = GetInstanceIDFromHMID($msg['Address']);
if(IPS_InstanceExists($id)) {
$name = IPS_GetLocation($id);
} else {
$name = "Device is not set up in IP Symcon";
}

$str .= "<tr><td>".$name."</td><td>".$msg['Address']."</td>
<td>".$text."</td></tr>";
}
$str .= "</table>";

$vid = CreateVariableByName($parentID, "Content", VARIABLETYPE_STRING);
IPS_SetIcon($vid, "Information");
IPS_SetVariableCustomProfile($vid, "~HTMLBox");
SetValue($vid, $str);

function GetInstanceIDFromHMID($sid)
{
$ids = IPS_GetInstanceListByModuleID("{EE4A81C6-5C90-4DB7-AD2F-
```

```
F6BBD521412E}");
foreach($ids as $id)
{
$a = explode(":", HM_GetAddress($id));
$b = explode(":", $sid);
if($a[0] == $b[0])
{
return $id;
}
}
return 0;
}

function CreateVariableByName($id, $name, $type)
{
$vid = @IPS_GetVariableIDByName($name, $id);
if($vid === false)
{
$vid = IPS_CreateVariable($type);
IPS_SetParent($vid, $id);
IPS_SetName($vid, $name);
IPS_SetInfo($vid, "this variable was created by script
#".$_IPS['SELF']);
}
return $vid;
}
```

HM_RequestStatus

```
bool HM_RequestStatus (int $InstanceID, string $Parameter)
```

PARAMETERS

InstanceID	ID of the device to be switched

Parameter

VALUE	DESCRIPTION
STATE	Status of an actuator
LEVEL	Dim value
...	An overview for all requestable "readable" parameters can be taken from the documentation of the Homematic Datapoints under Tipps & Tricks.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Requests the status variable with the name **Parameter** of the device with ID **InstanceID**.



The **Parameter** value you can derive based on the name of the state variable. In the parameter list some examples are given.



This fuction loads the radio traffic significantly.
It should be used solely to update critical devices to IP Symcon start. A requery of the runtime is not required because the HomeMatic forward all status changes directly to IP Symcon.



For battery powered units, only the stored state of the CCU/ LAN adapter will be required. Line-powered equipment can be queried directly by wireless.

EXAMPLE

```
// This requests an update of all parameters for all Homemeatic
instances.
// This strains the wireless communication extremely and is not
recommanded as general update.
$ids = IPS_GetInstanceListByModuleID("{EE4A81C6-5C90-4DB7-AD2F-
F6BBD521412E}");
echo "Geräte: ".sizeof($ids)."\n";
foreach($ids as $id)
{
$svs=IPS_GetStatusVariableIdents($id);
if(sizeof($svs) > 0) {
if(@HM_RequestStatus($id, $svs[0]) === false) {
echo "Fehler: ".IPS_GetLocation($id)."\n";
}
}
}
```

HM_WriteValueBoolean

```
bool HM_WriteValueBoolean (int $InstanceID, string $Parameter, bool
$Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Parameter	Parameter that should be set. The name of the parameter can be taken from the data point list or as "Ident" in the tab "Status Variables" in IP-Symcon.
Value	true or false

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the **Parameter** of the device with ID **InstanceID** to value **Value**.



The **Parameter** value you can derive based on the name of the state variable. In the parameter list some examples are given. Based on the type of variable you can also see the needed **HM_WriteValue** function.



Additional datapoints can be taken from the HomeMatic data point list.



If the error message **Failure** occurs, this is usually due to a bad wireless connection.

EXAMPLE

```
// Switch device on
HM_WriteValueBoolean(12345, "STATE", true);

// Lock device
HM_WriteValueBoolean(12345, "INHIBIT", true);

// Switch on display light
HM_WriteValueBoolean(12345, "BACKLIGHT", true);
```

HM_WriteValueFloat

```
bool HM_WriteValueFloat (int $InstanceID, string $Parameter, float
$Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Parameter	Parameter that should be set. The name of the parameter can be taken from the data point list or as "Ident" in the tab "Status Variables" in IP-Symcon.
Value	floating point number that should be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the **Parameter** of the device with ID **InstanceID** to value **Value**.



The **Parameter** value you can derive based on the name of the state variable. In the parameter list some examples are given. Based on the type of variable you can also see the needed **HM_WriteValue** function.



Additional datapoints can be taken from the HomeMatic data point list.



If the error message **Failure** occurs, this is usually due to a bad wireless connection.

EXAMPLE

```
// Dim device to 25%. Values allowed between 0.0 and 1.0
HM_WriteValueFloat(12345, "LEVEL", 0.25);

// Set the activation duration to 1 hour. Values allowed between 0 and
85825945.6 in seconds
HM_WriteValueFloat(12345, "ON_TIME", 3600);
```

HM_WriteValueInteger

```
bool HM_WriteValueInteger (int $InstanceID, string $Parameter, int
$Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Parameter	Parameter that should be set. The name of the parameter can be taken from the data point list or as "Ident" in the tab "Status Variables" in IP-Symcon.
Value	Whole number that should be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the **Parameter** of the device with ID **InstanceID** to value **Value**.



The **Parameter** value you can derive based on the name of the state variable. In the parameter list some examples are given. Based on the type of variable you can also see the needed **HM_WriteValue** function.



Additional datapoints can be taken from the HomeMatic data point list.



If the error message **Failure** occurs, this is usually due to a bad wireless connection.

EXAMPLE

```
// Set the number of alarms to the value 25. Values allowed between 0
and 255.
HM_WriteValueInteger(12345, ALARM_COUNT, 25);

// Set the number of service to the value 78. Values allowed between 0
and 255.
HM_WriteValueInteger(12345, SERVICE_COUNT, 78);
```

HM_WriteValueString

```
bool HM_WriteValueString (int $InstanceID, string $Parameter, string
$Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Parameter	Parameter that should be set. The name of the parameter can be taken from the data point list or as "Ident" in the tab "Status Variables" in IP-Symcon.
Value	string that should be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the **Parameter** of the device with ID **InstanceID** to value **Value**.



The **Parameter** value you can derive based on the name of the state variable. In the parameter list some examples are given. Based on the type of variable you can also see the needed **HM_WriteValue** function.



Further data points can be found in the HomeMatic data point list.

EXAMPLE

```
// Transfers the value "Hello World". Any string allowed.
HM_WriteValueString(12345, SUBMIT, "Hello World");
```

HomeMatic HCU (Beta)

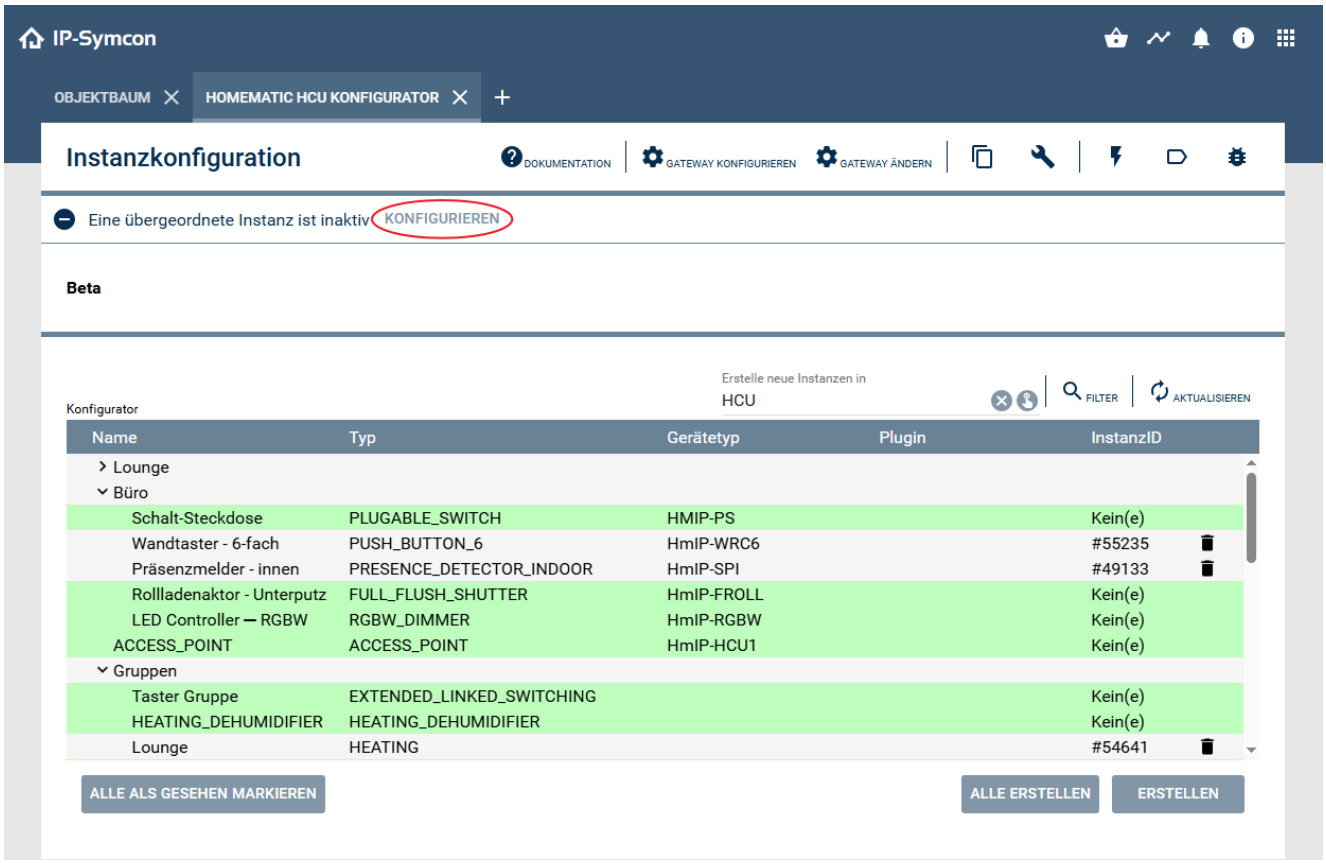
Require: Symcon >= 8.1

This module is currently still in the beta phase. To use it, developer mode must be activated via HCUWeb. Once this mode is activated, eQ-3 can no longer provide technical support. More information: Connect API FAQ

Due to a limitation in the HCU Connect API, some button channel assignments do not match those displayed in the Homematic IP app.

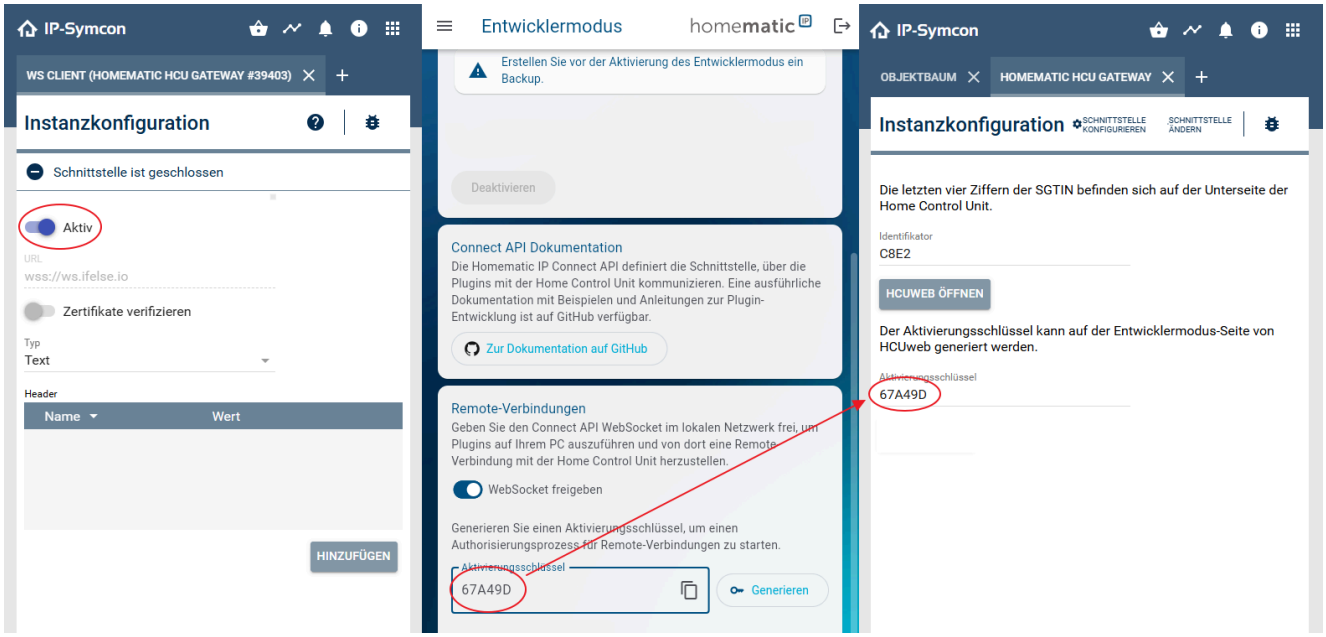
INTEGRATION IN IP-SYMCON

The HCU can be integrated via Device search. To do this, "Homematic HCU Discovery" must be selected as the system. The Discovery instance then offers the creation of a Homematic HCU Configurator. Once the configurator has been created, the individual devices and groups can be integrated as described below.



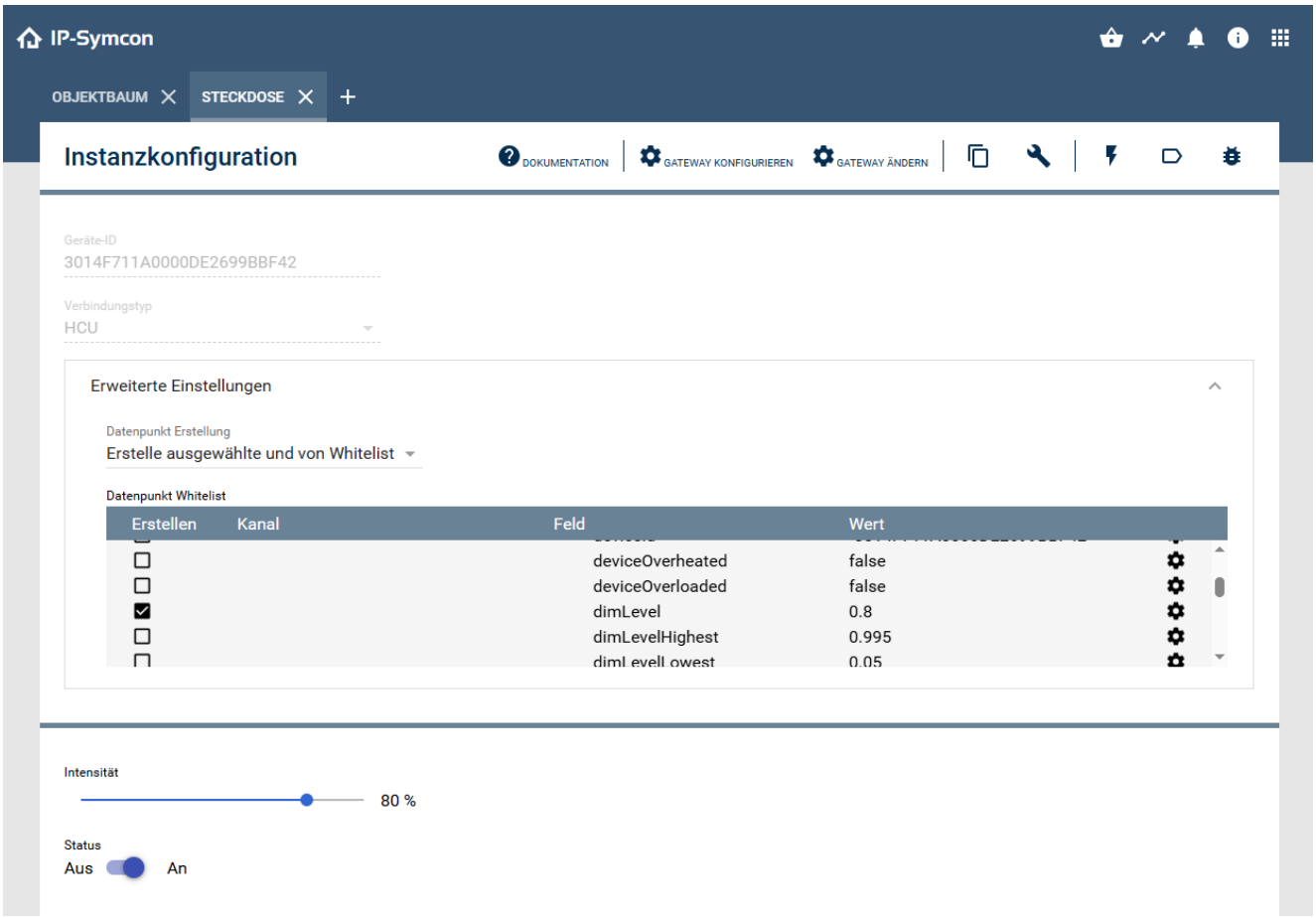
Within the configurator the configuration page of the "HomeMatic HCU Gateway" can be opened in via "Configure". The activation key from the HCUWeb must be entered here. To do this, developer mode must be activated in the HCUWeb and the WebSocket must be

enabled. Once the key has been generated and entered in the gateway, the process can be completed by clicking "Apply changes".



If the HCU needs to be registered again at a later date, a new activation key can be generated and entered in the corresponding field. To complete the process, click the 'Register Again' button in the gateway's expert options.
Various devices are now visible in the configurator and divided into rooms. The groups are summarized under a separate item. After selecting a device or a group, an instance can be created with "Create" and then the configuration page can be called up via "Configure".

DEVICE CONFIGURATION



On the configuration page, the connection type is displayed in addition to the device ID. Currently there is only the connection via the HCU. The "Advanced settings" allow you to influence the creation of a device's data points. The option "Create only selected" is active by default. If the device is not known to Symcon, all data points are created.

OPTION	DESCRIPTION
Create only selected	If the Symcon device is known, the data points created can be limited to those that are most relevant for the function of the device.
Create only selected and from whitelist	In addition to the data points selected by Symcon, further data points can be selected from a list of all data points.
Create only from whitelist	Each data point to be created can be selected individually.
Create all	All available data points are created. This option is selected if the Symcon device is not known

All switchable values of the device are displayed at the bottom of the action area.

The configuration of groups is structured identically

Image Grabber

The Image Grabber saves webcam images from a specific address. These are automatically added as media objects and are therefore also available to the Visualization.

As of IP-Symcon 4.0, the media created by the Image Grabber are only written to the hard disk after closing IP-Symcon. The intelligent caching protects the hard disk/ flash memory and the content is only stored in the working memory. The current content can be accessed at runtime via IPS_GetMediaContent.

INTEGRATION IN IP-SYMCON

Set up a new instance via "Add object" -> "Add instance" -> Manufacturer: "(Other)".
The following settings can be made in the configuration tab.

With a MJPG stream it is possible to display a webcam with moving images.

In contrast to most other instances, this module does not need a parent instance

SETTINGS

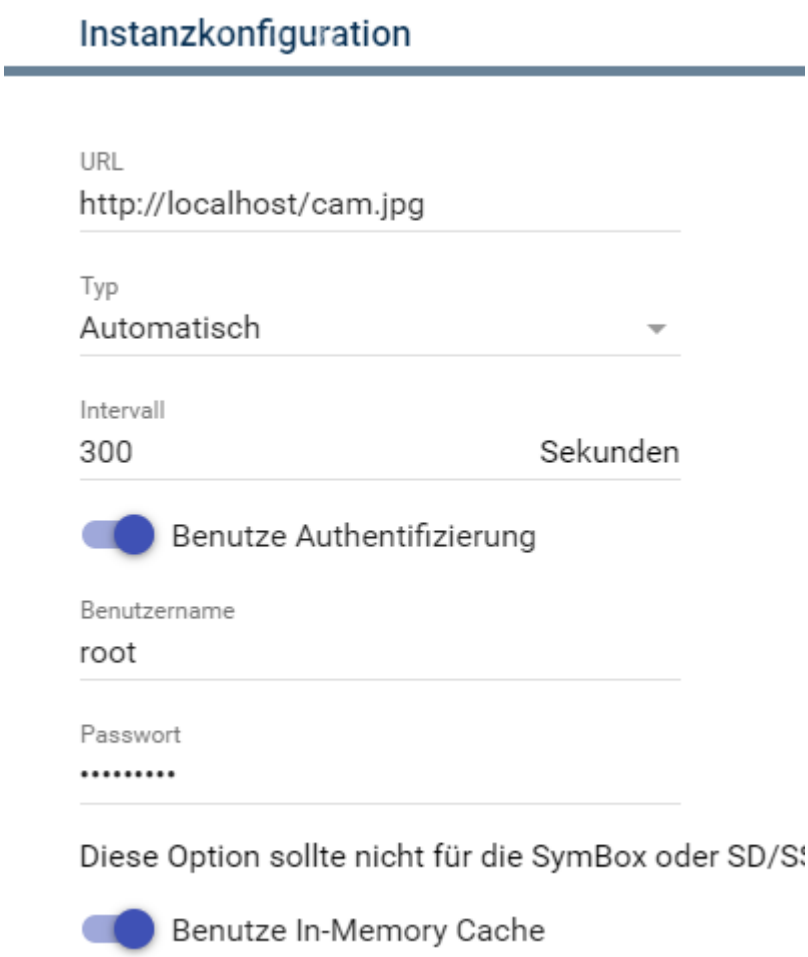
OPTION	DESCRIPTION
Address	The address of the webcam and the path/URL to the image. See list below
Type	In which format the image should be saved. If the file extension in the address is not ".jpg", ".gif", ".bmp", ".png", the file format must be specified explicitly.
Interval	In which cycle in seconds the image should be queried
Use authentication	Whether authentication should be used: Yes/No
Username	Username for authentication
Password	Password for authentication

ADDRESSES

All common manufacturers/models are listed under the link listed here. The standard stream addresses can also be seen under the respective model, which may have to be integrated via

a Stream.
List of manufacturers

EXAMPLE



IG_UpdateImage

```
bool IG_UpdateImage (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Updates the image of the device with the ID InstanceID.

EXAMPLE

```
// The picture of instance 12345 is updated
IG_UpdateImage(12345);
```

IPS-868

With its bidirectional radio components, the IP-Symcon radio system 868 supplements areas of application that are often not satisfactorily covered by other systems. This includes consumption recording via the S0 input, presence control via the PresenceControlModule including a tracker, analysis of the room air via an air quality sensor, and the control of LED strips.



The following devices are supported by IP-Symcon:
supported devices

CONTENT

Installation
Integration in IP-Symcon
Configurator
Configuration

INSTALLATION

The **USB Gateway** is connected to the PC via USB. The necessary drivers are available from SiLabs for download . After its installation, the gateway is ready for use.
The **LAN Gateway** is connected to the PC via LAN. By default, DHCP and port 5000 are set up. If an individual IP address is required, the IP-Symcon "Network Configuration Tool" is required. This is available for download . A simple description of how to configure the gateway is available. The gateway can then be reached via the corresponding IP address and port.



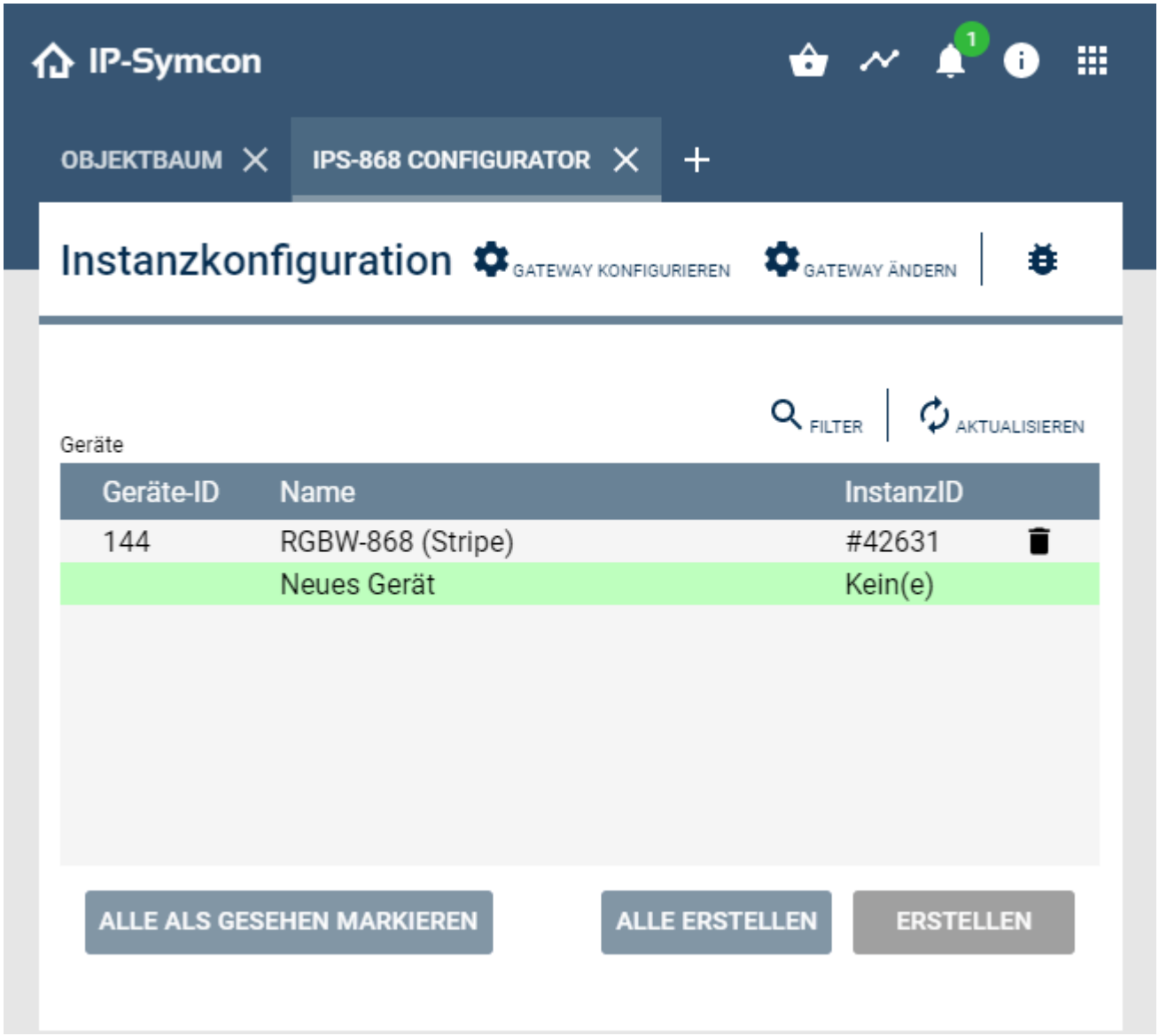
If no IP-Symcon house is printed on the front of the LAN gateway, it is an older model of the gateway. The installation instructions for this model can be found here .

INTEGRATION IN IP-SYMCON

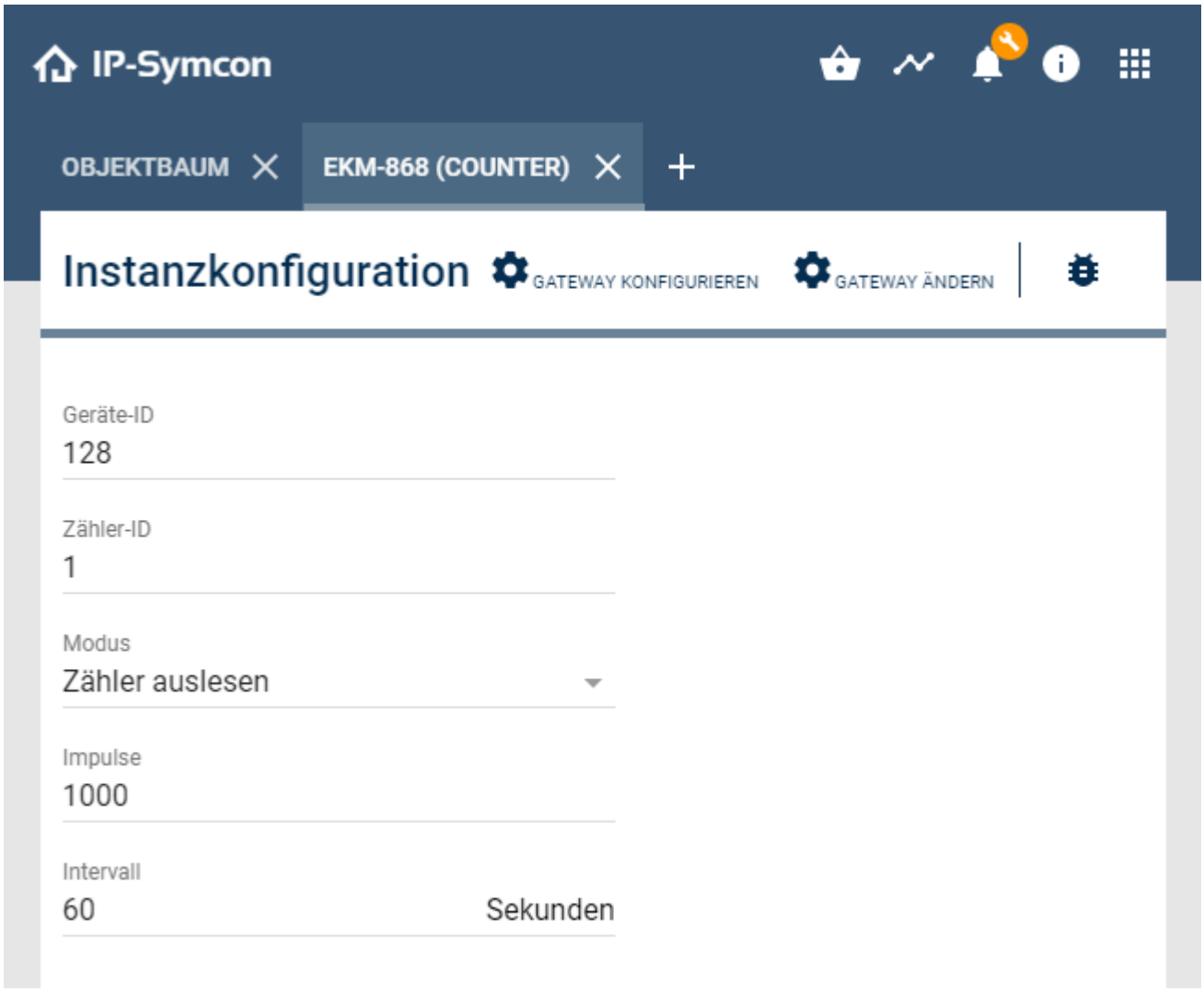
When using the LAN gateway, this can be integrated via the Device Search. To do this, "IPS-868 Discovery" must be selected as the system. The discovery instance then offers to create an IPS-868 Configurator. After the configurator has been created, the individual devices can be integrated there as described below.
The USB gateway is created directly as a configurator because it cannot be found via the Device search. For this purpose, an "IPS-868 Configurator" is created in the Object Tree. When configuring the interface, the "IPS-868 USB Gateway" mode must be selected and then the appropriate port selected and opened.

CONFIGURATOR

All IPS-868 devices can be created as Instance using the IPS-868 configurator. To do this, "New device" must be selected from the list and "Create" clicked on.



New Instances are created in the Object Tree in the main category. These created instances can then be renamed accordingly and sorted elsewhere. A newly added device can be configured via "Configure".
Here, on the one hand, the device ID must be entered here and, depending on the device, other device-specific properties may have to be defined. The exact procedure and the device ID can be found in the documentation of the respective device. (device list)



CONFIGURATION

Various properties can be set on the configuration page of the device.
These can be found in more detail in the respective device documentation.

DEVICE INSTANCE	DEVICE DESCRIPTION
ACC-868	----------
AKM-868 (Tracker)	AKM-868
ANA-868 (Analog-Output)	ANA-868
ANA-868 (Analog-Input)	ANA-868
DATA-868	----------
EKM-868 (Counter)	EKM-868
FD-868 (Display Input)	FD-868
FD-868 (Display Output)	FD-868
FD-868 (Display)	FD-868
JKM-868 (LevelJET)	JKM-868-LevelJET
JKM-868 (ThermoJET)	JKM-868-ThermoJET
LGS-868 (Air Quality Sensor)	LGS-868

DEVICE INSTANCE	DEVICE DESCRIPTION
RGBW-868 (Stripe)	RGBW-868
RGBW-868 (Stripe Input)	RGBW-868
SERVO-868	SERVO-868
SERVO-868 (Input)	SERVO-868
TFK-868 (Input)	----------
WDT-868 (WatchDog)	WDT-868

AKM-868

DESCRIPTION

The presence control module AKM 868 consists of a small remote control in keychain format in an oval housing (size: 55 *44* 15 mm) and a repeater. The remote has a button with a dual function: short and long press. This enables two devices (e.g. light and a garage door) to be switched on or off. The special feature of this handset is that it sends a 'ping' approximately every 40 seconds. It consists of a unique ID, which is analyzed by IP Symcon. If the tracker is no longer within range, a corresponding variable is set to FALSE after five minutes. In this way, certain scenarios can be started when leaving or entering the apartment.
The battery lasts at least six months - according to current knowledge even more than a year (pure ping operation). Data is transmitted via the 868 MHz frequency band.

INSTALLATION

In order to be able to receive data, an 868 Gateway (LAN or USB) must first be connected and set up. "AKM-868 (Tracker)" must be added in IP-Symcon. The parent instances (e.g. the gateway) are created automatically.

CONFIGURATION

To use the device in IP Symcon, add a new AKM-868 instance (manufacturer: IP Symcon/ ProJet):

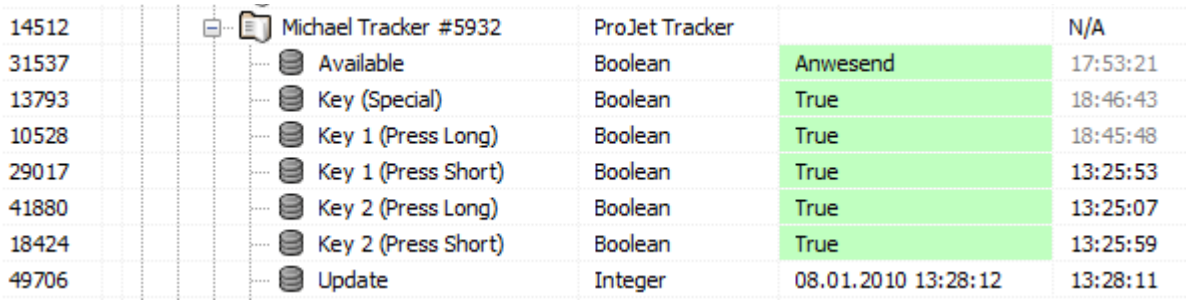
DeviceID:
As standard 0 is specified. This means that of all the repeaters data can be received. To specifically select a device, e.g. the 160 can be selected.
Jumper assignment (view in the device, antenna above, bottom 2 jumpers pairs):
160: both open
161: right closed
162: left closed
163: both closed

TrackerID:
To change the ID, the button must be held down while inserting the battery.
The ID is automatically generated by the tracker. This is retained even after a battery change.
Most of the time the TrackerID is: 8191 (Default)

If a 'ping' is received, 'Available' is automatically set to TRUE, and in addition, the time stamp in 'Update' updated. If no 'Ping' signals are received for more than five minutes (fixed, unchangeable), 'Available' is set to FALSE. Additionally, the key shots are evaluated. (For 'Special' both buttons are pressed simultaneously - old version: with two keys.)



Note that the key variables always remain TRUE and only the timestamp changes.



14512	Michael Tracker #5932	ProJet Tracker		N/A
31537	Available	Boolean	Anwesend	17:53:21
13793	Key (Special)	Boolean	True	18:46:43
10528	Key 1 (Press Long)	Boolean	True	18:45:48
29017	Key 1 (Press Short)	Boolean	True	13:25:53
41880	Key 2 (Press Long)	Boolean	True	13:25:07
18424	Key 2 (Press Short)	Boolean	True	13:25:59
49706	Update	Integer	08.01.2010 13:28:12	13:28:11

TIPS & TRICKS

The receiver sensitivity of the repeater can be set to "HI / LO". To do this, the jumper next to the ATMEL processor must be bridged (low reception sensitivity).
If the transmission power is low (LO), the approximate position of the remote control can be determined with a second or third AKM. Of course, this is highly dependent on the respective local conditions. However, a distinction - such as basement, garden or upper floor - should be possible.
In order to significantly increase the range of the remote control, the SMD antenna can be replaced by a wire antenna (*Lambda district: 87mm*). This is available from us on request.

TROUBLESHOOTING

If the tracker is not recognized despite a new battery, the TrackerID must be reassigned (see above) and then re-taught in the tracker module with "Search".

ANA-868

The analogue-digital/ digital-analogue converter module has 2 analogue inputs (10V, input impedance 10kOhm) and 2 analogue outputs (10V/10mA).

INSTALLATION

In order to read in the data from measured voltages or to be able to set voltages at the outputs, an 868 Gateway must first be connected and set up. An "ANA-868 (analog input/ analog output)" module must be added in IP-Symcon. The parent instances (e.g. the gateway) are created/connected automatically.

CONFIGURATION

VALUE	DESCRIPTION
Device Address	The ID:184 is specified by default. If there are more controllers in a system, a new ID must be assigned. The jumper assignment can be found in the operating instructions.
Channel Address	(1-4) Indicates which of the two output/input channels should be used.

PJ_RequestStatus

```
bool PJ_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the current values for the device with the ID **InstanceID** and writes them to the corresponding status variables.



This function places a significant load on radio traffic.
It should only be used if the internal timer is not to be used to implement uneven intervals. A constant query of the values may block the radio traffic of other participants.

EXAMPLE

```
//Queries the current values of instance 12345.
PJ_RequestStatus(12345);
```

PJ_SetVoltage

```
bool PJ_SetVoltage (int $InstanceID, float $Volt)
```

PARAMETERS

InstanceID	Device ID
Volt	Voltage to which the output is to be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the output to **Volts** for the device with ID **InstanceID**. Values higher than 10V are limited automatically.

EXAMPLE

```
//Sets the output voltage to 5.30 volts
PJ_SetVoltage(12345, 5.30);
```

EKM-868

DESCRIPTION

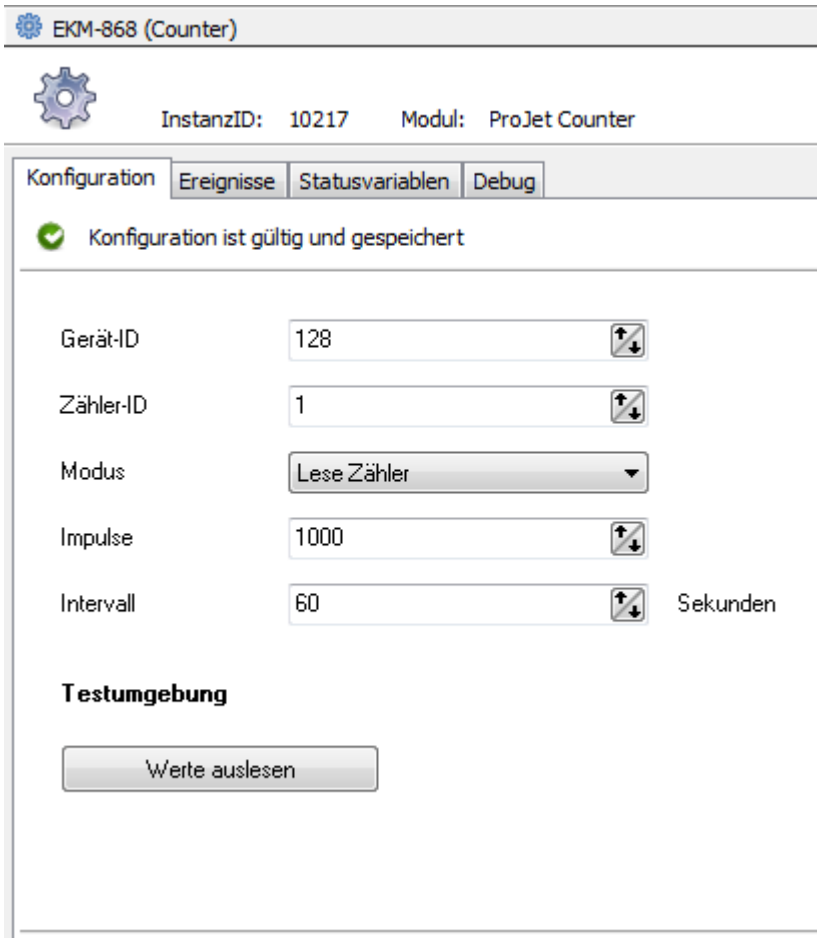
The power control module EKM 868 is a 4-channel module with 32-bit counter, which counts of four inputs and records their duration. The bidirectional data transmission takes place via a USB radio transceiver operating in the 868 MHz frequency band. It is used to visualize and projection of consumption costs in conjunction with IP Symcon. The count remains valid as long a power supply consists (9V = power adapter). A clear (reset) of the count by a power outage is detected by IP Symcon, a battery backup is therefore not required. The running PC you only need to provide a single value with a timestamp and to generate graphics. A special feature of the EKM 868 is that in addition the current power (consumption) is recorded.

INSTALLATION

To query counter data, first the virtual Comport driver on Windows must be installed and the USB radio transceiver must be connected.

CONFIGURATION

Now add a new EKM 868 (manufacturer: IP Symcon/ ProJet):



VALUE	DESCRIPTION
DeviceID	Default 128 is predetermined. When two EKM 868 are present in a system, in the device a jumpers can be set: the two lower pins (direction Header, upper PIN: NC). The new ID is then 129. ATTENTION: If red LEDs are installed (1 to 4), the ID is 129 by open Jumper! Devices with the ID 130/131, etc. are available on request.
CounterID	The ID corresponds to the respective inputs (channels): 1,2,3,4
Modus: Read counter	It is read out the count (Variable: Counter) and the time interval in 1/100 seconds between two impulses (Current).
Modus: Read on-time	The duty cycle (closed contact) is returned in 1/10 seconds (at 32 bits this results in a period of more than 13 years!)
Modus: Read off-time	The off-time (contact open) is returned in 1/10 seconds
Impulse	At electricity meters the impulses are entered per kWh. The unit of the variable "Current" is then also kWh. At 0 is the unit 1/100 second - ideal for e.g. to determine the volumetric flow of gas or water meter.
Timer activated	With a set tick values are queried in the specified time interval. This is limited to a minimum of 60 sec.
Test Center	A command for output values will be sent to the EMK. When properly installed, the variables in the object tree are updated accordingly.



Attention:
The inputs are not galvanically isolated - they must not exceed 5V. At plan view, the common ground of the four inputs is the right connection. At every entrance, an RC element was installed: 1K to as pull-up and 100nF mass. A switching e.g. with a reed contact to ground causes a count. The power supply must not exceed = 12V. Higher voltages or reversed polarity lead to destruction of the device. Then, an external protective circuit (Z-diode, resistor) is to provide.

TIPS & TRICKS

The internal timing (Current) has a resolution of 16-bit and 1/100 overclocking.
This means it can be detected pulse sequences of a maximum length of 655 seconds (almost 11 minutes). In case of an overflow, an -0- is returned.
Internally, the following formula is used to convert the current power in watt:
P = (3600 *1000) / (Current / 100* Impulses) [Watt]
Where "Current", is the time between two pulses in 1/100 seconds
To display the current energy consumption (energy-light), in practice intervals of 20 minutes revealed to be optimal.
Power supply: Please be sure to use the supplied = 9V power supply - no "bell transformer"
Internally, a low drop 3.3 V voltage transformer is installed.
The input voltage ranges from 5.0 V = to = 10.0 V. The inner contact of the DC socket is the positive pole.

If the counter e.g. is erased by a power failure, so this is not relevant because IP Symcon detects it automatically and takes it into account when generating the graphics



Problem: LED lights permanently
Solution: If necessary, swap the wires of the respective counter input (electronically)
or wait until the counter continues to rotate and releases the reed contact (mechanical).

TECHNICAL DATA

Dimensions (without antenna): 70 *70* 25 mm
Weight: 100g
Power supply: 9V bis 12V DC / 100 mA / Connector: DC socket for hollow plug 2,1mm
4 *S0 Inputs / 4* 32-Bit counter (TTL – 5V)
maximum count frequency: 100 Hz
minimum pulse length: 20 msec (debounced internally for mechanical contacts)
Protection class: IP 20
Radio module: 868MHz (PC radio interface: USB-T 868 is not included)

PJ_RequestStatus

```
bool PJ_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the current values for the device with the ID **InstanceID** and writes it to the appropriate state variables.



This function loads the radio traffic significantly.
It should be used only when the internal timer is not to be used to implement irregular intervals. A constant interrogation of the values can block the radio traffic from other participants.

EXAMPLE

```
PJ_RequestStatus(12345); //Query values
```

FD-868

FUNCTION

The radio display 868 is used to display text messages on a 2 x 16-character LC display.
In addition, three LEDs and a buzzer can be switched.
Optional is an IO box (relay board) with two outputs (10A/250V) and two inputs (TTL +5V) available.
The bidirectional data transmission takes place via an USB radio transceiver operating in the 868 MHz frequency band.

INSTALLATION

The radio display 868 is used to display text messages on a 2 x 16-character LC display.
In addition, three LEDs and a buzzer can be switched.
Optional is an IO box (relay board) with two outputs (10A/250V) and two inputs (TTL +5V) available.
The bidirectional data transmission takes place via an USB radio transceiver operating in the 868 MHz frequency band.

CONFIGURATION

The following properties can be changed as follows (not by command): hold left button on the display and switch the device on. Follow the further instructions on the screen.
Value	Description
Device-Address	By default, the ID: 1 is set. If more displays are available in a system, a new ID will be awarded.
Group-Address	Here all displays e.g. of the ground floor will be addressed together. This feature is not available in the first generation.
Broadcast-Address	Fixed: 255 - all displays will be addressed together
Key click	On/ Off



Only one device with the same ID may be present in a radio network.

Optional is an IO box (relay board) with two outputs (10A/250V) and two inputs (TTL +5V) available. RJ-11 socket assignment (Display):
1 – Input 1 / TTL
2 – GND
3 – Output 1 / Open Collector
4 – Output 2 / Open Collector
5 – VCC /+5V
6 – Input 2 / TTL
IO-BOX socket on the terminal of 'Output 1'

TIPS & TRICKS

Scripts for a menu control are available in our forums:
Link to the forum post

You can not turn off the key-click in the first generation.
It signaled to the operator that the command from the USB radio transceiver has been adopted (Acknowledgment).

PJ_Backlight

```
bool PJ_Backlight (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for on, FALSE for off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the display backlight in the FD-868 with the ID InstanceID on or off.

EXAMPLE

```
PJ_Backlight(12345, true);
```

PJ_Beep

```
bool PJ_Beep (int $InstanceID, int $TenthsOfSeconds)
```

PARAMETERS

InstanceID	ID of the device to be switched
TenthsOfSeconds	Length of the tone in 1/10 seconds

RETURNS

default

DESCRIPTION

Outputs a sound in the FD-868 with ID **InstanceID**.

EXAMPLE

```
PJ_Beep($id_lcd, 50); // Beep for 0,5 seconds
```

PJ_LCDText

```
bool PJ_LCDText (int $InstanceID, int $Line, string $Text)
```

PARAMETERS

InstanceID	ID of the device to be switched
Line	Line number of the display (1 or 2)
Text	Lines of text, 16 characters maximum - the rest is cut off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a text to the FD-868 of the ID **InstanceID**.
Since the 'comma' is used as a control character, it must not appear in the text.
The command will automatically replace it in a 'point'.
Special characters such as 'äöü' may not occur.

EXAMPLE

```
PJ_LCDText($id_lcd, 1, "ALARM - Meldung"); // Return text in line 1
```

PJ_SwitchLED

```
bool PJ_SwitchLED (int $InstanceID, int $LED, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
LED	1
Status	True = On / False = Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Turns on or off the light-emmiting diode of FD-868 with the ID **InstanceID**.

EXAMPLE

```
PJ_SwitchLED($id_lcd,1,False); // Red
PJ_SwitchLED($id_lcd,3,True); // Green
```

PJ_SwitchMode

```
bool PJ_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the Relay to be switched
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the relay with the ID InstanceID to value Status.

EXAMPLE

```
PJ_SwitchMode($id_lcd, True); //Turn on Relay
```

Device list

SUPPORTED GATEWAYS

GATEWAY		DESCRIPTION
Symcon LAN-T 868	Product Information	IP gateway with LAN connection
Symcon USB-T 868		USB gateway with USB port

SUPPORTED COMPONENTS

COMPONENTS		DESCRIPTION
AKM-868	Product Information	AttendanceControlModule
EKM-868	Product Information	EnergyControlModule (4-channel meter module)
FD-868		Wireless display
JKM-868 LevelJet		LevelJET
JKM-868 ThermoJET		ThermoJET
LGS-868	Product Information	AirQualitySensor
RGBW-868	Product Information	LED stripe controller (12V/24V)
SERVO-868		4-way servo controller
Tracker 868	Product Information	Key fob (used in combination with the AKM-868)
WDT-868		Watch-Dog-Timer

JKM-868 LevelJET

DESCRIPTION

The JKM-868 with connected LevelJET is used to read out the individual values of the LevelJET level indicator.

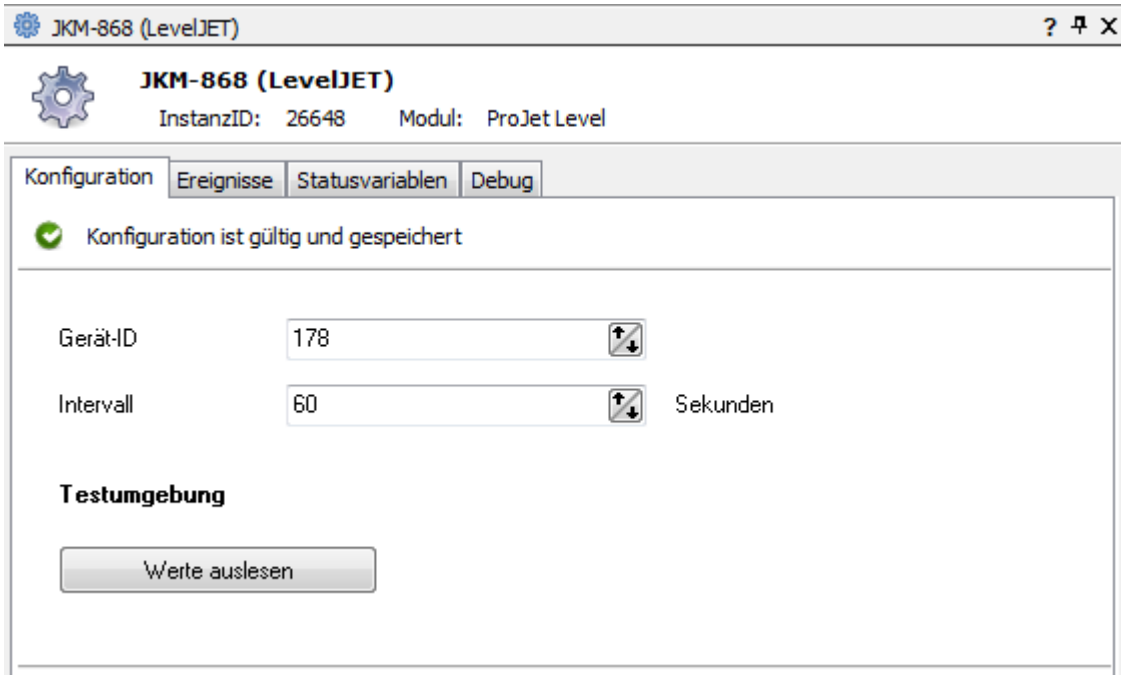
INSTALLATION

In order to be able to receive data, the virtual COM port driver from Silicon Labs (CP210x) must first be installed on Windows and the USB-T 868 radio transceiver must be connected. In IP-Symcon, “JKM-868 (LevelJET)” must be added. The parent instances (e.g. the gateway) are created automatically.
Alternatively, the 868 LAN gateway can also be used. A client socket with the appropriate settings must be selected as the “parent instance”.

CONFIGURATION

The DeviceID defaults to 166 and must be added to the address of the respective device. Example: the LevelJET has the address 12 - so 178 must be entered here. The address is parameterized in LevelJET according to the manufacturer's instructions.

VARIABLE	DESCRIPTION
Distance	Distance in cm
Output (1)	State of the 1st Relay
Output (2)	State of the 2nd Relay



The distance value (input) is queried cyclically in the set “interval”. The status of both outputs cannot be actively queried. It is sent automatically after the distance thresholds set in the LevelJET have been exceeded or undershot.

TIPS & TRICKS

In order to calculate the volume in liters, own PHP scripts must be created.

PJ_RequestStatus

```
bool PJ_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the current values for the device with the ID **InstanceID** and writes them to the corresponding status variables.



This function places a significant load on radio traffic.
It should only be used if the internal timer is not to be used to implement uneven intervals. A constant query of the values may block the radio traffic of other participants.

EXAMPLE

```
//Queries the current values of instance 12345.
PJ_RequestStatus(12345);
```

JKM-868 ThermoJET

DESCRIPTION

The JKM-868 with attached ThermoJET is used to read out the individual values of the ThermoJET temperature controller.

INSTALLATION

In order to be able to receive data, the virtual COM port driver from Silicon Labs (CP210x) must first be installed on Windows and the USB-T 868 radio transceiver must be connected. In IP-Symcon, “JKM-868 (ThermoJET)” must be added. The parent instances (e.g. the gateway) are created automatically.
Alternatively, the 868 LAN gateway can also be used. A “client socket” with the appropriate settings must be selected as the “parent instance”.

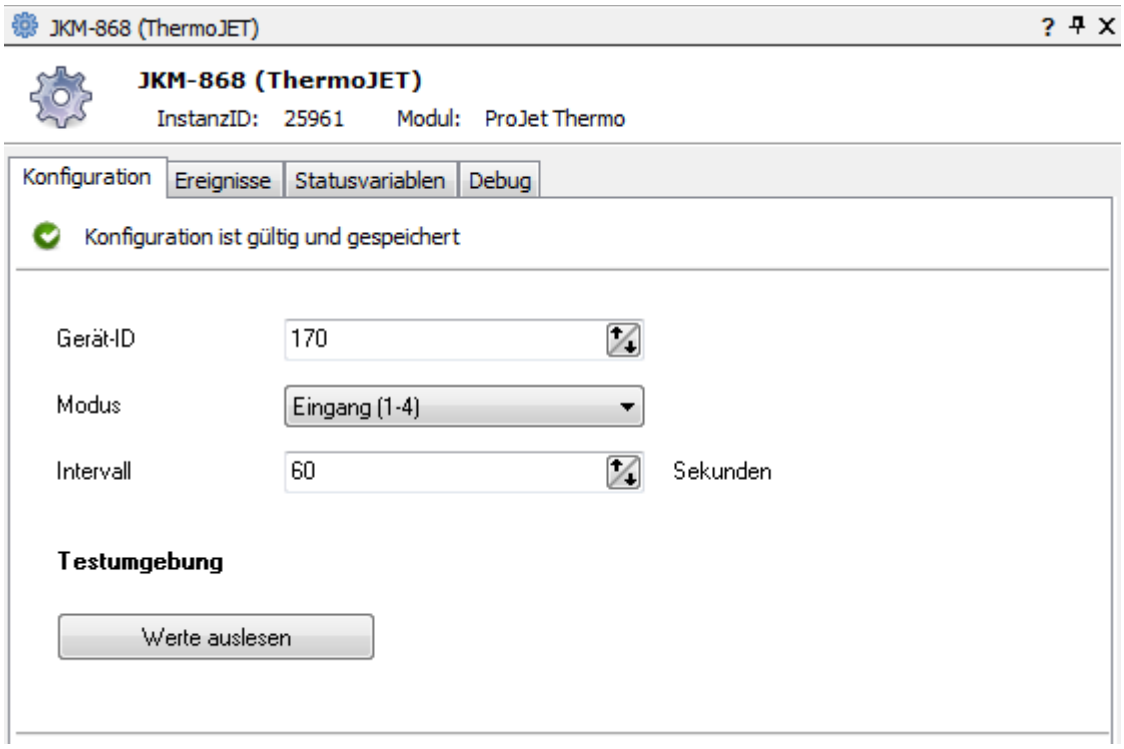
CONFIGURATION

The DeviceID defaults to 166 and must be added to the address of the respective device. Example: The ThermoJET has the address 17 - so 183 must be entered here. The ThermoJET has 8 inputs - if more than 4 temperature guides are connected, every second JKM instance with the mode setting: “Input (5-8)” must be added to IP-Symcon.

VARIABLE	DESCRIPTION
Output (1)	State of the 1st Relay
Output (2)	State of the 2nd Relay
Temperature (1..4)	Temperature value in °C

Note: If no sensor is connected, the output is -99.9°C

If a second instance is created for input 5-8, the new variables are still called temperature 1-4. These can be renamed if required. The variables for outputs 1+2 are the same in both modes.



The four temperature channels (inputs) are queried cyclically in the set “interval”. The status of both outputs cannot be actively queried. This is sent automatically after the temperature thresholds set in the ThermoJET have been exceeded or undershot.

PJ_RequestStatus

```
bool PJ_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the current values for the device with the ID **InstanceID** and writes them to the corresponding status variables.



This function places a significant load on radio traffic.
It should only be used if the internal timer is not to be used to implement uneven intervals. A constant query of the values may block the radio traffic of other participants.

EXAMPLE

```
//Queries the current values of instance 12345.
PJ_RequestStatus(12345);
```

LGS-868

The air quality sensor (LGS-868) analyses room air and summarizes the concentration of mixed gases in the air into a usable value (ppm - parts per million).

Technical information and advice on proper ventilation can be found in the following brochure: Download

INSTALLATION

In order to receive data, an IPS-868 gateway (LAN or USB) must first be connected and set up. The parent instances (e.g. the gateway) are created automatically.

CONFIGURATION

To use the device in IP-Symcon, a new LGS-868 instance (manufacturer: IP-SymCon/ ProJet) must be added.
If parent instances do not yet exist, they are created automatically.

VALUE	DESCRIPTION
Device address	The ID:176 is specified by default. If several controllers exist in a system, a new ID must be assigned. The jumper configuration can be found in the operating instructions.

Only one device with the same ID may be present in one wireless network at a time.

APPLICATION

The device returns an integer value that returns the measured ppm (parts per million/ millionth share). The official limits can be found in the table below.
The LGS-868 has 3 LEDs (green/yellow/red), which are automatically switched at the IP-Symcon limit values (450-1000/ 1001-1500/ 1501-2100). The automatic can be switched off by setting “Limit (yellow)” and “Limit (red)” to 0 in the configuration page.

PPM	AIR QUALITY
>2100	Very bad (heavily polluted room air; ventilation required)
2100... 1501	Bad (heavily polluted room air; ventilation required)
1500... 1001	Medium (polluted room air; ventilation recommended)
1000... 801	Satisfactory

PPM	AIR QUALITY
800... 601	Good
600... 450	Excellent



Extremely high values above 2100 ppm are rare and are based, for example, on possible alcohols in cleaning products or typical odors in a restroom.

PJ_RequestStatus

```
bool PJ_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the current values for the device with the ID **InstanceID** and writes them to the corresponding status variables.



This function places a significant load on radio traffic.
It should only be used if the internal timer is not to be used to implement uneven intervals. A constant query of the values may block the radio traffic of other participants.

EXAMPLE

```
//Queries the current values of instance 12345.
PJ_RequestStatus(12345);
```

PJ_SetLEDs

```
bool PJ_SetLEDs (int $InstanceID, bool $Green, bool $Yellow, bool $Red)
```

PARAMETERS

InstanceID	ID of the device to be switched
Green	Turns off the green LED
Yellow	Turns off the yellow LED
Red	Turns off the red LED

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the LEDs on the controller with the ID **InstanceID**.

EXAMPLE

```
//Turns the green and yellow LED on and the red LED off.
PJ_SetLEDs(12345,true,true,false);
```

RGBW-868

Manual as a PDF

DESCRIPTION

The LED stripe controller is used to control LED stripes to allow different lighting scenarios and moods in a residential environment.

INSTALLATION

To communicate with the controller, you must first install the virtual comport driver and connect the USB radio transceiver. In IP Symcon 'RGBW(Stripe)' must be added. The parent instances (e.g. the gateway) are automatically created. Please check if the right COM-Port was selected.

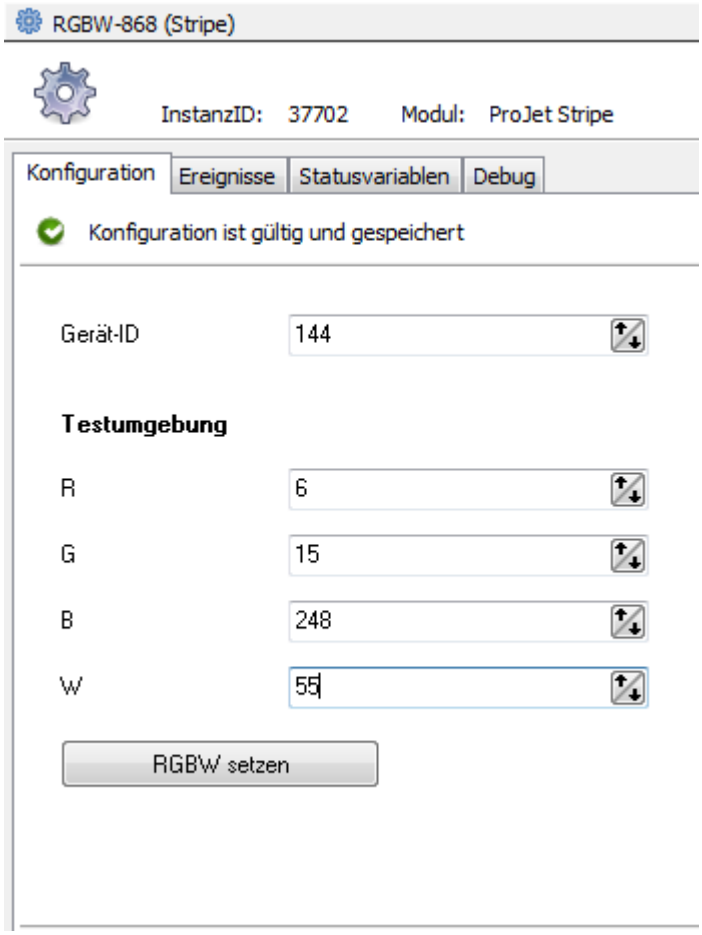
CONFIGURATION

VALUE	DESCRIPTION
Device-Address	By default, the ID: 144 is given. If more controllers are present in a system, a new ID will be awarded. Please check the jumper assignment in the instruction manual. Note: If necessary, the addresses: 149,150 and 151 are occupied by the system
Broadcast-Address	Fixed: 254 - all controller will be addressed together

Only one device with the same ID may be present in a radio network.

A FIRST TEST

For a quick check of basic functionality click the right mouse button on the 'RGBW(Stripe)' module, then 'Open Object' (or double-click) and choose a color, or, if connected, a brightness value for white-channel. Then click 'Insert'. The LED will light stripe in the desired color.



CHECK IN CASE OF PROBLEMS

When switching on the 12V supply the LED stripe flashes briefly.
The red LED on the board is also lit up brightly and then flashes further (very) slightly.
All jumpers open means address 144.
If commands are received, the LED lights as well.
If commands are sent, the Rx/Tx LED lights on the USB-T
The splitter 'ProJet Gateway' must be connected to the correct COM-Port

TIPS & TRICKS

In addition to the lighting scenes also for example flashing warning or flashing notes can be dislayed. These can be acknowledged by both of the controller pushbutton inputs as positive (as YES) or negative (as NO) since the used radio system is bidirectional. In IP Symcon the 'FD(Stripe) Input' module has to be added.

PJ_DimRGBW

```
bool PJ_DimRGBW (int $InstanceID, int $R, int $RTime, int $G, int
$GZeit, int $B, int $BZeit, int $W, int $WZeit)
```

PARAMETERS

InstanceID	ID of the device to be switched
R	Brightness value 0 (off) to 255 (on, maximum brightness)
RTime	In seconds in which the desired brightness level is to be achieved
G	Brightness value 0 (off) to 255 (on, maximum brightness)
GZeit	In seconds in which the desired brightness level is to be achieved
B	Brightness value 0 (off) to 255 (on, maximum brightness)
BZeit	In seconds in which the desired brightness level is to be achieved
W	Brightness value 0 (off) to 255 (on, maximum brightness)
WZeit	In seconds in which the desired brightness level is to be achieved

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a dimming command to the RGBW-868 controller of the ID **InstanceID**.



The value for the time is divided internally into two sections. When times are specified over 255 seconds, automatically all times that were less than 255 are set to 255. This area is intended for slow scenarios to wake up.

EXAMPLE

```
//dim in 2 seconds on yellow
PJ_DimRGBW(12345,255,2,255,2,0,0,0,0);
```

PJ_RunProgram

```
bool PJ_RunProgram (int $InstanceID, int $Program)
```

PARAMETERS

InstanceID	ID of the device to be switched
Program	0 – all RGBW off; 1 – all RGBW at the last brightness value; 2- blink slowly; 3- blink rapidly; 4- flash slowly ; 5- flash rapidly; 6- fade slowly

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a program command to the RGBW-868 controller of the ID **InstanceID**.

EXAMPLE

```
//let LEDs blink rapidly
PJ_RunProgram(12345, 3);
```

PJ_SetRGBW

```
bool PJ_SetRGBW (int $InstanceID, int $R, int $G, int $B, int $W)
```

PARAMETERS

InstanceID	ID of the device to be switched
R	Brightness value 0 (off) to 255 (on, maximum brightness)
G	Brightness value 0 (off) to 255 (on, maximum brightness)
B	Brightness value 0 (off) to 255 (on, maximum brightness)
W	Overall brightness value 0 (off) to 255 (on, maximum brightness)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets an **RGB** color value with brightness **W** on the controller with ID **InstanceID**.

EXAMPLE

```
// Switch to yellow at full brightness
PJ_SetRGBW(12345,255,255,0,255);
```

SERVO-868

DESCRIPTION

The 4-way servo controller is used to control commercially available model-making servos with the RJ connector system. As a second function, up to four buttons can be connected as control inputs.

INSTALLATION

In order to communicate with the controller, the USBT-868 or LAN-T-868 gateway must have been installed beforehand. The instance "Servo-868" must be added in IP-Symcon. The parent instances such as the "ProJetGateway" are created automatically. It must be checked whether the correct virtual COM port was selected by the USB gateway. With the LAN-T-868 Gateway, the correct IP address and port must be specified in the "Client Socket".

CONFIGURATION

The default device address is ID:160. If there are more controllers in a system, a new ID must be assigned. The jumper configuration can be found in the operating instructions.

A FIRST TEST

For a quick first function test, the device must be connected to a suitable power source. It is important to pay attention to the correct polarity. A servo is then connected to the servo output with a flat cable and a plug with a contact spacing of 2.54 mm. As a rule, the brown wire is "minus", red is "plus" and orange is the "PWM" signal. In IP-Symcon, the configuration page can be opened by right-clicking (or double-clicking) on the "Servo-868" instance and then on "Open object". A value between 0 and 255 can be entered in the respective "Channel" in the 'Test Environment'. After "Set Channel n" the servo moves to a new position.

TIPPS & TRICKS

en.wikipedia.org/wiki/Servo_(radio_control)
mikrocontroller.net/articles/Modellbauservo_Ansteuerung
electronicsplanet.ch/Roboter/Servo/intern/intern.htm

PJ_DimServo

```
bool PJ_DimServo (int $InstanceID, int $Channel, int $Value, int
$Steps)
```

PARAMETERS

InstanceID	ID of the device to be switched
Channel	Servo channel 1..4
Value	Value from 0..255
Steps	Steps from 0 (instant) through 1 (slow) to 255 (fast)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets a servo channel **Channel** on the controller with the ID **InstanceID** to the value **Value** in the speed **Steps**.

EXAMPLE

```
// Immediately sets servo channel 1 to the value 255
PJ_DimServo(12345, 1, 255, 0);
```

PJ_SetServo

```
bool PJ_SetServo (int $InstanceID, int $Channel, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Channel	Servo channel 1..4
Value	Value from 0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets a servo channel **Channel** on the controller with ID **InstanceID** to **Value**.

EXAMPLE

```
PJ_SetServo(12345, 1, 255);
```

WDT-868

DESCRIPTION

The watchdog timer is used to monitor computer systems and individual programs. A "sign of life" is sent cyclic from the PC to the WDT. If this is not received correctly, can via a potential free contact e.g. a siren ca be activated or a hardware reset can be executed. Additionally, the UM-relay (5A/230V) can be turned on and off, or to be addressed by a timer (65,535 sec). The unit also has a pushbutton input at leisure. The status is shown as a variable in IP Symcon.

INSTALLATION

To receive data, previously on Windows must be the [virtual comport driver from Silicon Labs (CP210x)] installed[1] and the T-USB 868 radio transceiver connected. In IP Symcon 'WDT-868' must be added. The parent instances (e.g. the gateway) are automatically created.

CONFIGURATION

In the DeviceID 165 will be set (Position 0). This can be changed by turning the coding switch to 174 (Position 9).



VARIABLE	DESCRIPTION
Input	False: Button is open / True: Button is pressed
Status	False: Relay de-energized / True: Relay energized
Timer	Time in seconds, after the relay changes its status

After the timer expires, the WDT sends the new state of the relay. The state variable is updated accordingly (no simulation). By an active timer the LED blinks every second.
A brief flash signals a radio commands in IP Symcon-868 radio system[/i]



Only one device with the same ID may be present in a radio network.

TIPPS & TRICKS

To the two contacts of the switch e.g. an external switch (doorbell, motion sensors, photocells, etc.) can be connected.

PJ_SwitchDuration

```
bool PJ_SwitchDuration (int $InstanceID, bool $Status, int $Dauer)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	**TRUE** for On, **FALSE** for Off
Dauer	On or off times in seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the relay from the WDT with ID **InstanceID** to value **Status** for **Duration** seconds.



The IP Symcon 868 radio system is bidirectional and thus the WDT sends the new status of the relay after the expiration of the timer.

EXAMPLE

```
//The following command turns on the relay for 60 seconds:
PJ_SwitchDuration(12345, True, 60);
```

PJ_SwitchMode

```
bool PJ_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sends a switching command to the WDT-868 module of the ID InstanceID.

EXAMPLE

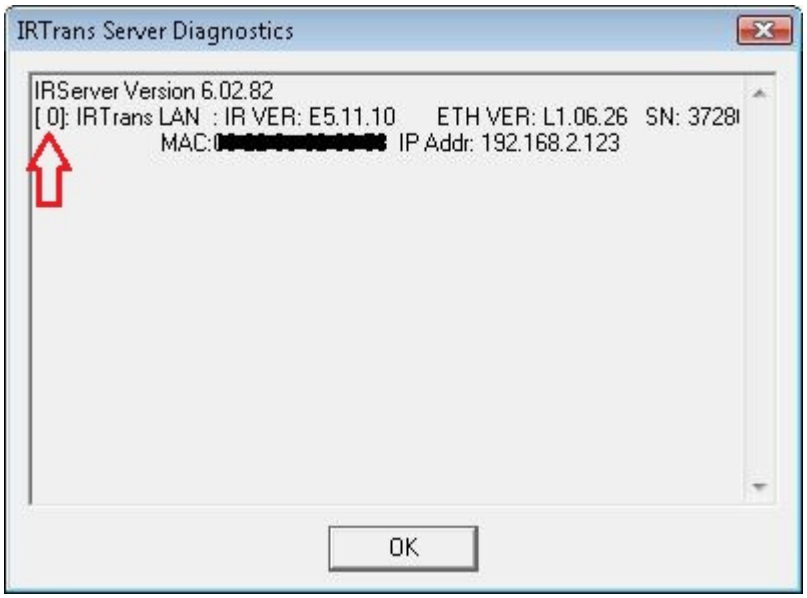
```
PJ_SwitchMode(12345, True);
```

IR Trans

Using infrared signals allows an inexpensive and easy integration of existing audio and video equipment into home automation. Moreover, e.g. light scenarios using the infrared remote control can be activated or blinds can be lowered. For this, in principle, only one transmitter and receiver diode is necessary - as you want as a finished device or for crafting ...
Manufacturer: IRTrans: "… turns your PC into a programmable remote control"
irtrans.de/de/index.php

INSTALLATION

To send/receive IR signals over IRTrans, a IRTrans compatible device, such as USB, LAN or WLAN variant is needed. Once the devices operate within the IRTrans Tray software, you can begin the installation within IP-Symcon. www.irtrans.de/download/setup.exe
With the IRTrans Tray Icon (right click -> Diagnostics), the IDs of the devices can be queried, that were found by the IRTrans server application, which can then be controlled via IP-Symcon. The following figure shows an IRTrans LAN device that has the ID 0.

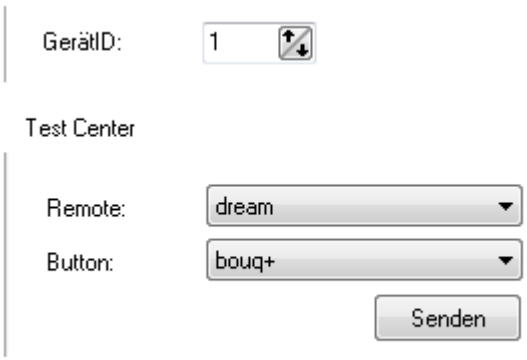


PROGRAMMING THE REMOTE CONTROL

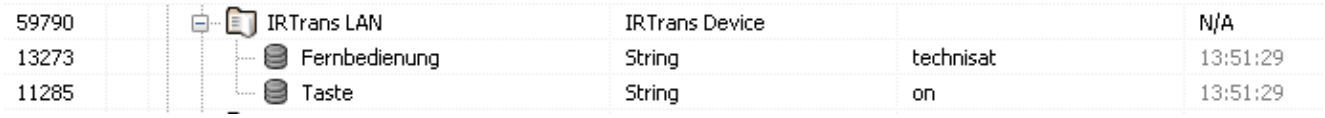
The manufacturer recommends the learning of the commands via the "IRTrans GUI Client". First, you must enter the name of the remote control (e.g. "sat") and the appropriate key (e.g. "power"). A teaching about IP-Symcon is not possible.



In the configuration of the IRTrans instance the device ID must be specified, which can be removed from the upper diagnosis window. Each instance can control a single device. Any number of IRTrans instances can be created. If you want to send to multiple devices, multiple PHP commands can simply be sent to multiple instances in a row. The function can be checked in the Test Center.



The IRTrans module provides two variables in the object tree. With these variables it can reacted on the currently depressed remote control and button. For that, a triggered event must be created that responds to the update of the "key" variable.



Device list

SUPPORTED GATEWAYS

GATEWAY	DESCRIPTION
IRTrans Ethernet Module	IR-Trans LAN adapter
IRTrans USB Module	IR-Trans USB adapter
IRTrans WiFi Module	IR-Trans wireless adapter

SUPPORTED COMPONENTS

The list of supported infrared devices can be found on the homepage HERE .



The SymBox only supports IR-Trans devices with IR database.

IRT_SendOnce

```
bool IRT_SendOnce (int $InstanceID, string $RemoteControl, string
$Button)
```

PARAMETERS

InstanceID	ID of the device to be switched
RemoteControl	Name of the remote control, which is/ was registered in the remote database.
Button	Name of the remote control, which is/ was registered in the remote database.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a remote command from the instance with ID **InstanceID**. The values remote conrol and button specify, which command should be sent. These must be set up accordingly in the client software (IRTrans Tray).

EXAMPLE

```
IRT_SendOnce(37279,"ccf", "v+");
```

KEBA

KEBA offers various wallboxes/charging stations. These can be read out via ModBus RTU/TCP. A connection with IP-Symcon is possible via LAN or RS485, depending on the version.

The following devices are supported by IP-Symcon:
Supported components

INSTALLATION

In order to be able to use KEBA wallboxes/charging stations in IP-Symcon, a connection via Ethernet or RS485 to the wallbox must be available.

IP-SYMCON INTEGRATION

First, a "ModBus Device" instance must be added within the IP-Symcon object tree. The DeviceID must be set to 255 and the IP address of the wallbox must be entered in the following dialog. The port is 502 by default. Both pieces of information can be viewed on the web interface/app of the WallBox/charging pole.

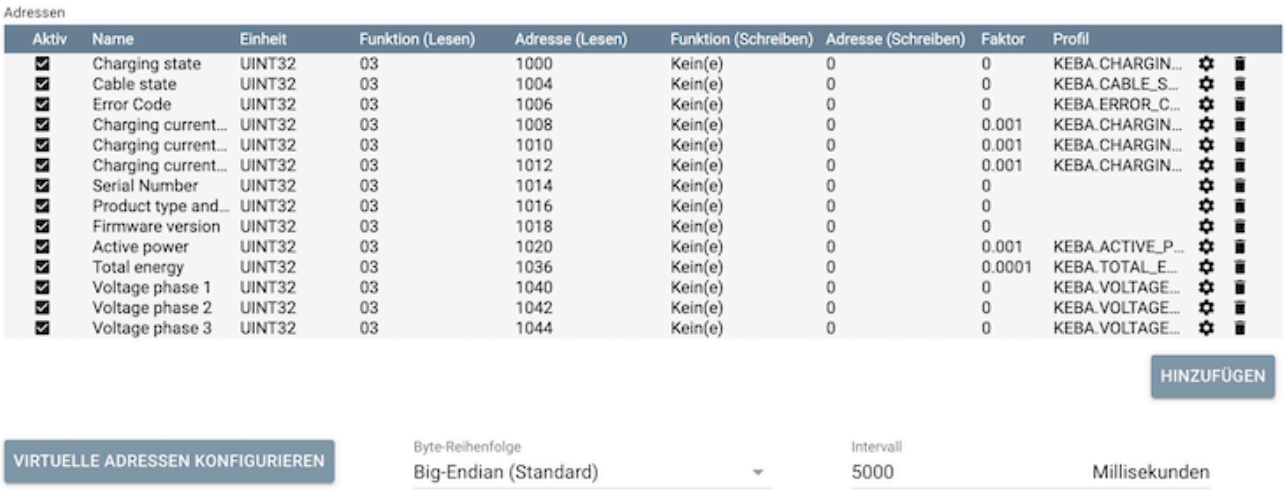






The ModBus template for KEBA wallboxes/charging stations can then be downloaded. This contains the entire configuration of the ModBus device. After downloading, the "KebaModBus_vx.json" can be loaded via "Import". All ModBus addresses of the common KEBA wallboxes/charging stations are then set up.

ModBus template for KEBA wallboxes/charging stations:
Download



ADD ADDRESSES

If further addresses are to be added, this can be done via "Add".
Depending on the wallbox version, individual addresses may need to be deactivated/activated. This can be controlled via the Active column.

PROTOCOL DESCRIPTION



Protocol description for KEBA wallboxes/charging stations:

KEBA KeContact P30: Download

KEBA KeContact P40: Download

Device List

Require: Symcon >= 7.0

SUPPORTED COMPONENTS

SYSTEM	MODEL	DESCRIPTION
KEBA KeContact P30	All, without a-series	Supported in all versions
KEBA KeContact P40	All	Supported in all versions

KNX

KNX is a wired as well as radio based system. A connection with IP-Symcon is possible via a serial or KNX IP interface. For the configuration of the system an engineering tool software (short "ETS") is needed.



The following devices are supported by IP-Symcon:
Supported components (excerpt)
KNX Data Secure and KNX IP Secure are supported!

SUMMARY

Introduction
Installation
- XML Export in ETS
- Integration IP-Symcon
  - Integration without KNX Data Secure
  - Integration with KNX Data Secure
  - Set up single actuators
  - Automatic detection of feedback address

Examples
- Example: Send temperature values to the bus
- Better roller shutter control with KNX Shutter

Tips and Tricks
- Conversion of group addresses
- KNX with QNAP, Synology or Docker
- Read out states
- Output of group addresses

Legacy
- OPC Export in ETS
- Assignment EIS type / DPT

INTRODUCTION

KNX is a network of home and building control systems according to EN 50090 and ISO/IEC 14543. It controls, for example, heating, lighting, blinds, ventilation and security systems across different trades. The system usually uses cables (designation e.g. J-Y (St) Y 2x2x0.8 EIB or YCYM 2x2x0.8) with two wire pairs (red-black and white-yellow). The KNX system is supplied by a power supply with 30 V nominal voltage [DC voltage]. This voltage supplies the bus couplers via which each KNX device communicates with the other networked KNX devices. The data exchange between the KNX devices takes place via telegrams.
The programming of the participants and the assignment of the group addresses is done with the Engineering Tool Software (ETS). The ETS is provided by the umbrella organization Konnex and ensures the trouble-free cooperation of components from different manufacturers.
The connection to IP-Symcon is done via a serial or IP gateway.
As a rule, all serial gateways "PEI type 10″ with FT1.2 protocol (interfaces of type PEI 16 do

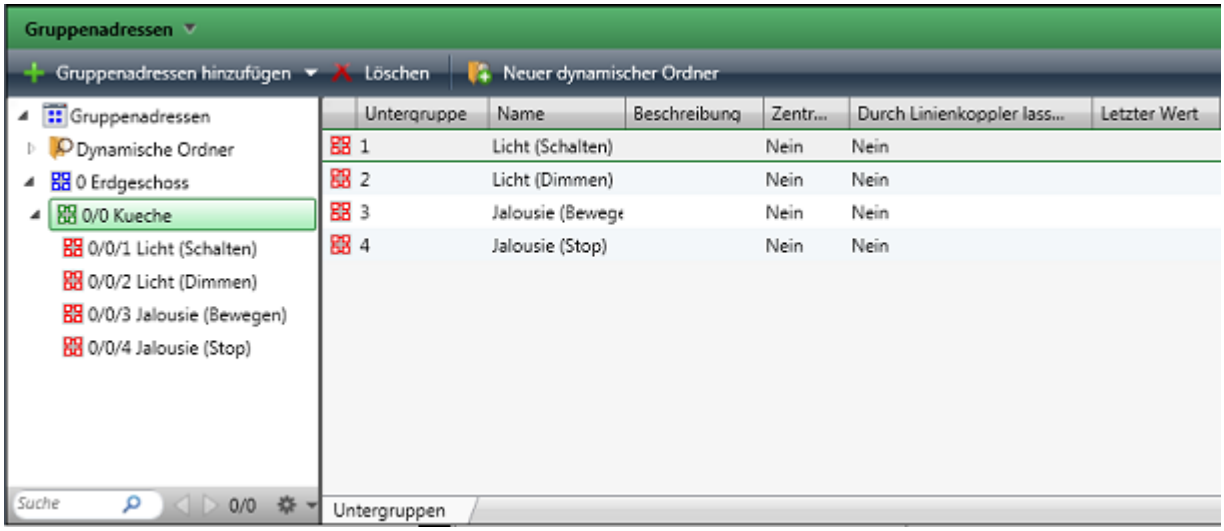
not work) are supported and all KNX/IP gateways that offer the tunneling protocol. If a KNX gateway meets these requirements and does not work, Symcon support should be contacted. The problem will be solved as soon as possible.
Since IP-Symcon 6.4, KNX Data Secure and KNX IP Secure are also fully supported. In addition, all KNX IP Secure Interfaces also support the new KNX/IP over TCP, which is much more robust than the older KNX/IP over UDP.

INSTALLATION

When all KNX devices are correctly set up and connected, they must be configured with the ETS. The three-stage address assignment is used for this purpose.

Currently only three-level addresses (e.g. 10/2/5) are supported. Two-level or free addresses are not supported by IP-Symcon at the moment. About the Conversion any two-level address can be controlled via the translated three-level address

If the installation was done by an electrician, he should include the OPC/XML export explained below and the associated file of type XML.
To use KNX in IP-Symcon, either a serial FT1.2 gateway or a compatible KNX/IP gateway is required. The setup of the different gateways differs slightly. A KNX instance can be created directly in IP-Symcon and the gateway can be configured from there. Alternatively, the XML export from the ETS can be used.

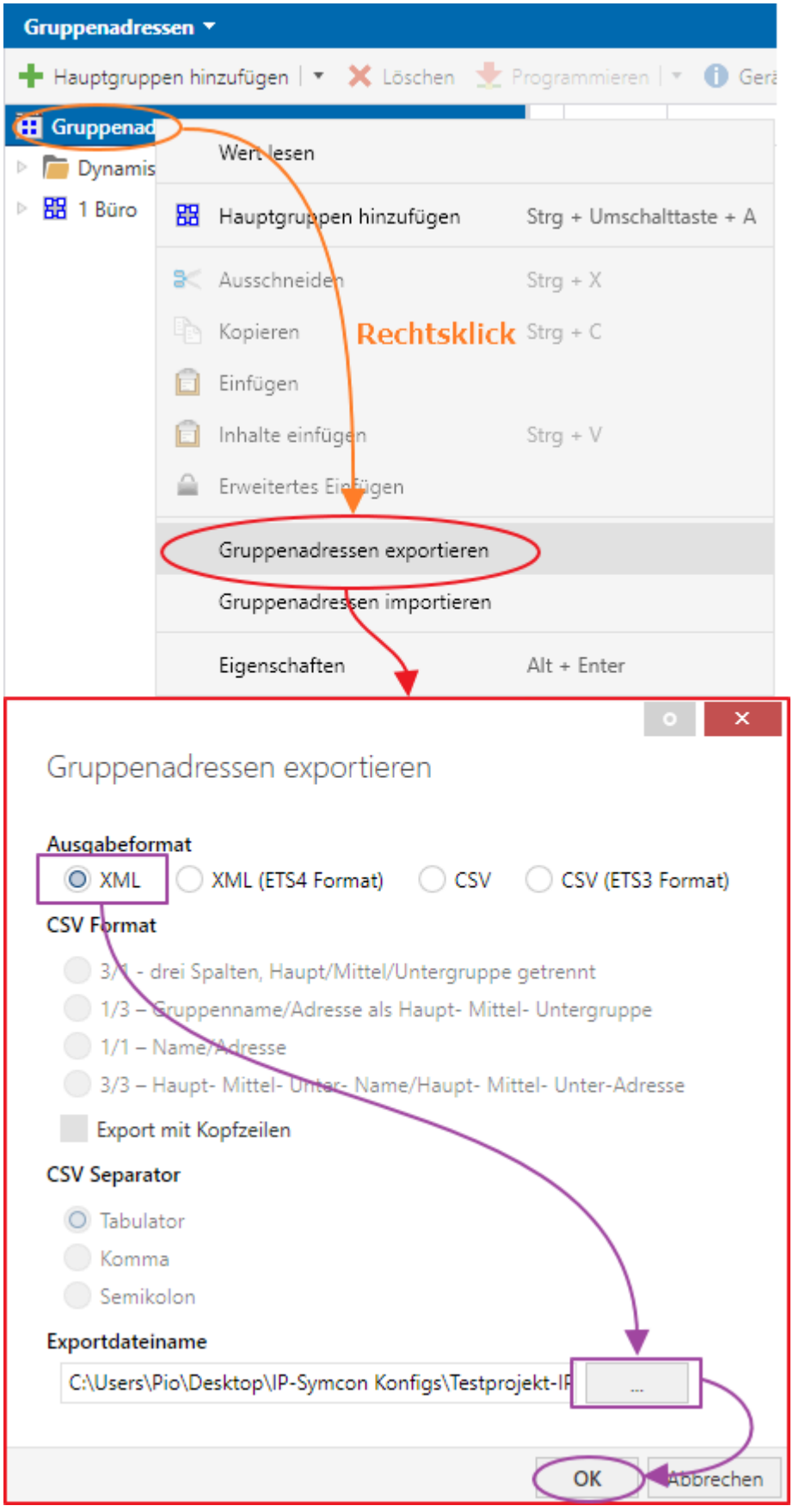


Using the XML export with the IP-Symcon configurator allows a fast integration of all KNX components.

From IP-Symcon 5.3 the export as XML is recommended, because it supports the new KNX DPT instances

XML EXPORT TO ETS

An XML export can be saved within the ETS in the opened project via right click on the group addresses.

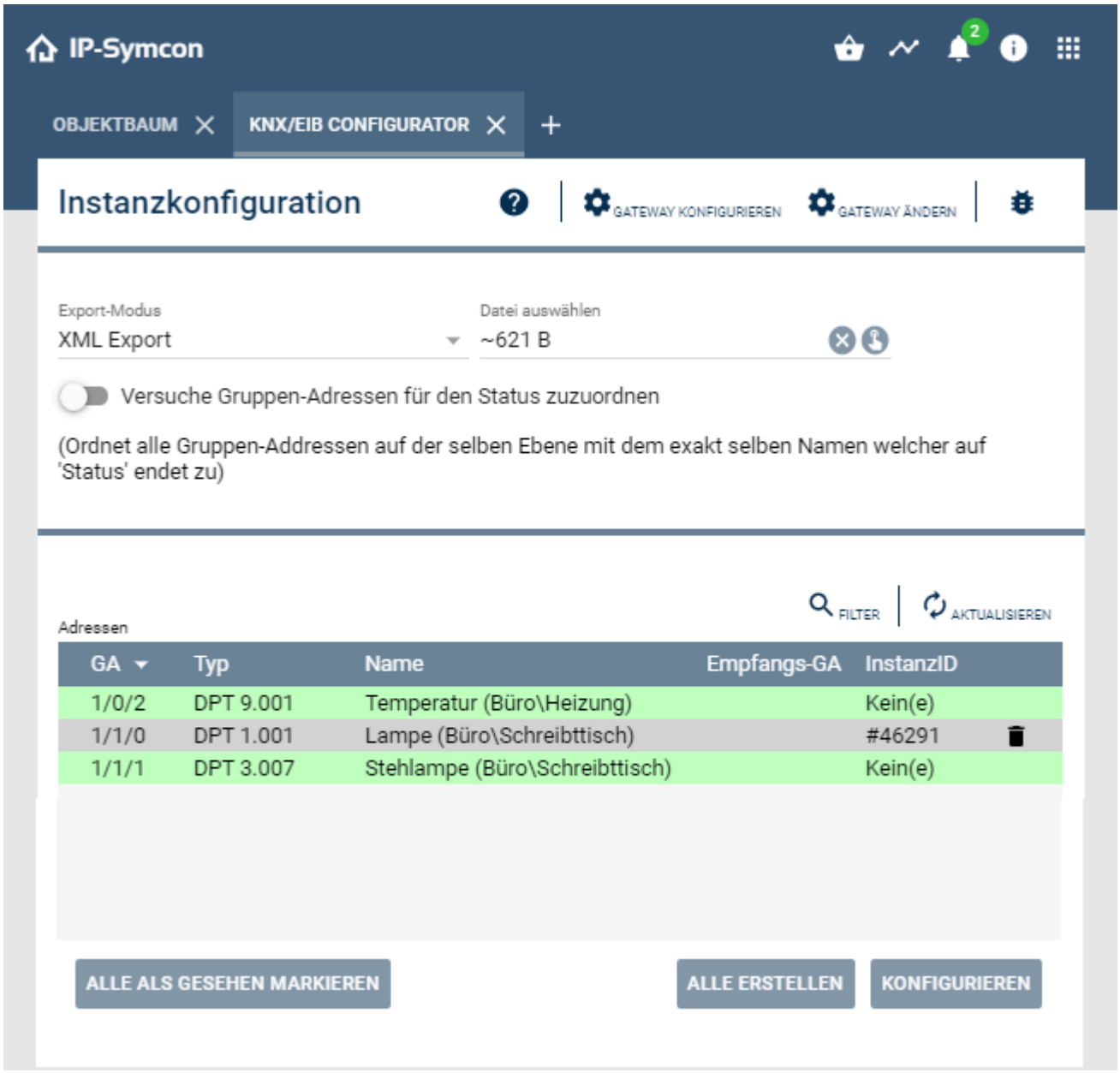


Video tutorial for export:
From version 6.0.0

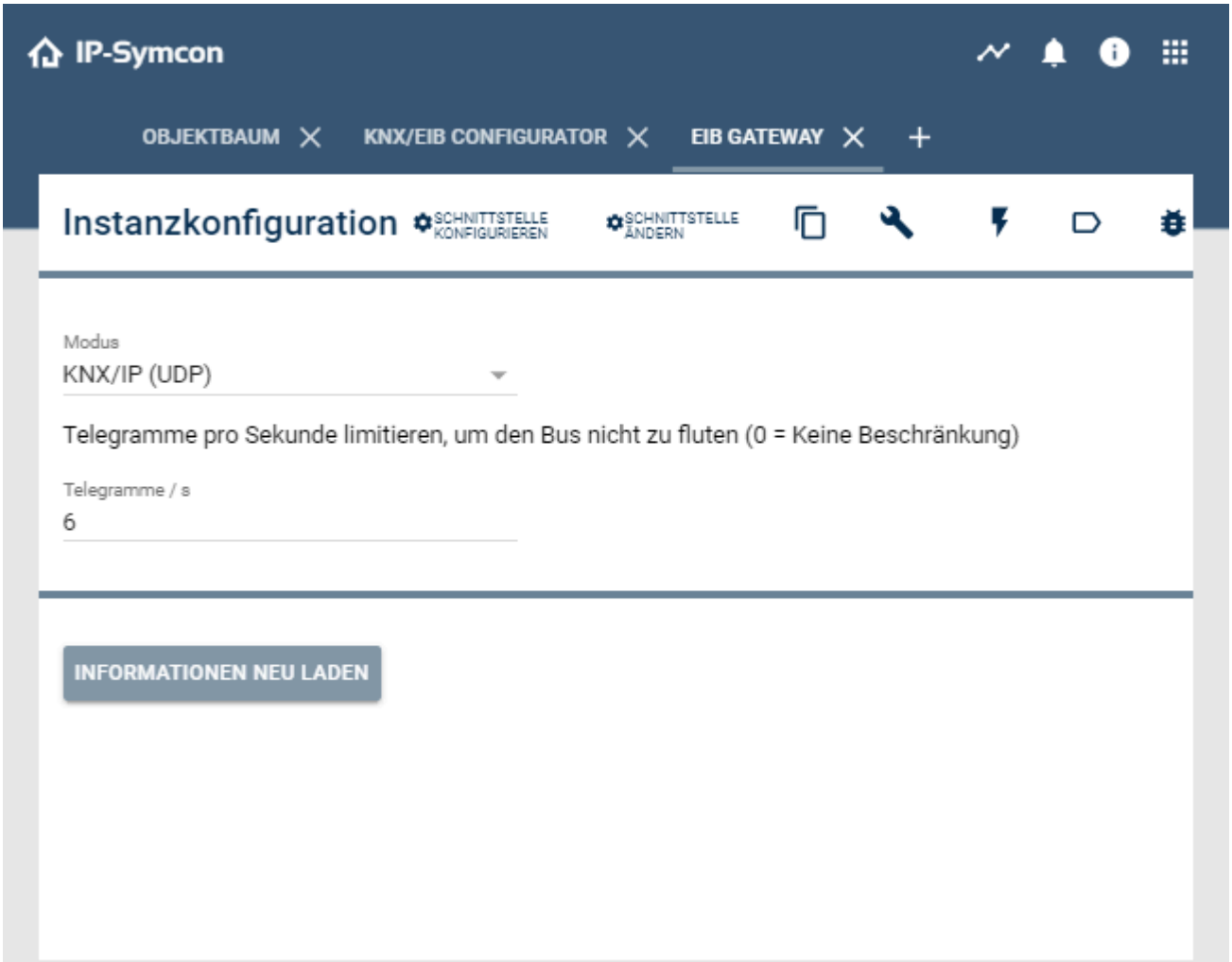
From version 5.6.5

INTEGRATION IN IP-SYMCON

The KNX interface can be integrated via the device-search. For this, "KNX Discovery" must be selected as the system. The Discovery instance then offers the creation of a KNX Configurator. After the configurator has been created, the individual devices can be added via this as described below.



Before the import of the group addresses can be started, the interface must be correctly selected and configured. To do this, click on the "cogwheel" in the upper part of the configuration. The following dialog should open.



If KNX Data Secure should be used, you can read more in the chapter KNX Data Secure

INTEGRATION WITHOUT KNX DATA SECURE

Under Mode the type of gateway has to be selected. For all common IP gateways the mode "KNX/IP (UDP)" may be selected. For the KNX IP interfaces with secure support, you can also choose the more reliable mode “KNX/IP (TCP)” or the appropriate secure equivalent. Changes are saved with the "Apply" button. To be able to enter the address for the interface, the "Gearwheel" in the upper area must be clicked again. Depending on the mode, either the IP address or the serial port belonging to the interface must be entered. In case of doubt, the IP address can also be found out with the help of the ETS. Further settings are not necessary. The changes are confirmed with "Apply". In the configuration of the gateway, the parameters of the gateway can now be read out via "Reload information". If these have been read out correctly, the configuration is complete.
Video tutorial:

INTEGRATION IN IP-SYMCON WITH KNX IP SECURE / KNX DATA SECURE

To use KNX Data Secure and/or KNX IP Secure in IP-Symcon, a backup of a keychain is needed in addition to the XML file.
Video tutorial to export the keychain:

In the KNX Gateway the mode "KNX/IP Secure (TCP)" must be selected. In the field "Select keychain" the keychain file is imported and the password is entered in the field "Keychain password". Changes will be saved with the button "Apply". Now the field "Secure tunnel" appears over which the configured tunnel of the interface can be selected. To be able to enter the address for the interface, the "Gearwheel" in the upper area must be clicked again. Here the IP address must be entered belongs. In case of doubt, the IP address can also be found out with the help of the ETS. Further settings are not necessary. The changes are confirmed with "Apply". In the configuration of the gateway, the gateway parameters can now be read out via "Reload information". If these have been read out correctly, the configuration is complete.
Video tutorial for KNX Data Secure with KNX IP Secure Interface:

Video tutorial for KNX Data Secure with SymBox with KNX extension:

find IP address

SINGLE ACTUATORS SETUP

In the KNX Configurator after uploading the XML file all previously configured KNX actuators can be integrated and visualized in IP-Symcon.
For this purpose devices have to be created by pressing "Create". The devices are now operable.
The ID that appears on the right side is the instance ID assigned by IP-Symcon. This is unique and unchangeable.
After clicking on a selected KNX device and then clicking on "Configure", the functionality of each individual device can be tested.
The devices can be found again in the object tree.
The created devices can now be switched via the "Visualization".

AUTOMATIC DETECTION OF FEEDBACK ADDRESSES

If a device has a feedback address set up, this can be automatically detected and set as a feedback address. To do this, click on the button "Detect feedback GA automatically" in the KNX Configurator and activate this via the item "Activate detection".
Then you must select the levels via which the group addresses are to be linked. If the feedback address and the device are in the same middle group, the item "Link on the same level (1/1/x)" is selected. If the feedback address and the device are in different middle groups, the item "Link over two levels (1/x/x)" is selected. If the addresses are in different main groups, the item "Link on all levels (x/x/x)" is selected.
The linking of the group addresses is done by the specified keywords. If the name of a feedback address is identical with the name of a group address after removing the keyword, they will be linked. The upper/lower case is ignored. The feedback address is then set as the feedback address of the group address.
Video about the setup with automatic feedback addresses:

EXAMPLES

SENDING TEMPERATURE VALUES TO THE KNX BUS

In order to be able to send values such as the temperature to the KNX bus, an action script is required. This forms the link between the visualization and the actuators. A script with the following content must be created:

```
EIB_Value(IPS_GetParent($_IPS['VARIABLE']), $_IPS['VALUE']);
```

For each desired device that should be changeable, the corresponding variable must be linked to this action script. This must be done in the configuration of the respective variable. To do this, the variable must be double-clicked. The selection of the action script is located in the "Profile settings" area. Here, the script just created must be selected. In addition, a suitable profile (e.g. ~temperature) should be selected via "Own profile".

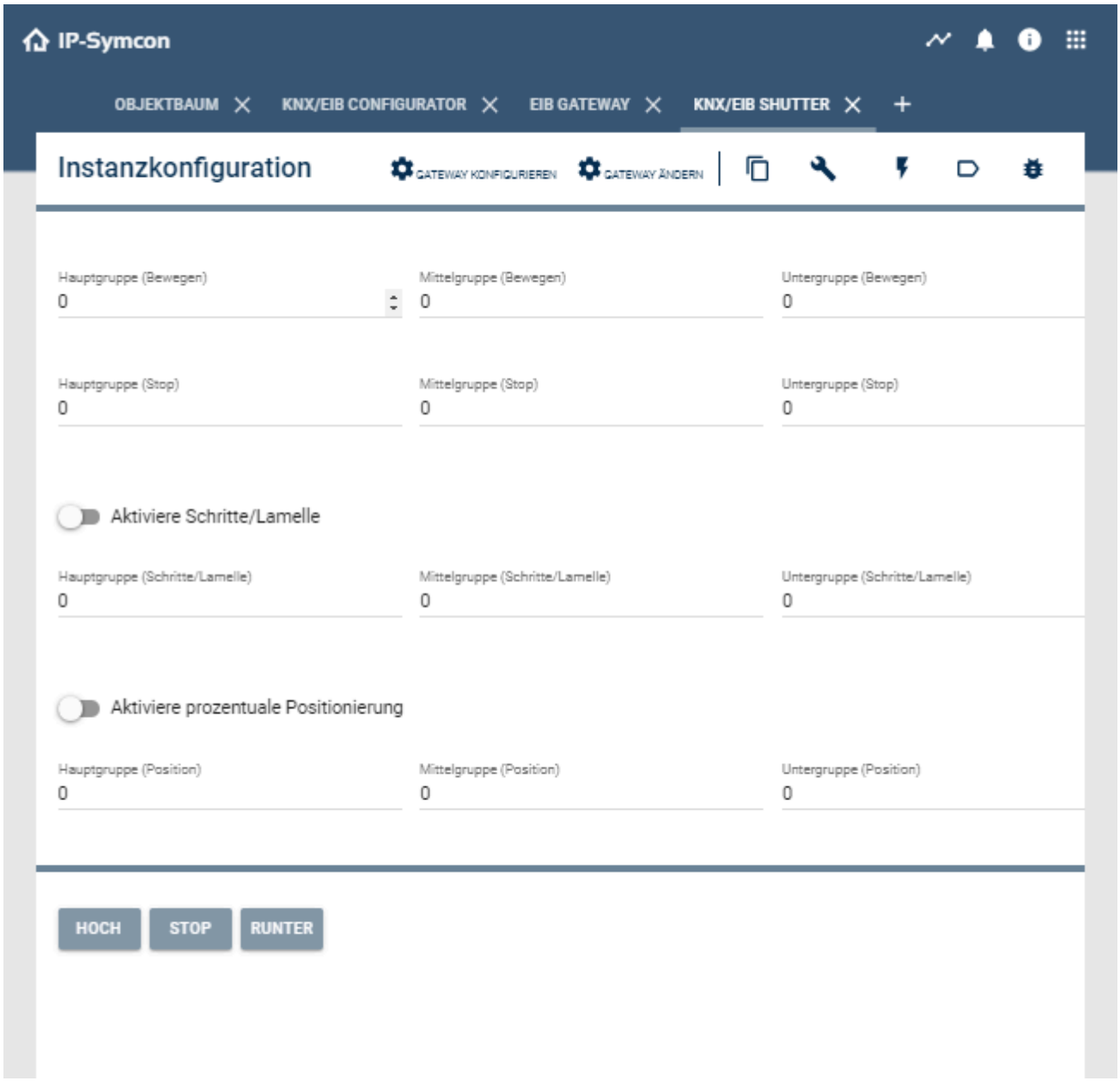




The last step can be repeated as often as needed. It is only necessary to create the script once with the above mentioned content

BETTER SHUTTER CONTROL WITH KNX SHUTTER

Due to the many setting options and linked addresses, shutters cannot be added via the configurator.



MOVE

Here the address for the up and down movement of the shutter must be entered. In the area "More?" further addresses can be entered on which the instance for moving up/down should listen.

STOP

Here the address for stopping the shutter must be entered. In the "More?" area, further addresses can be entered to which the instance is to listen for stopping.

ACTIVATE_STEPS/SLAT

This address can be enabled and controls the stepwise movement of the shutter. This is used to adjust the slats of the shutter. In the "More?" area, further addresses can be entered to which the instance for step control should listen.

ACTIVATE PERCENTAGE POSITIONING

These two addresses can be enabled and control either the percentage positioning of the shutter or the slats.

TEST CENTER

In the Test Center basic functions can be tested adhoc.

TIPS AND TRICKS

Is the ETS mandatory?
Synchronize time with KNX

CONVERSION OF GROUP ADDRESSES

Convert from two-level group addresses to three-level group addresses
If a two-level representation of the group addresses has been selected in the ETS, the addresses can be converted with the help of this tool. If the OPC export is used, the ETS takes over this process automatically.

Convert >>>

Convert free group addresses to three level group addresses
If you have selected the free representation of the group addresses in the ETS, unfortunately an OPC export is not possible. With the help of this tool the addresses can be converted.

Convert>>>

KNX WITH QNAP, SYNOLOGY OR DOCKER

If KNX is to be used with QNAP, Synology or Docker, NATSupport must be set up to receive responses from the system. This process is explained here.

SCRIPT TO READ STATES FROM THE BUS

Since only variable changes are perceived by IP-Symcon, it is necessary to query IP-Symcon when restarting to see if there have been any status changes in order to ensure a consistent display.
For this a start script can be added in the Event-Control .

```
$gatewayIDs = IPS_GetInstanceListByModuleID("{1C902193-B044-43B8-9433-
419F09C641B8}");
$instanceIDs = IPS_GetInstanceList();
foreach($gatewayIDs as $gatewayID) {
foreach($instanceIDs as $instanceID) {
```

```
$i = IPS_GetInstance($instanceID);
if($i['ConnectionID'] == $gatewayID) {
switch($i['ModuleInfo']['ModuleID']) {
case "{24A9D68D-7B98-4D74-9BAE-3645D435A9EF}": //Shutter
case "81F54858-72B1-4C2C-8CE3-7E00A3168378": //RGB (Legacy until IP-
Symcon 5.0)
case "{81F54858-72B1-4C2C-8CE3-7E00A3168378}": //RGB (IP-Symcon 5.1+)
case "4D7F7548-0979-4ABD-9BB3-81F9477C0903": //RGBW (Legacy until IP-
Symcon 5.0)
case "{4D7F7548-0979-4ABD-9BB3-81F9477C0903}": //RGBW (IP-Symcon 5.1+)
EIB_RequestStatus($instanceID);
break;

case "{D62B95D3-0C5E-406E-B1D9-8D102E50F64B}": //Group
if(IPS_GetProperty($instanceID, "GroupCapabilityRead")) {
EIB_RequestStatus($instanceID);
}
break;
case "{FB223058-3084-C5D0-C7A2-3B8D2E73FE8A}": //Device
KNX_RequestStatus($instanceID);
break;
default:
//DPTs
if(strpos($i['ModuleInfo']['ModuleName'], "DPT") !== false) {
if(IPS_GetProperty($instanceID, "CapabilityRead")) {
KNX_RequestStatus($instanceID);
}
}
break;
}
}
}
}
```

SCRIPT TO OUTPUT THE GROUP ADDRESSES

The following script can be used to output the group addresses including id, name and location.

```
echo "ID,Name,Location,GA,MoreGA" . PHP_EOL;

$gatewayIDs = IPS_GetInstanceListByModuleID("{1C902193-B044-43B8-9433-
419F09C641B8}");
$instanceIDs = IPS_GetInstanceList();
foreach($gatewayIDs as $gatewayID) {
foreach($instanceIDs as $instanceID) {
$i = IPS_GetInstance($instanceID);
if($i['ConnectionID'] == $gatewayID) {
switch($i['ModuleInfo']['ModuleID']) {
case "{24A9D68D-7B98-4D74-9BAE-3645D435A9EF}": //Shutter
printGA($instanceID, "GroupMove");
if (IPS_GetProperty($instanceID, "EnableStep")) {
printGA($instanceID, "GroupStep");
}
printGA($instanceID, "GroupStop");
```

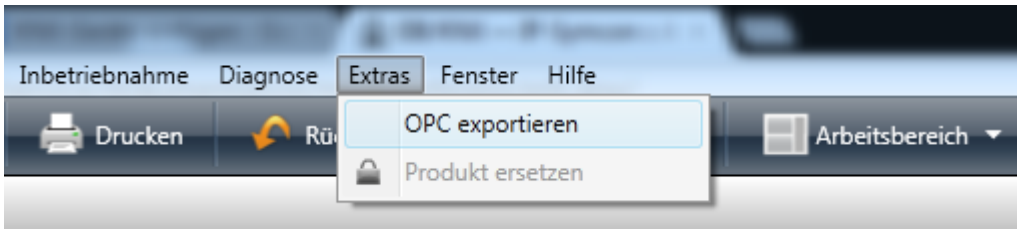
```
if (IPS_GetProperty($instanceID, "EnablePosition")) {
printGA($instanceID, "GroupPosition");
}
if (IPS_GetProperty($instanceID, "EnableBladePosition")) {
printGA($instanceID, "GroupBladePosition");
}
break;
case "81F54858-72B1-4C2C-8CE3-7E00A3168378": //RGB (Legacy until IP-
Symcon 5.0)
case "{81F54858-72B1-4C2C-8CE3-7E00A3168378}": //RGB (IP-Symcon 5.1+)
case "4D7F7548-0979-4ABD-9BB3-81F9477C0903": //RGBW (Legacy until IP-
Symcon 5.0)
case "{4D7F7548-0979-4ABD-9BB3-81F9477C0903}": //RGBW (IP-Symcon 5.1+)
case "{D62B95D3-0C5E-406E-B1D9-8D102E50F64B}": //Group
printGA($instanceID, "Group");
break;
case "{FB223058-3084-C5D0-C7A2-3B8D2E73FE8A}": //Device
printGAs($instanceID);
break;
default:
//DPTs
if(strpos($i['ModuleInfo']['ModuleName'], "DPT") !== false) {
printGA($instanceID, "");
}
break;
}
}
}
}

function printGA($id, $prefix) {
$c = json_decode(IPS_GetConfiguration($id), true);
echo sprintf("%d,%s,%s,", $id, IPS_GetName($id), IPS_GetLocation($id));
echo sprintf("%d/%d/%d", $c[$prefix . "Address1"], $c[$prefix .
"Address2"], $c[$prefix . "Address3"]);
foreach (json_decode($c[$prefix . "Mapping"], true) as $a) {
echo "," . sprintf("%d/%d/%d", $a[$prefix . "Address1"], $a[$prefix .
"Address2"], $a[$prefix . "Address3"]);
}
echo PHP_EOL;
}

function printGAs($id) {
$c = json_decode(IPS_GetConfiguration($id), true);
foreach(json_decode($c['GroupAddresses'], true) as $ga) {
echo sprintf("%d,%s,%s,", $id, IPS_GetName($id), IPS_GetLocation($id));
echo sprintf("%d/%d/%d", $ga["Address1"], $ga["Address2"],
$ga["Address3"]);
foreach ($ga["Mapping"] as $a) {
echo "," . sprintf("%d/%d/%d", $a["Address1"], $a["Address2"],
$a["Address3"]);
}
echo PHP_EOL;
}
}
```

LEGACY

OPC Export in ETS



Video tutorial for export:

Setup with OPC

ASSIGNMENT OF INDIVIDUAL FUNCTIONS TO EIS TYPE/DPT

FUNCTION	EIS TYPE	DPT
EIB_Char	EIS13	DPT4
EIB_Counter8bit	EIS14	DPT5, DPT6
EIB_Counter16bit	EIS10	DPT7, DPT8
EIB_Counter32bit	EIS11	DPT12, DPT13
EIB_Date	EIS4	DPT11
EIB_DimControl	EIS2	DPT3
EIB_DimValue	EIS6	DPT5
EIB_DriveBladeValue	EIS6	DPT5
EIB_DriveMove	EIS7	DPT1
EIB_DriveShutterValue	EIS6	DPT5
EIB_DriveStep	EIS7	DPT1
EIB_FloatValue	EIS9	DPT14
EIB_Move	EIS7	DPT1
EIB_Position	EIS6	DPT5
EIB_PriorityControl	EIS8	DPT2
EIB_PriorityPosition	EIS1	DPT1
EIB_Scale	EIS6	DPT5
EIB_Str	EIS15	DPT16
EIB_Switch	EIS1	DPT1
EIB_Time	EIS3	DPT10
EIB_Value	EIS5	DPT9
N/A	EIS12	DPT15

EIB_Char

```
bool EIB_Char (int $InstanceID, string $Letter)
```

PARAMETERS

InstanceID	ID of the device to be switched
Letter	Single letter

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a single letter **Letter** to device with ID **InstanceID**.

EXAMPLE

```
EIB_Char(12345, "A"); //Sends the letter A onto the bus
```

EIB_Counter8Bit

```
bool EIB_Counter8Bit (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Default = 0..255; Signed = -127..127

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the counter value **Value** to the device with the ID **InstanceID**.

EXAMPLE

```
EIB_Counter8Bit(12345, 10); //Sends the value 10 on the bus
```

EIB_Counter16Bit

```
bool EIB_Counter16Bit (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Standard = 0..65535; Signed = -32768..32767

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the counter value **Value** to the device with the ID **InstanceID**.

EXAMPLE

```
EIB_Counter16Bit(12345, 10); //Sends the value 10 on the bus
```

EIB_Counter32bit

```
bool EIB_Counter32Bit (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Standard = -2147483648..2147483647

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the counter value **Value** to the device with the ID **InstanceID**.

EXAMPLE

```
EIB_Counter32bit (12345, 10);//Sends the value 10 to the bus
```

EIB_Date

```
bool EIB_Date (int $InstanceID, string $Date)
```

PARAMETERS

InstanceID	ID of the device to be switched
Date	Default = yyyymmdd

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the date value **Date** to the device with ID **InstanceID**.

EXAMPLE

```
EIB_Date(12345, date("Ymd")); //Sends the current date on the bus
```

EIB_DimControl

```
bool EIB_DimControl (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Dim value date between 0..15 or -7..7

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a dim value date **Value** to the device with the ID **InstanceID**. Depending on the configuration 0..15 (Standard) or -7..7 (Enhanced) can be sent.



Since this function performs only a relative change of the dimmer, the brightness can not be specified absolutely. It is recommended to create an 8-bit address in order to set direct light levels.

EXAMPLE

```
EIB_DimControl(12345, 0);
```

EIB_DimValue

```
bool EIB_DimValue (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Value 0..255, 0..100%

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the dim value **Value** to the device with the ID **InstanceID**. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.

EXAMPLE

```
EIB_DimValue(12345, 100); //Dimming to 100%
```

EIB_DriveBladeValue

```
bool EIB_DriveBladeValue (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Value 0..255, 0..100%

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the position value **Value** to the device with the ID **InstanceID**. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.

EXAMPLE

```
EIB_DriveBladeValue(12345, 100); //Go to 100%
```

EIB_DriveMove

```
bool EIB_DriveMove (int $InstanceID, bool $Direction)
```

PARAMETERS

InstanceID	ID of the device to be switched
Direction	Standard

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moving the device with ID **InstanceID** in a particular **Direction**. Depending on configuration the movement direction can be inverted.

EXAMPLE

```

```

EIB_DriveShutterValue

```
bool EIB_DriveShutterValue (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Value 0..255, 0..100%

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the position value **Vert** to the device with the ID **InstanceID**. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.

EXAMPLE

```
EIB_DriveShutterValue(12345, 100); //Go to 100%
```

EIB_DriveStep

```
bool EIB_DriveStep (int $InstanceID, bool $StepTowards)
```

PARAMETERS

InstanceID	ID of the device to be switched
StepTowards	Standard

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moving the device **Stepwise** with ID **InstanceID** into a determined **Direction**. Depending on the configuration, the movement direction can be inverted.

EXAMPLE

```
EIB_DriveStep(12345, true); //Shutters move stepwise
```

EIB_FloatValue

```
bool EIB_FloatValue (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Floating point value

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the value **Value** to the device with the ID **InstanceID**.

EXAMPLE

```
EIB_FloatValue(12345, 22.1); //Sends 22.1°C to the controller
```

EIB_Move

```
bool EIB_Move (int $InstanceID, int $Direction)
```

PARAMETERS

InstanceID	ID of the KNX shutter instance
Direction	0 = up, 1 = step (up), 2 = stop, 3 = step (down), 4 = down

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Use the **InstanceID** to move the shutter in the desired **Direction**.

EXAMPLE

```
EIB_move (12345, 3); //Stop shutters
```

EIB_Position

```
bool EIB_Position (int $InstanceID, int $Position)
```

PARAMETERS

InstanceID	ID of the KNX shutter instance
Position	0-100%

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter with the **InstanceID** to the desired **Position**.



This function only works if the KNX device supports absolute positioning

EXAMPLE

```
EIB_Position(12345, 50); //Move roller shutters to 50%
```

EIB_PriorityControl

```
bool EIB_PriorityControl (int $InstanceID, int $State)
```

PARAMETERS

InstanceID	ID of the device to be switched
State	Default

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the device with ID **InstanceID** to a specific **State**.

EXAMPLE

```
EIB_PriorityControl(12345, 0);
```

EIB_PriorityPosition

```
bool EIB_PriorityPosition (int $InstanceID, bool $Direction)
```

PARAMETERS

InstanceID	ID of the device to be switched
Direction	Standard

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the device with ID **InstanceID** to a certain **Direction**. Depending on configuration the direction can be inverted.

EXAMPLE

```
EIB_PriorityPosition(12345, true); //Move shutters
```

EIB_RequestStatus

```
bool EIB_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a read request for device with ID **InstanceID** onto the bus.



The read request is only sent for the main address.



The EIB_RequestStatus function works for EIB instances, i.e. instances with the manufacturer “KNX/EIB”. For DPT instances, i.e. instances with the manufacturer “KNX”, the function KNX_RequestStatus can be used analogously.

EXAMPLE

```
EIB_RequestStatus(12345); //Send read request
```

EIB_Scale

```
bool EIB_Scale (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Value 0..255, 0..100, 0..360

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the Value **Value** to the device with the ID **InstanceID**. Depending on configuration 0..255 (Standard) or 0..100 (Percent) will be passed as the value parameter.

EXAMPLE

```
EIB_Scale(12345, 360);
```

EIB_SetRGB

```
bool EIB_SetRGB (int $InstanceID, int $Red, int $Green, int $Blue)
```

PARAMETERS

InstanceID	ID of the device to be switched
Red	0 .. 255 (0 = off, 255 = bright)
Green	0 .. 255 (0 = off, 255 = bright)
Blue	0 .. 255 (0 = off, 255 = bright)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the RGB stripe of the device with ID **InstanceID** to red value **Red**, green value **Green** and blue value **Blue**.

EXAMPLE

```
EIB_SetRGB(12345, 255, 0, 0); //Sets the RGB stripe to red
```

EIB_SetRGBW

```
bool EIB_SetRGBW (int $InstanceID, int $Red, int $Green, int $Blue, int $White)
```

PARAMETERS

InstanceID	ID of the device to be switched
Red	0 .. 255 (0 = off, 255 = bright)
Green	0 .. 255 (0 = off, 255 = bright)
Blue	0 .. 255 (0 = off, 255 = bright)
White	0 .. 255 (0 = off, 255 = bright)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the RGBW stripe of the device with the ID **InstanceID** to red **Red**, the green value **Green**, the blue level **Blue** and the white content to **White**.

EXAMPLE

```
EIB_SetRGBW(12345, 255, 0, 0, 0); //Sets the RGB stripe to red
```

EIB_Str

```
bool EIB_Str (int $InstanceID, string $String)
```

PARAMETERS

InstanceID	ID of the device to be switched
String	Text to send (14 characters maximum)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a text **String** to the device with the ID **InstanceID**.

EXAMPLE

```
EIB_Str(12345, “It rings! “); //Send the string to the bus
```

EIB_Switch

```
bool EIB_Switch (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the device with the ID InstanceID to value Status

EXAMPLE

```
EIB_Switch(12345, true); //Turns on the device
```

EIB_Time

```
bool EIB_Time (int $InstanceID, string $Time)
```

PARAMETERS

InstanceID	ID of the device to be switched
Time	Standard = dhhmmss; TimeOnly = hhmmss

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the time value **Time** to the device with ID **InstanceID**. The d value is between 1 (Monday) and 7 (Sunday).

EXAMPLE

```
EIB_Time(12345, date("NHis")); //Sends the current time on the bus
(Standard)
EIB_Time(12345, date("His")); //Sends the current time on the bus
(TimeOnly)
```

EIB_Value

```
bool EIB_Value (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-671088,64 .. 670760,96

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the value **Value** to the device with ID **InstanceID**.

EXAMPLE

```
EIB_Value(12345, 22.1); //Sends 22.1 ° C to the controller
```

Device list

SUPPORTED GATEWAYS

All serial gateways with FT1.2 protocol (and cEMI messages) and all KNX/IP gateways, that offer the tunneling protocol, are supported. For KNX gateways that meet these requirements but do not work, please contact our support. We will solve the problem as quickly as possible.
Since IP-Symcon 6.4 we also support KNX Data Secure and KNX IP Secure.

COMPONENT	MANUFACTURER	CONNECTION TYPE	DESCRIPTION
SymBox with KNX extension	Symcon GmbH	Serial	Integrated module from Weinzierl Engineering GmbH
1a EIB KNX IP interface PoE	EIBMARKT	IP	EIB KNX IP interface PoE (ArtNr. N000401)
i-bus® KNX/IP interface	ABB	IP	REG IPS/S 2.1, 2CDG 110 098 R0011
MDT SCN-IP000.02	MDT	IP	
MDT SCN-IP000.03	MDT	IP (Secure)	
1168 KNX IP Secure Interface	Enertex	IP (Secure)	
KNX BAOS Module 838 kBerry	Weinzierl Engineering GmbH	IP (Secure)	
KNX BAOS Module 838 kBerry	Weinzierl Engineering GmbH	Seriell	
KNX IP Interface 730	Weinzierl Engineering GmbH	IP	
KNX IP Interface 731	Weinzierl Engineering GmbH	IP	
KNX IP Interface 732 Secure	Weinzierl Engineering GmbH	IP (Secure)	
KNX IP Interface 740 wireless	Weinzierl Engineering GmbH	IP	
KNX Router Weinzierl 750	Weinzierl Engineering GmbH	IP	
N146	Siemens	IP	
N148	Siemens	IP	

SUPPORTED DPT

SINCE VERSION 5.0

DPT-TYPE	VERSION
DPT 001.x	since 5.0
DPT 002.x	since 5.0
DPT 003.x	since 5.0
DPT 004.x	since 5.0
DPT 005.x	since 5.0
DPT 006.x	since 5.0
DPT 007.x	since 5.0
DPT 008.x	since 5.0
DPT 009.x	since 5.0
DPT 010.x	since 5.0
DPT 011.x	since 5.0
DPT 012.x	since 5.0
DPT 013.x	since 5.0
DPT 014.x	since 5.0
DPT 015.x	since 5.0
DPT 016.x	since 5.0
DPT 017.x	since 5.0
DPT 018.x	since 5.0
DPT 019.x	since 5.0
DPT 020.x	since 5.0
DPT 021.x	since 5.0
DPT 022.x	since 5.0
DPT 023.x	since 5.0
DPT 024.x	since 5.0
DPT 025.x	since 5.0
DPT 026.x	since 5.0
DPT 027.x	since 5.0
DPT 028.x	since 5.0
DPT 029.x	since 5.0
DPT 030.x	since 5.0
DPT 031.x	since 5.0
DPT 200.x	since 5.0
DPT 201.x	since 5.0
DPT 202.x	since 5.0
DPT 203.x	since 5.0
DPT 204.x	since 5.0

DPT-TYPE	VERSION
DPT 205.x	since 5.0
DPT 206.x	since 5.0
DPT 207.x	since 5.0
DPT 209.x	since 5.0
DPT 210.x	since 5.0
DPT 211.x	since 5.0
DPT 212.x	since 5.0
DPT 213.x	since 5.0
DPT 214.x	since 5.0
DPT 215.x	since 5.0
DPT 216.x	since 5.0
DPT 217.x	since 5.0
DPT 218.x	since 5.0
DPT 219.x	since 5.0
DPT 220.x	since 5.0
DPT 221.x	since 5.0
DPT 222.x	since 5.0
DPT 223.x	since 5.0
DPT 224.x	since 5.0
DPT 225.x	since 5.0
DPT 229.x	since 5.0
DPT 230.x	since 5.0
DPT 231.x	since 5.0
DPT 232.x	since 5.0
DPT 234.x	since 5.0
DPT 235.x	since 5.0
DPT 236.x	since 5.0
DPT 237.x	since 5.0
DPT 238.x	since 5.0
DPT 239.x	since 5.0
DPT 240.x	since 5.0
DPT 241.x	since 5.0
DPT 242.x	since 5.0
DPT 249.x	since 5.0
DPT 251.x	since 5.0

DPT 001.X

DPT	DPT_NAME
1.001	DPT_Switch
1.002	DPT_Bool
1.003	DPT_Enable
1.004	DPT_Ramp
1.005	DPT_Alarm
1.006	DPT_BinaryValue
1.007	DPT_Step
1.008	DPT_UpDown
1.009	DPT_OpenClose
1.010	DPT_Start
1.011	DPT_State
1.012	DPT_Invert
1.013	DPT_DimSendStyle
1.014	DPT_InputSource
1.015	DPT_Reset
1.016	DPT_Ack
1.017	DPT_Trigger
1.018	DPT_Occupancy
1.019	DPT_Window_Door
1.021	DPT_LogicalFunction
1.022	DPT_Scene_AB
1.023	DPT_ShutterBlinds_Mode
1.024	DPT_DayNight
1.100	DPT_Heat/Cool

DPT 002.X

DPT	DPT_NAME
2.001	DPT_Switch_Control
2.002	DPT_Bool_Control
2.003	DPT_Enable_Control
2.004	DPT_Ramp_Control
2.005	DPT_Alarm_Control
2.006	DPT_BinaryValue_Control
2.007	DPT_Step_Control
2.008	DPT_Direction1_Control
2.009	DPT_Direction2_Control

DPT	DPT_NAME
2.010	DPT_Start_Control
2.011	DPT_State_Control
2.012	DPT_Invert_Control

DPT 003.X

DPT	DPT_NAME
3.007	DPT_Control_Dimming
3.008	DPT_Control_Blinds

DPT 004.X

DPT	DPT_NAME
4.001	DPT_Char_ASCII
4.002	DPT_Char_8859_1

DPT 005.X

DPT	DPT_NAME
5.001	DPT_Scaling
5.003	DPT_Angle
5.004	DPT_Percent_U8
5.005	DPT_DecimalFactor
5.006	DPT_Tariff
5.010	DPT_Value_1_Ucount

DPT 006.X

DPT	DPT_NAME
6.001	DPT_Percent_V8
6.010	DPT_Value_1_Count
6.020	DPT_Status_Mode3

DPT 007.X

DPT	DPT_NAME
7.001	DPT_Value_2_Ucount
7.002	DPT_TimePeriodMsec
7.003	DPT_TimePeriod10MSec
7.004	DPT_TimePeriod100MSec
7.005	DPT_TimePeriodSec
7.006	DPT_TimePeriodMin
7.007	DPT_TimePeriodHrs
7.010	DPT_PropDataType
7.011	DPT_Length_mm
7.012	DPT_UElCurrentmA
7.013	DPT_Brightness

DPT 008.X

DPT	DPT_NAME
8.001	DPT_Value_2_Count
8.002	DPT_DeltaTimeMsec
8.003	DPT_DeltaTime10MSec
8.004	DPT_DeltaTime100MSec
8.005	DPT_DeltaTimeSec
8.006	DPT_DeltaTimeMin
8.007	DPT_DeltaTimeHrs
8.010	DPT_Percent_V16
8.011	DPT_Rotation_Angle
8.012	DPT_Length_m

DPT 009.X

DPT	DPT_NAME
9.001	DPT_Value_Temp
9.002	DPT_Value_Tempd
9.003	DPT_Value_Tempa
9.004	DPT_Value_Lux
9.005	DPT_Value_Wsp
9.006	DPT_Value_Pres
9.007	DPT_Value_Humidity
9.008	DPT_Value_AirQuality

DPT	DPT_NAME
9.010	DPT_Value_Time1
9.011	DPT_Value_Time2
9.020	DPT_Value_Volt
9.021	DPT_Value_Curr
9.022	DPT_PowerDensity
9.023	DPT_KelvinPerPercent
9.024	DPT_Power
9.025	DPT_Value_Volume_Flow
9.026	DPT_Rain_Amount
9.027	DPT_Value_Temp_F
9.028	DPT_Value_Wsp_kmh

DPT 010.X

DPT	DPT_NAME
10.001	DPT_TimeOfDay

DPT 011.X

DPT	DPT_NAME
11.001	DPT_Date

DPT 012.X

DPT	DPT_NAME
12.001	DPT_Value_4_Ucount
12.1200	DPT_VolumeLiquid_Litre
12.1201	DPT_Volume_m3

DPT 013.X

DPT	DPT_NAME
13.001	DPT_Value_4_Count
13.002	DPT_FlowRate_m3/h
13.010	DPT_ActiveEnergy
13.011	DPT_ApparantEnergy
13.012	DPT_ReactiveEnergy

DPT	DPT_NAME
13.013	DPT_ActiveEnergy_kWh
13.014	DPT_ApparantEnergy_kVAh
13.015	DPT_ReactiveEnergy_kVARh
13.100	DPT_LongDeltaTimeSec
13.1200	DPT_DeltaVolumeLiquid_Litre
13.1201	DPT_DeltaVolume_m3
13.16	DPT_ActiveEnergy_MWh

DPT 014.X

DPT	DPT_NAME
14.000	DPT_Value_Acceleration
14.001	DPT_Value_Acceleration_Angular
14.002	DPT_Value_Activation_Energy
14.003	DPT_Value_Activity
14.004	DPT_Value_Mol
14.005	DPT_Value_Amplitude
14.006	DPT_Value_AngleRad
14.007	DPT_Value_AngleDeg
14.008	DPT_Value_Angular_Momentum
14.009	DPT_Value_Angular_Velocity
14.010	DPT_Value_Area
14.011	DPT_Value_Capacitance
14.012	DPT_Value_Charge_DensitySurface
14.013	DPT_Value_Charge_DensityVolume
14.014	DPT_Value_Compressibility
14.015	DPT_Value_Conductance
14.016	DPT_Value_Electrical_Conductivity
14.017	DPT_Value_Density
14.018	DPT_Value_Electric_Charge
14.019	DPT_Value_Electric_Current
14.020	DPT_Value_Electric_CurrentDensity
14.021	DPT_Value_Electric_DipoleMoment
14.022	DPT_Value_Electric_Displacement
14.023	DPT_Value_Electric_FieldStrength
14.024	DPT_Value_Electric_Flux
14.025	DPT_Value_Electric_FluxDensity

DPT	DPT_NAME
14.026	DPT_Value_Electric_Polarization
14.027	DPT_Value_Electric_Potential
14.028	DPT_Value_Electric_PotentialDifference
14.029	DPT_Value_ElectromagneticMoment
14.030	DPT_Value_Electromotive_Force
14.031	DPT_Value_Energy
14.032	DPT_Value_Force
14.033	DPT_Value_Frequency
14.034	DPT_Value_Angular_Frequency
14.035	DPT_Value_Heat_Capacity
14.035	DPT_Value_Heat_Capacity
14.036	DPT_Value_Heat_FlowRate
14.037	DPT_Value_Heat_Quantity
14.038	DPT_Value_Impedance
14.039	DPT_Value_Length
14.040	DPT_Value_Light_Quantity
14.041	DPT_Value_Luminance
14.042	DPT_Value_Luminous_Flux
14.043	DPT_Value_Luminous_Intensity
14.044	DPT_Value_Magnetic_FieldStrength
14.045	DPT_Value_Magnetic_Flux
14.046	DPT_Value_Magnetic_FluxDensity
14.047	DPT_Value_Magnetic_Moment
14.048	DPT_Value_Magnetic_Polarization
14.049	DPT_Value_Magnetization
14.050	DPT_Value_MagnetomotiveForce
14.051	DPT_Value_Mass
14.052	DPT_Value_MassFlux
14.053	DPT_Value_Momentum
14.054	DPT_Value_Phase_AngleRad
14.055	DPT_Value_Phase_AngleDeg
14.056	DPT_Value_Power
14.057	DPT_Value_Power_Factor
14.058	DPT_Value_Pressure
14.059	DPT_Value_Reactance
14.060	DPT_Value_Resistance
14.061	DPT_Value_Resistivity

DPT	DPT_NAME
14.062	DPT_Value_SelfInductance
14.063	DPT_Value_SolidAngle
14.064	DPT_Value_Sound_Intensity
14.065	DPT_Value_Speed
14.066	DPT_Value_Stress
14.067	DPT_Value_Surface_Tension
14.068	DPT_Value_Common_Temperature
14.069	DPT_Value_Absolute_Temperature
14.070	DPT_Value_TemperatureDifference
14.071	DPT_Value_Thermal_Capacity
14.072	DPT_Value_Thermal_Conductivity
14.073	DPT_Value_ThermoelectricPower
14.074	DPT_Value_Time
14.075	DPT_Value_Torque
14.076	DPT_Value_Volume
14.077	DPT_Value_Volume_Flux
14.078	DPT_Value_Weight
14.079	DPT_Value_Work
14.1200	DPT_Volume_Flux_Meter
14.1201	DPT_Volume_Flux_ls

DPT 015.X

DPT	DPT_NAME
15.000	DPT_Access_Data

DPT 016.X

DPT	DPT_NAME
16.000	DPT_String_ASCII
16.001	DPT_String_8859_1

DPT 017.X

DPT	DPT_NAME
17.001	DPT_SceneNumber

DPT 018.X

DPT	DPT_NAME
18.001	DPT_SceneControl

DPT 019.X

DPT	DPT_NAME
19.001	DPT_DateTime

DPT 020.X

DPT	DPT_NAME
20.001	DPT_SCLOMode
20.002	DPT_BuildingMode
20.003	DPT_OccMode
20.004	DPT_Priority
20.005	DPT_LightApplicationMode
20.006	DPT_ApplicationArea
20.007	DPT_AlarmClassType
20.008	DPT_PSUMode
20.011	DPT_ErrorClass_System
20.012	DPT_ErrorClass_HVAC
20.013	DPT_Time_Delay
20.014	DPT_Beaufort_Wind_Force_Scale
20.017	DPT_SensorSelect
20.020	DPT_ActuatorConnectType
20.100	DPT_FuelType
20.101	DPT_BurnerType
20.102	DPT_HVACMode
20.103	DPT_DHWMode
20.104	DPT_LoadPriority
20.105	DPT_HVACContrMode
20.106	DPT_HVACEmergMode
20.107	DPT_ChangeoverMode
20.108	DPT_ValveMode
20.109	DPT_DamperMode
20.110	DPT_HeaterMode

DPT	DPT_NAME
20.111	DPT_FanMode
20.112	DPT_MasterSlaveMode
20.113	DPT_StatusRoomSetp
20.120	DPT_ADAType
20.121	DPT_BackupMode
20.122	DPT_StartSynchronization
20.600	DPT_Behaviour_Lock_Unlock
20.601	DPT_Behaviour_Bus_Power_Up_Down
20.602	DPT_DALI_Fade_Time
20.603	DPT_BlinkingMode
20.604	DPT_LightControlMode
20.605	DPT_SwitchPBModel
20.606	DPT_PBAction
20.607	DPT_DimmPBModel
20.608	DPT_SwitchOnMode
20.609	DPT_LoadTypeSet
20.610	DPT_LoadTypeDetected
20.801	DPT_SABExceptBehaviour
20.802	DPT_SABBehaviour_Lock_Unlock
20.803	DPT_SSSBMode
20.804	DPT_BlindsControlMode
20.1000	DPT_CommMode
20.1001	DPT_AddInfoTypes
20.1002	DPT_RF_ModeSelect
20.1003	DPT_RF_FilterSelect

DPT 021.X

DPT	DPT_NAME
21.001	DPT_StatusGen
21.002	DPT_Device_Control
21.100	DPT_ForceSign
21.101	DPT_ForceSignCool
21.102	DPT_StatusRHC
21.103	DPT_StatusSDHWC
21.104	DPT_FuelTypeSet
21.105	DPT_StatusRCC

DPT	DPT_NAME
21.106	DPT_StatusAHU
21.601	DPT_LightActuatorErrorInfo
21.1000	DPT_RF_ModeInfo
21.1001	DPT_RF_FilterInfo
21.1010	DPT_Channel_Activation_8

DPT 022.X

DPT	DPT_NAME
22.100	DPT_StatusDHWC
22.101	DPT_StatusRHCC
22.1000	DPT_Media
22.1010	DPT_Channel_Activation_16

DPT 023.X

DPT	DPT_NAME
23.001	DPT_OnOff_Action
23.002	DPT_Alarm_Reaction
23.003	DPT_UpDown_Action
23.102	DPT_HVAC_PB_Action

DPT 024.X

DPT	DPT_NAME
24.001	DPT_VarString_8859_1

DPT 025.X

DPT	DPT_NAME
25.1000	DPT_DoubleNibble

DPT 026.X

DPT	DPT_NAME
26.001	DPT_SceneInfo

DPT 027.X

DPT	DPT_NAME
27.001	DPT_CombinedInfoOnOff

DPT 028.X

DPT	DPT_NAME
28.001	DPT_UTF-8

DPT 029.X

DPT	DPT_NAME
29.010	DPT_ActiveEnergy_V64
29.011	DPT_ApparantEnergy_V64
29.012	DPT_ReactiveEnergy_V64

DPT 030.X

DPT	DPT_NAME
30.1010	DPT_Channel_Activation_24

DPT 031.X

DPT	DPT_NAME
31.101	DPT_PB_Action_HVAC_Extended

DPT 200.X

DPT	DPT_NAME
200.100	DPT_Heat/Cool_Z
200.101	DPT_BinaryValue_Z

DPT 201.X

DPT	DPT_NAME
201.100	DPT_HVACMode_Z

DPT	DPT_NAME
201.102	DPT_DHWMode_Z
201.104	DPT_HVACContrMode_Z
201.105	DPT_EnablH/Cstage_Z DPT_EnablH/CStage
201.107	DPT_BuildingMode_Z
201.108	DPT_OccMode_Z
201.109	DPT_HVACEmergMode_Z

DPT 202.X

DPT	DPT_NAME
202.001	DPT_RelValue_Z
202.002	DPT_UCountValue8_Z

DPT 203.X

DPT	DPT_NAME
203.002	DPT_TimePeriodMsec_Z
203.003	DPT_TimePeriod10Msec_Z
203.004	DPT_TimePeriod100Msec_Z
203.005	DPT_TimePeriodSec_Z
203.006	DPT_TimePeriodMin_Z
203.007	DPT_TimePeriodHrs_Z
203.011	DPT_UFlowRateLiter/h_Z
203.012	DPT_UCountValue16_Z
203.013	DPT_UElCurrentµA_Z
203.014	DPT_PowerKW_Z
203.015	DPT_AtmPressureAbs_Z
203.017	DPT_PercentU16_Z
203.100	DPT_HVACAirQual_Z
203.101	DPT_WindSpeed_Z DPT_WindSpeed
203.102	DPT_SunIntensity_Z
203.104	DPT_HVACAirFlowAbs_Z

DPT 204.X

DPT	DPT_NAME
204.001	DPT_RelSignedValue_Z

DPT 205.X

DPT	DPT_NAME
205.002	DPT_DeltaTimeMsec_Z
205.003	DPT_DeltaTime10Msec_Z
205.004	DPT_DeltaTime100Msec_Z
205.005	DPT_DeltaTimeSec_Z
205.006	DPT_DeltaTimeMin_Z
205.007	DPT_DeltaTimeHrs_Z
205.017	DPT_Percent_V16_Z
205.100	DPT_TempHVACAbs_Z
205.101	DPT_TempHVACRel_Z
205.102	DPT_HVACAirFlowRel_Z

DPT 206.X

DPT	DPT_NAME
206.100	DPT_HVACModeNext
206.102	DPT_DHWModeNext
206.104	DPT_OccModeNext
206.105	DPT_BuildingModeNext

DPT 207.X

DPT	DPT_NAME
207.100	DPT_StatusBUC
207.101	DPT_LockSign
207.102	DPT_ValueDemBOC
207.104	DPT_ActPosDemAbs
207.105	DPT_StatusAct
207.600	DPT_StatusLightingActuator

DPT 209.X

DPT	DPT_NAME
209.100	DPT_StatusHPM
209.101	DPT_TempRoomDemAbs
209.102	DPT_StatusCPM

DPT	DPT_NAME
209.103	DPT_StatusWTC

DPT 210.X

DPT	DPT_NAME
210.100	DPT_TempFlowWaterDemAbs

DPT 211.X

DPT	DPT_NAME
211.100	DPT_EnergyDemWater

DPT 212.X

DPT	DPT_NAME
212.100	DPT_TempRoomSetpSetShift[3]
212.101	DPT_TempRoomSetpSet[3]

DPT 213.X

DPT	DPT_NAME
213.100	DPT_TempRoomSetpSet[4]
213.101	DPT_TempDHWSetpSet[4]
213.102	DPT_TempRoomSetpSetShift[4]

DPT 214.X

DPT	DPT_NAME
214.100	DPT_PowerFlowWaterDemHPM
214.101	DPT_PowerFlowWaterDemCPM

DPT 215.X

DPT	DPT_NAME
215.100	DPT_StatusBOC
215.101	DPT_StatusCC

DPT 216.X

DPT	DPT_NAME
216.100	DPT_SpecHeatProd

DPT 217.X

DPT	DPT_NAME
217.001	DPT_Version

DPT 218.X

DPT	DPT_NAME
218.001	DPT_VolumeLiter_Z
218.002	DPT_FlowRate_m3/h_Z

DPT 219.X

DPT	DPT_NAME
219.001	DPT_AlarmInfo

DPT 220.X

DPT	DPT_NAME
220.100	DPT_TempHVACAbsNext

DPT 221.X

DPT	DPT_NAME
221.001	DPT_SerNum

DPT 222.X

DPT	DPT_NAME
222.100	DPT_TempRoomSetpSetF16[3]
222.101	DPT_TempRoomSetpSetShiftF16[3]

DPT 223.X

DPT	DPT_NAME
223.100	DPT_EnergyDemAir

DPT 224.X

DPT	DPT_NAME
224.100	DPT_TempSupply AirSetpSet

DPT 225.X

DPT	DPT_NAME
225.001	DPT_ScalingSpeed
225.002	DPT_Scaling_Step_Time
225.003	DPT_TariffNext

DPT 229.X

DPT	DPT_NAME
229.001	DPT_MeteringValue

DPT 230.X

DPT	DPT_NAME
230.1000	DPT_MBus_Address

DPT 231.X

DPT	DPT_NAME
231.001	DPT_Locale_ASCII

DPT 232.X

DPT	DPT_NAME
232.600	DPT_Colour_RGB

DPT 234.X

DPT	DPT_NAME
234.001	DPT_LanguageCodeAlpha2_ASCII
234.002	DPT_RegionCodeAlpha2_ASCII

DPT 235.X

DPT	DPT_NAME
235.001	DPT_Tariff_ActiveEnergy

DPT 236.X

DPT	DPT_NAME
236.001	DPT_Prioritised_Mode_Control

DPT 237.X

DPT	DPT_NAME
237.600	DPT_DALI_Control_Gear_Diagnostic

DPT 238.X

DPT	DPT_NAME
238.001	DPT_SceneConfig
238.600	DPT_DALI_Diagnostics

DPT 239.X

DPT	DPT_NAME
239.001	DPT_FlaggedScaling

DPT 240.X

DPT	DPT_NAME
240.800	DPT_CombinedPosition

DPT 241.X

DPT	DPT_NAME
241.800	DPT_StatusSAB

DPT 242.X

DPT	DPT_NAME
242.600	DPT_Colour_xyY

DPT 249.X

DPT	DPT_NAME
249.600	DPT_Brightness_Colour_Temperature_Transition

DPT 251.X

DPT	DPT_NAME
251.600	DPT_Colour_RGBW

SUPPORTED EIS

EIS-TYPE	VERSION	MATCHING DPT
EIS1	since 3.4	DPT 001.x
EIS2	since 3.4	DPT 003.x
EIS3	since 3.4	DPT 010.x
EIS4	since 3.4	DPT 011.x
EIS5	since 3.4	DPT 009.x
EIS6	since 3.4	DPT 005.x
EIS7	since 3.4	DPT 001.x
EIS8	since 3.4	DPT 002.x
EIS9	since 3.4	DPT 014.x
EIS10	since 3.4	DPT 007.x , DPT 008.x
EIS11	since 3.4	DPT 012.x , DPT 013.x
EIS12	since 3.4	DPT 015.x
EIS13	since 3.4	DPT 004.x
EIS14	since 3.4	DPT 005.x , DPT 006.x
EIS15	since 3.4	DPT 016.x

KNX_RequestStatus

```
bool KNX_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a read request for device with ID **InstanceID** onto the bus.



The read request is only sent for the main address.



The function KNX_RequestStatus works for DPT instances, i.e. instances with the manufacturer "KNX". The EIB_RequestStatus function can be used analogously for EIB instances, i.e. instances with the manufacturer "KNX/EIB".

EXAMPLE

```
KNX_RequestStatus(12345); //Send read request
```

LCN

LCN (Local Control Network) is a wired bus system.



The following devices are supported by IP-Symcon:
Supported devices



The new modules with 12 variables (since firmware 160B13) are supported since IP-Symcon 3.0.

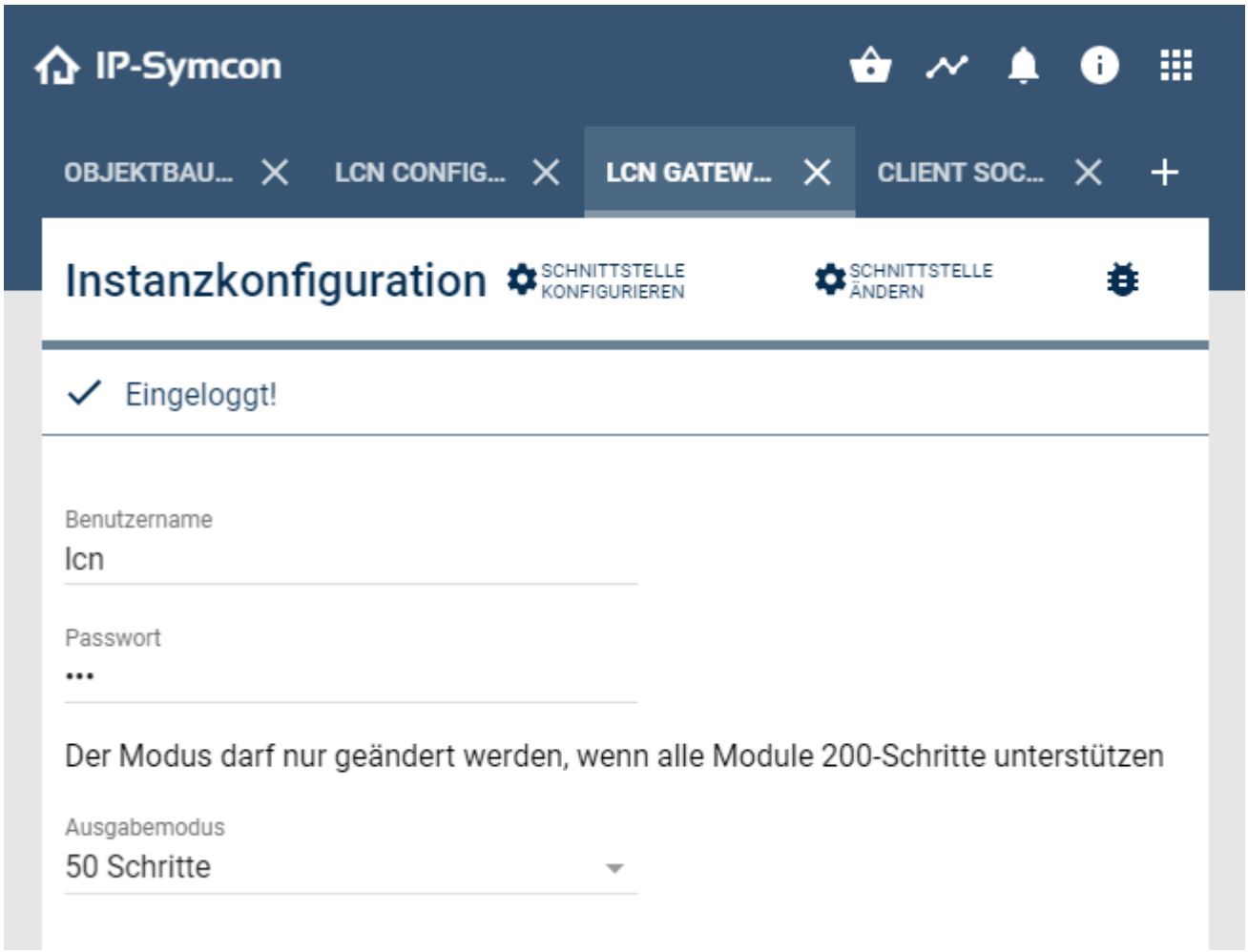
INSTALLATION

In order to be able to connect an LCN bus to IP-Symcon, a Bus-Coupler (LCN-PKU) with coupling program PCHK or an Ethernet-Coupler (LCN-VISU) is required.

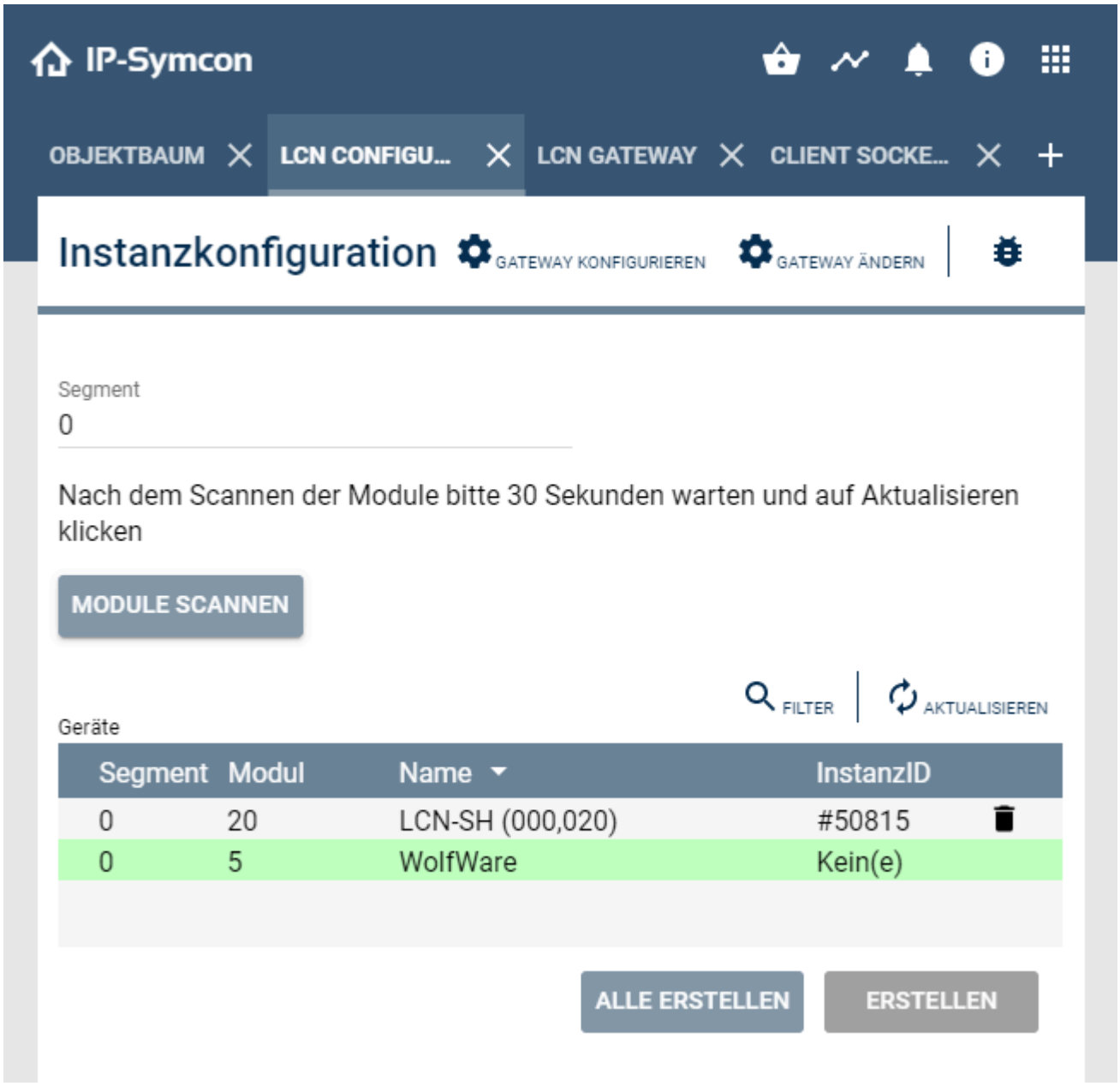
SETUP VIDEO-TUTORIAL

INTEGRATION IN IP-SYMCON

A configurator for the LCN system can be created via the Management Console.
After a configurator has been created, a red message may appear. This indicates an incorrect or incomplete configuration. To correct the error, the user name and password must be specified in the parent instance. The change is saved by clicking on "Apply".

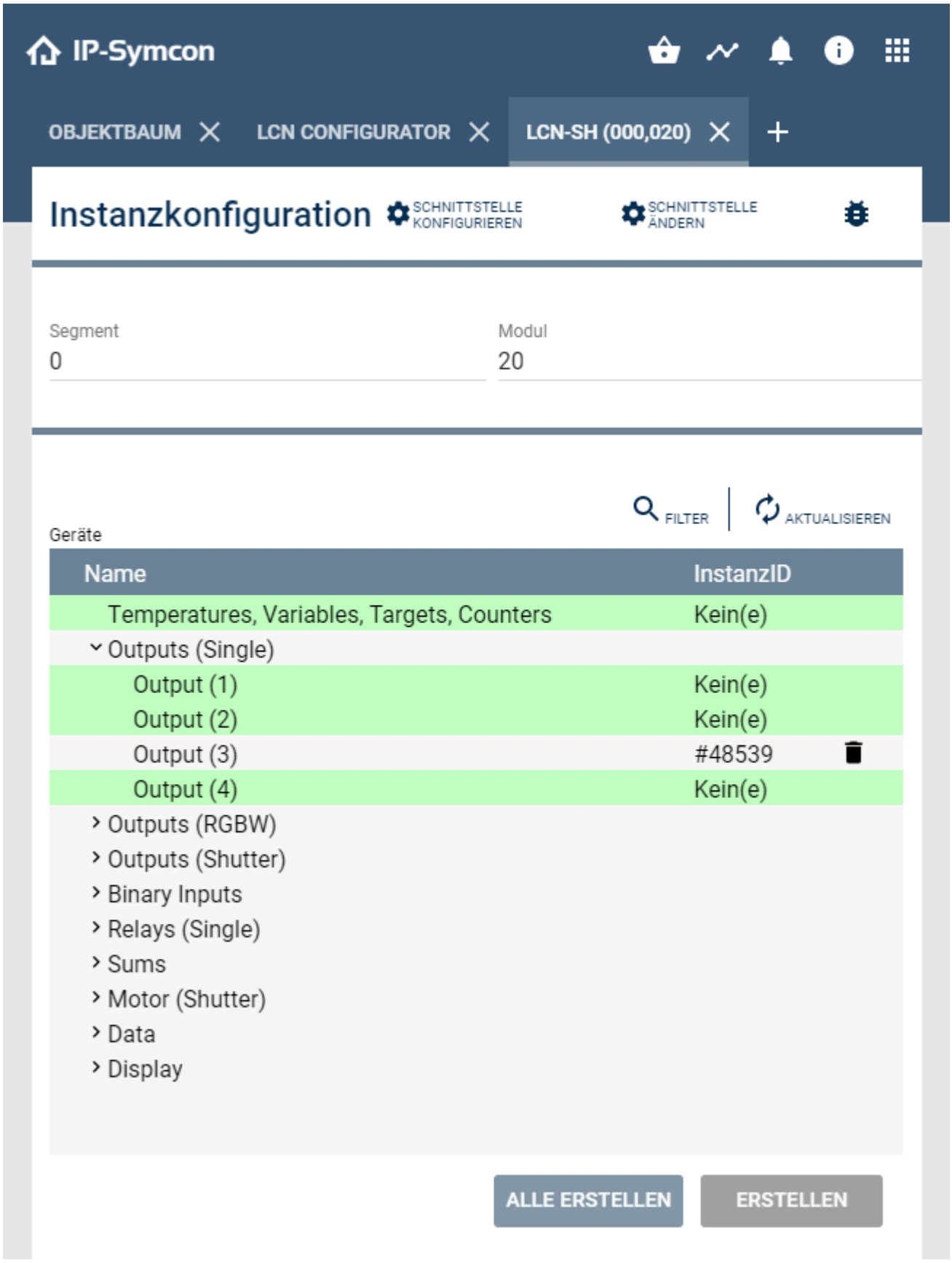


If it is not possible to log in after applying, the network configuration within the interface must be checked. Also, in the gateway, the output mode can also be selected between 50 and 200 steps.
After successfully logging in, modules can be searched for on the bus in the configurator, created in IP-Symcon and the individual modules can be created as IP-Symcon splitter instances.

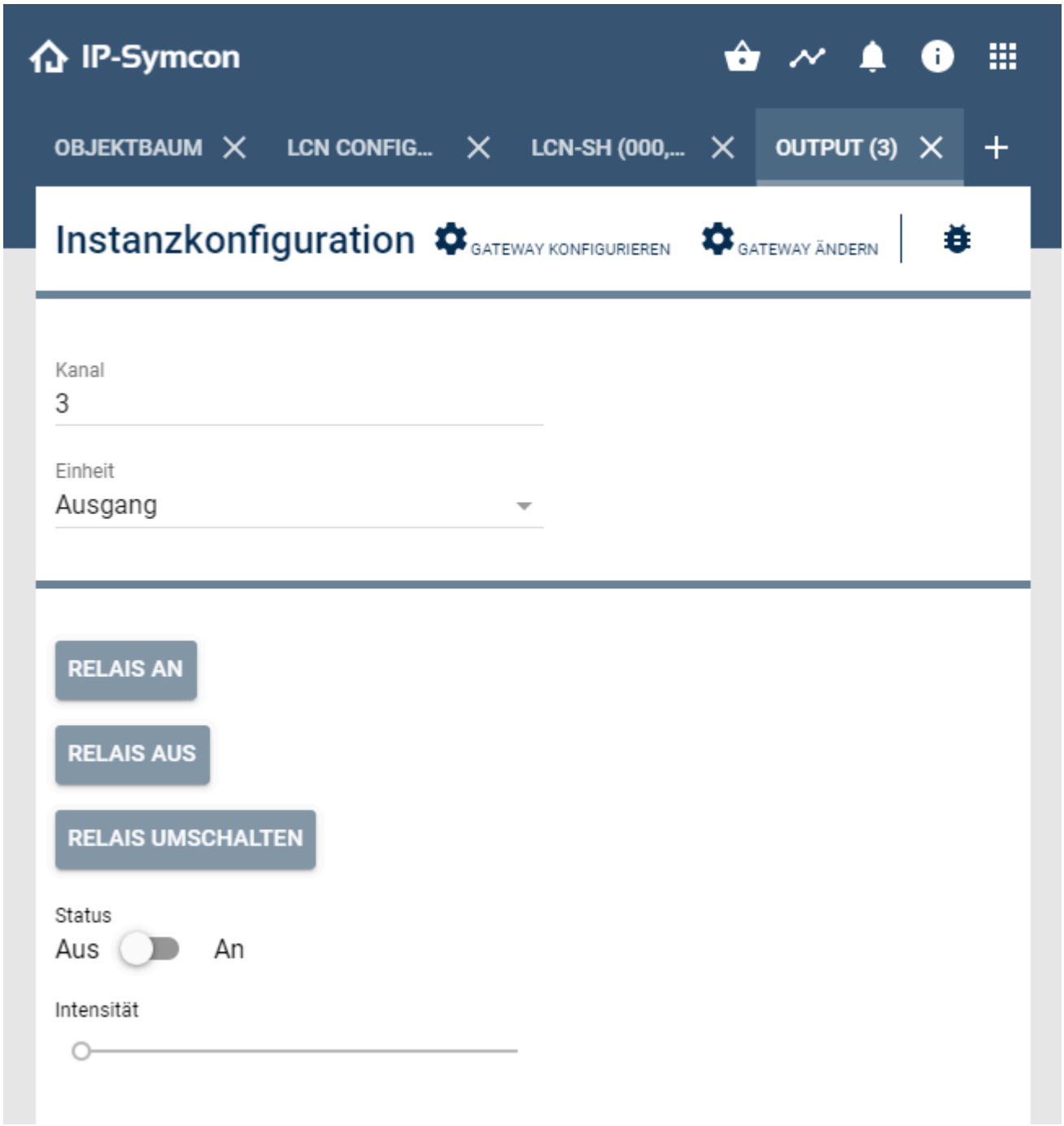


CONFIGURE MODULES

As soon as a module is created in the configurator and the configuration button is pressed, a new dialog opens.



In order to create the individual instances, the desired relays, inputs/outputs, etc. must be selected and "Create" must be clicked on.
New devices are created and sorted within the object tree into the category that matches the name of the module. These created instances can then be renamed accordingly and sorted elsewhere. It is also possible to call up the respective instance configuration via "Configure".


IP-Symcon
OBJEKTBAUM
LCN CONFIG...
LCN-SH (000,...
OUTPUT (3)
Instanzkonfiguration
GATEWAY KONFIGURIEREN
GATEWAY ÄNDERN
Kanal
3
Einheit
Ausgang
RELAIS AN
RELAIS AUS
RELAIS UMSCHALTEN
Status
Aus
An
Intensität

Device list

SUPPORTED GATEWAYS

GATEWAY	DESCRIPTION
LCN-PK (via LCN PCHK)	Serial port, communication via PCHK with IP-Symcon
LCN-PKU (via LCN PCHK)	USB connection, communication via PCHK with IP-Symcon
LCN-PKE	Ethernet connection, communication directly with IP-Symcon
LCN-VISU	Ethernet connection, communication directly with IP-Symcon

SUPPORTED COMPONENTS

IP-Symcon supports all components.



The LCN-HL4 only supports the RGB color model (switch position 2)

LCN_AddGroup

```
bool LCN_AddGroup (int $InstanceID, int $Group)
```

PARAMETERS

InstanceID	Instance ID; the ID of the splitter instance, not the device instance
Group	Group number (1..255)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Adds the device with ID **InstanceID** to the group with group number **Group**.

EXAMPLE

```
//The device with ID 12345 is added to group 123
LCN_AddGroup(12345, 123);
```

LCN_AddIntensity

```
bool LCN_AddIntensity (int $InstanceID, int $Intensity)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	0-100%

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** brighter by the value **Intensity**.

EXAMPLE

```
LCN_AddIntensity(12345, 50); // Dim device brighter by 50 percentage
points
```

LCN_AddThresholdCurrent

```
bool LCN_AddThresholdCurrent (int $InstanceID, int $Register, int
$Threshold, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Register	Register of the threshold value to be switched (1x5
Threshold	Threshold to be set (1x5
Value	Threshold value (0..1000)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Adds the value **Value** of the threshold **Threshold** in the register **Register** relative to the current value of the LCN module with the ID **InstanceID**.

EXAMPLE

```
// 1x5 threshold
// Adds the value 2.5 to the threshold 2 of the instance with ID 12345
LCN_AddThresholdCurrent(12345, 0, 2, 2.5);

// 4x4 threshold
// Adds the value 2.5 to the threshold 2 in register 3 of the instance
with ID 12345
LCN_AddThresholdCurrent(12345, 3, 2, 2.5);
```

LCN_AddThresholdDefined

```
bool LCN_AddThresholdDefined (int $InstanceID, int $Register, int
$Threshold, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Register	Register of the threshold value to be switched (1x5
Threshold	Threshold to be set (1x5
Value	Value of the threshold

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Adds the value **Value** of the threshold **Threshold** in the register **Register** relative to the programmed value of the LCN module with the ID **InstanceID**.

EXAMPLE

```
// 1x5 threshold
// Adds the value 2.5 relative to the programmed value into threshold 2
of the instance with ID 12345
LCN_AddThresholdDefined(12345, 0, 2, 2.5);

// 4x4 threshold
// Adds the value 2.5 relative to the programmed value into threshold 2
in register 3 of the instance with ID 12345
LCN_AddThresholdDefined(12345, 3, 2, 2.5);
```

LCN_Beep

```
bool LCN_Beep (int $InstanceID, bool $SpecialTone, int $Number)
```

PARAMETERS

InstanceID	ID of the Instance; the ID of the splitter instance, not the device instance
SpecialTone	TRUE for On, FALSE for Off
Number	Number between 1..15

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Lets the instance with the ID InstanceID beep

EXAMPLE

```
LCN_Beep(12345, true, 1); //Beep once (SpecialTone)
```

LCN_DeductIntensity

```
bool LCN_DeductIntensity (int $InstanceID, int $Intensity)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	0-100%

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** darker by the value **Intensity**.

EXAMPLE

```
LCN_DeductIntensity(12345, 10); //Dim device darker by 10 percentage
points
```

LCN_DeductThresholdCurrent

```
bool LCN_DeductThresholdCurrent (int $InstanceID, int $Register, int
$Threshold, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Register	Register of the threshold value to be switched (1x5
Threshold	Threshold to be set (1x5
Value	Threshold value (0..1000)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Deducts the value **Value** of the threshold **Threshold** in the register **Register** relative to the current value of the LCN module with the ID **InstanceID**.

EXAMPLE

```
// 1x5 threshold
// Deducts the value 2.5 from the threshold 2 of the instance with ID
12345
LCN_DeductThresholdCurrent(12345, 0, 2, 2.5);

// 4x4 threshold
// Deducts the value 2.5 from the threshold 2 in register 3 of the
instance with ID 12345
LCN_DeductThresholdCurrent(12345, 3, 2, 2.5);
```

LCN_DeductThresholdDefined

```
bool LCN_DeductThresholdDefined (int $InstanceID, int $Register, int
$Threshold, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Register	Register of the threshold value to be switched (1x5
Threshold	Threshold to be set (1x5
Value	Value of the threshold

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Deducts the value **Value** of the threshold **Threshold** in the register **Register** relative to the programmed value of the LCN module with the ID **InstanceID**.

EXAMPLE

```
// 1x5 threshold
// Deducts the value 2.5 relative to the programmed value into
Threshold 2 of instance with ID 12345
LCN_AddThresholdDefined(12345, 0, 2, 2.5);

// 4x4 threshold
// Deducts the value 2.5 relative to the programmed value into
threshold 2 in register 3 of instance with ID 12345
LCN_AddThresholdDefined(12345, 3, 2, 2.5);
```

LCN_FadeOut

```
bool LCN_FadeOut (int $InstanceID, int $Intensity, int $Ramp)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	0-100%
Ramp	0-200 seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the devive with ID **InstanceID** to value **Intensity** and dims then with the **Ramp** seconds down to 0%.

EXAMPLE

```
LCN_FadeOut(12345, 100, 5); //Dim down device in 5 seconds of 100% to
0%
```

LCN_FlipRelay

```
bool LCN_FlipRelay (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Relay to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches over the relay with the ID InstanceID

EXAMPLE

```
LCN_FlipRelais(12345, true); //Switch relay
```

LCN_LimitOutput

```
bool LCN_LimitOutput (int $InstanceID, int $Value, int $Time, string $TimeType)
```

PARAMETERS

InstanceID	ID of the output to be switched
Value	Percentage value to be limited to (0..100)
Time	Period for which the output should be limited
TimeType	Indicates whether the limit should apply to seconds, minutes, hours or days

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Limits the output with the ID **InstanceID** for the period **Time** and time type **TimeType** to the value **Value**.

TIME TYPE	POSSIBLE PERIOD
'S' (seconds)	1..60
'M' (minutes)	1..90
'H' (hours)	1..50
'D' (days)	1..45

EXAMPLE

```
//Limits the output with ID 12345 to 36% for 8 hours
LCN_LimitOutput(12345, 36, 8, 'H');
```

LCN_LoadScene

```
bool LCN_LoadScene (int $InstanceID, int $Scene)
```

PARAMETERS

InstanceID	ID of the output to be switched
Scene	Which scene should be loaded (0.. 9, 15)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Calls the scene **Scene** of the output with the ID **InstanceID**.



If the special value 15 is specified as the scene, the value is taken from the counter Tvar/Var1

EXAMPLE

```
//Calls scene 5 of the output with ID 12345
LCN_LoadScene(12345, 5);
```

LCN_LockTargetValue

```
bool LCN_LockTargetValue (int $InstanceID, int $Target)
```

PARAMETERS

InstanceID	ID of the device to be switched
Target	Which controller should be locked (0,1,2,3)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Locks controller **Target** of device with ID **InstanceID**.

EXAMPLE

```
//Locks controller 0 of device 12345
LCN_LockTargetValue(12345, 0);
```

LCN_RampStop

```
bool LCN_RampStop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Stops the dimming process of the device with the ID **InstanceID**

EXAMPLE

```
LCN_RampStop(12345);
```

LCN_ReleaseTargetValue

```
bool LCN_ReleaseTargetValue (int $InstanceID, int $Target)
```

PARAMETERS

InstanceID	ID of the device to be switched
Target	Which controller to unlock (0,1,2,3)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Unlocks controller **Target** of device with ID **InstanceID**.

EXAMPLE

```
//Unlocks controller 0 of device 12345
LCN_ReleaseTargetValue(12345, 0);
```

LCN_RemoveGroup

```
bool LCN_RemoveGroup (int $InstanceID, int $Group)
```

PARAMETERS

InstanceID	Instance ID; the ID of the splitter instance, not the device instance
Group	Group number (0..255)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Removes the device with ID **InstanceID** from the group with group number **Group**.



If group 0 is specified, the device will be removed from all groups

EXAMPLE

```
//The device with ID 12345 is removed from group 123
LCN_RemoveGroup(12345, 123);
```

LCN_RequestLights

```
bool LCN_RequestLights (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the panel to be queried

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Queries the LEDs (panel lights) of the panel with the ID InstanceID.

EXAMPLE

```
//The LEDs of the panel with ID 12345 are queried
LCN_RequestLights(12345);
```

LCN_RequestRead

```
bool LCN_RequestRead (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to query

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Queries the variables, targets and counters of the temperature instance with the ID InstanceID.

EXAMPLE

```
//Query the device with ID 12345
LCN_RequestRead(12345);
```

LCN_RequestStatus

```
bool LCN_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the module to query

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the status of all outputs, relays and binary sensors of the module with ID **InstanceID**.

EXAMPLE

```
// Get all the statuses of the module with ID 12345
LCN_RequestStatus(12345);
```

LCN_RequestThresholds

```
bool LCN_RequestThresholds (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the threshold to query

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the thresholds and hysteresis ID **InstanceID**.

EXAMPLE

```
//Get the thresholds and hysteresis of threshold 12345
LCN_RequestThresholds(12345);
```

LCN_SaveScene

```
bool LCN_SaveScene (int $InstanceID, int $Scene)
```

PARAMETERS

InstanceID	ID of the output to be switched
Scene	Which scene to save in (0..9, 15)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Stores scene **Scene** for the outlet with ID **InstanceID**.



If the special value 15 is specified as the scene, the value is taken from the counter Tvar/Var1

EXAMPLE

```
// Saves scene 5 of the exit with ID 12345
LCN_SaveScene(12345, 5);
```

LCN_SelectSceneRegister

```
bool LCN_SelectSceneRegister (int $InstanceID, int $Register)
```

PARAMETERS

InstanceID	ID of the device to be switched
Register	Which register should be selected (0..9)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Selects register with number **Register** of device with ID **InstanceID**.

EXAMPLE

```
// Selects register 2 of device with ID 12345
ULCN_SelectSceneRegister(12345, 2);
```

LCN_SendCommand

```
bool LCN_SendCommand (int $InstanceID, string $Function, string $Data)
```

PARAMETERS

InstanceID	ID of the Instance; the ID of the splitter instance, not the device instance
Function	The first two (2) signs of PCHK command (e.g. A1)
Data	The remaining characters of PCHK command (e.g. DI000000). Any numbers must be sent in the decimal representation (3 characters).

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a custom command to the instance with the ID **InstanceID**. For possible commands, refer to the PCHK documentation. If you do not have this, you can possibly refer to your dealer. We do not provide this option.

EXAMPLE

```
LCN_SendCommand(12345, "A1", "DI000000"); //Dim output 1 immediately to
0%.
```

LCN_SetDisplayText

```
bool LCN_SetDisplayText (int $InstanceID, int $Line, string $Text)
```

PARAMETERS

InstanceID	ID of the device to be switched
Line	Display line between 1..4
Text	Text with a maximum of 60 characters

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Displays the text Text on line Line of the display with ID InstanceID.

EXAMPLE

```
//Shows "Hello World" on line 1 of the display with the InstanceID
12345.
LCN_SetDisplayText(12345, 1, "Hello World");
```

LCN_SetDisplayTime

```
bool LCN_SetDisplayTime (int $InstanceID, int $Line, int $DisplayTime)
```

PARAMETERS

InstanceID	ID of the device to be switched
Line	Display line between 1..4
DisplayTime	Time units between 1..30

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Configures line **Line** of the display with ID **InstanceID** to display time **DisplayTime**.



Whether the display duration is defined as seconds/minutes/hours/days is set in the LCN module, i.e.: In the device itself.

EXAMPLE

```
//Configure line 1 of display instance 12345 to 30 time units
LCN_SetDisplayTime(12345, 1, 30);
```

LCN_SetIntensity

```
bool LCN_SetIntensity (int $InstanceID, int $Intensity, int $Ramp)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	0-100%
Ramp	0 = Immediately; n = Seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with the ID **InstanceID** to value **Intensity** in **Ramp** seconds.

EXAMPLE

```
LCN_SetIntensity(12345, 100, 0); //Dim device to 100% immediately
```

LCN_SetLamp

```
bool LCN_SetLamp (int $InstanceID, int $Lamp, string $Action)
```

PARAMETERS

InstanceID	Instance ID; the ID of the splitter instance, not the device instance
Lamp	Selection of the panel lamps to be switched (1..12)
Action	What action to take.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the panel lamp **Lamp** of the instance with the ID **InstanceID** to the value **Action**.

ACTION	DESCRIPTION
'A'	Turn off
'B'	Flash
'E'	Turn on
'F'	Flicker

EXAMPLE

```
// Blink panel light 2 of module with ID 12345.
LCN_SetLamp(12345, 2, 'B');
```

LCN_SetRelay

```
bool LCN_SetRelay (int $InstanceID, string $Wert)
```

PARAMETERS

InstanceID	ID of the Instance; the ID of the splitter instance, not the device instance
Wert	8 times character combination of the following values

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Set all relays of the instance with the ID **InstanceID** to **Value**

EXAMPLE

```
LCN_SetRelay(12345, "--1100UU");
```

LCN_SetRGBW

```
bool LCN_SetRGBW (int $InstanceID, int $Red, int $Green, int $Blue, int $White)
```

PARAMETERS

InstanceID	ID of the device to be switched
Red	Intensity between 1..100
Green	Intensity between 1..100
Blue	Intensity between 1..100
White	Intensity between 1..100

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the color and white value of the instance with ID **InstanceID** to **Red**, **Green** and **Blue**, and the intensity to **White**.

EXAMPLE

```
//Set instance 12345 to full intensity red.
LCN_SetRGBW(12345, 100, 0, 0, 100);
```

LCN_SetTargetValue

```
bool LCN_SetTargetValue (int $InstanceID, int $Target, float $Value)
```

PARAMETERS

InstanceID	ID of the Instance
Target	0 = Regulator R1; 1 = Regulator R2; 2 = Regulator S1; 3 = Regulator S2
Value	Value

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the regulator **Target** of the instance with the ID **InstanceID** to the value **Value**

EXAMPLE

```
LCN_SetTargetValue(12345, 2, 24.3); //Set Regulator S1 to 24.3°C
```

LCN_ShiftTargetValue

```
bool LCN_ShiftTargetValue (int $InstanceID, int $Target, float
$RelativeValue)
```

PARAMETERS

InstanceID	ID of the device to be switched
Target	Which controller should be shifted (0,1,2,3)
RelativeValue	Value by which to shift

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Shifts the current target value of the controller **Target** of the device with the ID **InstanceID** by the value **RelativeValue**.

EXAMPLE

```
// Shifts the target value of the controller 1 by -1.7°C
LCN_ShiftTargetValue(12345, 1, -1.7);
```

LCN_ShutterMove

```
bool LCN_ShutterMove (int $InstanceID, int $Position)
```

PARAMETERS

InstanceID	ID of the device to be switched
Position	Percentage position of the shutter (0-100%)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts the movement of the shutter with the ID **InstanceID** to the position **Position**.

EXAMPLE

```
//Roller shutter movement to 80% (20% closed)
LCN_ShutterMove(12345, 80);
```

LCN_ShutterMoveDown

```
bool LCN_ShutterMoveDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts the downward movement of the roller shutter with ID **InstanceID**.

EXAMPLE

```
//Start roller shutter downward movement
LCN_ShutterMoveDown(12345);
```

LCN_ShutterMoveUp

```
bool LCN_ShutterMoveUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts the upward movement of the shutter with the ID **InstanceID**.

EXAMPLE

```
//Start roller shutter upward movement
LCN_ShutterMoveUp(12345);
```

LCN_ShutterStop

```
bool LCN_ShutterStop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Stops the movement of the shutter with the ID **InstanceID**.

EXAMPLE

```
//Stop shutter movement
LCN_ShutterStop(12345);
```

LCN_StartFlicker

```
bool LCN_StartFlicker (int $InstanceID, string $Flicker, string $Speed,
int $FlickerCount)
```

PARAMETERS

InstanceID	ID of the device to be switched
Flicker	G = Low; M = Medium; S = Start
Speed	L = Slow (ca. 0,5x / Sec); M = Medium (ca. 1x / Sec); S = Fast (ca. 2x / Sec)
FlickerCount	Number between 1..15

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Let the device with the ID **InstanceID** flicker

EXAMPLE

```
LCN_StartFlicker(12345, "G", "L", 5);
```

LCN_StopFlicker

```
bool LCN_StopFlicker (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Stop flickering of the device with the ID **InstanceID** immediately

EXAMPLE

```
LCN_StopFlicker(12345);
```

LCN_SwitchDurationMin

```
bool LCN_SwitchDurationMin (int $InstanceID, int $Minutes, string
$DimmingTime, bool $Preserving)
```

PARAMETERS

InstanceID	ID of the device to be switched
Minutes	After how many minutes it should be dimmed down
DimmingTime	How fast to dim down
Preserving	The device is only shut down after the time elapsed if it has not been on before.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** brighter after **Minutes** minutes. Other options are the dimming speed **DimmingTime** and whether it is **Preserving**.

DIMMINGTIME	DESCRIPTION
'K'	Short DimmingTime
'M'	Medium DimmingTime
'L'	Long DimmingTime

EXAMPLE

```
// Slowly dim device 12345 after 10 minutes
LCN_SwitchDurationMin(12345, 10, 'L', false);
```

LCN_SwitchDurationSec

```
bool LCN_SwitchDurationSec (int $InstanceID, int $Seconds, string
$DimmingTime, bool $Preserving)
```

PARAMETERS

InstanceID	ID of the device to be switched
Seconds	After how many seconds it should be dimmed down
DimmingTime	How fast to dim down
Preserving	The device is only shut down after the time elapsed if it has not been on before.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** brighter after **Seconds** seconds. Other options are the dimming speed **DimmingTime** and whether it is **Preserving**.

DIMMINGTIME	DESCRIPTION
'K'	Short DimmingTime
'M'	Medium DimmingTime
'L'	Long DimmingTime

EXAMPLE

```
// Quickly dim device 12345 after 5 seconds
LCN_SwitchDurationSec(12345, 5, 'K', false);
```

LCN_SwitchMemory

```
bool LCN_SwitchMemory (int $InstanceID, int $Ramp)
```

PARAMETERS

InstanceID	ID of the device to be switched
Ramp	Number of seconds, 0 = Immediately

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with the ID **InstanceID**to a stored value

EXAMPLE

```
LCN_SwitchMemory(12345, 0); //Turn off device immediately
```

LCN_SwitchMode

```
bool LCN_SwitchMode (int $InstanceID, int $Ramp)
```

PARAMETERS

InstanceID	ID of the device to be switched
Ramp	Number of seconds, 0 = Immediately

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches over the device with the ID **InstanceID**.

EXAMPLE

```
LCN_SwitchMode(12345, 0); //Switch device immediately
```

LCN_SwitchRelay

```
bool LCN_SwitchRelay (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the relay with the ID InstanceID to value Status

EXAMPLE

```
LCN_SwitchRelay(12345, true); //Turn on relay
```

LCN_SwitchRelayTimer

```
bool LCN_SwitchRelayTimer (int $InstanceID, int $TimeFactor)
```

PARAMETERS

InstanceID	ID of the device to be switched
TimeFactor	Time factor between 1..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the relay with ID **InstanceID** to time factor **TimeFactor**



The time factor is not a linear value. 1 = 30ms and 255 = 240s

EXAMPLE

```
//Set relay time factor to 255.
LCN_SwitchRelayTimer(12345, 255);
```

LJQuick

Require: Symcon >= 7.2
The KNX quick (Lingg&Janke) module helps to add KNX quick instances quickly and easily. By selecting a group, channel and clicking on the 'Create' button, the instances are created with the correct addresses. Date and time are sent to KNX address 30/3/254 every 10 minutes.

FUNCTIONAL SCOPE

Creation of instances based on group settings

SOFTWARE INSTALLATION

Install the 'KNX quick (Lingg&Janke)' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance' the 'KNX quick (Lingg&Janke)' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

The configurator is structured as a tree. First the device type, then the various groups under which the various channels are located. If one or more channels are selected, the corresponding KNX device instances with the corresponding group addresses can be created by clicking on the 'Create' button.
Available devices:

- Switches
- Dimming
- Shading
- Energy
- Water, Gas, Oil
- Heat Quantity
- Temperature
- Temperature/ Humidity

Each device contains group 1-F.
Each group contains channels 1-9.
The device groups switch, dimming, shading, temperature and temperature/humidity also contain channel 0.

LJ_SendDateTime

Require: Symcon >= 7.2

```
bool LJ_SendDateTime (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Sends the current date/time to the KNX bus

EXAMPLE

```
LJ_SendDateTime(12345);
```

Matter

Require: Symcon >= 8.2

The Matter integration is still under development and will be released with version 8.2. Matter is currently not supported on Windows.

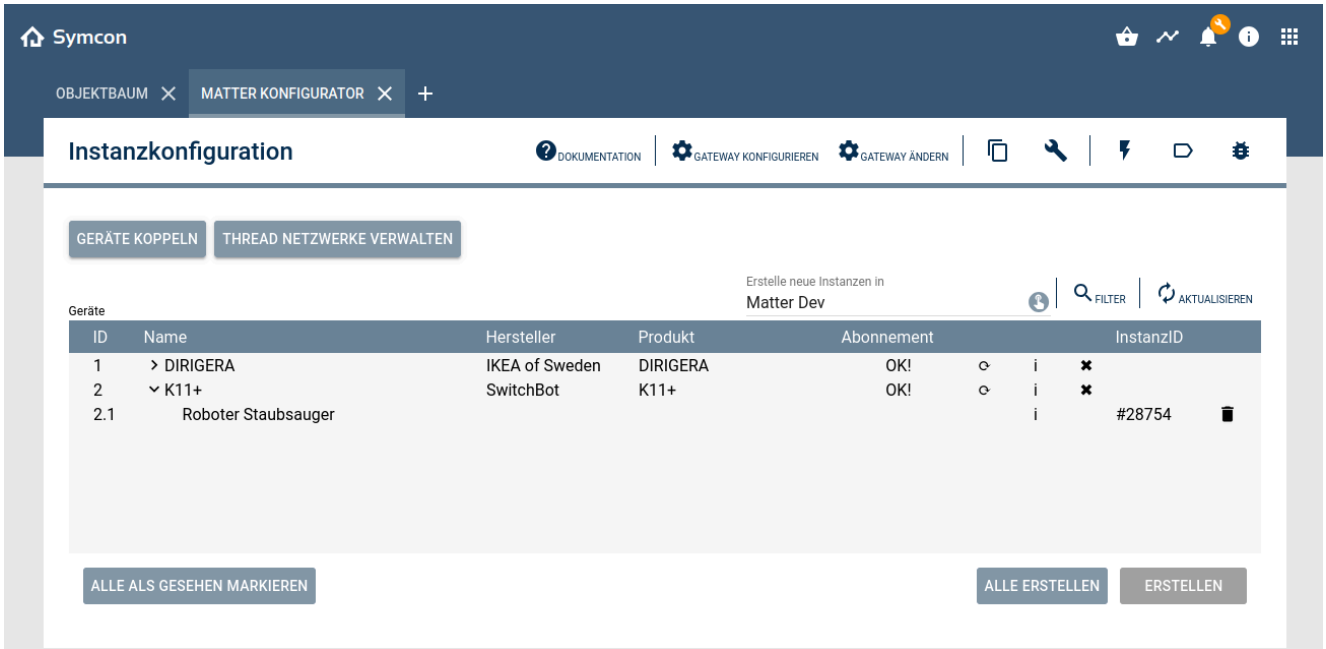
In general, we do not recommend using the Matter integration with Docker, as there are various requirements and limitations involved, some of which are highly technical and nuanced. For experienced users, we have put together a guide.

INTEGRATION INTO SYMCON

A Matter configurator is required to integrate Matter devices into Symcon. This can be added to the object tree via "+"->"Instance" using the quick filter "Matter Configurator".

SETUP

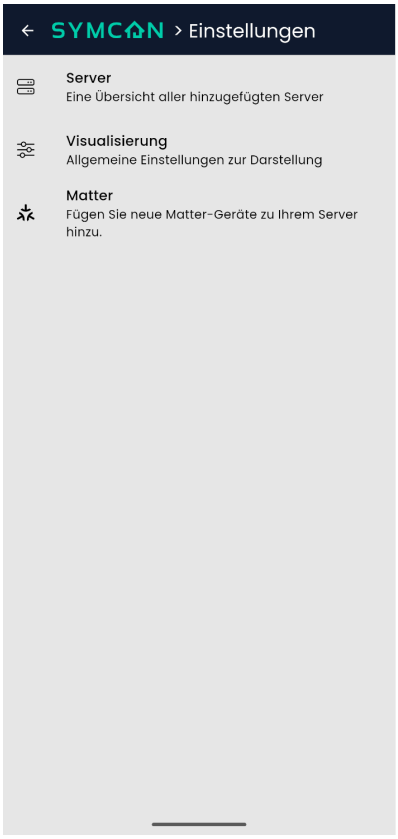
There are 2 ways to connect Matter devices to Symcon.

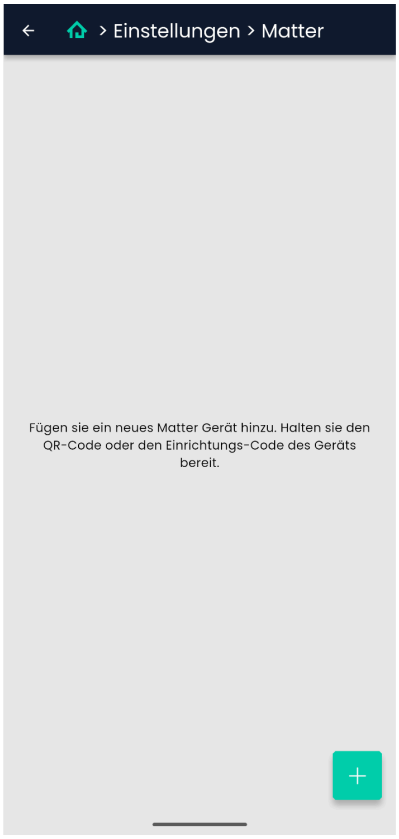


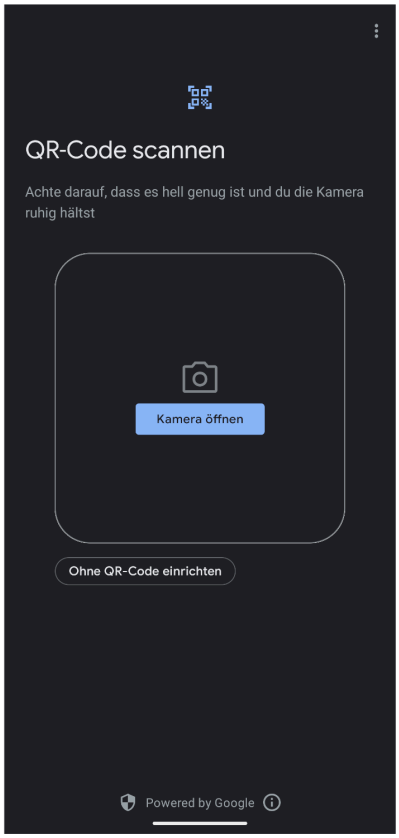
1. THE SYMCON VISUALIZATION APP

If the device is brand new or has been reset to factory settings, it can be added to the server via the Symcon app.

To do this, the Matter configurator must be opened and the "Pair via app" option selected via the "Pair device" button. The app can then be opened. Once the app is connected to the server, the setup page can be opened in the settings via the Matter menu item. The plus in the bottom right-hand corner can be used to start the pairing process for the respective operating system. After the QR code has been scanned or the pairing code has been entered manually, the device is released for the Symcon server. When the configurator signals that the device has been successfully connected, the available instances can be created.







2. SHARING WITH ANOTHER CONTROLLER

If the device is already connected to another controller (IKEA DIRIGERA Hub, Aqara Hub M3, Google Nest Hub or Apple Home Pod), a pairing code can be generated via the respective apps to release the device for other systems. In the configurator, the "Add device" option can be selected via the "Pair device" button. The pairing code can be entered in this dialog. The device is then set up and the available instances can be created.

Some devices (such as Shelly) that are in the same network as the Symcon server without being set up by a controller can be set up as described in option 2. The pairing code corresponds to that printed directly on the device under the Matter QR.

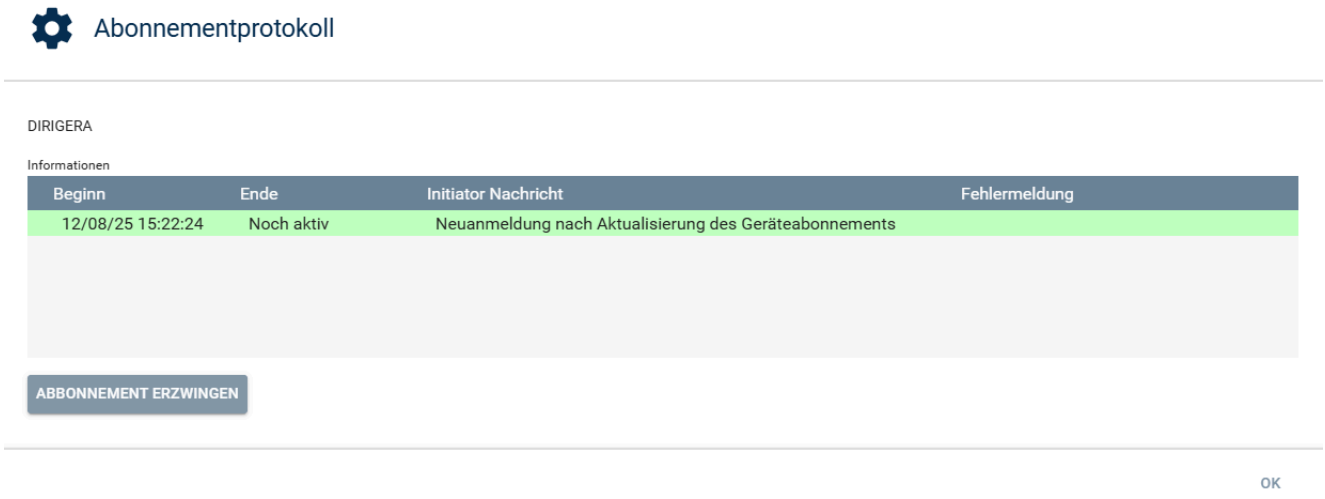


CONFIGURATOR

The configurator can be used to manage devices that have already been added.

SUBSCRIPTIONS

The subscription column displays the connection status of the individual nodes. Clicking on the text opens a dialog box that lists past and current connections and their status. In addition, a new connection can be established manually using the “Force subscription” button. This is only necessary in exceptional cases, as interrupted connections are resumed automatically.

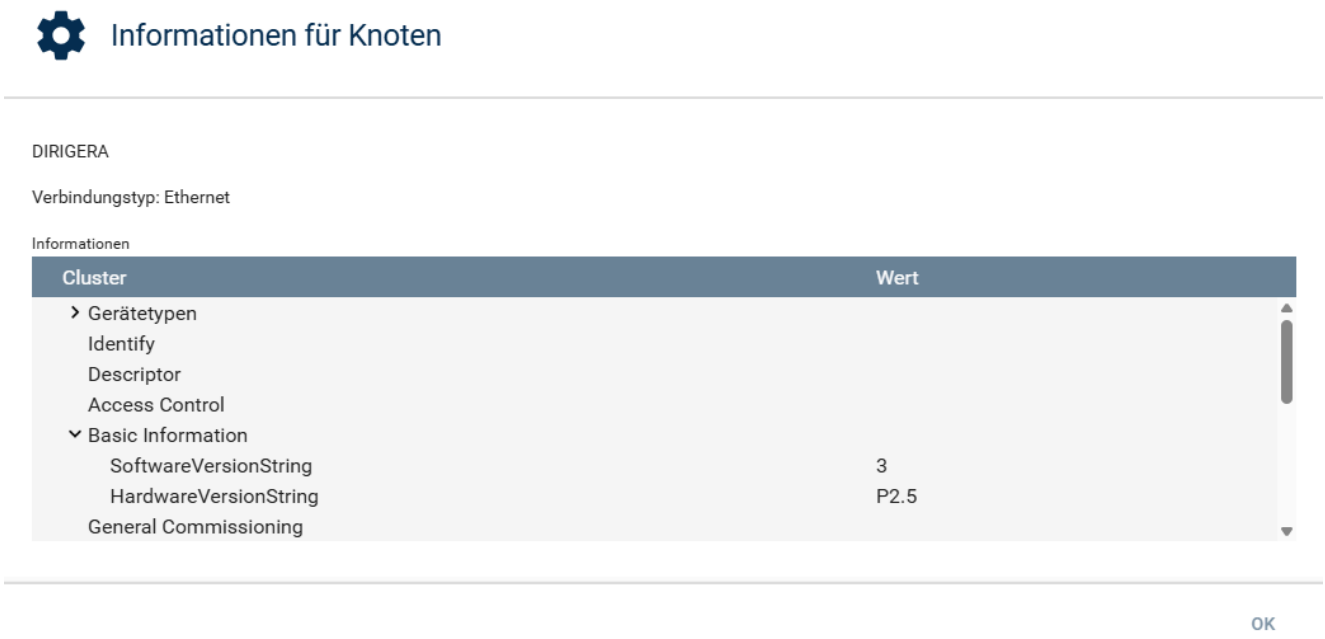


UPDATE DEVICE

All information about a device can be retrieved again by clicking on the round arrow icon. This is necessary if new devices have been added to an integrated bridge (Hue Bridge, IKEA DIRIGERA) that are not yet listed in the configurator.

DISPLAY ADVANCED INFORMATION

Clicking on the "i" in the configurator opens a dialog box that displays various information about the main node and the individual endpoints. This includes the type of connection (Ethernet, Wi-Fi, Thread), the software and hardware version, and all controllers to which the device is connected. All clusters of the selected device are also displayed.

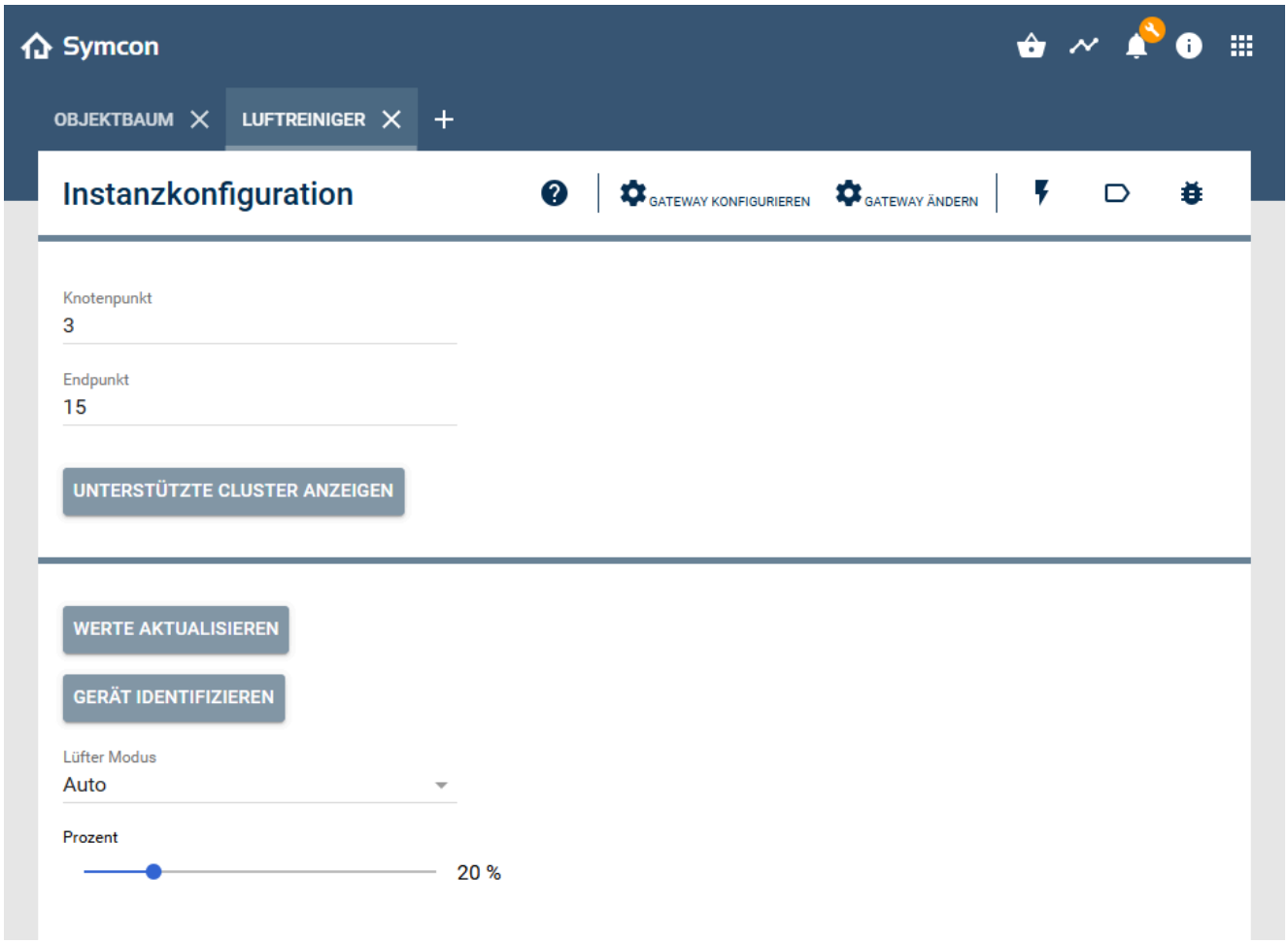


REMOVE DEVICES

The connection to the device can be disconnected by clicking on the "x" in the configurator.

MATTER DEVICE INSTANCE

A Matter device consists of one or more endpoints. An instance of the type "Matter Device" is created for each endpoint. An endpoint consists of several clusters, which represent the different functions of a device.



The "Show supported clusters" button opens a dialog box that displays the clusters available to the endpoint and whether they are supported by Symcon. The "Feature Map" is also displayed, which provides information about the range of functions of the individual clusters.
The "Update values" button reads the individual values of the device and writes them to the variables.
If a device supports the function, a time can be set using the "Identify Device" button. Upon confirmation, the physical device will respond for the specified duration, for example by flashing or buzzing.
All switchable variables are displayed in the lower area of the instance configuration and can be changed.

BATTERY-POWERED DEVICES

Some battery-powered devices are considered intermittently connected devices (ICDs). These devices may be unavailable for extended periods of time. Therefore, Symcon cannot reliably subscribe to them. The values of the ICDs can still be updated manually as long as Symcon can reach the device. Even if the device is paired with Symcon, it may not be displayed in the configurator temporarily.

SUPPORTED CLUSTERS

A Matter device consists of one or more clusters.
Not all functions offered by the individual clusters are yet supported.

- OnOff
- LevelControl
- ColorControl

OccupancySensing
TemperatureMeasurement
RelativeHumidityMeasurement
IlluminanceMeasurement
Switch
BooleanState
Pm25ConcentrationMeasurement
FanControl
RvcCleanMode
RvcOperationalState
RvcRunMode
ServiceArea
AirQuality
PowerSource
DoorLock
Thermostat
HepaFilterMonitoring
ElectricalEnergyMeasurement
ElectricalPowerMeasurement
WindowCovering
Descriptor
Identify

Note on Using Matter with Docker

In general, we do not recommend using the Matter integration under Docker, as there are various requirements and restrictions here, some of which are also very technical and subtle. Synology/QNAP NAS have a kernel that is too old, so that Thread devices cannot run there at all, and LAN/Wi-Fi devices can only run in host/MacVLAN network mode.

If Matter is to be used under Docker, one of the two requirements **must** be met:

Docker runs in MacVLAN network mode, so that the container has its own IP address in the network.

Docker runs in host network mode. The host must not have Avahi installed. Otherwise there will be server collision error messages and the mDNS network resolution will not be reliable because two Avahi stacks are running in parallel.

Matter still does not work reliably with the following restrictions:

Docker in bridge network mode does not work at all, as mDNS is not available as a result.

Docker runs correctly in host network mode, but Avahi also runs on the host. As a result, Avahi runs twice. This is particularly the case with Synology/QNAP NAS. This will interfere with finding the devices via mDNS and devices will be sporadically unreachable/unreliable.

If the kernel is too old, router announcements are not supported. This is particularly the case with Synology/QNAP NAS. In this case, Matter integration only works for LAN/WLAN devices and for Thread devices only if an IPv6 DHCP server is available that correctly assigns all addresses/routes to all devices/thread border routers.

Incorrect or unset Sysctl settings. The configurator also points this out accordingly. In particular, this configures the router announcements so that Thread devices function reliably.

Device list

Require: Symcon >= 8.2

MATTER OVER THREAD

If Matter devices are to be connected via Thread, a Thread Border Router (TBR) is required. To be able to add a "Matter over Thread" device to Symcon, the operating system (iOS or Android) must know the thread credentials of the TBR. Alternatively, learned devices can be released for Symcon via the manufacturer's app.
The TBRs of the following manufacturers were tested:

MANUFACTURER	PRODUCT
IKEA	DERIGERA Hub
Apple	e.g. HomePod mini
Google	e.g. Google TV Streamer (4K)
Aeotec	e.g. Smart Home Hub

M-Bus

M-Bus is a standardized protocol and bus system for recording consumption data. A LAN, serial or USB gateway is required to connect to IP-Symcon.



The following devices are supported by IP-Symcon:
Supported Devices

CONTENTS

Installation
Integration in IP-Symcon
Configurator
Configuration

INSTALLATION

The **M-Bus LAN Gateway** is connected to the PC via LAN. By default, DHCP and port 5000 are set up. If an individual IP address is required, the IP-Symcon "Network Configuration Tool" is required. This is available for download . A simple description of how to configure the gateway is available. The gateway can then be reached via the configured IP address and port.

INTEGRATION IN IP-SYMCON

The LAN gateway can be integrated via the Device Search. To do this, "M-Bus Discovery" must be selected as the system. The discovery instance then offers to create an M-Bus Configurator. After the configurator has been created, devices can be searched for and integrated there.

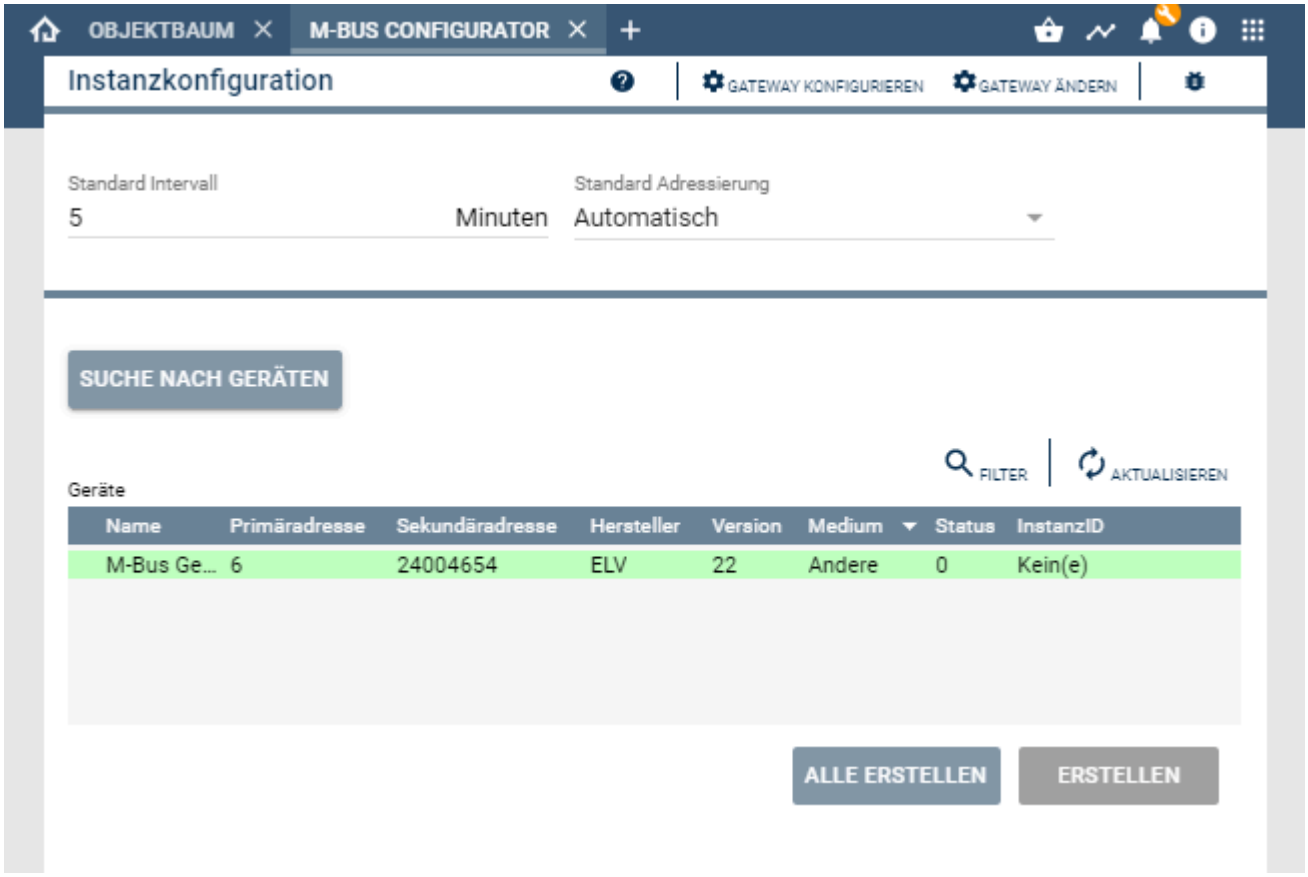
CONFIGURATOR

Default settings for newly created devices can be defined in the configurator:

PROPERTY	DESCRIPTION
Default Addressing	Selection of whether devices are to be addressed via the primary address or secondary address. With "Automatic", the secondary address is used if the primary address is 0, otherwise the secondary address is used
Default Interval	At what intervals the values should be queried (in minutes)

The default settings are only applied when creating new Instances. A change does not affect already existing Instances.

After defining the standard settings, the search for M-Bus devices can be started by clicking on the "Search for devices" button. The search may take a few minutes.
After the search, the devices found appear as instances that can be created in the configurator. To do this, a device must be selected from the list and "Create" must be clicked on.

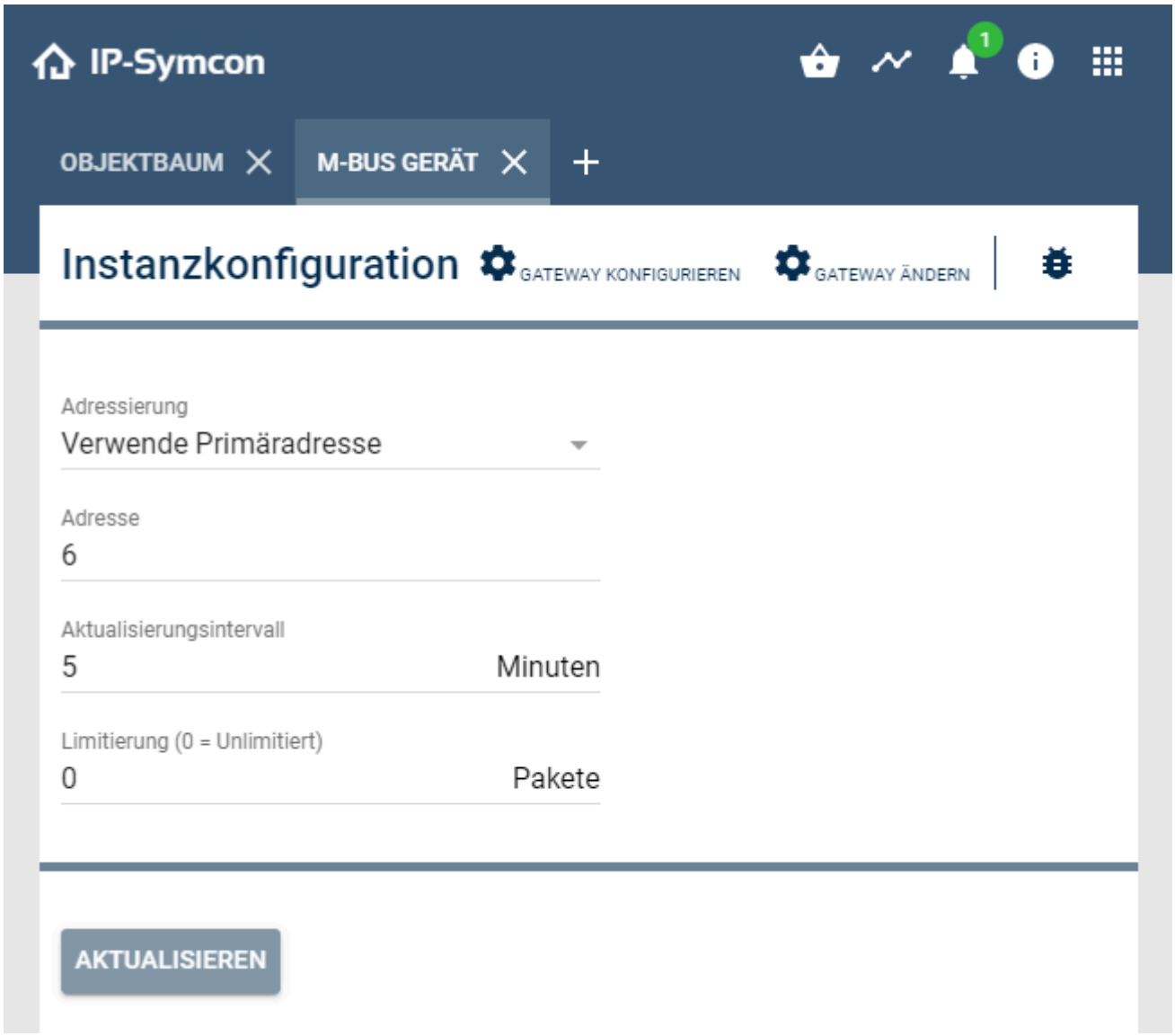


New instances are created in the object tree in the main category. These created instances can then be renamed accordingly and sorted elsewhere. It is also possible to call up the respective instance configuration via "Configure" in the configurator.

CONFIGURATION

Various properties can be set on the configuration page of the device.

PROPERTY	DESCRIPTION
Addressing	Selection of whether the device should be addressed via the primary address or the secondary address
Address	Address or ID of the device
Refresh Interval	At what intervals the values should be queried in minutes
Limitation	Limitation of packets on the bus (0 = unlimited)


IP-Symcon
1
OBJEKTBAUM
M-BUS GERÄT
Instanzkonfiguration
GATEWAY KONFIGURIEREN
GATEWAY ÄNDERN
Adressierung
Verwende Primäradresse
Adresse
6
Aktualisierungsintervall
5
Minuten
Limitierung (0 = Unlimitiert)
0
Pakete
AKTUALISIEREN

Device list

SUPPORTED GATEWAYS

In general, all M-Bus Gateways are supported.
Explicitly tested gateways:

MANUFACTURER	DESCRIPTION
SymBox with M-Bus extension Order now	Serial integrated module
Symcon M-Bus LAN Gateway order now	IP Gateway
PiiGAB	M-Bus 800
Relay	M-Bus Micro-Master
Relay	Level converter PW3
Relay	Level converter PW20
Relay	Level converter PW60
Relay	RelAir R2M HOME
Relay	RelAir R2M Pro

SUPPORTED COMPONENTS

IP-Symcon supports all devices that use the M-Bus protocol.

MBUS_UpdateValues

```
bool MBUS_UpdateValues (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to update

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Updates the values with the ID InstanceID

EXAMPLE

```
MBUS_UpdateValues(12345);
```

Mennekes

Require: Symcon >= 7.0
Mennekes offers various wallboxes/charging stations. These can be read out via ModBus TCP. A connection with Symcon is possible via a LAN-IP gateway or W-LAN, depending on the version.

The following devices are supported by Symcon:
Supported components

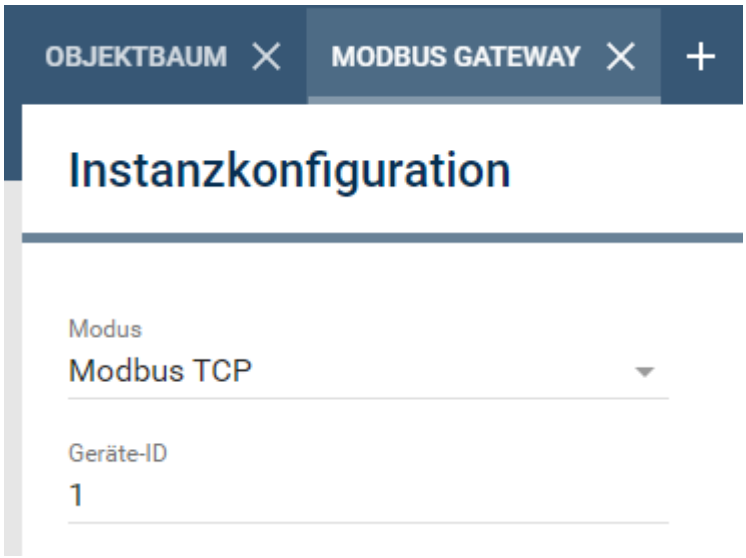
INSTALLATION

In order to use Mennekes wallboxes/charging columns in Symcon, a connection via Ethernet or W-LAN to the wallbox must be available. Within the wallbox/charging station, a USB->LAN converter is required for a wired solution, which is supplied with the wallbox/charging station.

INTEGRATION SYMCON

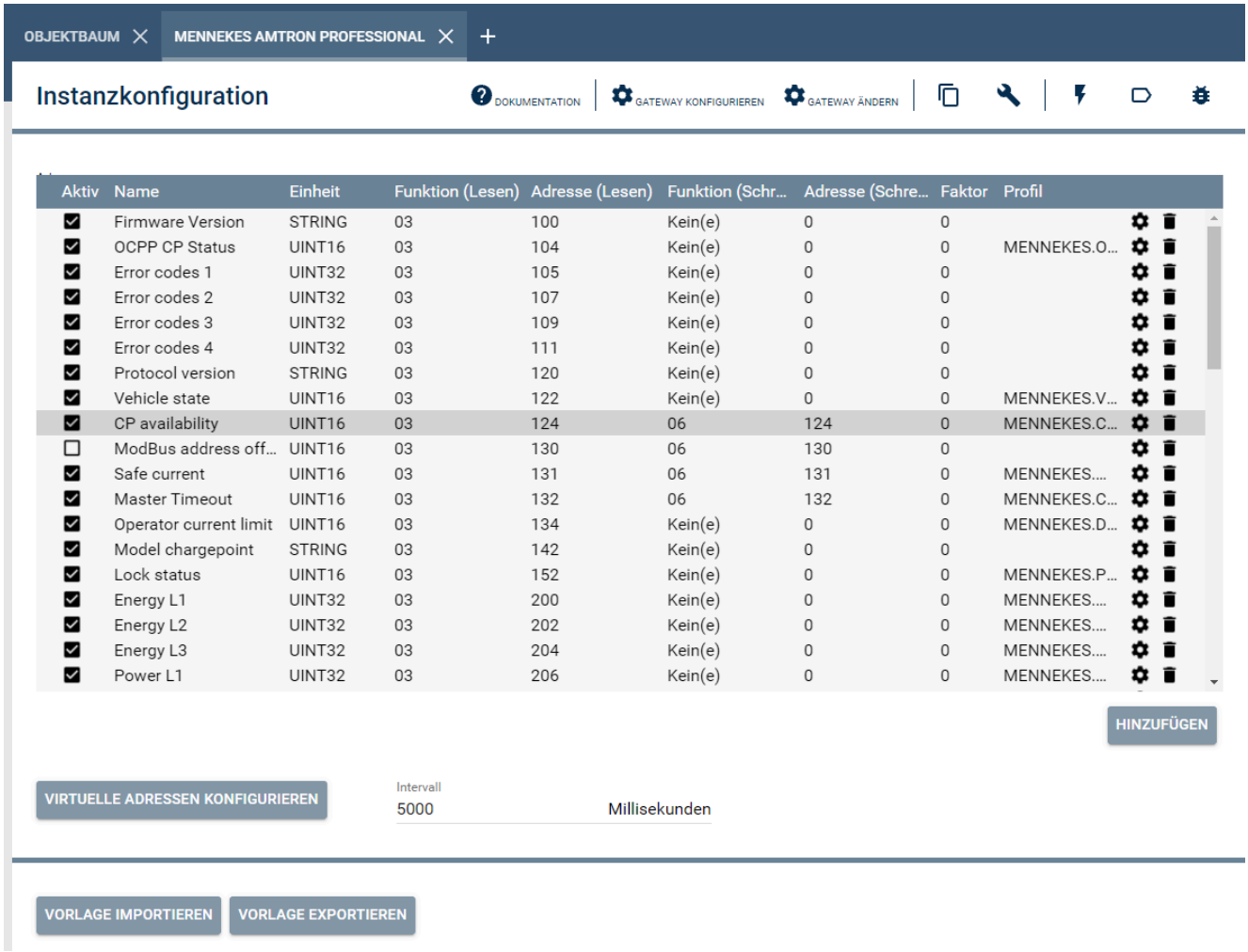
First a "ModBus Device" instance must be added within the object tree of Symcon. In the following dialog the IP address of the wallbox has to be entered. The port is 502 by default. Both information can be viewed on the web interface of the wallbox/charging station.





Afterwards the ModBus template for Mennekes wallboxes/charging stations can be downloaded. This contains the whole configuration of the ModBus device. After downloading, the "MennekesModBusx_vx.json" can be loaded via "Import". Afterwards all ModBus addresses of the common Mennekes wallboxes/charging stations are configured.

Templates in the device overview



ADD ADDRESSES

If more addresses are to be added, this can be done via "Add".
Depending on the design of the wallbox, it may be that individual addresses must be de-/activated. This can be controlled via the Active column.

PROTOCOL DESCRIPTION



Protocol description in thedevice overview

Device List

Require: Symcon >= 7.0

SUPPORTED COMPONENTS

AMEDIO PROFESSIONAL

All variants of the product family are supported.
Protocol description of all datapoints.

SYSTEM	MODBUS TEMPLATE
AMEDIO Professional 22	Download
AMEDIO Professional PnC 22	Download
AMEDIO Professional+ 22	Download
AMEDIO Professional+ PnC 22	Download
AMEDIO Professional PnC 22 C2	Download
AMEDIO Professional+ PnC 22 C2	Download

AMTRON CHARGE CONTROL

All variants of the product family are supported.
Protocol description of all datapoints.

SYSTEM	MODBUS TEMPLATE
AMTRON Charge Control 11	Download
AMTRON Charge Control 11 C2	Download

AMTRON PROFESSIONAL

All variants of the product family are supported.
Protocol description of all datapoints.

SYSTEM	MODBUS TEMPLATE
AMTRON Professional 22 C2	Download
AMTRON Professional 22	Download
AMTRON Professional PnC 22 C2	Download
AMTRON Professional PnC 22	Download
AMTRON Professional+ 22 C2	Download
AMTRON Professional+ 22	Download

SYSTEM	MODBUS TEMPLATE
AMTRON Professional+ PnC 22 C2	Download
AMTRON Professional+ PnC 22	Download
AMTRON Professional TCX 22 C2	Download
AMTRON Professional TCX 22	Download
AMTRON Professional TCX PnC 22 C2	Download
AMTRON Professional TCX PnC 22	Download
AMTRON Professional+ TCX 22 C2	Download
AMTRON Professional+ TCX 22	Download
AMTRON Professional+ TCX PnC 22 C2	Download
AMTRON Professional+ TCX PnC 22	Download

AMTRON XTRA

All variants of the product family are supported.
Protocol description of all datapoints.

SYSTEM	MODBUS TEMPLATE
AMTRON Xtra 11 C2	Download
AMTRON Xtra 22 C2	Download
AMTRON Xtra E 11/22 T2	Download
AMTRON Xtra R 11 T2	Download
AMTRON Xtra R 3,7/7,4 T2S	Download

AMTRON 4YOU

All variants of the product family are supported.
Protocol description of all datapoints.

SYSTEM	MODBUS TEMPLATE
AMTRON 4You 410 11 C2	Download
AMTRON 4You 410 7.4/22 C2	Download
AMTRON 4You 410 7.4/22 T2S	Download
AMTRON 4You 460 11 C2	Download
AMTRON 4You 510 7.4 C2	Download
AMTRON 4You 510 11 C2	Download
AMTRON 4You 510 11 T2S	Download
AMTRON 4You 510 7.4/22 T2S	Download
AMTRON 4You 510 22 C2	Download
AMTRON 4You 560 11 C2	Download

SYSTEM	MODBUS TEMPLATE
AMTRON 4You 560 22 C2	Download

AMTRON 4BUSINESS

All variants of the product family are supported.
Protocol description of all datapoints.

SYSTEM	MODBUS TEMPLATE
AMTRON 4Business 610 22 C2	Download
AMTRON 4Business 610 22 T2	Download
AMTRON 4Business 610 22 T2S	Download
AMTRON 4Business 620 22 T2	Download
AMTRON 4Business 630 22 C2	Download
AMTRON 4Business 630 22 T2	Download
AMTRON 4Business 630 22 T2S	Download
AMTRON 4Business 710 7.4 C2	Download
AMTRON 4Business 710 11 C2	Download
AMTRON 4Business 710 22 C2	Download
AMTRON 4Business 710 22 T2	Download
AMTRON 4Business 710 22 T2S	Download
AMTRON 4Business 730 7.4 C2	Download
AMTRON 4Business 730 11 C2	Download
AMTRON 4Business 730 22 C2	Download
AMTRON 4Business 730 22 T2	Download
AMTRON 4Business 730 22 T2S	Download
AMTRON 4Business 760 11 C2	Download
AMTRON 4Business 760 22 C2	Download
AMTRON 4Business 760 22 T2	Download
AMTRON 4Business 780 11 C2	Download
AMTRON 4Business 780 22 C2	Download
AMTRON 4Business 780 22 T2	Download

Modbus RTU/TCP

ModBus is a protocol based on RTU (serial binary) and TCP/IP packets. A connection with IP-Symcon is possible via LAN or serial gateway.



The following devices are supported by IP-Symcon:
Supported devices

INTEGRATION IN IP-SYMCON

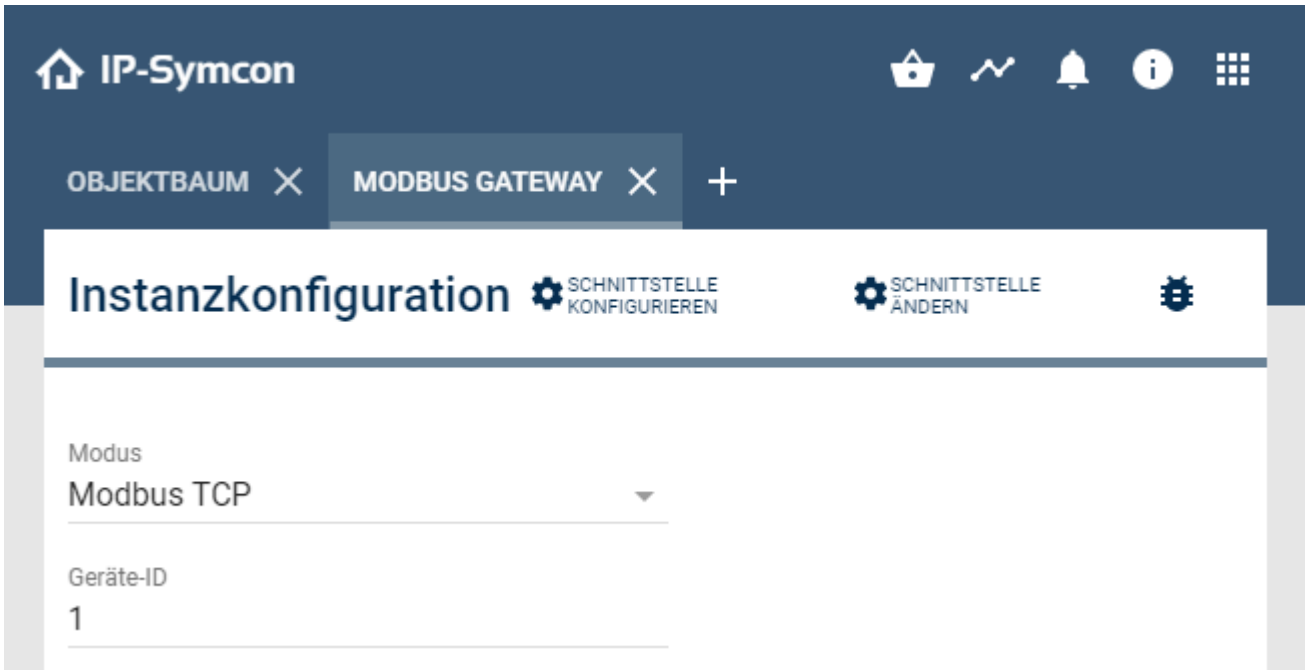
First a "ModBus" instance has to be added within IP-Symcon. There are several variants that can be created:

INSTANCE	DESCRIPTION
ModBus Device	Instance, which can map multiple addresses (coils/registers) and supports templates (7.0 and higher)
ModBus Coil.	Instance which represents a single address (coil)
ModBus Address	Instance which represents a single address (register)



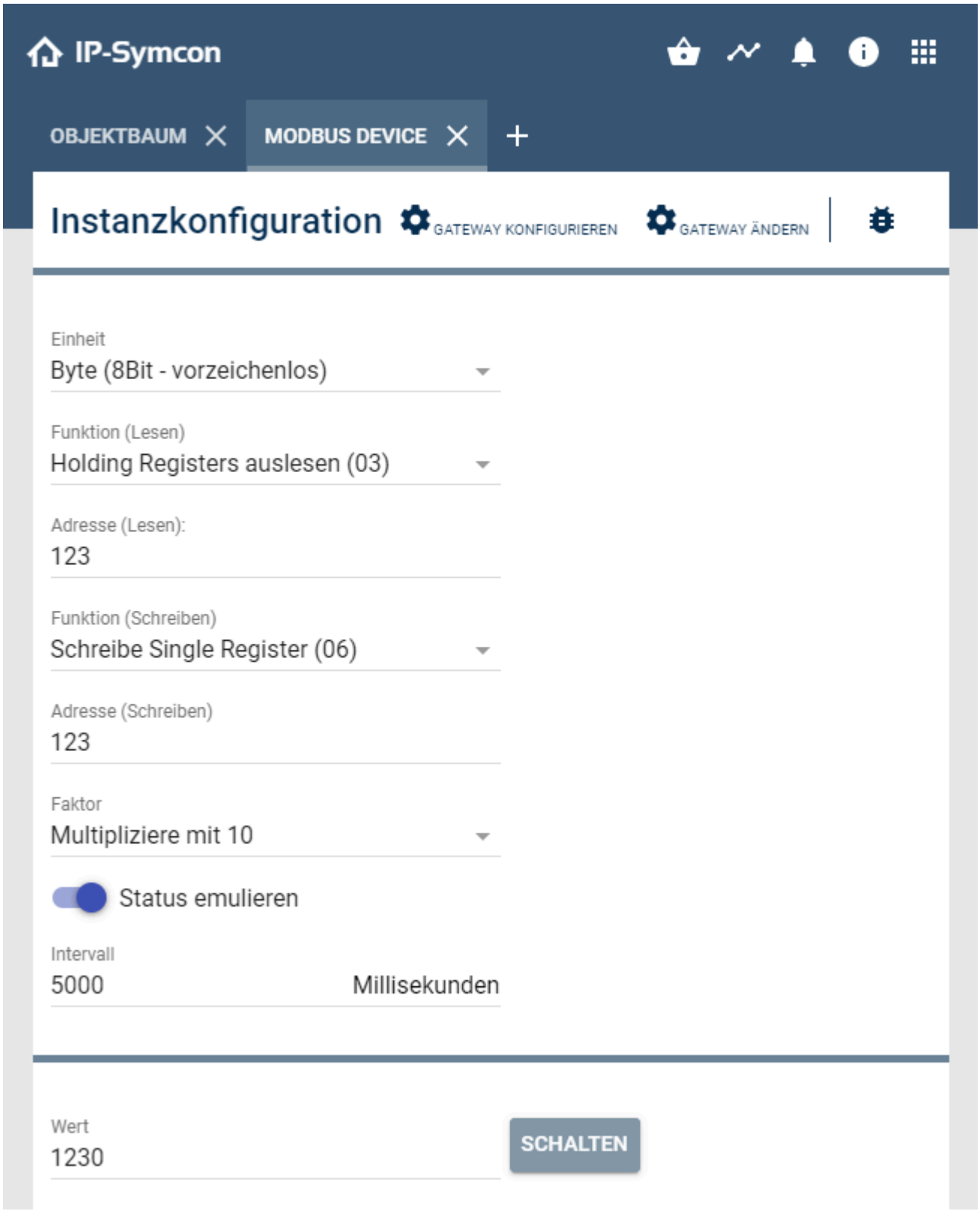
Many templates can be found in our community: Show templates

The configuration of the parent gateway and I/O instance must then be adapted to the connected device. It must be selected via which protocol and device ID the device is addressed. The device ID is particularly important for RTU connected devices. For devices connected to TCP, this is often device ID 1.



In order to be able to establish a connection to the device, IP address and port (default:502) must be entered in the gateway's parent I/O instance.
Further configuration (screenshot below) is as follows.

OPTION	DESCRIPTION
Unit	Data type of the variable of the device
Function (Read)	Function according to which the values are to be read
Address (Read)	Device address of the register to be read from
Function (Write)	Function according to which the values are to be written
Address (Write)	Device address of the register to be written to
Factor (Numeric values)	The factor multiplies or divides the received value and writes it to the device variable
Length (Strings)	If the length is greater than 0, it defines the precise number of characters (Bytes) queried. Note: 1 register = 2 characters (Bytes)
Byte order	Depending on the device, the byte order must be adapted. Often this is not documented, but must be tried out. Big Endian and Little Endian (byte swap) are the common values.
Emulate status	"Emulate status" means that the value of the variable is set to the new value if the write command is successful and does not depend on a read command
Interval	If the interval is greater than 0, the address (read) is queried cyclically on the ModBus device and the variable of the device is updated



Due to the fact that we still support 32 bit systems Int64 values are mapped as Float64.

MODBUS DEVICE

If multiple addresses are to be queried for a device, the Modbus device should be used. Here several addresses can be entered as a table and virtual addresses can be defined.

Configuration:

NAME	DESCRIPTION
Addresses	Tabular listing of the defined addresses
Configure virtual addresses	Button with which a list of virtual addresses can be defined
Byte order	The byte order must be adjusted depending on the device. This is often not documented but has to be tried out. Big Endian and Little Endian (bytes swapped) are the common values.
Interval	If the interval is set to greater than 0, the address (read) of the ModBus device is queried cyclically and the device variable is updated.
Import template	Using this button, a template of the tables can be quickly inserted.
Export template	Through this button,



The structure of template files is described in detail under Templates (file format).

| A template can be created from the settings made | |
Contents of the address list:

NAME	DESCRIPTION
Active	Specifies whether the variable should be created
Name	Name, used as variable name
Unit	Data type of the register
Function (Read)	Function according to which the values should be read
Address (Read)	Register to be read from
Function (Writing)	Function according to which the values should be written
Address (writing)	Register to be written to
Profile	Profile, which should get the variable

Under the expert options, the ident can be customized and a translation for the variable name can also be provided.
Contents of the virtual address list:

NAME	DESCRIPTION
Active	The variable is created
Name	Name, used as variable name
Variable type	Type of variable
Profile	Profile of the variable
Read	Script that provides the values of the virtual address

NAME	DESCRIPTION
Write	Script that passes the values to the variables.

In the read script, all values with their respective idents are available in $VALUES. The new value,
which should load in the variable of the virtual address is simply returned with return. If no value is to be written (e.g. because invalid), null can be returned. (from 7.1)

```
return ($VALUES["A_3_3_23296"] + $VALUES["A_3_3_23298"] +
$VALUES["A_3_3_23300"])/3;
```

The write script expects an associative array as return, which contains all the idents to be written to. $VALUE contains the new value,
which was requested via RequestAction.

```
return ["A_3_3_23296" => $VALUE];
```

FUNCTIONCODES

Depending on the FunctionCode, a suitable address must also be entered. This can be taken from the table.
The table below also provides an overview of which FunctionCode is sent with which parameterization within IP-Symcon:

FUNCTIONCODE	DEVICE ADDRESS	READ/WRITE ADDRESS	SURNAME
0x01 (1)	1 - 10000	Device address - 1	Read Coils
0x05 (5)	1 - 10000	Device address - 1	Write Single Coil
0x02 (2)	10001 - 20000	Device address - 10001	Read Discrete Inputs
0x03 (3)	40001 - 50000	Device address - 40001	Read Holding Registers
0x10 (16)	40001 - 50000	Device address - 40001	Write Multiple registers
0x04 (4)	30001 - 40000	Device address - 30001	Read Input Registers



If, for example, address 40123 is to be written to (FunctionCode = 0x10), the appropriate data type (unit) must be selected and write address 122 (40123 - 40001) entered.



Some manufacturers do not adhere to this convention. It must be checked in the respective instructions or data sheets whether the device address must be subtracted depending on the function code or whether absolute addresses apply

DATA TYPES

DATA TYPE	SIGN	BITS
BOOL	unsigned	1
UINT8MSB	unsigned	8
UINT8LSB	unsigned	8
UINT16	unsigned	16
UINT32	unsigned	32
UINT64	unsigned	64
INT8MSB	signed	8
INT8LSB	signed	8
INT16	signed	16
INT32	signed	32
INT64	signed	64
FLOAT32	signed	32
FLOAT64	signed	64
STRING		

Device list

SUPPORTED GATEWAYS/COMPONENTS

IP-Symcon supports all devices that use the ModBus RTU or ModBus TCP protocol.
In particular, devices and PLCs from Wago/Beckhoff/ABB.

SUPPORTED COMPONENTS

IP-Symcon supports all components.

MANUFACTURER	DESCRIPTION
SymBox with RS485 (ModBus RTU) extension Order now	Serial integrated module
Exsys EX-6051 (nonbinding recommendation)	Serial to LAN converter for mode "ModBus RTU over TCP"
Waveshare RS485 to Ethernet Converter (nonbinding recommendation)	Serial to LAN converter for mode "ModBus RTU over TCP"

ModBus_RequestRead

```
bool ModBus_RequestRead (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Executes a new read for the device with the ID InstanceID.
Once the function has been performed, the new values are available in the state variables.

EXAMPLE

```
//Read out device with the ID 12345
ModBus_RequestRead(12345);
```

ModBus_WriteCoil

```
bool ModBus_WriteCoil (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sets address with ID InstanceID to Value

EXAMPLE

```
//Turns instance with the ID "12345" on
ModBus_WriteCoil(12345, true);
```

ModBus_WriteRegister

```
bool ModBus_WriteRegister (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	32-bit floating point value according to IEEE754 or integer

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Writes the value **Value** to the configured address (write) of the instance with the ID **InstanceID** without knowing the exact data type.



This function cannot be used for bit values. ModBus_WriteCoil must be used for bit values.

EXAMPLE

```
////Writes the value 23.5 to the register of the instance with ID 12345
ModBus_WriteRegister(12345, 23.5);

////Writes the value 12 to the register of the instance with ID 23456
ModBus_WriteRegister(23456, 12);
```

ModBus_WriteRegisterByte

```
bool ModBus_WriteRegisterByte (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	0-255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes 123 into the register of the instance with the ID 12345
ModBus_WriteRegisterByte(12345, 123);
```

ModBus_WriteRegisterChar

Require: Symcon >= 4.4

```
bool ModBus_WriteRegisterChar (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-128 to 127

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function was named **ModBus_WriteRegisterShortInt** up to and including Version 4.3

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes -123 into the register of the instance with the ID 12345
ModBus_WriteRegisterChar(12345, -123);
```

ModBus_WriteRegisterDWord

```
bool ModBus_WriteRegisterDWord (int $InstanceID, int $Wert)
```

PARAMETERS

InstanceID	ID of the device to be switched
Wert	0-4294967295

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**



On armhf systems, only the range up to 2147483647 can be used due to the limitation to 32-bit integers in PHP. armhf is supported until Symcon 10.0.

EXAMPLE

```
//Writes 123 into the register of the instance with the ID 12345
ModBus_WriteRegisterDWord(12345, 123);
```

ModBus_WriteRegisterInt64

```
bool ModBus_WriteRegisterInt64 (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-9223372036854775808 bis 9223372036854775807

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes -123 into the register of the instance with the ID 12345
ModBus_WriteRegisterInt64(12345, -123);
```

ModBus_WriteRegisterInteger

```
bool ModBus_WriteRegisterInteger (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-2147483648 bis 2147483647

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes -123 into the register of the instance with the ID 12345
ModBus_WriteRegisterInteger(12345, -123);
```

ModBus_WriteRegisterReal

```
bool ModBus_WriteRegisterReal (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	32bit floating point value according to IEEE754

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes 23.5 into the register of the instance with the ID 12345
ModBus_WriteRegisterReal(12345, 23.5);
```

ModBus_WriteRegisterReal64

```
bool ModBus_WriteRegisterReal64 (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	64bit floating point value according to IEEE754

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes 23.5 into the register of the instance with the ID 12345
ModBus_WriteRegisterReal64(12345, 23.5);
```

ModBus_WriteRegisterShort

Require: Symcon >= 4.4

```
bool ModBus_WriteRegisterShort (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-32768 to 32767

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function was named **ModBus_WriteRegisterSmallInt** up to and including Version 4.3

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes -123 into the register of the instance with the ID 12345
ModBus_WriteRegisterShort(12345, -123);
```

ModBus_WriteRegisterString

```
bool ModBus_WriteRegisterString (int $InstanceID, string $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	String which should be written into register

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes "Hello world" into the register of the instance with the ID
12345
ModBus_WriteRegisterString(12345, "Hello world");
```

ModBus_WriteRegisterWord

```
bool ModBus_WriteRegisterWord (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	0-65535

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets address with ID **InstanceID** to **Value**

EXAMPLE

```
//Writes 123 into the register of the instance with the ID 12345
ModBus_WriteRegisterWord(12345, 123);
```

Templates (file format)

Require: Symcon >= 7.0
The "ModBus Device" instance can export its complete configuration as a template and import it again. A template is a JSON file containing all addresses, virtual addresses, required variable profiles, the byte order and the polling settings. A device that has been set up once can thus be set up on other systems with a few clicks or shared with other users.



Many ready-made templates can be found in our community: Show templates

This page describes the file format completely, so that templates can also be created directly from a manufacturer's data sheet or generated by a script.

IMPORT AND EXPORT

ACTION	DESCRIPTION
Export template	Creates a JSON file from the current configuration. All used profiles that do not start with "~" are exported as well.
Import template	Replaces addresses, virtual addresses, byte order and polling settings in the configuration form. The changes are only saved with "Apply". Missing profiles are created in the process, existing profiles are not modified.

The file is checked before the import. If it does not contain the three keys "Addresses", "VirtualAddresses" and "Profiles", the import is aborted with the message "This is not a valid ModBus template!". If a profile already exists with different settings, a hint is shown (e.g. "2 profiles do not match!"). In this case the existing profile is used.



The import overwrites the existing configuration of the instance. Variables whose ident no longer occurs in the new configuration are deleted when applying - including their archive data.

STRUCTURE

A template is a JSON object with the following keys:

KEY	TYPE	REQUIRED	DESCRIPTION
Addresses	Array	Yes	List of Modbus addresses, see Addresses
VirtualAddresses	Array	Yes	List of virtual addresses, see Virtual addresses, may be empty
Profiles	Object	Yes	Variable profiles created on import, see Profiles, may be empty

KEY	TYPE	REQUIRED	DESCRIPTION
ByteOrder	Number	No	Byte order of the instance, see Byte order
Requests	Object	No	Polling settings, see Polling

The order of the keys is arbitrary. Unknown keys are ignored. A minimal, valid template with a single address looks like this:

```
{
"Addresses": [
{
"Active": true,
"Name": "Battery state of charge",
"Ident": "battery_soc",
"Translation": [
{ "Language": "de", "Name": "Batterieladezustand" }
],
"DataType": 2,
"ReadFunctionCode": 4,
"ReadAddress": 33139,
"WriteFunctionCode": 0,
"WriteAddress": 0,
"Factor": 0,
"Length": 0,
"ByteOrder": -1,
"Profile": "~Battery.100"
}
],
"VirtualAddresses": [],
"Profiles": {},
"ByteOrder": 0,
"Requests": {
"Type": 0,
"Interval": 5000,
"DataBlocks": []
}
}
```

ADDRESSES

Each entry in "Addresses" describes one value of the device. A variable is created below the instance for each active entry.

FIELD	TYPE	DESCRIPTION
Active	Bool	Defines whether the variable is created and the value is read. Inactive entries remain in the template and can be activated by the user later.
Name	String	Name of the variable. An English name translated via "Translation" is recommended.
Ident	String	Ident of the variable. If the field is empty, the ident is generated automatically, see Ident.

FIELD	TYPE	DESCRIPTION
Translation	Array	Translations of the name as a list of objects with "Language" (e.g. "de") and "Name".
DataType	Number	Data type of the register, see Data types.
ReadFunctionCode	Number	Function code for reading: 0 (do not read), 1, 2, 3 or 4, see Function codes.
ReadAddress	Number	Read address (start register or coil), see Addressing.
WriteFunctionCode	Number	Function code for writing: 0 (do not write), 5, 6, 15 or 16. If a function code is set, the variable gets a standard action.
WriteAddress	Number	Write address. Usually identical to the read address.
Factor	Number	Factor the read value is multiplied with. 0 means "no factor", see Factor.
Length	Number	Strings only: length in bytes (2 bytes per register). 0 for all other data types.
ByteOrder	Number	Byte order of this address. -1 uses the setting of the instance, see Byte order.
Profile	String	Name of the variable profile or empty. The profile type must match the variable type, see Data types.



All fields should always be specified completely and with the correct JSON type. Numbers must not be given as a string ("3") or as null, otherwise applying the configuration fails. The exception is "Active": if it is missing, the address is considered active.

Older exports sometimes contain additional fields like "SwapBytes" or "CustomFactor". These are ignored and can be omitted. The value of a custom factor is always stored in the field "Factor".

DATA TYPES IN TEMPLATES

In the template the data type is stored as a number. For compatibility reasons the numbers are not sorted by size.

DATATYPE	SHOWN IN THE CONSOLE	REGISTERS	VARIABLE TYPE (WITHOUT FACTOR)
0	BOOL	1 bit	Boolean
1	UINT8 (MSB)	1	Integer
12	UINT8 (LSB)	1	Integer
2	UINT16	1	Integer
3	UINT32	2	Integer
11	UINT64	4	Float
4	INT8 (MSB)	1	Integer
13	INT8 (LSB)	1	Integer
5	INT16	1	Integer

DATATYPE	SHOWN IN THE CONSOLE	REGISTERS	VARIABLE TYPE (WITHOUT FACTOR)
6	INT32	2	Integer
8	INT64	4	Float
7	FLOAT32	2	Float
9	FLOAT64	4	Float
10	STRING (PLAIN)	Length/2	String
14	STRING (HEX)	Length/2	String

MSB/LSB: UINT8/INT8 read a whole register and use the high-order (MSB) or the low-order byte (LSB). This way two 8 bit values can be read from one register with two entries on the same address.
64 bit: INT64 and UINT64 are mapped to a float variable because Symcon also supports 32 bit systems.
STRING (PLAIN): The bytes are interpreted as text. Spaces and null bytes at the beginning and end are removed.
STRING (HEX): The bytes are shown as a hex string (upper case), e.g. "0A1B". This is useful for bit fields, version numbers or MAC addresses.
Factor: As soon as a factor other than 0 is set, the variable is always created as float - also for integer data types and also for the factor 1.



The type of a profile must match the variable type. A UINT16 register without factor results in an integer variable and can therefore not get the float profile "~Watt", for example. If an integer value should get a float profile, the factor 1 can be set.

FUNCTION CODES AND ADDRESSES

FUNCTION CODE	DIRECTION	NAME	ALLOWED DATA TYPES
1	Read	Read Coils	BOOL
2	Read	Read Discrete Inputs	BOOL
3	Read	Read Holding Registers	all except BOOL
4	Read	Read Input Registers	all except BOOL
5	Write	Write Single Coil	BOOL
15	Write	Write Multiple Coils	BOOL
6	Write	Write Single Register	8/16 bit types only
16	Write	Write Multiple Registers	all except BOOL

Invalid combinations are rejected with an error message when applying, e.g. "Non-Bit values must use function Read Holding Registers/Read Input Registers" or "Writing in only one register is not possible for multi register values like Int32/UInt32, Int64/UInt64, Float32/Float64, String".
Addressing: "ReadAddress" and "WriteAddress" are the addresses that are actually transmitted in the Modbus telegram, starting at 0. Many data sheets use the classic notation with a prefix instead (e.g. 40001 for the first holding register). In this case the prefix has to

be subtracted, see Function codes. Other manufacturers already specify the addresses directly (e.g. Solis with 33000 for an input register). If in doubt, a test with a known value like the serial number or the grid frequency helps.

BYTE ORDER

The byte order is defined for the whole instance in the top-level key "ByteOrder". Single addresses can override it with their own "ByteOrder" field. The value -1 uses the setting of the instance. The examples show how the 32 bit value 0x11223344 is transmitted by the device:

BYTEORDER	NAME	TRANSMITTED BYTES	TYPICAL DESCRIPTION IN THE DATA SHEET
-1	Inherited from device	----------	address level only
0	Big-Endian (Standard)	11 22 33 44	"High word first", "ABCD", Modbus standard
1	Little-Endian	44 33 22 11	"DCBA"
2	Big-Endian (Byte Swap)	22 11 44 33	"BADC"
3	Little-Endian (Byte Swap)	33 44 11 22	"Low word first", "Word swap", "CDAB"

For 16 bit values only 1 and 2 have an effect (bytes swapped within the register). The byte order has no meaning for BOOL values.

FACTOR

The raw value is multiplied by "Factor" before it is written into the variable. When writing, the value is divided by the factor first. The factor is always a multiplier: a division by 10 is specified as 0.1.

DATA SHEET	FACTOR
Unit 0.1 V	0.1
Unit 0.01 Hz	0.01
Unit 10 W	10
Wh, wanted in kWh	0.001
no factor	0

The factor must be 0 for BOOL and string addresses. If the device provides its own scale factor in a register (e.g. SunSpec "Scale Factor"), it can be applied using a virtual address.

IDENT

The ident identifies the variable permanently. If "Ident" is empty, it is generated from data type, function code and read address: A*[DataType]*[ReadFunctionCode]_[ReadAddress], e.g. "A_7_3*100". For virtual addresses the name is used, with all characters except letters, digits and underscore replaced by "*".



Without a fixed ident, the ident changes as soon as data type, function code or address (or the name for virtual addresses) are modified. The old variable is then deleted including its archive data and a new one is created. Templates should therefore always set a fixed, meaningful ident like "battery_soc". Each ident must only occur once - also not between addresses and virtual addresses.

The ident is also used in the scripts of the virtual addresses to access the values.

VIRTUAL ADDRESSES

Virtual addresses calculate a variable with PHP from the values of the other addresses or distribute a written value to one or more addresses.

FIELD	TYPE	DESCRIPTION
Active	Bool	Defines whether the variable is created and the scripts are executed.
Name	String	Name of the variable
Ident	String	Ident of the variable. If the field is empty, the ident is generated from the name.
Translation	Array	Translations of the name, like for addresses
VariableType	Number	0 = Boolean, 1 = Integer, 2 = Float, 3 = String
Profile	String	Name of the variable profile or empty. The profile type must match the VariableType.
ReadAction	String	PHP code to calculate the value, empty for no calculation
WriteAction	String	PHP code for writing. If it is set, the variable gets a standard action.

The scripts only contain the body of a function, i.e. without `<?php`. Line breaks are given as \n (or \r\n) in JSON.
ReadAction: Is executed after each poll. $VALUES contains all address values as an array with the ident as key. Virtual addresses are calculated in order and their result is added to $VALUES as well - a virtual address can therefore access the results of the previous ones. The return value is written into the variable. If null is returned, the variable stays unchanged. The return value must match the VariableType.

```
return ($VALUES['pv_voltage_1'] ?? 0) * ($VALUES['pv_current_1'] ?? 0);
```

WriteAction: Is executed when the variable is switched. $VALUE contains the new value, $VALUES the current values of all readable addresses. An array with the ident as key and the value to be written is returned. For each entry the action of the corresponding address is executed, i.e. including factor and data type. If null is returned, nothing is written.

```
// Only change bits 0-3 of a bit field, keep all other bits
$current = $VALUES['control_register'] ?? 0;
return ['control_register' => ($current & ~0x0F) | ($VALUE & 0x0F)];
```



The operator ?? 0 prevents errors as long as an address has not been read yet or is inactive. If a script outputs errors, they can be found in the debug of the instance under the ident of the virtual address.

Values are only written if they change or the variable is older than 60 seconds. This applies to read addresses as well as virtual addresses.

PROFILES

"Profiles" is an object with the profile name as key. Profiles whose name starts with "~" are system profiles. They are not exported and do not have to be contained in the template. Custom profiles should get a unique prefix (e.g. "Vendor.Name") so that they do not collide with profiles of other templates.

FIELD	TYPE	DESCRIPTION
Type	Number	0 = Boolean, 1 = Integer, 2 = Float, 3 = String
Prefix	String	Prefix
Suffix	String	Suffix, including a leading space, e.g. " W"
MinValue	Number	Minimum value
MaxValue	Number	Maximum value
StepSize	Number	Step size. For switchable variables it defines the display as slider.
Digits	Number	Number of decimal places
Icon	String	Name of the icon or empty
Associations	Array	Associations, each with "Value", "Name", "Icon" and "Color" (-1 = no color)

All fields must be specified. The JSON type of "Value" in the associations should match the profile type (true/false for Boolean, whole number for Integer).

```
"Profiles": {
"Vendor.OperatingMode": {
"Type": 1,
"Prefix": "",
"Suffix": "",
```

```
"MinValue": 0.0,
"MaxValue": 0.0,
"StepSize": 0.0,
"Digits": 0,
"Icon": "Information",
"Associations": [
{ "Value": 0, "Name": "Standby", "Icon": "", "Color": -1 },
{ "Value": 1, "Name": "Running", "Icon": "", "Color": 65280 }
]
}
}
```



Only missing profiles are created on import. An existing profile with the same name is not modified, even if it differs.

POLLING

The key "Requests" contains the polling settings:

FIELD	TYPE	DESCRIPTION
Type	Number	0 = Single addresses, 1 = Data blocks
Interval	Number	Polling interval in milliseconds for the type "Single addresses". 0 disables polling.
DataBlocks	Array	Data blocks for the type "Data blocks"

Single addresses: In each interval every active address with a read function code is polled individually. This is simple, but creates a lot of requests for many addresses. Especially with Modbus RTU or slow data loggers the time is often not sufficient.
Data blocks: Each data block reads a contiguous range with a single request and has its own interval. Afterwards all addresses that are completely within the block and use the same function code are updated. This way measured values can be read every 5 seconds and meter readings only every minute, for example.

FIELD	TYPE	DESCRIPTION
Function	Number	Function code 1, 2, 3 or 4
Address	Number	Start address
Quantity	Number	Number of registers or coils (at most 125 registers)
Poller	Number	Interval in milliseconds

```
"Requests": {
"Type": 1,
"Interval": 5000,
"DataBlocks": [
{ "Function": 4, "Address": 33049, "Quantity": 10, "Poller": 5000 },
{ "Function": 4, "Address": 33161, "Quantity": 20, "Poller": 60000 },
{ "Function": 3, "Address": 43110, "Quantity": 1, "Poller": 10000 }
```

```
]
}
```



With the type "Data blocks", addresses that are not within any data block are not updated. A data block should only cover registers that the device actually supports - many devices answer a request with a gap in the register range with an error (ILLEGAL_DATA_ADDRESS) for the whole block.

CHECKLIST

All three required keys "Addresses", "VirtualAddresses" and "Profiles" are present, even if they are empty.
Each address contains all fields with the correct JSON type (no strings or null instead of numbers).
Each address and virtual address has a fixed, unique ident.
Names are English and have a German translation in "Translation".
Addresses are specified as they are transmitted in the telegram (without 30001/40001 prefix).
Function code and data type match (BOOL only with 1, 2, 5, 15; multi register values not with 6).
The profile type matches the variable type (a factor other than 0 always results in float).
Custom profiles have a unique prefix and are completely contained in "Profiles".
Rarely needed values are included with "Active": false instead of being omitted.
Data blocks are used for many addresses and every active address is within a block.
The template has been imported, applied and exported again once. The export must contain the same addresses.

Möhlenhoff Alpha 2

Require: Symcon >= 4.0
The module is used for receiving and switching Möhlenhoff Alpha2 data.

FUNCTION SCOPE

Setting variables via script command or Visualization.
Reading out the data of the Möhlenhoff Alpha 2

SOFTWARE INSTALLATION

Install the Möhlenhoff Alpha 2 module via the Module Store.

INSTANCE SETUP IN IP-SYMCON

Under "Add Instance" the 'Möhlenhoff Alpha 2' module can be found using the quick filter.
More information about adding instances in the Documentation of Instances

CONFIGURATION PAGE:

NAME	DESCRIPTION
IP address	IP address of the Alpha 2
Interval	Adjustable interval in seconds in which the data of the Alpha 2 should be queried. (Default: 0)
Heating zone 1..12	Activates the readout of the respective heating zone. (The heating circuits are always all read out)
"Read out device"	Reads out the data manually. Must also be done once at the beginning.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

The variables are created automatically.

NAME (BASE)	TYPE	DESCRIPTION
Base Setback Temperature	Float	Serves as a "category" in which all monitored locations, as well as the time stamp and longitude/latitude are located. Created per device.
Base Setback Input	Boolean	
Basic Automatic Time Change	Boolean	On/Off of automatic time change
Basic Operating Mode Heating/Cooling (CO input)	Boolean	Controls the operating mode (On = cooling, Off = heating)
Basic Frost Protection	Boolean	On/Off of Frost Protection
Basic Frost Protection Temperature	Integer	Temperature of Frost Protection
Base HW Version	String	Hardware Version of the Base
Base ID	String	Base ID
Base Cooling Mode	Boolean	Cooling Mode On/Off
Base Rank in System Network	Integer	Edge of Base (0 = Standalone, 1 = Master, 2 = Slave)
Base Smartstart on/off	Boolean	On/off of Smartstart
Base SW ETH Version	String	Software Version Ethernet
Basic SW STM Version	String	Software Version STM
Basic Dewpoint Sensor	Boolean	On/Off of Dewpoint Sensor
Basic Temperature Limiter	Boolean	On/Off of Temperature Limiter
Basic Temperature Unit	Integer	Setting whether Fahrenheit or Celsius (0 = °C, 1 = °F)
Basic Vacation Status	Integer	Vacation Status (0 = off, 1 = scheduled, 2 = active)
Base Vacation Temperature Heating	Float	Temperature during vacation

NAME (HEATING ZONE 1..12)	TYPE	DESCRIPTION
Heating Zone 01..12 Actual Temperature	Float	Actual Temperature
Heating zone 01..12 set temperature	float	set temperature
Heating zone 01..12 Operating mode	Integer	Operating mode of heating zone (0 = Auto, 1 = Day, 2 = Night)
Heating Zone 01..12 Actual Temperature Ext. Sensor	Float	Current Actual Temperature of External Sensor
Heating zone 01..12 Presence	Boolean	On/Off of presence

NAME (HEATING ZONE 1..12)	TYPE	DESCRIPTION
Heating zone 01..12 Heating system	Integer	The heating system used (0 = FBH standard, 1 = FHB low energy, 2 = radiator, 3 = convector passive, 4 = convector active)
Heating zone 01..12 Correction of actual value recording	Float	Correction of actual value (-2.0 .. +2.0)
Heating zone 01..12 Name	String	Name of heating zone
Heating zone 01..12 Party switching (hours)	Integer	How many hours the party switching should run
Heating zone 01..12 Party time remaining (min)	Integer	How long the party mode is still active
Heating zone 01..12 Weekend program	Integer	Which time program is active at the weekend
Heating zone 01..12 Program weekdays	Integer	Which time program is active during the week
Heating zone 01..12 Setpoint temperature max	Float	Setting range setpoint temperature (maximum)
Heating zone 01..12 Set temperature Min	Float	Adjustment range set temperature (minimum)
Heating zone 01..12 Status	Boolean	Status display (0 = OK, 1 = Error)

NAME (HEATING ZONE 1..12)	TYPE	DESCRIPTION
Heating circuit 1..12 Active	Boolean	Whether the heating circuit is active (0 = No, 1 = Yes)
Heating circuit 1..12 Actuator	Boolean	On/Off of actuator
Heating circuit 1..12 actuator percent	Integer	Percentage activity of the actuator (Only with SW STM version 2.02 or higher)
Heating Circuit 1..12 Status	Integer	Status (0 = Off, 1 = On, 2 = Error)
Heating circuit 1..12 assigned heating zone	Integer	Which heating zone is connected to the heating circuit

PROFILE:

NAME	TYPE
MH.AntifreezeTemp	Integer
MH.TemperatureUnit	Integer
MH.SummerWinter	Boolean
MH.ChangeOver	Boolean
MH.Mode	Integer
MH.EcoDiff	Float

NAME	TYPE
MH.VacationState	Integer
MH.PumpTime	Integer
MH.RelayTime	Integer
MH.EmergencyTime	Integer
MH.PWMCycle	Integer
MH.PWMPercent	Integer
MH.HeatAreaMode	Integer
MH.HeatAreaProgram	Integer
MH.HeatAreaParty	Integer
MH.HeatAreaPartyRemainingTime	Integer
MH.HeatAreaState	Boolean
MH.HeatAreaRPMMotor	Integer
MH.HeatingSystem	Integer
MH.HeatAreaTActualTemp	Float
MH.HeatAreaTTarget	Float
MH.HeatAreaTHeatCool	Float
MH.HeatAreaOffset	Float
MH.HeatAreaBlockHC	Integer
MH.HeatAreaHeatCTRLState	Integer
MH.HeatAreaNo	Integer
MH.HeatCtrlActorPercent	Integer

VISUALIZATION

Via the Visualization or in the mobile apps, values are displayed and can be changed if possible.
Due to the amount of variables we recommend to disable the display of the whole Alpha 2 in the admin console and display all desired variables via links in a separate category.

MA2_RequestStatus

Require: Symcon >= 4.0

```
bool MA2_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Gets the values stored in the Alpha2 with the InstanceID **InstanceID** and sets the associated variables.

EXAMPLE

```
MA2_RequestStatus(12345);
```

MA2_WriteValue

Require: Symcon >= 4.0

```
bool MA2_WriteValue (int $InstanceID, string $Ident, mixed $Value)
```

PARAMETERS

InstanceID	ID of the Instance
Ident	Ident of the variable
Value	Value of the variable

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Writes the Value **Value** into the variable with the Ident **Ident** in the Möhlenhoff Alpha 2 with the InstanceID **InstanceID**.

EXAMPLE

```
MA2_WriteValue(12345 , "HEATAREA1_T_TARGET", 23.3);
```

MQTT

MQTT is an open message protocol for machine-to-machine communication (M2M).
In this context, IP-Symcon acts both as a server (broker) and as a client.



The following devices are supported by IP-Symcon:
Supported Devices

INTEGRATION IN IP-SYMCON

IP-Symcon both clients and servers (brokers) are available for using MQTT.
Configure a MQTT-Server
Configure a MQTT-Client

Device list

SUPPORTED GATEWAYS/COMPONENTS

IP-Symcon supports all MQTT client devices.

SUPPORTED FEATURES

SINCE IP-SYMCON 5.1

MQTT 3.1/3.1.1 compatible
Clients can connect/disconnect
Clients can send messages on topics (QoS 0-2)
Clients can subscribe to topics (including filters +/#) (These are always sent with QoS 0)
Clients can authenticate with username/password

SINCE IP-SYMCON 5.3

Retain (persistent since 5.5)
Session Management (See timeout and queue limit settings in the server)
Last Will/Testament
Publish of a client is only evaluated in IP-Symcon and not forwarded to the other clients

SINCE IP-SYMCON 5.5

Persistent retain
MQTT clients
TLS (Server Sockets)
Keep Alive (+ Last Will send on timeout)

CURRENTLY MISSING FUNCTIONS

The following functions are missing but planned
QoS 1-2 when sending

MQTT Client

MQTT clients subscribe to an MQTT server and can receive and display values from it.

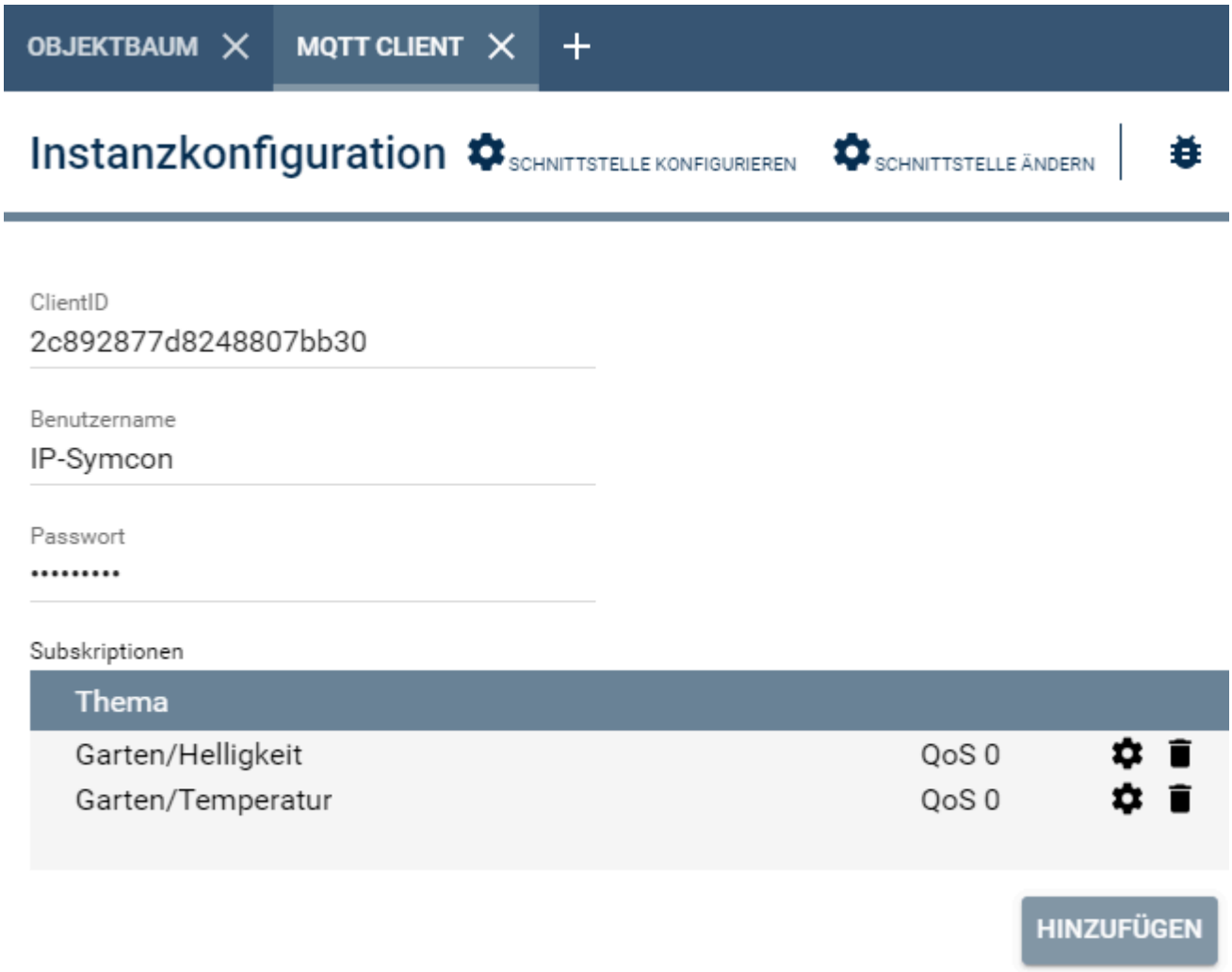
SETUP VIDEO-TUTORIAL

INTEGRATION IN IP-SYMCON

In order to enable easy and convenient integration, it is recommended to use the Configurator for the MQTT client. This can be added in the object tree via "+"->"Instance" via the "MQTT Configurator" quick filter.
The desired subscriptions can be created in the instance configuration of the associated MQTT client (gateway). A "#" is the standard and all data of the MQTT server is subscribed to.
In the instance configuration of the associated client socket the host address and the port can be defined and the connection activated. The port must be identical to the port of the MQTT server.
After setting up the MQTT client configurator, the next step is to add and create device instances.

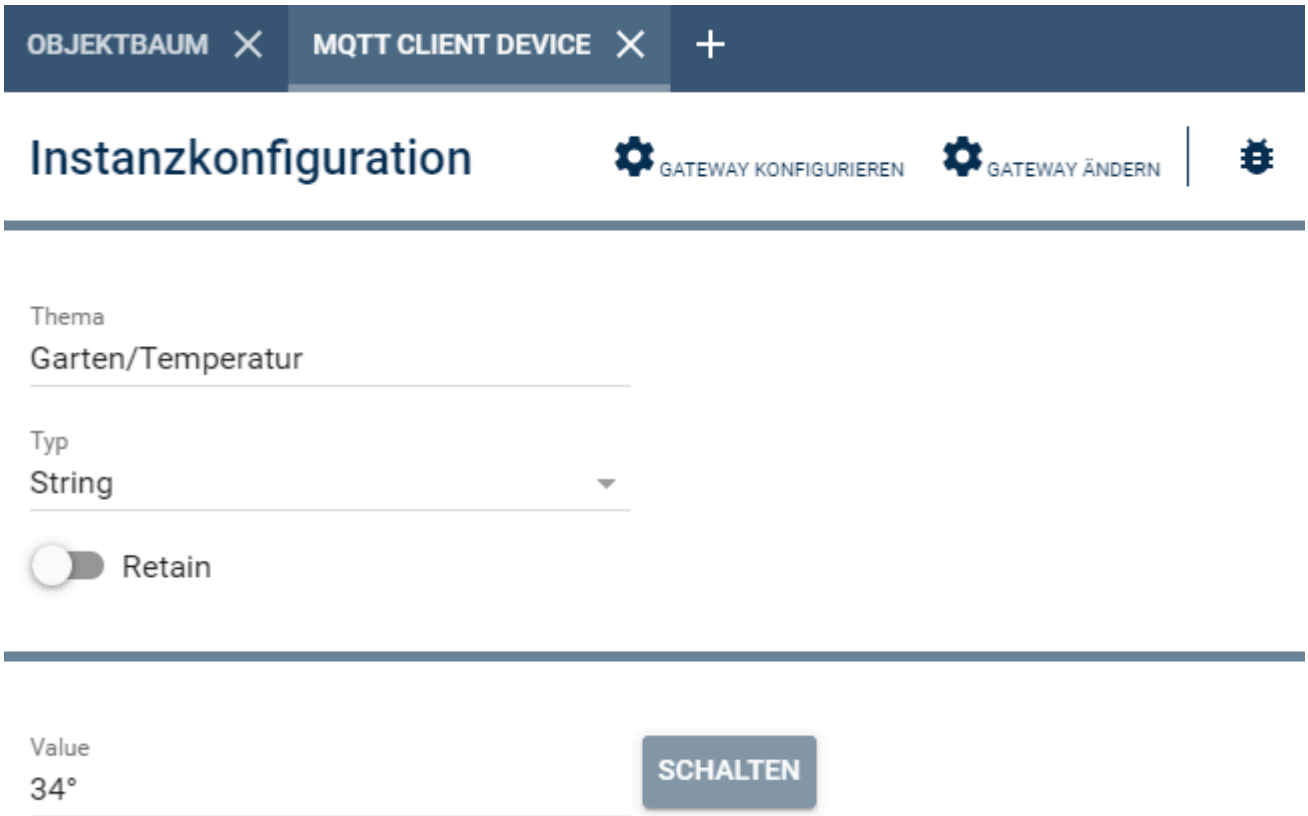
ADDING DEVICES

The client receives the values of the subscribed topics of the MQTT server. The configurator can read this data and displays all the topics that have reported.
"Refresh" updates the display of all received topics and their last received user data.
The selected instance can be created via "Create".
The MQTT client has a preconfigured ClientID. The username and password must match those of the server.



If the "#" is entered as a subscription, all topics of an MQTT server are received.

Messages can then be sent to the MQTT server via the configurator or manually created devices. This then forwards it to all other clients.



EXAMPLE

Script example for publishing a value on a topic.
The topic is defined by the MQTT device instance.

```
//Publish of "AGreatValueForPublishing" to the variable with ID 12345
RequestAction(12345, "A great value for publishing");
```

MQTT-Server

The MQTT-Server in IP-Symcon acts as a so-called MQTT-Broker. Clients can register with this and get their data from this.

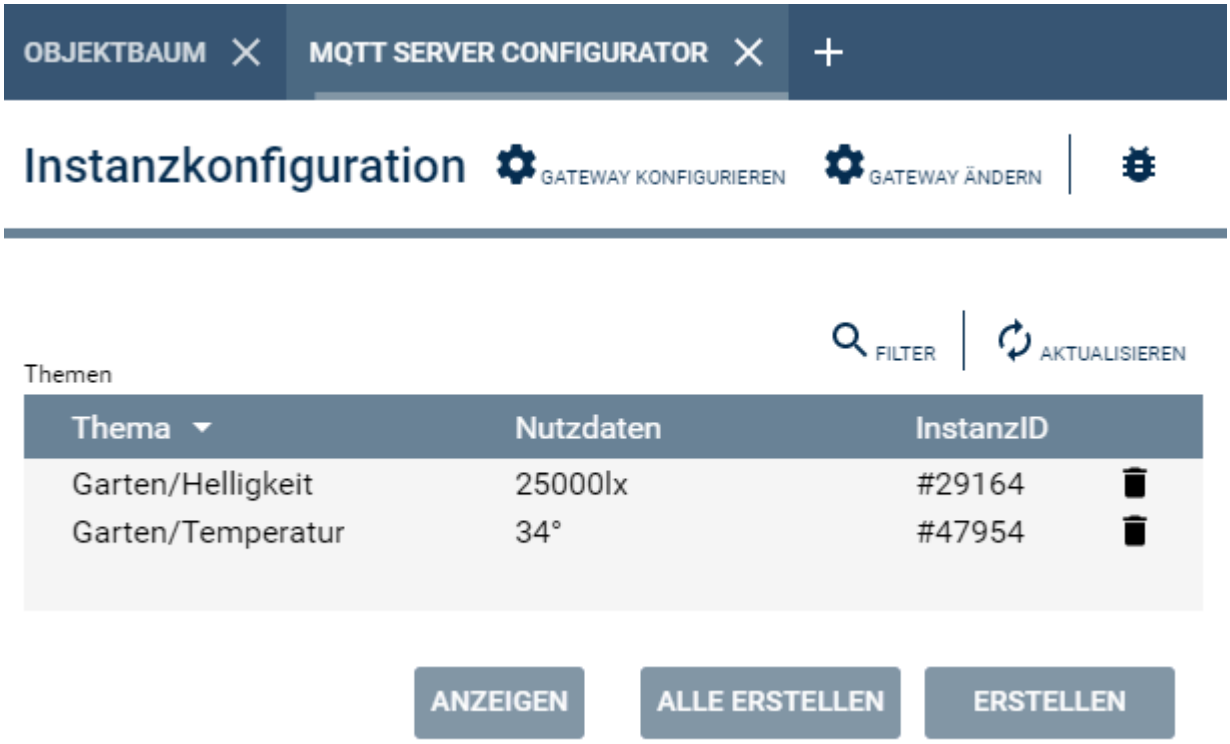
SETUP VIDEO-TUTORIAL

INTEGRATION IN IP-SYMCON

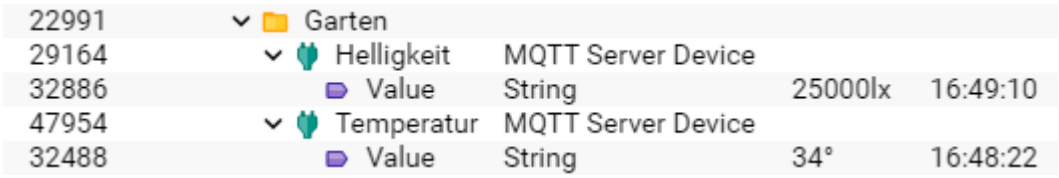
In order to enable easy and convenient integration, it is recommended to use the Configurator for MQTT-Servers. This can be added in the object tree via "+"->"Instance" via the "MQTT Server" quick filter.
IP-Symcon acts as an MQTT-Server (broker) and requires an open server socket interface for this. In the instance configuration of the associated server socket the port can be defined and the connection activated.
Furthermore, a user name and can be specified in the instance configuration of the MQTT-Server. The MQTT clients must then send to the configured port.
After setting up the MQTT configurator, the next step is to add and create device instances.

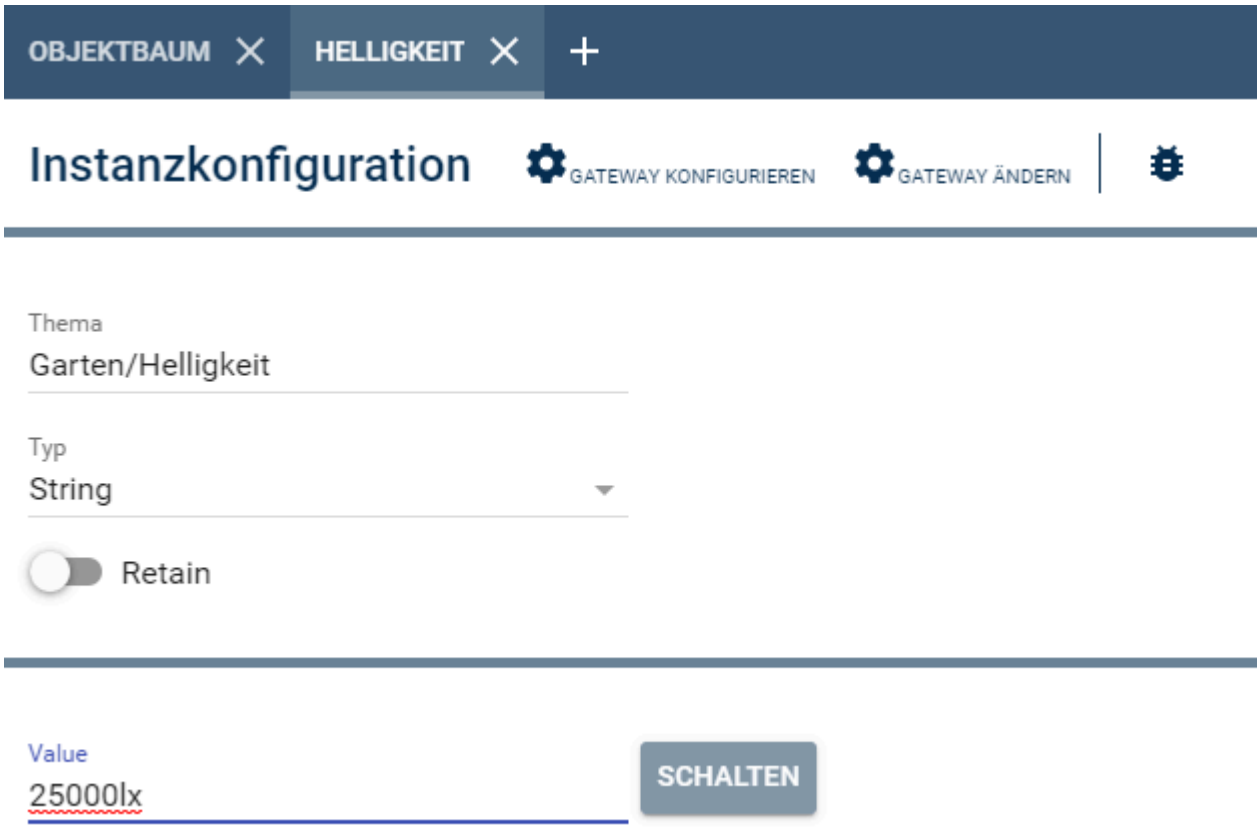
ADDING DEVICES

The server (broker) receives the values and properties of the MQTT devices. The configurator can read this data and displays all devices that have reported.
If a device is not to be displayed, it must be ensured that this device has sent a message to the server.
"Refresh" updates the display of all received topics and their last received user data.
The selected instance can be created via "Create".



The configurator creates the path of the topic with the appropriate categories and the user data variable of the "String" data type. If a different data type is required, this can be changed in the instance.





Devices that can only receive ("subscribe") and not send ("publish") cannot be created using the configurator, as it never receives anything from the devices. For these devices, an MQTT-Device must be created, a topic entered and a data type specified. (See also: Create instance)

EXAMPLE

Script example for publishing a value on a topic.
The topic is defined by the MQTT device instance.

```
//Publish of "AGreatValueForPublishing" to the variable with ID 12345
RequestAction(12345, "A great value for publishing");
```

NEA Smart

Require: Symcon >= 4.0



This module is only compatible with NEA Smart 1.0
and is not compatible with NEA Smart 2.0 or newer

The module is used for receiving and switching NEA Smart data.

SCOPE OF FUNCTIONS

Setting variables via script command or Visualization.
Reading out the data of the NEA Smart

SOFTWARE INSTALLATION

Install the REHAU Nea Smart module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add instance" the 'NEA Smart' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
IP address	IP address of NEA Smart
Interval	Adjustable interval in seconds in which the data of the NEA Smart should be queried. (Default: 0)
Heating zone 1..12	Activates the readout of the respective heating zone. (The heating circuits are always all read out)
"Read out device"	Reads out the data manually. Must also be done once at the beginning.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

The variables are created automatically.

NAME (BASE)	TYPE	DESCRIPTION
Base Setback Difference Temperature	Float	Serves as a "category" in which all monitored locations, as well as the time stamp and longitude/latitude are located. Created per device.
Base Setback Input	Boolean	
Basic Automatic Time Change	Boolean	On/Off of automatic time change
Basic Operating Mode Heating/Cooling (CO input)	Boolean	Controls the operating mode (On = cooling, Off = heating)
Basic Frost Protection	Boolean	On/Off of Frost Protection
Basic Frost Protection Temperature	Integer	Temperature of Frost Protection
Base HW Version	String	Hardware Version of the Base
Base ID	String	Base ID
Base Cooling Mode	Boolean	Cooling Mode On/Off
Base Rank in System Network	Integer	Edge of Base (0 = Standalone, 1 = Master, 2 = Slave)
Base Smartstart on/off	Boolean	On/off of Smartstart
Base SW ETH Version	String	Software Version Ethernet
Basic SW STM Version	String	Software Version STM
Basic Dewpoint Sensor	Boolean	On/Off of Dewpoint Sensor
Basic Temperature Limiter	Boolean	On/Off of Temperature Limiter
Basic Temperature Unit	Integer	Setting whether Fahrenheit or Celsius (0 = °C, 1 = °F)
Basic Vacation Status	Integer	Vacation Status (0 = off, 1 = scheduled, 2 = active)
Base Vacation Temperature Heating	Float	Temperature during vacation

NAME (HEATING ZONE 1..12)	TYPE	DESCRIPTION
Heating zone 01..12 actual temperature	Float	Current actual temperature
heating zone 01..12 set temperature	float	set temperature
Heating zone 01..12 Operating mode	Integer	Operating mode of heating zone (0 = Auto, 1 = Day, 2 = Night)
Heating Zone 01..12 Actual Temperature Ext. Sensor	Float	Current Actual Temperature of External Sensor
Heating zone 01..12 Presence	Boolean	On/Off of presence

NAME (HEATING ZONE 1..12)	TYPE	DESCRIPTION
Heating zone 01..12 Heating system	Integer	The heating system used (0 = FBH standard, 1 = FHB low energy, 2 = radiator, 3 = convector passive, 4 = convector active)
Heating zone 01..12 Correction of actual value recording	Float	Correction of actual value (-2.0 .. +2.0)
Heating zone 01..12 Name	String	Name of heating zone
Heating zone 01..12 Party switching (hours)	Integer	How many hours the party switching should run
Heating zone 01..12 Party time remaining (min)	Integer	How long the party mode is still active
Heating zone 01..12 Weekend program	Integer	Which time program is active at the weekend
Heating zone 01..12 Program weekdays	Integer	Which time program is active during the week
Heating zone 01..12 Setpoint temperature max	Float	Setting range setpoint temperature (maximum)
Heating zone 01..12 Set temperature Min	Float	Adjustment range set temperature (minimum)
Heating zone 01..12 Status	Boolean	Status display (0 = OK, 1 = Error)

NAME (HEATING ZONE 1..12)	TYPE	DESCRIPTION
Heating circuit 1..12 Active	Boolean	Whether the heating circuit is active (0 = No, 1 = Yes)
Heating circuit 1..12 Actuator	Boolean	On/Off of actuator
Heating circuit 1..12 actuator percent	Integer	Percentage activity of the actuator (Only with SW STM version 2.02 or higher)
Heating Circuit 1..12 Status	Integer	Status (0 = Off, 1 = On, 2 = Error)
Heating circuit 1..12 assigned heating zone	Integer	Which heating zone is connected to the heating circuit

PROFILE:

NAME	TYPE
NEAS.AntifreezeTemp	Integer
NEAS.TemperatureUnit	Integer
NEAS.SummerWinter	Boolean
NEAS.ChangeOver	Boolean
NEAS.Mode	Integer
NEAS.EcoDiff	Float

NAME	TYPE
NEAS.VacationState	Integer
NEAS.PumpTime	Integer
NEAS.RelayTime	Integer
NEAS.EmergencyTime	Integer
NEAS.PWMCycle	Integer
NEAS.PWMPercent	Integer
NEAS.HeatAreaMode	Integer
NEAS.HeatAreaProgram	Integer
NEAS.HeatAreaParty	Integer
NEAS.HeatAreaPartyRemainingTime	Integer
NEAS.HeatAreaState	Boolean
NEAS.HeatAreaRPMMotor	Integer
NEAS.HeatingSystem	Integer
NEAS.HeatAreaTActualTemp	Float
NEAS.HeatAreaTTarget	Float
NEAS.HeatAreaTHeatCool	Float
NEAS.HeatAreaOffset	Float
NEAS.HeatAreaBlockHC	Integer
NEAS.HeatAreaHeatCTRLState	Integer
NEAS.HeatAreaNo	Integer
NEAS.HeatCtrlActorPercent	Integer

VISUALIZATION

Through the Visualization or in the mobile apps, values are displayed and can be changed if possible.
Due to the amount of variables, we recommend to disable the display of the entire NEA Smart in the management console and to display all desired variables via links in a separate category.

NEAS_RequestStatus

Require: Symcon >= 4.0

```
bool NEAS_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

Gets the values stored in the NEA Smart with the InstanceID **InstanceID** and sets the associated variables.

EXAMPLE

```
NEAS_RequestStatus(12345);
```

NEAS_WriteValue

Require: Symcon >= 4.0

```
bool NEAS_WriteValue (int $InstanceID, string $Ident, mixed $Value)
```

PARAMETERS

InstanceID	ID of the Instance
Ident	Ident of the variable
Value	Value of the variable

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Writes the Value **Value** to the variable with the ident **Ident** in the NEA Smart with the InstanceID **InstanceID**.

EXAMPLE

```
NEAS_WriteValue(12345 , "HEATAREA1_T_TARGET", 23.3);
```

OCPP

Via the OCPP protocol, wallboxes can be conveniently monitored and switched via Symcon.

SCOPE OF FUNCTIONS

Display of consumption of individual charging points
Display of whether a transaction is currently in progress for charging points
Enable charging process via Id Tag, typically RFID card

SOFTWARE INSTALLATION

Install the 'OCPP' module via the Module Store.
Under 'Add instance' the 'OCPP Configurator' module can be found using the quick filter.
Further information on adding instances in the Instance documentation
On the configuration page of the splitter, the information for setting up a charging point can be taken.
Depending on the wallbox, the parameters must be specified individually or as a complete URL.
Once the connection is established, the charging points are displayed in the configurator and can be created.

CONFIGURATION PAGE (CHARGING POINT)

NAME	DESCRIPTION
Charge Point Identity	Identifier of the charging station
Start transaction...	Defines when the charging process should be started automatically: "... automatically on connection": The charging process is started directly when a vehicle is connected. "... when an Id Tag is validated - Allow all Id Tags": The charging process is started when any Id Tag is scanned. The tag itself is not further validated. "... when an Id Tag is validated - Allow only Central Id Tag list": The charging process is started when an Id Tag from the central Id Tag list is scanned. This list can be configured in the splitter. "... when an Id Tag is validated - Allow only Local Id Tag list": The charging process is started when an Id Tag from the local Id Tag list is scanned. This list can be configured directly below this selection. "... when an Id Tag is validated - Allow both Id Tag lists": The charging process is started when an Id Tag from the local or central Id Tag list is scanned. The local list can be configured directly below this selection and the central list in the splitter.
Local list of valid Id Tags (e.g. RFID cards)	The local list for Id Tags at which the charging process should be started. In addition to the Id Tag, first name, last name, e-mail and general information about the Id Tag can be specified. These help to assign the tag. (Only displayed if "Start transaction..." is set to "... when an Id Tag is validated - Allow only Local Id Tag list" or "... when an Id Tag is validated - Allow both Id Tag lists".)

CONFIGURATION PAGE (SPLITTER)

NAME	DESCRIPTION
Central list of valid Id Tags (e.g. RFID cards)	The central list for Id Tags which can be used to automatically start the charging process. In addition to the Id Tag, first name, last name, e-mail and general information about the Id Tag can be specified. These help to assign the tag.

Notes

The exact Id Tags of an RFID card can be determined by scanning it once at the charging station. The Id Tag can then be taken from the status variable "Last Id Tag".
For the Pulsar Plus, all schedules must be removed before activating OCPP. Control via OCPP cannot overwrite the configured time schedules, as these always have priority.

STATUS VARIABLES: CHARGING POINT

GENERAL STATUS VARIABLES

These status variables exist once, independent of the connectors of the charging point.

NAME	TYPE	DESCRIPTION
Vendor	string	Name of the manufacturer
Model	string	Model of the charging point
Serial Number	string	Serial number of the charging point
Last Id Tag	string	Last read Id Tag

STATUS VARIABLES PER CONNECTOR

A charging point can manage multiple connectors. In this case, each connector may create its own status variables. These are created at runtime based on the interaction with the connector. Therefore, not all variables must exist.

NAME	TYPE	DESCRIPTION
Meter Value	float	The meter reading of the connector
Available	boolean	The availability of the connector - If the connector has been set to unavailable, charging is not possible there
Status	string	Current status of the connector
ErrorCode	string	Current error code, "" if there is currently no error
Transaction	boolean	If true, a transaction is active, so the connected vehicle is being charged.
Transaction Id	integer	ID of the current or last transaction
Transaction Meter Start	integer	Meter reading in Wh at the start of the current or last transaction

NAME	TYPE	DESCRIPTION
Transaction Meter End	integer	Meter reading in Wh at the end of the current or last transaction
Transaction Id Tag	string	Id Tag of the current or last transaction
Transaction Consumption	integer	Energy consumption in Wh of the current or last transaction

OCPP_RemoteStartTransaction

```
bool OCPP_RemoteStartTransaction (int $InstanceID, int $ConnectorID)
```

PARAMETERS

InstanceID	Instance ID
ConnectorID	ID of the connector

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

If the status is 'Preparing' or 'Finishing', loading can be enabled by this command.

EXAMPLE

```
OCPP_RemoteStartTransaction(12345, 1);
```

OCPP_RemoteStopCurrentTransaction

```
bool OCPP_RemoteStopCurrentTransaction (int $InstanzID, int
$ConnectorID)
```

PARAMETERS

InstanzID	Instance ID
ConnectorID	ID of the connector

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

If the state is 'Charging', this command can be used to stop the charging. The current TransactionId is taken, which is in the matching variable to the ConnectorId!

EXAMPLE

```
OCPP_RemoteStopCurrentTransaction(12345, 1);
```

OCPP_RemoteStopTransaction

```
bool OCPP_RemoteStopTransaction (int $InstanzID, int $TransactionID)
```

PARAMETERS

InstanzID	Instance ID
TransactionID	ID of the transaction

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

If the state is set to 'Charging', this command can be used to stop charging. However, not the ConnectorId, but the TransactionId, which is in the appropriate variable, must be passed!

EXAMPLE

```
OCPP_RemoteStopTransaction(12345, 4479);
```

OPC UA

Require: Symcon >= 6.4



The OPC UA module is a paid extension that can be purchased for any existing Symcon license in the Shop. Up to 3 OPC UA nodes can be created free of charge for each Symcon license.

The OPC UA (Client) connection allows the comfortable setup of OPC UA devices within Symcon. The OPC UA address space of a device can be determined and traversed via an OPC UA configurator. Any OPC UA addresses (nodes) can be created. The nodes show the current value, which is automatically monitored and updated if the device supports this. Alternatively, the value can also be queried cyclically. The value can also be changed via visualization, script and flowchart.

VIDEO TUTORIAL FOR SETUP

Follows

SUPPORTED PROTOCOL

OPC UA (Binary Encoding)

SUPPORTED TRANSPORT

TCP/IP

SUPPORTED SECURITY MODES

None
Sign
SignAndEncrypt

SUPPORTED AUTHENTICATION METHODS

Anonymous
Username/Password

SUPPORTED SECURITYPOLICIES

None
Basic256
Basic256Sha256

SUPPORTED DATA TYPES

Boolean, SByte, Byte, Int16, UInt16, Int32, UIn32, Int64, UIn64, Float, Double, String, DateTime, GUID, ByteString, XMLElement, NodeId, StatusCode, QualifiedName, LocalizedText, Variant
Array of the respective types (represented as separate variables)

SageGlass (BACnet)

Require: Symcon >= 6.0
This module enables the integration of SageGlass SIM II controllers via BACnet/IP.

FUNCTION SCOPE

Integrates the control of SageGlass zones and displays the current status including the transmitted lux values.

REQUIREMENTS

IP-Symcon Enterprise from version 6.0 with BACnet extension

SOFTWARE INSTALLATION

Via the Module Store install the 'SageGlass' module.

SETUP IN IP-SYMCON

It is recommended to have all BACnet objects automatically created on the SageGlass SIM II controller, making the individual data points available in IP-Symcon. In the second step, the following script can be used to create the SageGlass instances for the zones, which will prepare the information more nicely. It is also possible to do the linking manually by linking all properties to the correct BACnet objects. The names of the properties correspond to the descriptions of the BACnet objects and should be linked to the "Preset Value" variable.

```
<?php

// Not every zone has a sensor - Normally each facade will have a
vertical/horizontal sensor
// Fill these arrays with the sensor -> zone information
// e.g. Sensor 1 (Vertical) is used for Zone 10, Zone 20 and Zone 25
$sensorVertical = [
1 => [10, 20, 25]
];
$sensorHorizontal = [];

$zones = 250;
for ($i = 1; $i <= $zones; $i++) {
$variableTintID = searchBACnet(2 /* Analog Value */, $i + 1 /* Zone 1 =
2 */);
$automodeStateID = searchBACnet(2 /* Analog Value */, $i + 2001 /* Zone
1 = 2002 */);
$luxLevelSetPointID = searchBACnet(2 /* Analog Value */, $i + 4001 /*
Zone 1 = 4002 */);
$statusID = searchBACnet(0 /* Analog Input */, $i + 4000 /* Zone 1 =
```

```
4001 */);

// Search for the our sensor the vertical sensor matrix
$verticalSensorID = searchSensorID($i, $sensorVertical);
if ($verticalSensorID) {
$verticalSensorID = searchBACnet(0 /* Analog input */,
$verticalSensorID + 2000 /* Sensor 1 = 2001 */);
}

// Search for the our sensor the vertical sensor matrix
$horizontalSensorID = searchSensorID($i, $sensorHorizontal);
if ($horizontalSensorID) {
$horizontalSensorID = searchBACnet(0 /* Analog input */,
$horizontalSensorID + 2000 /* Sensor 1 = 2001 */);
}

if (!@IPS_GetObjectIDByIdent("SageGlassZone" . $i)) {
if ($variableTintID && $automodeStateID && $luxLevelSetPointID &&
$verticalSensorID && $horizontalSensorID && $statusID) {
echo "Creating Zone " . $i . "..." . PHP_EOL;
$id = IPS_CreateInstance("{67CEA419-A625-703E-2BE6-BF51B3C913B9}");
IPS_SetName($id, "Zone " . $i);
IPS_SetIdent($id, "SageGlassZone" . $i);
IPS_SetProperty($id, "VariableTint", $variableTintID);
IPS_SetProperty($id, "AutomodeState", $automodeStateID);
IPS_SetProperty($id, "LuxLevelSetPoint", $luxLevelSetPointID);
IPS_SetProperty($id, "Status", $statusID);
IPS_SetProperty($id, "SensorVertical", $verticalSensorID);
IPS_SetProperty($id, "SensorHorizontal", $horizontalSensorID);
IPS_ApplyChanges($id);
}
}
}

function searchSensorID($zone, $sensorMatrix) {
foreach($sensorMatrix as $sensor => $matrix) {
foreach($matrix as $sensorZone) {
if ($zone == $sensorZone) {
return $sensor;
}
}
}
return 0;
}

function searchBACnet($objectType, $instanceNumber) {
$ids = IPS_GetInstanceListByModuleID("{CD5D5D10-3743-DA88-F16C-
8B65CF4103F9}");
foreach ($ids as $id) {
if ((IPS_GetProperty($id, "ObjectType") == $objectType) &&
(IPS_GetProperty($id, "InstanceNumber") == $instanceNumber)) {
return IPS_GetObjectIDByIdent("PresentValue", $id);
}
}
return 0;
}
```

Shutter Control

Require: Symcon >= 5.0
The "Shutter Control" offers the possibility to move a roller shutter/awning to a specific percentage position.
Furthermore, the position is updated when the shutter is controlled via function or manually.
In contrast to Shutter Control (legacy), the position can also be adjusted while in motion.

REQUIREMENTS

The IP-Symcon Shutter module needs a variable or an instance with a status variable that uses the ~ShutterMoveStop or ~ShutterMoveStep profile.
The travel times and start-up delay must also be entered in the module. The easiest way to do this is with a stopwatch.



By default, dS Shutter, EnOcean Shutter, KNX Shutter, LCN Shutter, xComfort Shutter and Z-Wave Shutter support these profiles. This means that when they are created, a variable with the required profile is available.

SETTINGS

OPTION	POSITION	DESCRIPTION
Type	--------	Shutter or awning
Delay	--------	Starting delay of the engine after switching
Target	--------	Actuator variable that is to be controlled
Move down (middle)	50%	Measured time from open to center.
Move down (bottom)	99%	(Shutter only) Measured time from open to down. (bars still open)
Move down (Closed)	100%	Measured time from open to end position reached. Slats closed.
Move up (Bottom)	99%	(Shutter only) Measured time from closed to bars open.
Move up (middle)	50%	Measured time from closed to mid reached.
Move Up (Open)	0%	Measured time from closed to end position reached. Fully open.

ADD AN INDIVIDUAL DEVICE

If the device does not belong to one of the supported variants, an integer variable can be created by oneself.
This must have the ~ShutterMoveStop or ~ShutterMoveStep profile assigned.
Furthermore, the following action script must be stored.

```
// Template for action script
switch($_IPS['VALUE']) {

case 0:
// Here is the open function
break;

case 1: // Only with ~ShutterMoveStep
// Here is the step-open function
break;

case 2:
// Here is the stop function
break;

case 3: // Only with ~ShutterMoveStep
// Here is the step-close function
break;

case 4:
// Here is the close function
break;
}

SetValue($_IPS['VARIABLE'], $_IPS['VALUE']);
```

SC_Move

```
bool SC_Move (int $InstanceID, int $Position)
```

PARAMETERS

InstanceID	ID of Shutter Control Instance
Position	0%-100%, 99% = Auf Spalt fahren

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutters for the device with the ID **InstanceID** to the **Position**.



Because some radio systems are unidirektonal, can not be determined whether a command succeeded. The exact positioning can not be guaranteed.

EXAMPLE

```
SC_Move(12345, 50); //Go to 50%
```

SC_MoveDown

```
bool SC_MoveDown (int $InstanceID, int $Duration)
```

PARAMETERS

InstanceID	ID of Shutter Control Instance
Duration	0 = End position, >1 = Duration in ms

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutters for the device with the ID **InstanceID** for **Duration** milliseconds downwards.



The state variable is reset regardless of the elapsed time to 100.

EXAMPLE

```
SC_MoveDown(12345, 0); //Go downwards
```

SC_MoveUp

```
bool SC_MoveUp (int $InstanceID, int $Duration)
```

PARAMETERS

InstanceID	ID of Shutter Control Instance
Duration	0 = End position, >1 = Duration in ms

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutters for the device with the ID **InstanceID** for **Duration** milliseconds upwards.



The state variable is reset regardless of the elapsed time to 0.

EXAMPLE

```
SC_MoveUp(12345, 0); //Go upwards
```

SC_Stop

```
bool SC_Stop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of Shutter Control Instance

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Stops a moving operation for the device with the ID **InstanceID**.
The state variable is reset to -1.

EXAMPLE

```
SC_Stop(12345); //Stop
```

Siemens OZW

Siemens OZW is a web server that allows remote control and remote monitoring of KNX devices via web. IP-Symcon is connected to Siemens OZW via LAN(IP).



The following devices are supported by IP-Symcon:
Supported devices

INTRODUCTION

This manual shall help to connect a **Siemens OZW 672 / Siemens OZW 772 Web-Server** to the computer as fast and uncomplicated as possible. This should create the prerequisite to cover various application areas with IP-Symcon and to visualize them. The OZW web server enables remote control and remote monitoring of plants via the web. It is available in four versions: for the connection of one, four, 16 or 250 KNX devices of the ranges Synco 700, Synco RXB/RXL, room thermostats RDG/RDF/RDU and the Synco living central apartment units QAX9.

VIDEO TUTORIAL FOR SETUP

CONNECTION

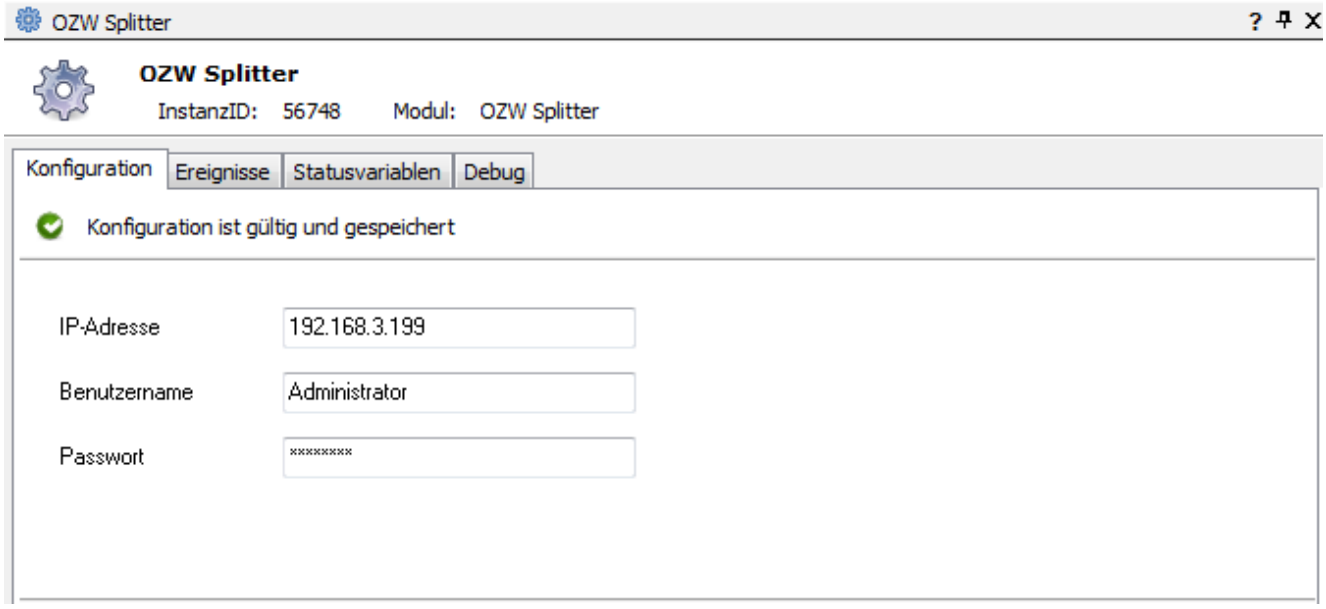
It should be correctly connected all KNX devices before connecting the device through a network cable to the network and connect it to the power supply.
It is also possible to connect to the PC via USB, provided that there is a connection to the Internet, so that the RNDIS driver can be installed automatically when connecting via USB (provided that the Microsoft online update service is enabled). A manual installation of the driver is also possible.

INSTALLATION

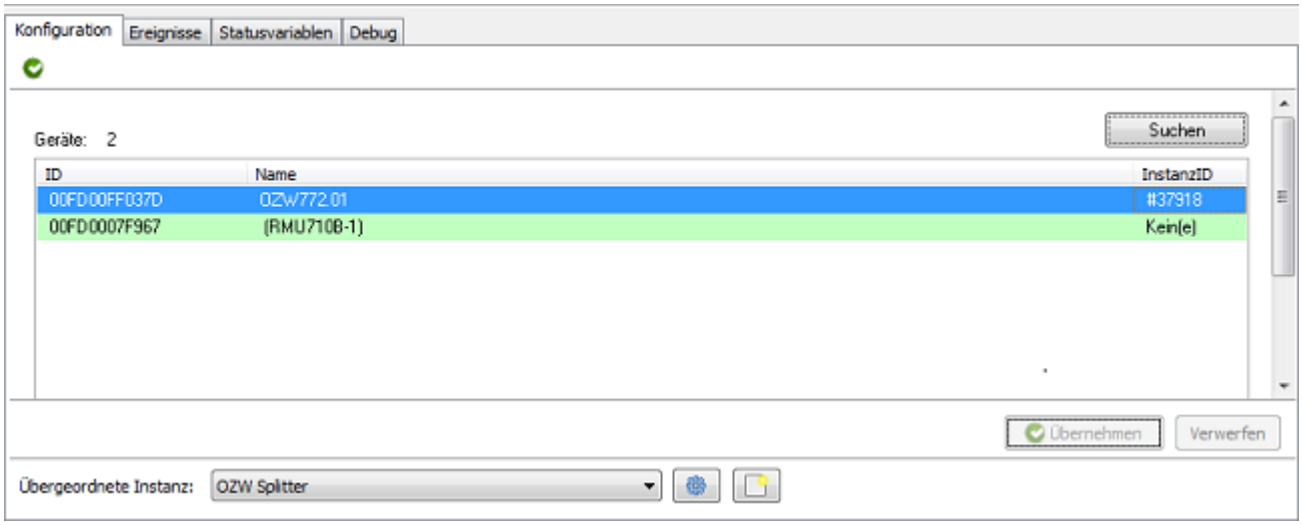
If the OZW is connected correctly and a DHCP server is used, it is now connected to the local network. *(The DHCP server automatically takes over the address assignment for the network participants.)*
The OZW now appears under the network devices in the category "Other devices".
Clicking on "OZW..." opens a window in which the IP address of the OZW is displayed in the lower area. By copying this address into any browser the Siemens web page opens.
At the first login the following applies: **"Username" = "Administrator" and "Password" = "Password"**.
The password should be changed after the first login.
The password is case sensitive. After the first login the web server language is "English", but this can be adjusted. On the left side are all connected KNX devices.
If no DHCP server is used, the IP address of the OZW is 192.168.2.10 (USB: 192.168.250.1 (not changeable)).

INTEGRATION IN IP-SYMCON

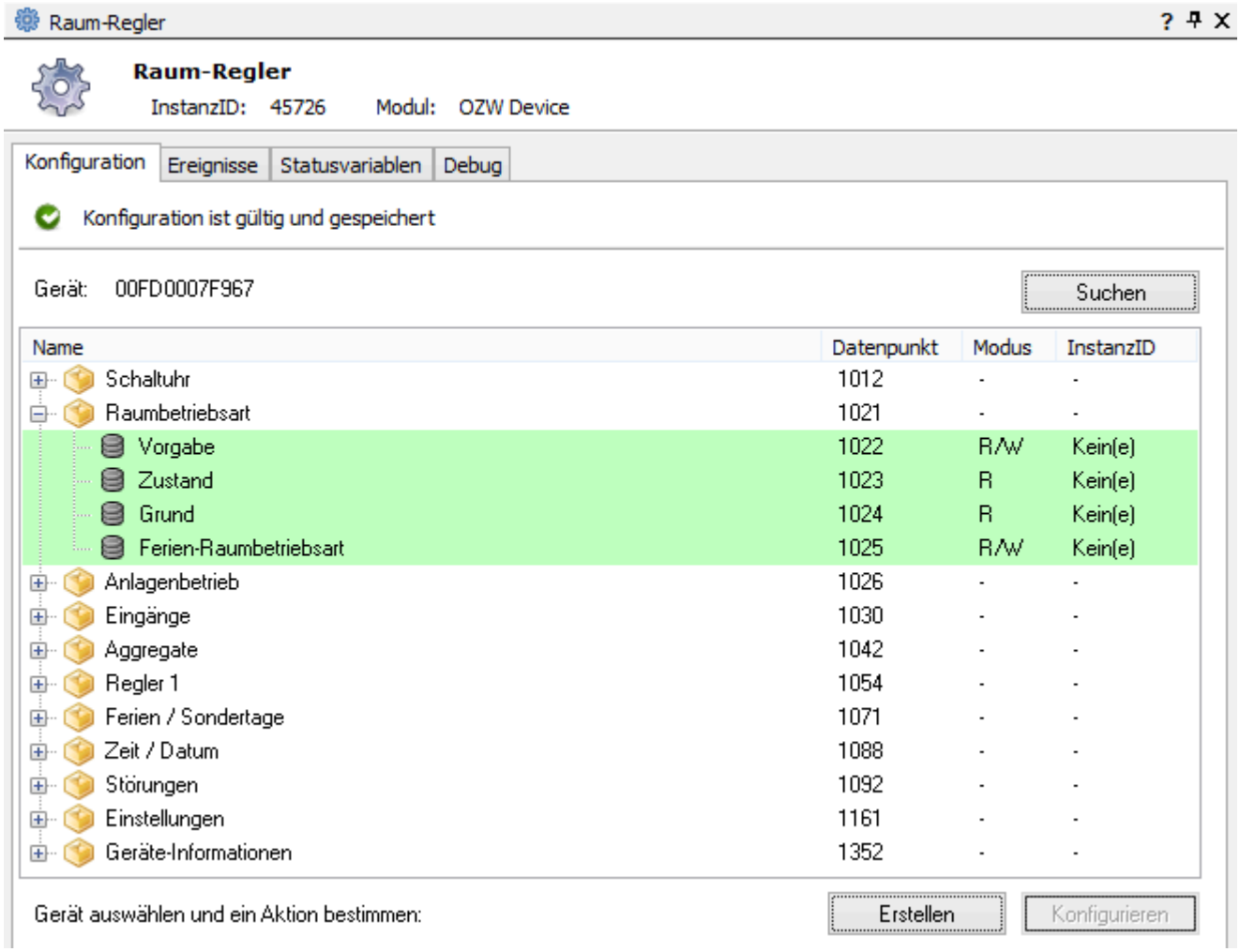
In the Management Console a "OZW Configurator" can be created via "Create Configurator". To do this, click "Next" twice and "OK" once. The following configuration page can be accessed via the cogwheel (bottom left).



The configuration of the "OZW splitter" opens. Here the default values from Siemens are already preset. If necessary, changed passwords or a different IP can be set here. With the click on "Apply" the changes are saved. In the OZW configurator you can now click on "Search". The OZW and all KNX devices connected to it then appear.



The ID on the left side is the identification number that can also be found on the back of each device. In case of a large number of installed KNX devices, this ensures easy assignment and configuration.
The ID on the right side is the instance ID assigned by IP-Symcon. This is unique and unchangeable.
Each device must be selected one after the other and set up in IP-Symcon by "Create". These devices are now available and operable within IP-Symcon.
After a click on a selected KNX device and then a click on "Configure" the device can furthermore be set exactly. It is also possible to set what should be displayed in the Visualization.



To do this, click on the "+" and select the datapoint to be displayed in the Visualization. By clicking on "Create", the data point will be set up in IP-Symcon. The view can now be closed.

CONFIGURE QUERY INTERVALS

Once the individual desired data points have been selected, they can be opened via "Configure" and a query interval can be set. This is necessary, because IP-Symcon must actively query the values from the OZW.

A query takes about 800ms. So e.g. 13 data points can be queried every 10 seconds. If the intervals are too short, the queries are processed one after the other, delayed

Device list

SUPPORTED GATEWAYS/COMPONENTS

All KNX devices that are compatible with the Siemens OZW 672 / Siemens OZW 772 web server are supported.

OZW_RequestStatus

```
bool OZW_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to query

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Requests the status of the device with InstanceID **InstanceID**.
Sets the value of the data point with the ID **InstanceID** to the specified value.

EXAMPLE

```
OZW_RequestStatus(12345);
```

OZW_WriteDataPoint

```
bool OZW_WriteDataPoint (int $InstanceID, mixed $Value)
```

PARAMETERS

InstanceID	ID of the device to be modified
Value	E.g. 23.5

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the value of the data point with the ID **InstanceID** to the specified value.



If the displayed profile has associations, the integer value of the desired association must be specified (e.g. 0) instead of the string value (e.g. off).

EXAMPLE

```
OZW_WriteDataPoint(12345, 1);
```

SNMP

Require: Symcon >= 6.0
The SMNP module reads all available OIDs of a device and represents any of them as variables in IP-Symcon. These are updated cyclically and can also be written to if required. Support for OIDLib files gives the OIDs more information/context to simplify setup.

FEATURE SCOPE

Automatic walk of all OIDs of a device
Creating individual OIDs as variables
Writing values to the OIDs
Additional information if OIDLib files are available
SNMPv1/SNMPv2/SNMPv3 incl. authentication and encryption are supported

SOFTWARE-INSTALLATION

Install the 'SMNP' module via the Module Store.

INSTANCE SETUP IN IP-SYMCON

Under 'Add Instance' the 'SMNP' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE

NAME	DESCRIPTION
host	address of host
Version	Select between SNMPv1, SNMPv2, SNMPv3
Start at	OID where the walk should start
User	Username
Community	Required for SNMPv1, SNMPv2
Enable Authentication	Enable authentication option
Authentication Password	Password for authentication
Authentication Type	Select the authentication type: MD5, SHA1, SHA224, SHA256, SHA384, SHA512
Enable encryption	Enable encryption
Encryption password	Password for encryption
Encryption type	Selection for encryption type: DES, AES128, 3DES, AES192, AES256, AES192blue, AES256blue

NAME	DESCRIPTION
Show only known OIDs from the OIDLibs	If enabled, only OIDs are shown that are available in the OIDLibs
OIDLibs	List of files which provides the OIDs with description and name. The following tool can be used to convert MIB files to OIDLib files: MIB files to OIDLibs
Update interval	Interval in seconds, in which time interval the values are updated

NAME	DESCRIPTION
Start Walk / Stop Walk	Button to start and stop the walk
OID	Display of the OID
Name	Name given by the OIDLib
Description	Description given by the OIDLib
Value	Value of the OID given at the time of the walk
Active?	Creates a variable for the selected OID
Writeable?	Enables writing for the selected OID

SNMP_UpdateValues

Require: Symcon >= 6.0

```
bool SNMP_UpdateValues (int $InstanzID)
```

PARAMETERS

InstanzID	Instance ID

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Updates the variable values which are below the instance.

EXAMPLE

```
SNMP_UpdateValues(12345);
```

Snom

Require: Symcon >= 7.0



The Snom module was created in cooperation between Symcon and Snom. The module is available as open source and the current Documentation is available on GitHub.

PLC: Siemens, Vipa, Logo

DESCRIPTION

Programmable Logic Controllers (PLC) are independent electronic assemblies for controlling machines and systems. These work independently of a PC. The functions for control and regulation are stored directly in the PLC.

The following devices are supported by IP-Symcon:
Supported devices

Due to the fact that PHP offers integers as the largest integer value, DWORDs (unsigend ints) can be read in IP-Symcon, but they are mapped to integers, which means that only the lower 31 bits can be evaluated.

CONTENT

Integration in IP-Symcon
S7 Connection Configuration
Logo Connection
Logo Configurator
Example with TIA Portal
Example with Logo Soft Comfort
Data types
Logo VM-Addresses

INTEGRATION IN IP-SYMCON

The Siemens PLCs can only be integrated via Ethernet connection.

The support of MPI and ProfiBus via LIBNODAVE is discontinued since version 4.0

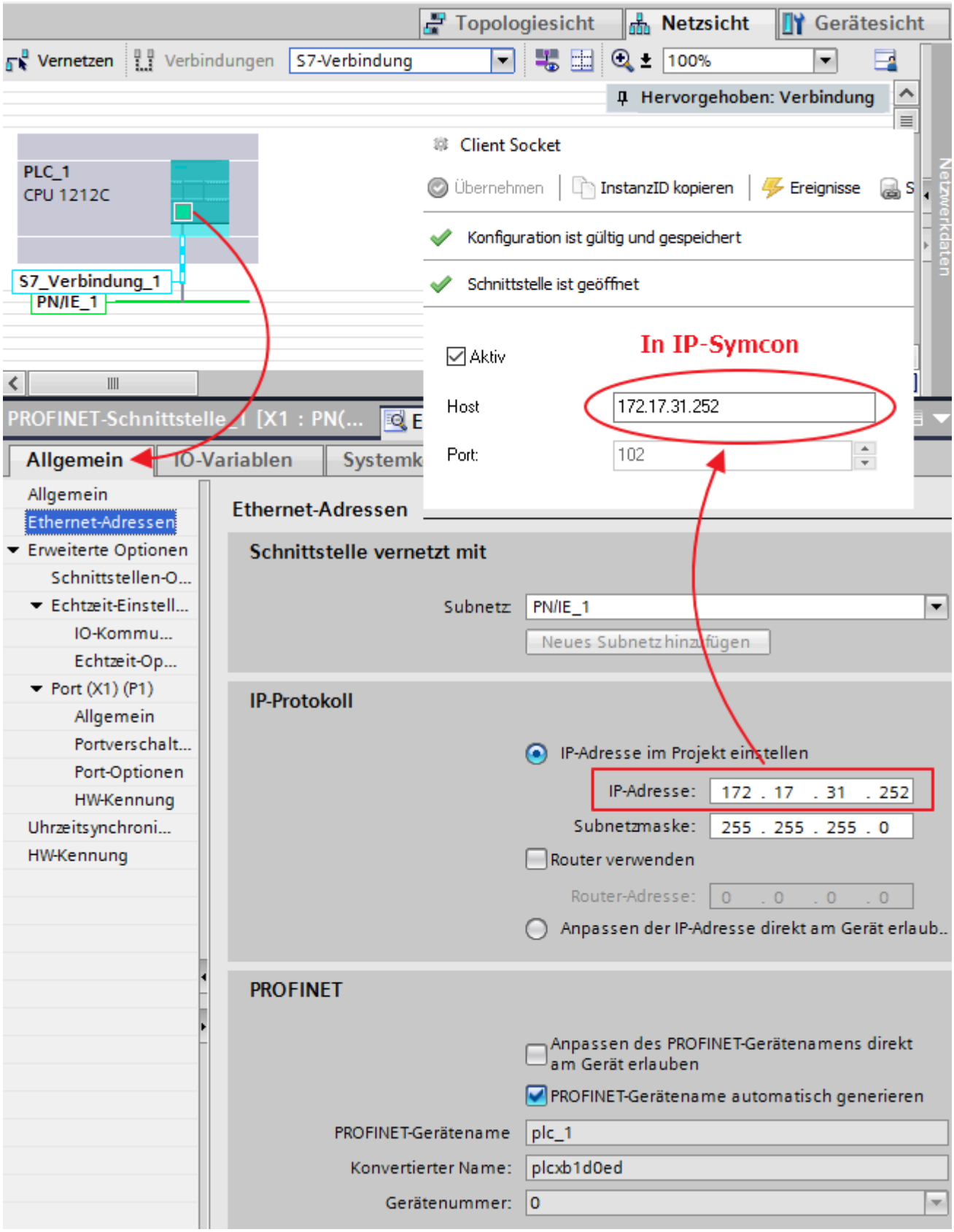
With newer S7 (1200/1500) the "Optimized block access" attribute must be deactivated. Further information can be found HERE

S7 CONNECTION CONFIGURATION

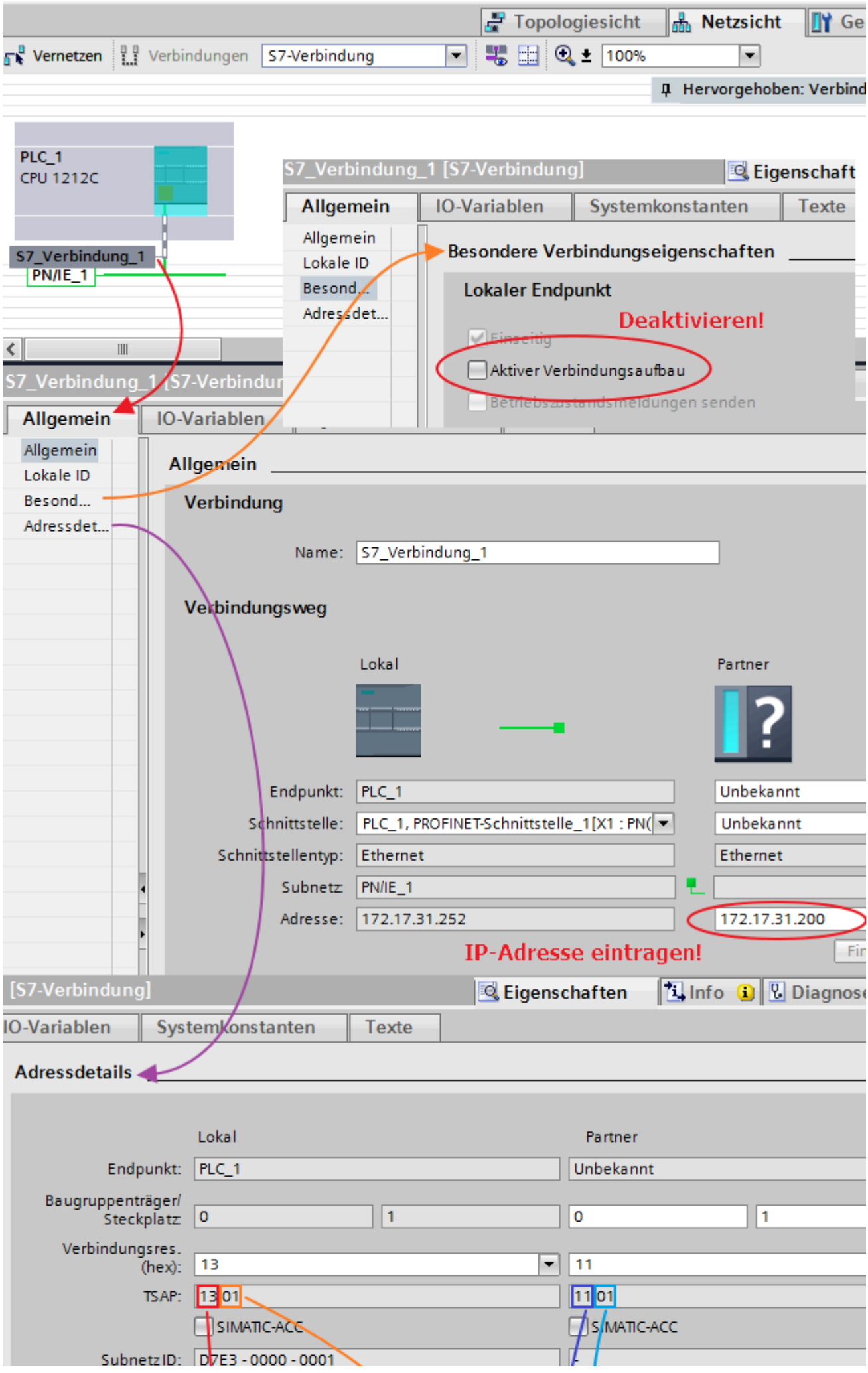
The connection is configured via "TIA Portal" software (Totally Integrated Automation) from Siemens.
The following settings must be made.

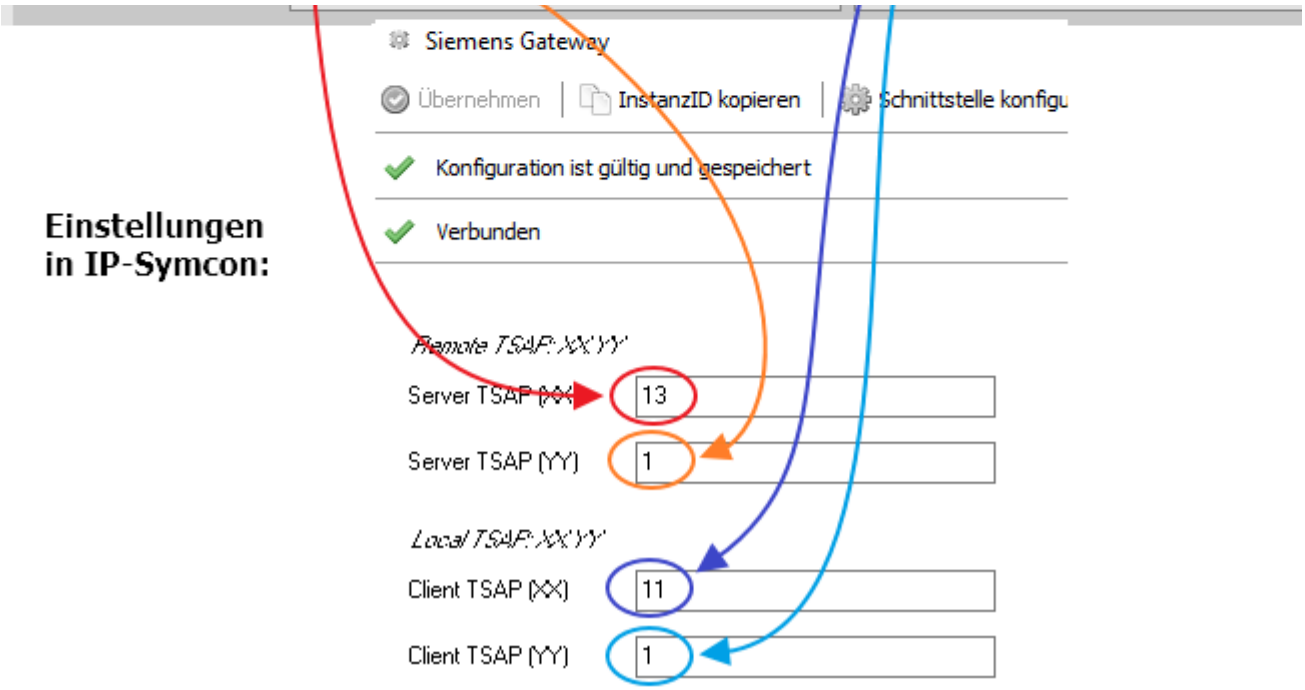
- The IP address must be entered in the PROFINET interface (see screenshot)
- PN/IE subnet must be connected
- S7 connection to PROFINET must be inserted and the following points configured (see screenshot)
- Connection data must be entered in IP-Symcon (see screenshot)

PROFINET Interface



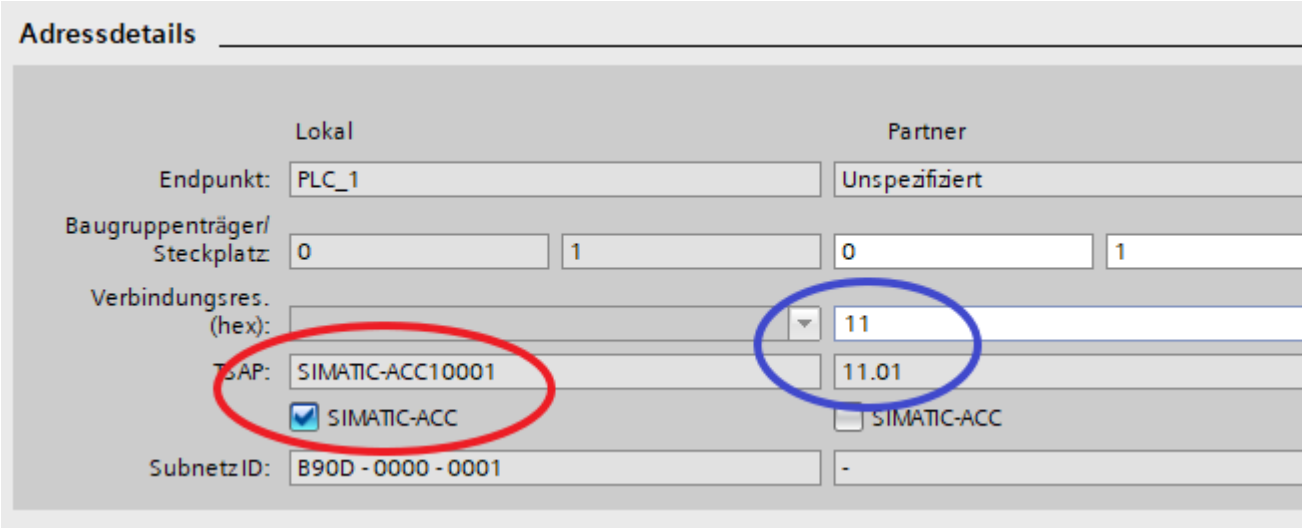
S7 Connection

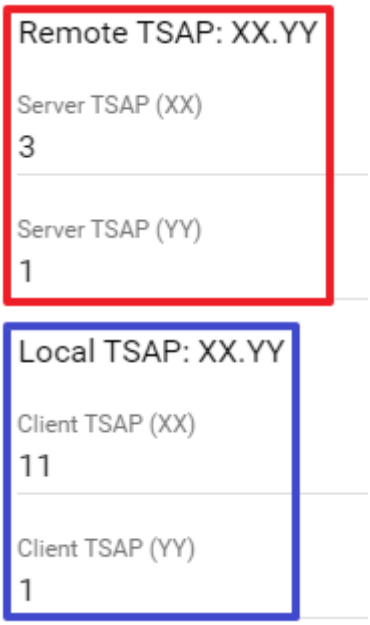

Topologiesicht
Netzsicht
Vernetzen
Verbindungen
S7-Verbindung
100%
Hervorgehoben: Verbind
PLC_1
CPU 1212C
S7_Verbindung_1
PN/IE_1
S7_Verbindung_1 [S7-Verbindung]
Eigenschaft
Allgemein
IO-Variablen
Systemkonstanten
Texte
Allgemein
Lokale ID
Besond...
Adressdet...
Besondere Verbindungseigenschaften
Lokaler Endpunkt
Deaktivieren!
Einseitig
Aktiver Verbindungsaufbau
Betriebszustandsmeldungen senden
Allgemein
Verbindung
Name:
S7_Verbindung_1
Verbindungsweg
Lokal
Partner
Endpunkt:
PLC_1
Unbekannt
Schnittstelle:
PLC_1, PROFINET-Schnittstelle_1[X1 : PN(
Unbekannt
Schnittstellentyp:
Ethernet
Ethernet
Subnetz:
PN/IE_1
Adresse:
172.17.31.252
172.17.31.200
IP-Adresse eintragen!
[S7-Verbindung]
Eigenschaften
Info
Diagnose
IO-Variablen
Systemkonstanten
Texte
Adressdetails
Lokal
Partner
Endpunkt:
PLC_1
Unbekannt
Baugruppenträger/ Steckplatz:
0
1
0
1
Verbindungsres. (hex):
13
11
TSAP:
13 01
11 01
SIMATIC-ACC
SIMATIC-ACC
Subnetz ID:
D7E3 - 0000 - 0001



S7 Connection Secure

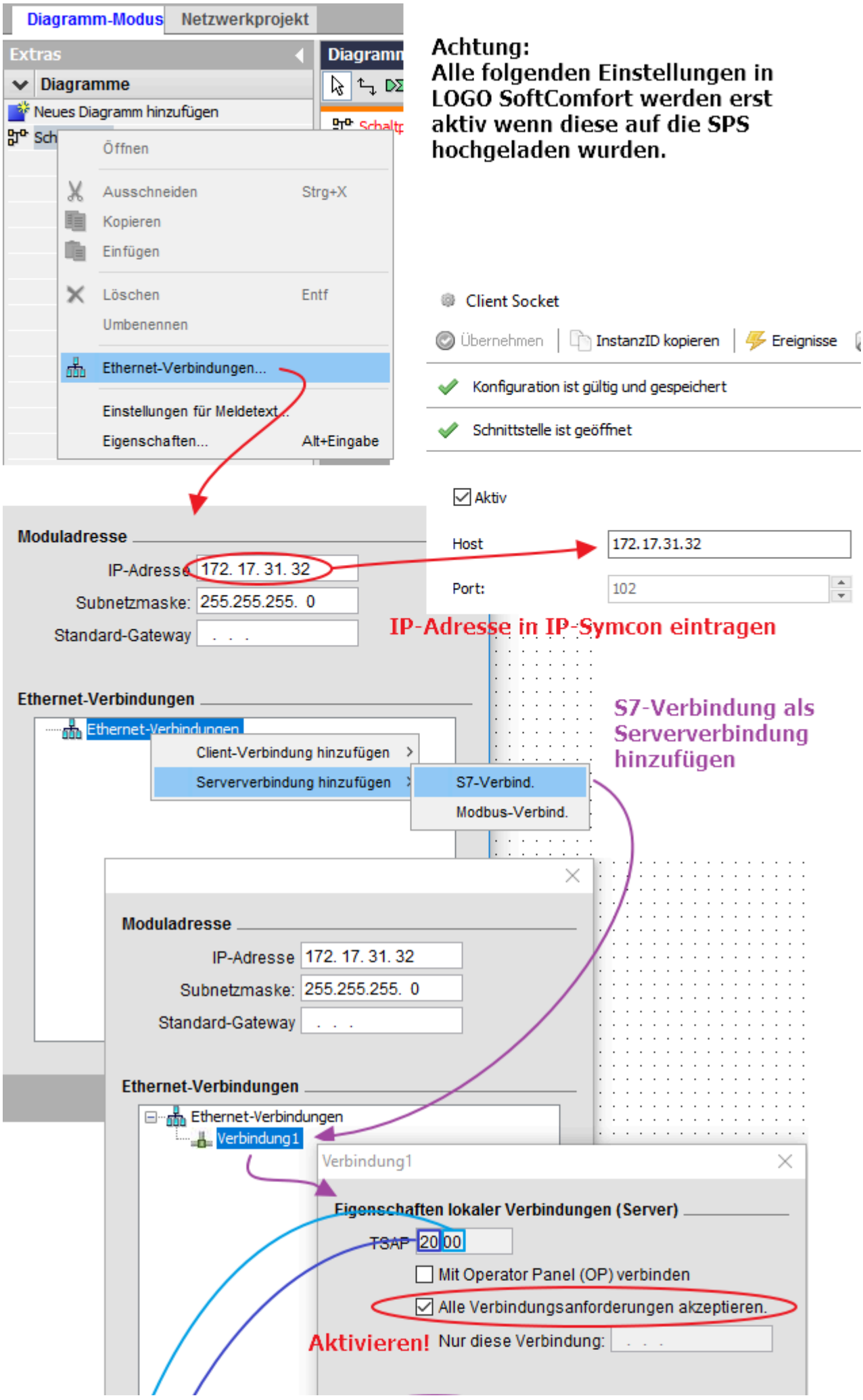
If a Siemens PLC with additional security settings is used (e.g. S7-1517f), a connection can only be used with activated SIMATIC ACC on the PLC side. This means that the "SIMATIC-ACC" option must be activated for the local TSAP in the S7 connection of the PLC. This results in the remote TSAP 03.01 in the gateway configuration in IP-Symcon.

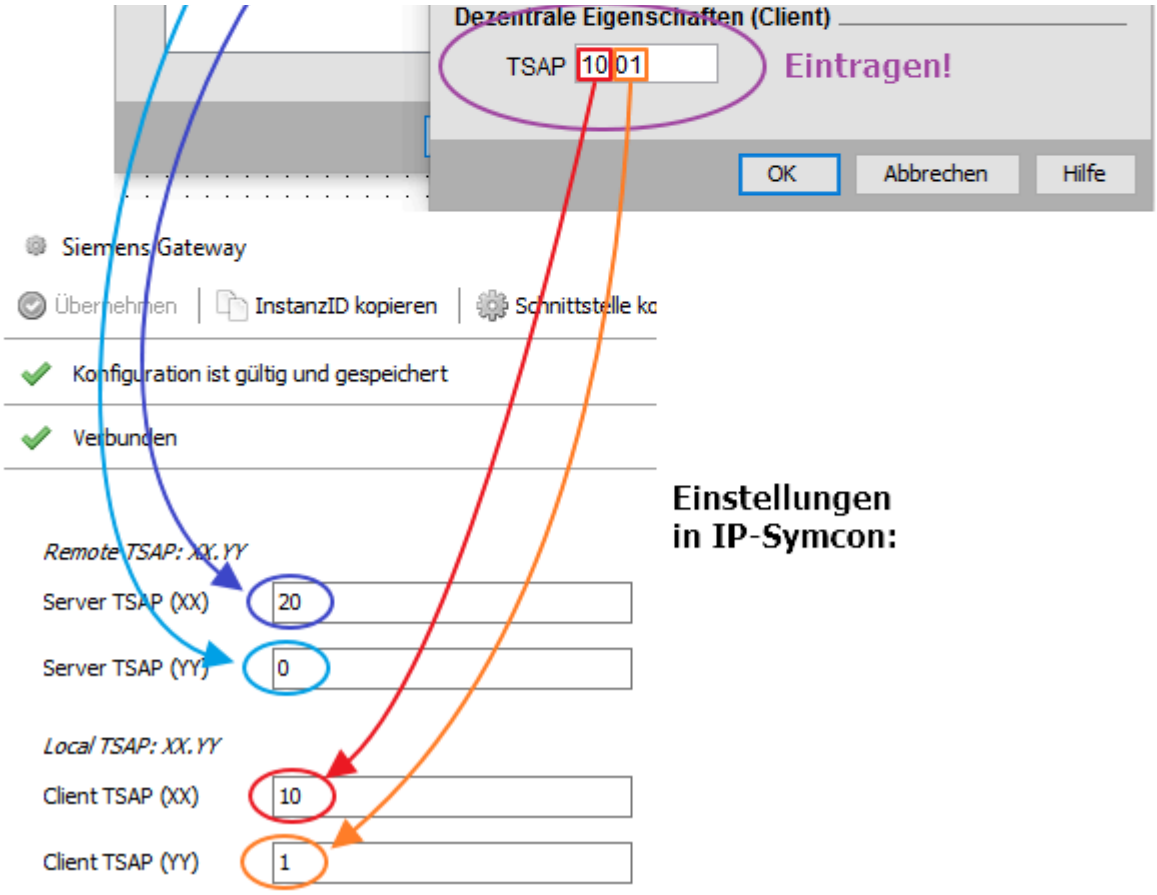




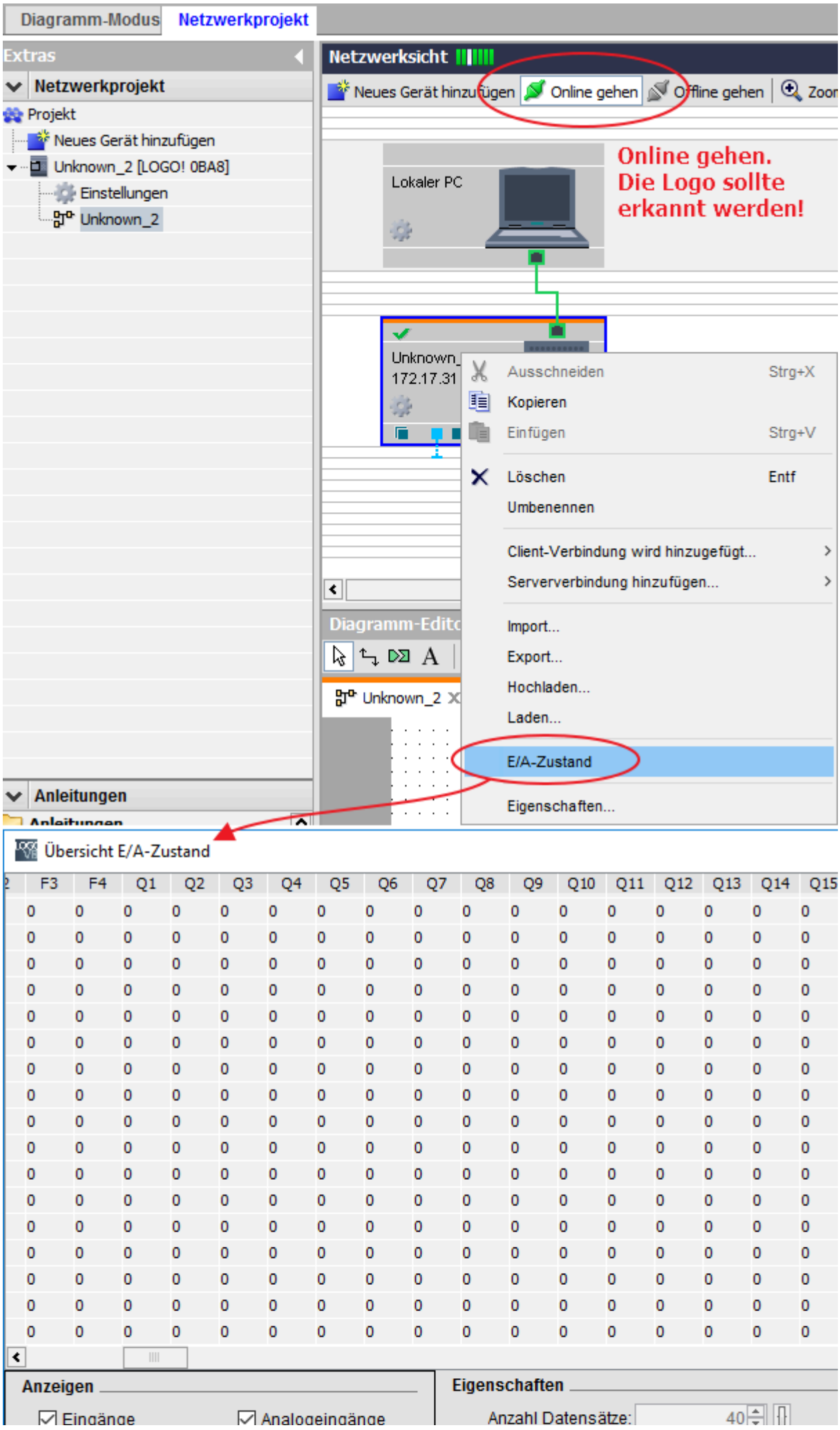
LOGO 7/8 CONFIGURATION

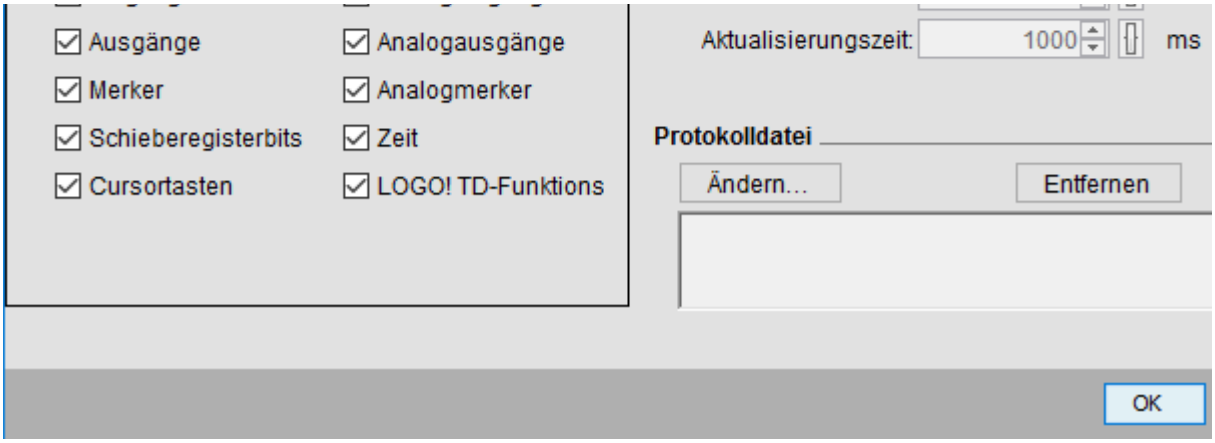
The configuration of the PLC and the gateway must be set up as follows.
Configuration via LOGO!Soft Comfort 8.1.


Diagramm-Modus
Netzwerkprojekt
Extras
Diagramme
Neues Diagramm hinzufügen
Öffnen
Ausschneiden Strg+X
Kopieren
Einfügen
Löschen Entf
Umbenennen
Ethernet-Verbindungen...
Einstellungen für Meldetext...
Eigenschaften... Alt+Eingabe
Achtung:
Alle folgenden Einstellungen in LOGO SoftComfort werden erst aktiv wenn diese auf die SPS hochgeladen wurden.
Client Socket
Übernehmen
InstanzID kopieren
Ereignisse
Konfiguration ist gültig und gespeichert
Schnittstelle ist geöffnet
Aktiv
Host 172.17.31.32
Port: 102
Moduladresse
IP-Adresse 172. 17. 31. 32
Subnetzmaske: 255.255.255. 0
Standard-Gateway
IP-Adresse in IP-Symcon eintragen
Ethernet-Verbindungen
Client-Verbindung hinzufügen
Serververbindung hinzufügen
S7-Verbind.
Modbus-Verbind.
S7-Verbindung als Serververbindung hinzufügen
Verbindung1
Eigenschaften lokaler Verbindungen (Server)
TSAP 20.00
Mit Operator Panel (OP) verbinden
Alle Verbindungsanforderungen akzeptieren.
Aktivieren!
Nur diese Verbindung:



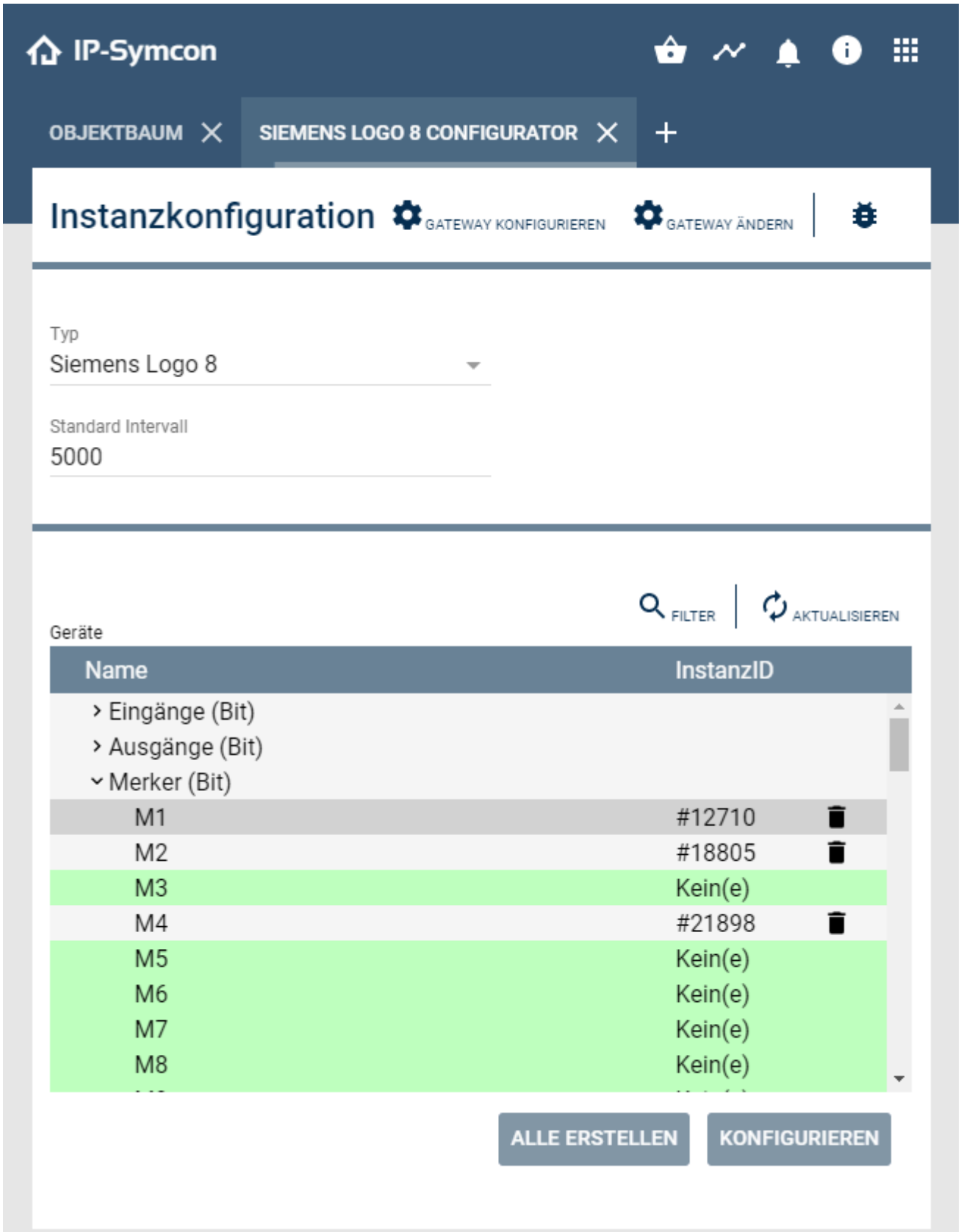
The functions can then be tested. For a simple overview of whether everything is working correctly, the SoftComfort software offers an overview of the I/O states.
This can be called as follows.


Diagramm-Modus
Netzwerkprojekt
Extras
Netzwerkprojekt
Projekt
Neues Gerät hinzufügen
Unknown_2 [LOGO! 0BA8]
Einstellungen
Unknown_2
Netzwerksicht
Neues Gerät hinzufügen
Online gehen
Offline gehen
Lokaler PC
Online gehen.
Die Logo sollte
erkannt werden!
Unknown_
172.17.31
Ausschneiden Strg+X
Kopieren
Einfügen Strg+V
Löschen Entf
Umbenennen
Client-Verbindung wird hinzugefügt...
Serververbindung hinzufügen...
Import...
Export...
Hochladen...
Laden...
E/A-Zustand
Eigenschaften...
Diagramm-Editc
Unknown_2
Anleitungen
Übersicht E/A-Zustand
F3 F4 Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15
Anzeigen
Eingänge
Analogeingänge
Eigenschaften
Anzahl Datensätze:
40

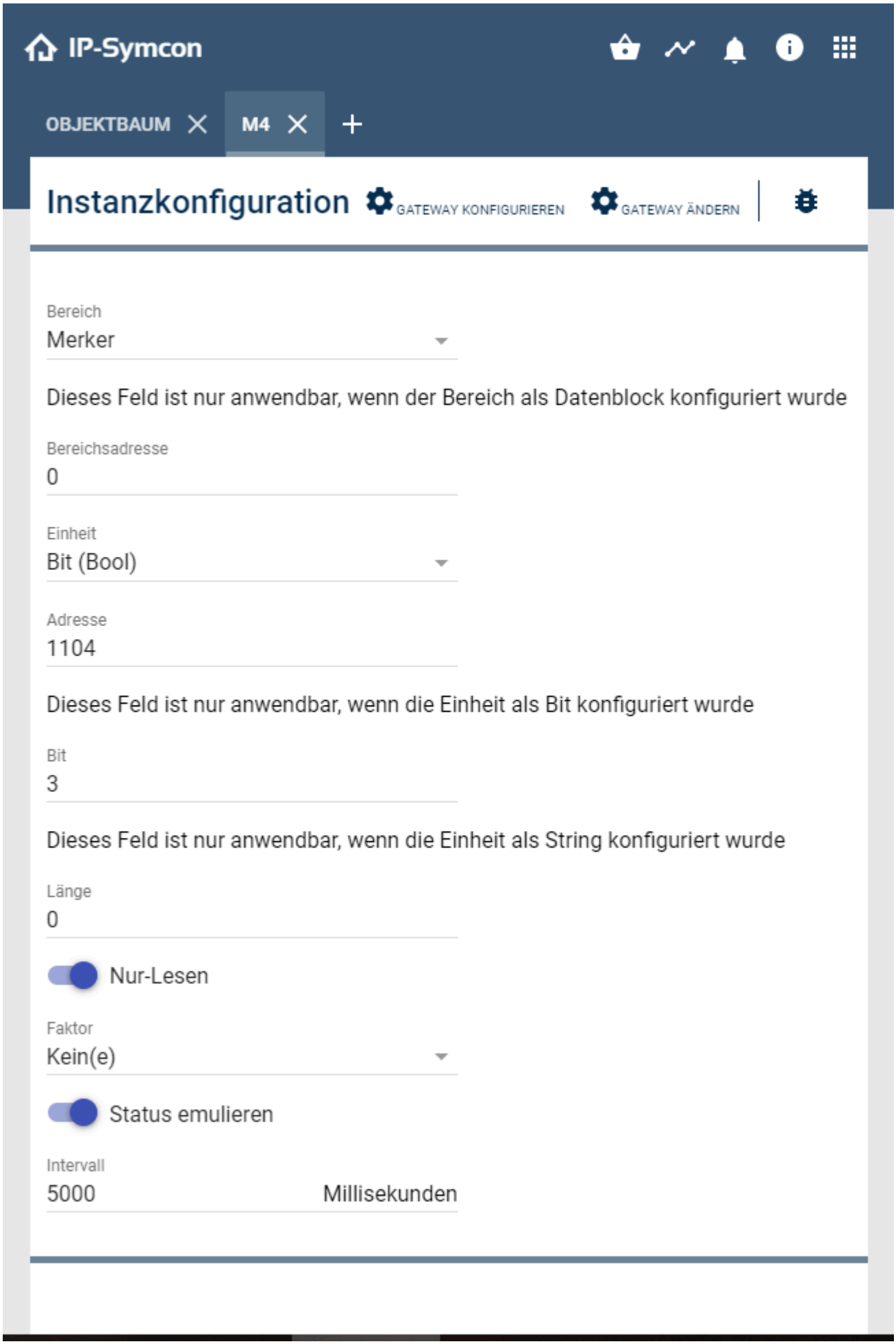


LOGO 7/8 CONFIGURATOR

If the Logo has been integrated properly, the respective memory addresses can be created as an instance via the Logo 7/8 Configurator. To do this, the respective input, output or marker must be selected and "Create" clicked on.



As soon as a module is created in the configurator and the configuration button is pressed, a new dialog opens.
New instances are created in the object tree in the main category. These created instances can then be renamed accordingly and sorted elsewhere. It is also possible to call up the respective instance configuration via "Configure" in the configurator.

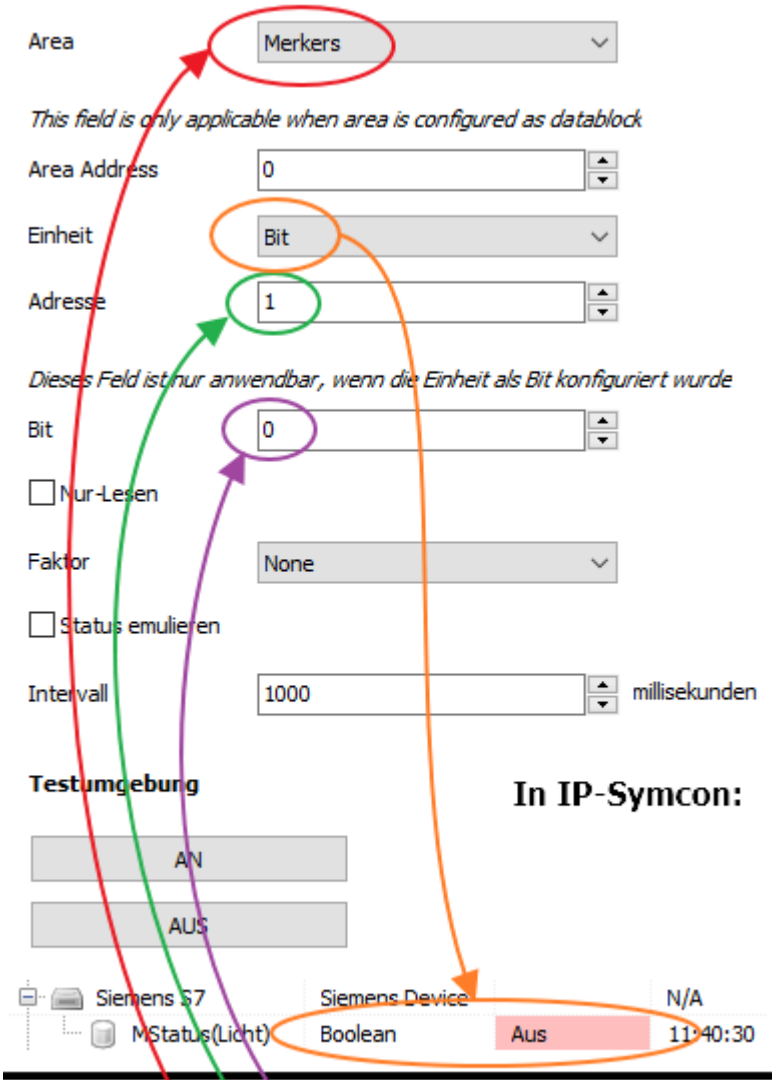

IP-Symcon
OBJEKTBAUM
M4
Instanzkonfiguration
GATEWAY KONFIGURIEREN
GATEWAY ÄNDERN
Bereich
Merker
Dieses Feld ist nur anwendbar, wenn der Bereich als Datenblock konfiguriert wurde
Bereichsadresse
0
Einheit
Bit (Bool)
Adresse
1104
Dieses Feld ist nur anwendbar, wenn die Einheit als Bit konfiguriert wurde
Bit
3
Dieses Feld ist nur anwendbar, wenn die Einheit als String konfiguriert wurde
Länge
0
Nur-Lesen
Faktor
Kein(e)
Status emulieren
Intervall
5000
Millisekunden

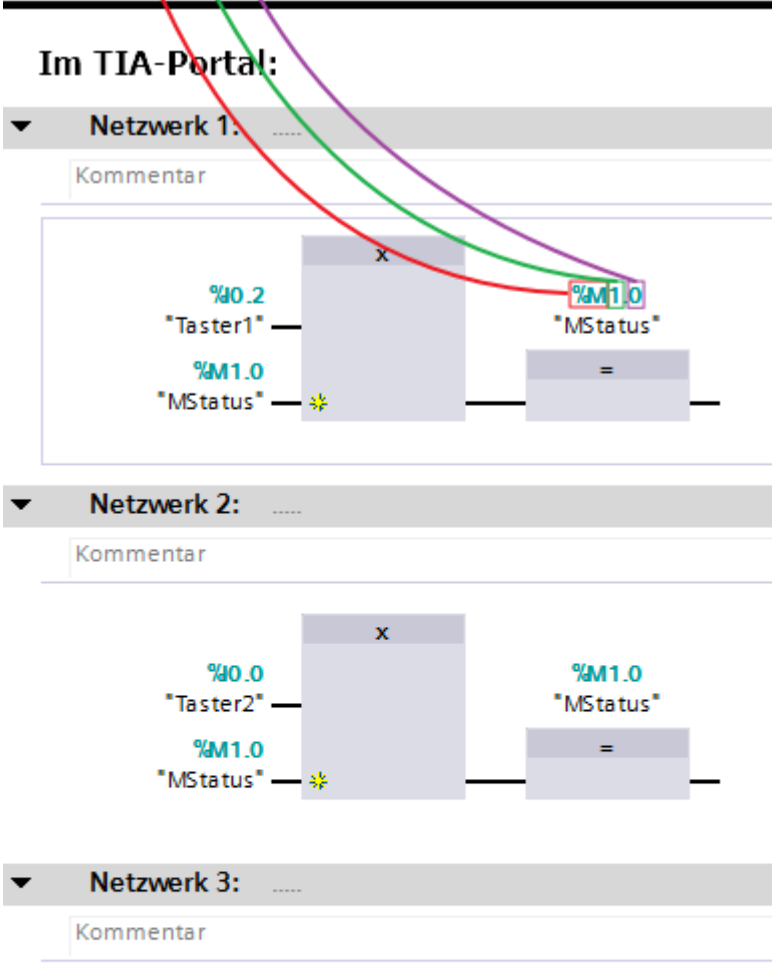
EXAMPLE WITH TIA PORTAL

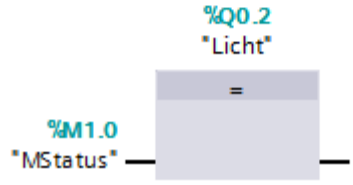
This example on the S7-1200 reflects 2 buttons, which should switch a light on/off independently of each other. At the same time, it should also be possible to control the light via IP-Symcon.

- Configure Main (OB1) as on the screenshot. (See screenshot)
- Load the program onto the PLC and start it
- Add "Siemens S7" instance in IPS-Symcon
- Configure the configuration page of the "Siemens S7" instance (see screenshot)
- Address and bit result from the TIA Portal (see screenshot)
- The type is specified under the "Area" field (see screenshot)
- The type of variable in IP-Symcon is specified via unit (see screenshot)
- The marker status is queried at the entered interval. (See screenshot)
- The test environment (ON/OFF) can be used to test directly whether the marker is switched correctly

With the XOR circuit and the marker, it is possible to switch over with a button and IP-Symcon will notice the variable change. Thus, IP-Symcon always shows the status of the light without having to use a separate variable to read out the PLC output. It is also possible for IP-Symcon to switch the light directly via the marker.
Better results can be achieved in this example if an AND (edge) is placed in front of the respective XOR to get just one cycle. However, whether this is required must be tested on site and is not included in this example.


Area
Merkers
This field is only applicable when area is configured as datablock
Area Address
0
Einheit
Bit
Adresse
1
Dieses Feld ist nur anwendbar, wenn die Einheit als Bit konfiguriert wurde
Bit
0
Nur-Lesen
Faktor
None
Status emulieren
Intervall
1000
millisekunden
Testumgebung
In IP-Symcon:
AN
AUS
Siemens S7
Siemens Device
N/A
MStatus(Licht)
Boolean
Aus
11:40:30


Im TIA-Portal:
Netzwerk 1:
Kommentar
%I0.2
"Taster1"
%M1.0
"MStatus"
%M1.0
"MStatus"
%M1.0
"MStatus"
=
Netzwerk 2:
Kommentar
%I0.0
"Taster2"
%M1.0
"MStatus"
%M1.0
"MStatus"
=
Netzwerk 3:
Kommentar

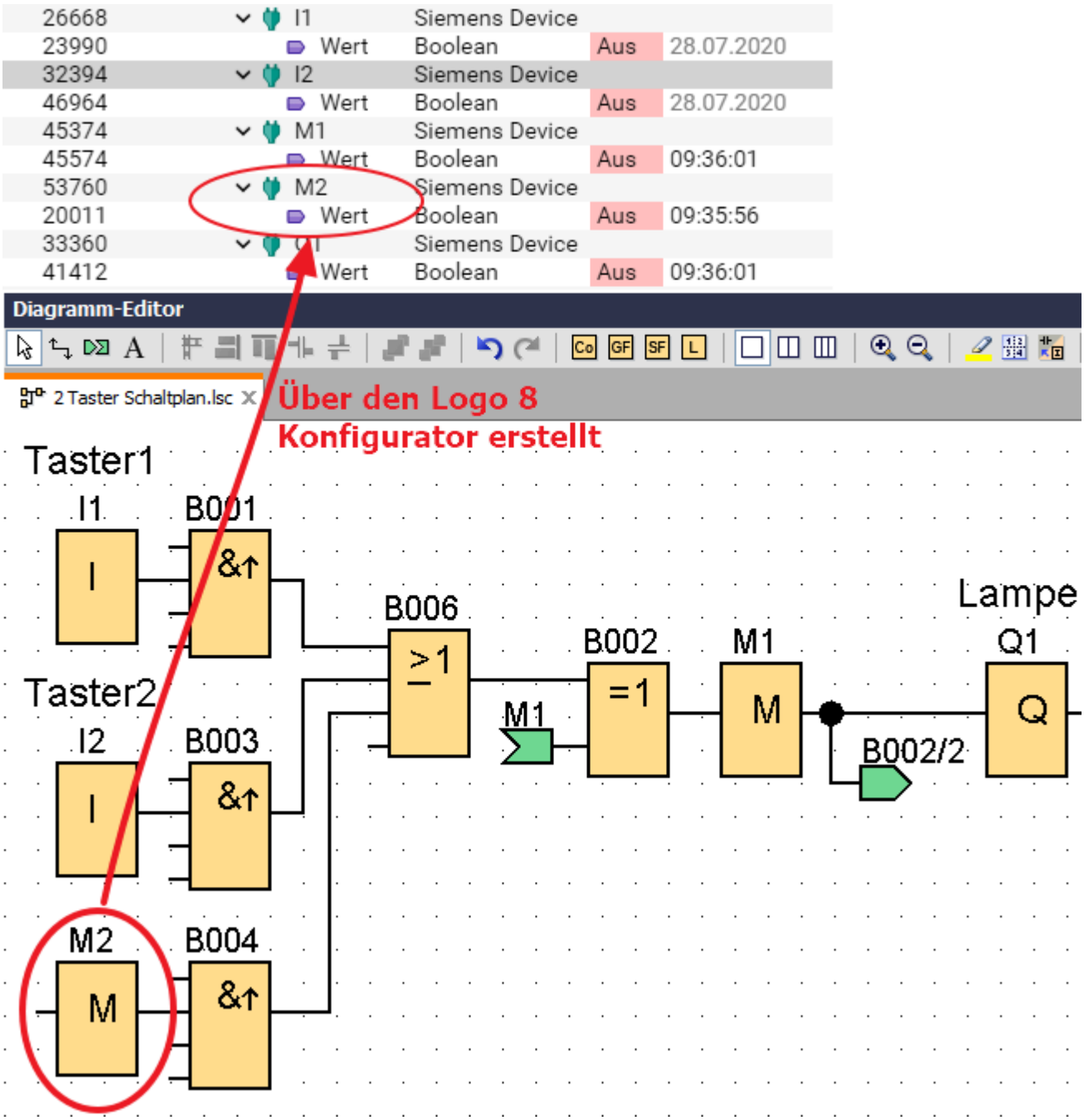


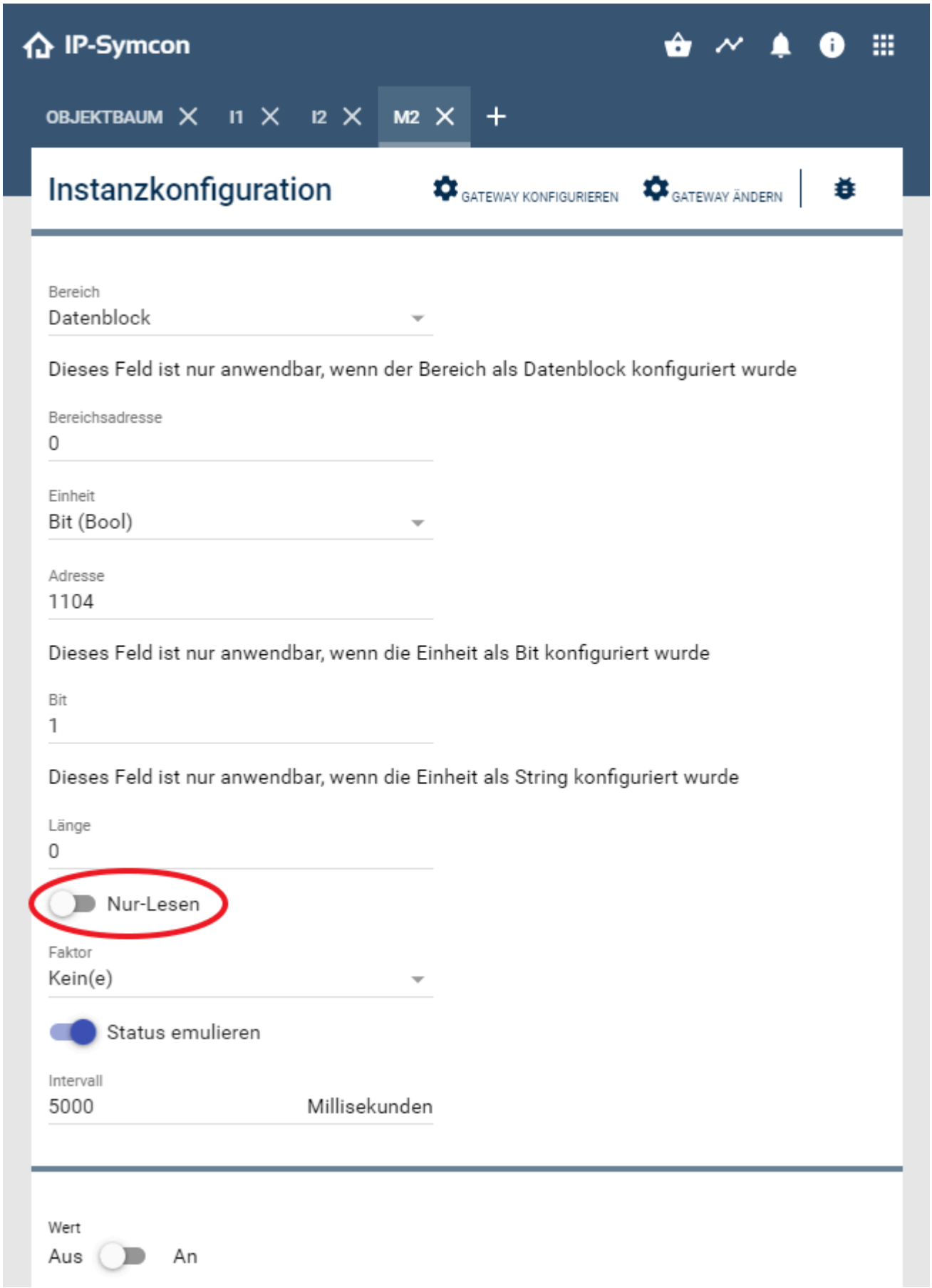
EXAMPLE WITH LOGO SOFT COMFORT

This example on the Logo 8 reflects 2 buttons, which should switch a light on/off independently of each other. At the same time, it should also be possible to control the light via IP-Symcon.

- Configure 2 button circuit diagram as shown on the screenshot. (See screenshot)
- Load the program onto the PLC and start it
- Add instance in IPS-Symcon "Logo Configurator" and configure interface as described above
- Create the desired variables using the configurator
- Deactivate the "Read only" option for switchable marker M2
- The type of variable in IP-Symcon is specified via unit
- The marker status is queried at the entered interval. (See screenshot)
- The test environment (ON/OFF) can be used to test directly whether the marker is switched correctly

With the XOR circuit and the marker, it is possible to switch over with a button and IP-Symcon will notice the variable change if the Q2 is set up. It is also possible for IP-Symcon to switch the light directly via marker M2. This is not possible via marker M1 because the logo does not support this in this configuration. Marker M2 acts as a switch in IP-Symcon and as a button in the Logo.


26668 I1 Siemens Device
23990 Wert Boolean Aus 28.07.2020
32394 I2 Siemens Device
46964 Wert Boolean Aus 28.07.2020
45374 M1 Siemens Device
45574 Wert Boolean Aus 09:36:01
53760 M2 Siemens Device
20011 Wert Boolean Aus 09:35:56
33360 Q1 Siemens Device
41412 Wert Boolean Aus 09:36:01
Diagramm-Editor
2 Taster Schaltplan.lsc
Über den Logo 8
Konfigurator erstellt
Taster1
I1
B001
&↑
B006
≥1
B002
=1
M1
M
Lampe
Q1
Q
B002/2
Taster2
I2
B003
&↑
M1
I
M2
B004
&↑
M


IP-Symcon
OBJEKTBAUM
I1
I2
M2
Instanzkonfiguration
GATEWAY KONFIGURIEREN
GATEWAY ÄNDERN
Bereich
Datenblock
Dieses Feld ist nur anwendbar, wenn der Bereich als Datenblock konfiguriert wurde
Bereichsadresse
0
Einheit
Bit (Bool)
Adresse
1104
Dieses Feld ist nur anwendbar, wenn die Einheit als Bit konfiguriert wurde
Bit
1
Dieses Feld ist nur anwendbar, wenn die Einheit als String konfiguriert wurde
Länge
0
Nur-Lesen
Faktor
Kein(e)
Status emulieren
Intervall
5000
Millisekunden
Wert
Aus
An

DATA TYPES

DATA TYPE	SIGN	BIT	NAME IP-SYMCON
BOOL	unsigned	1	Bit
BYTE	dependent	8	Char/Byte
WORD	dependent	16	Short/Word
DWORD	dependent	32	Integer/DWord
SINT	signed	8	Char
INT	signed	16	Short
DINT	signed	32	Integer
USINT	unsigned	8	Byte
UINT	unsigned	16	Word
UDINT	unsigned	32	DWord
REAL	signed	32	Real

LOGO-VM-ADDRESSES

The Logo has reserved fixed memory addresses in the VM memory above the 850th byte for inputs/outputs and markers.
These vary depending on the model (Logo7/8).
If the logo is to be read out with IP-Symcon, it is helpful to know these addresses.
A complete list can be found here: Logo-VM-Addresses

Device list

SUPPORTED GATEWAYS/COMPONENTS

MANUFACTURER	DEVICE NAME	DESCRIPTION
Siemens	Logo 7	
Siemens	Logo 8	
Siemens	S7 200/300/400	Connected via Ethernet
Siemens	S7 1200/1500	More information can be found HERE
Siemens	S7 1517f	More information can be found HERE Furthermore, the special S7 Configuration of the S7 must be considered

Logo-VM-Addresses

The Logo has fixed memory addresses for inputs/outputs and markers.
These vary depending on the model (Logo7/8). (Source LOGO!Soft Comfort Online Help)



These addresses must be entered within the logo configuration in IP-Symcon under "Area address". The respective bits under "Bit"

LOGO 7

Inputs Bit
Outputs Bit
Analog Inputs Word
Analog Outputs Word
Marker Bit
Analog Marker Word

LOGO 8

Inputs Bit
Outputs Bit
Analog Inputs Word
Analog Outputs Word
Marker Bit
Analog Marker Word
Network Inputs Bit
Network Outputs Bit
Analog Network Inputs Word
Analog Network Outputs Word

LOGO 7 INPUTS BIT

INPUT	ADDRESS	BIT
I1	923	0
I2	923	1
I3	923	2
I4	923	3
I5	923	4
I6	923	5
I7	923	6
I8	923	7
I9	924	0

INPUT	ADDRESS	BIT
I10	924	1
I11	924	2
I12	924	3
I13	924	4
I14	924	5
I15	924	6
I16	924	7
I17	925	0
I18	925	1
I19	925	2
I20	925	3
I21	925	4
I22	925	5
I23	925	6
I24	925	7

LOGO 7 OUTPUTS BIT

OUTPUT	ADDRESS	BIT
Q1	942	0
Q2	942	1
Q3	942	2
Q4	942	3
Q5	942	4
Q6	942	5
Q7	942	6
Q8	942	7
Q9	943	0
Q10	943	1
Q11	943	2
Q12	943	3
Q13	943	4
Q14	943	5
Q15	943	6
Q16	943	7

LOGO 7 ANALOG INPUTS WORD

INPUT	ADDRESS	BIT
AI1	926	0
AI2	928	0
AI3	930	0
AI4	932	0
AI5	934	0
AI6	936	0
AI7	938	0
AI8	940	0

LOGO 7 ANALOG OUTPUTS WORD

WORD	ADDRESS	BIT
AQ1	944	0
AQ2	946	0

LOGO 7 MARKER BIT

MARKER	ADDRESS	BIT
M1	948	0
M2	948	1
M3	948	2
M4	948	3
M5	948	4
M6	948	5
M7	948	6
M8	948	7
M9	949	0
M10	949	1
M11	949	2
M12	949	3
M13	949	4
M14	949	5
M15	949	6
M16	949	7
M17	950	0
M18	950	1
M19	950	2

MARKER	ADDRESS	BIT
M20	950	3
M21	950	4
M22	950	5
M23	950	6
M24	950	7
M25	951	0
M26	951	1
M27	951	2

LOGO 7 ANALOG MARKER WORD

MARKER	ADDRESS	BIT
AM1	952	0
AM2	954	0
AM3	956	0
AM4	958	0
AM5	960	0
AM6	962	0
AM7	964	0
AM8	966	0
AM9	968	0
M10	970	0
AM11	972	0
AM12	974	0
AM13	976	0
AM14	978	0
AM15	980	0
AM16	982	0

LOGO 8

LOGO 8 INPUTS BIT

INPUT	ADDRESS	BIT
I1	1024	0
I2	1024	1
I3	1024	2

INPUT	ADDRESS	BIT
I4	1024	3
I5	1024	4
I6	1024	5
I7	1024	6
I8	1024	7
I9	1025	0
I10	1025	1
I11	1025	2
I12	1025	3
I13	1025	4
I14	1025	5
I15	1025	6
I16	1025	7
I17	1026	0
I18	1026	1
I19	1026	2
I20	1026	3
I21	1026	4
I22	1026	5
I23	1026	6
I24	1026	7

LOGO 8 OUTPUTS BIT

OUTPUT	ADDRESS	BIT
Q1	1064	0
Q2	1064	1
Q3	1064	2
Q4	1064	3
Q5	1064	4
Q6	1064	5
Q7	1064	6
Q8	1064	7
Q9	1065	0
Q10	1065	1
Q11	1065	2
Q12	1065	3

OUTPUT	ADDRESS	BIT
Q13	1065	4
Q14	1065	5
Q15	1065	6
Q16	1065	7
Q17	1066	0
Q18	1066	1
Q19	1066	2
Q20	1066	3

LOGO 8 ANALOG INPUTS WORD

INPUT	ADDRESS	BIT
AI1	1032	0
AI2	1034	0
AI3	1036	0
AI4	1038	0
AI5	1040	0
AI6	1042	0
AI7	1044	0
AI8	1046	0

LOGO 8 ANALOG OUTPUTS WORD

OUTPUTS	ADDRESS	BIT
AQ1	1072	0
AQ2	1074	0
AQ3	1076	0
AQ4	1078	0
AQ5	1080	0
AQ6	1082	0
AQ7	1084	0
AQ8	1086	0

LOGO 8 MARKER BIT

MARKER	ADDRESS	BIT
M1	1104	0

MARKER	ADDRESS	BIT
M2	1104	1
M3	1104	2
M4	1104	3
M5	1104	4
M6	1104	5
M7	1104	6
M8	1104	7
M9	1105	0
M10	1105	1
M11	1105	2
M12	1105	3
M13	1105	4
M14	1105	5
M15	1105	6
M16	1105	7
M17	1106	0
M18	1106	1
M19	1106	2
M20	1106	3
M21	1106	4
M22	1106	5
M23	1106	6
M24	1106	7
M25	1107	0
M26	1107	1
M27	1107	2
M28	1107	3
M29	1107	4
M30	1107	5
M31	1107	6
M32	1107	7
M33	1108	0
M34	1108	1
M35	1108	2
M36	1108	3
M37	1108	4
M38	1108	5

MARKER	ADDRESS	BIT
M39	1108	6
M40	1108	7
M41	1109	0
M42	1109	1
M43	1109	2
M44	1109	3
M45	1109	4
M46	1109	5
M47	1109	6
M48	1109	7
M49	1110	0
M50	1110	1
M51	1110	2
M52	1110	3
M53	1110	4
M54	1110	5
M55	1110	6
M56	1110	7
M57	1111	0
M58	1111	1
M59	1111	2
M60	1111	3
M61	1111	4
M62	1111	5
M63	1111	6
M64	1111	7

LOGO 8 ANALOG MARKER WORD

MARKER	ADDRESS	BIT
AM1	1118	0
AM2	1120	0
AM3	1122	0
AM4	1124	0
AM5	1126	0
AM6	1128	0
AM7	1130	0

MARKER	ADDRESS	BIT
AM8	1132	0
AM9	1134	0
AM10	1136	0
AM11	1138	0
AM12	1140	0
AM13	1142	0
AM14	1144	0
AM15	1146	0
AM16	1148	0
AM17	1150	0
AM18	1152	0
AM19	1154	0
AM20	1156	0
AM21	1158	0
AM22	1160	0
AM23	1162	0
AM24	1164	0
AM25	1166	0
AM26	1168	0
AM27	1170	0
AM28	1172	0
AM29	1174	0
AM30	1176	0
AM31	1178	0
AM32	1180	0
AM33	1182	0
AM34	1184	0
AM35	1186	0
AM36	1188	0
AM37	1190	0
AM38	1192	0
AM39	1194	0
AM40	1196	0
AM41	1198	0
AM42	1200	0
AM43	1202	0
AM44	1204	0

MARKER	ADDRESS	BIT
AM45	1206	0
AM46	1208	0
AM47	1210	0
AM48	1212	0
AM49	1214	0
AM50	1216	0
AM51	1218	0
AM52	1220	0
AM53	1222	0
AM54	1224	0
AM55	1226	0
AM56	1228	0
AM57	1230	0
AM58	1232	0
AM59	1234	0
AM60	1236	0
AM61	1238	0
AM62	1240	0
AM63	1242	0
AM64	1244	0

LOGO 8 NETWORK INPUTS BIT

INPUT	ADDRESS	BIT
NI1	1246	0
NI2	1246	1
NI3	1246	2
NI4	1246	3
NI5	1246	4
NI6	1246	5
NI7	1246	6
NI8	1246	7
NI9	1247	0
NI10	1247	1
NI11	1247	2
NI12	1247	3
NI13	1247	4

INPUT	ADDRESS	BIT
NI14	1247	5
NI15	1247	6
NI16	1247	7
NI17	1248	0
NI18	1248	1
NI19	1248	2
NI20	1248	3
NI21	1248	4
NI22	1248	5
NI23	1248	6
NI24	1248	7
NI25	1249	0
NI26	1249	1
NI27	1249	2
NI28	1249	3
NI29	1249	4
NI30	1249	5
NI31	1249	6
NI32	1249	7
NI33	1250	0
NI34	1250	1
NI35	1250	2
NI36	1250	3
NI37	1250	4
NI38	1250	5
NI39	1250	6
NI40	1250	7
NI41	1251	0
NI42	1251	1
NI43	1251	2
NI44	1251	3
NI45	1251	4
NI46	1251	5
NI47	1251	6
NI48	1251	7
NI49	1252	0
NI50	1252	1

INPUT	ADDRESS	BIT
NI51	1252	2
NI52	1252	3
NI53	1252	4
NI54	1252	5
NI55	1252	6
NI56	1252	7
NI57	1253	0
NI58	1253	1
NI59	1253	2
NI60	1253	3
NI61	1253	4
NI62	1253	5
NI63	1253	6
NI64	1253	7

LOGO 8 NETWORK OUTPUTS BIT

OUTPUT	ADDRESS	BIT
NQ1	1390	0
NQ2	1390	1
NQ3	1390	2
NQ4	1390	3
NQ5	1390	4
NQ6	1390	5
NQ7	1390	6
NQ8	1390	7
NQ9	1391	0
NQ10	1391	1
NQ11	1391	2
NQ12	1391	3
NQ13	1391	4
NQ14	1391	5
NQ15	1391	6
NQ16	1391	7
NQ17	1392	0
NQ18	1392	1
NQ19	1392	2

OUTPUT	ADDRESS	BIT
NQ20	1392	3
NQ21	1392	4
NQ22	1392	5
NQ23	1392	6
NQ24	1392	7
NQ25	1393	0
NQ26	1393	1
NQ27	1393	2
NQ28	1393	3
NQ29	1393	4
NQ30	1393	5
NQ31	1393	6
NQ32	1393	7
NQ33	1394	0
NQ34	1394	1
NQ35	1394	2
NQ36	1394	3
NQ37	1394	4
NQ38	1394	5
NQ39	1394	6
NQ40	1394	7
NQ41	1395	0
NQ42	1395	1
NQ43	1395	2
NQ44	1395	3
NQ45	1395	4
NQ46	1395	5
NQ47	1395	6
NQ48	1395	7
NQ49	1396	0
NQ50	1396	1
NQ51	1396	2
NQ52	1396	3
NQ53	1396	4
NQ54	1396	5
NQ55	1396	6
NQ56	1396	7

OUTPUT	ADDRESS	BIT
NQ57	1397	0
NQ58	1397	1
NQ59	1397	2
NQ60	1397	3
NQ61	1397	4
NQ62	1397	5
NQ63	1397	6
NQ64	1397	7

LOGO 8 ANALOG NETWORK INPUTS WORD

INPUT	ADDRESS	BIT
NAI1	1262	0
NAI2	1264	0
NAI3	1266	0
NAI4	1268	0
NAI5	1270	0
NAI6	1272	0
NAI7	1274	0
NAI8	1276	0
NAI9	1278	0
NAI10	1280	0
NAI11	1282	0
NAI12	1284	0
NAI13	1286	0
NAI14	1288	0
NAI15	1290	0
NAI16	1292	0
NAI17	1294	0
NAI18	1296	0
NAI19	1298	0
NAI20	1300	0
NAI21	1302	0
NAI22	1304	0
NAI23	1306	0
NAI24	1308	0
NAI25	1310	0

INPUT	ADDRESS	BIT
NAI26	1312	0
NAI27	1314	0
NAI28	1316	0
NAI29	1318	0
NAI30	1320	0
NAI31	1322	0
NAI32	1324	0

LOGO 8 ANALOG NETWORK OUTPUTS WORD

OUTPUT	ADDRESS	BIT
NAQ1	1406	0
NAQ2	1408	0
NAQ3	1410	0
NAQ4	1412	0
NAQ5	1414	0
NAQ6	1416	0
NAQ7	1418	0
NAQ8	1420	0
NAQ9	1422	0
NAQ10	1424	0
NAQ11	1426	0
NAQ12	1428	0
NAQ13	1430	0
NAQ14	1432	0
NAQ15	1434	0
NAQ16	1436	0

S7_RequestRead

```
bool S7_RequestRead (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Re-reads the device with ID InstanceID.
As soon as the function has been executed, the new values can be called up in the status variables.

EXAMPLE

```
// Read device
S7_RequestRead(12345);
```

S7_Write

```
bool S7_Write (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	32-bit floating point value according to IEEE754 or integer

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Writes the value **Value** to the configured address of the instance with the ID **InstanceID** without knowing the exact data type.



This function also supports setting values when a factor has been set.



This function cannot be used for bit values. S7_WriteBit must be used for bit values.

EXAMPLE

```
// Float value
S7_Write(12345, 23.5);

// Integer
S7_Write(12345, 123);
```

S7_WriteBit

```
bool S7_WriteBit (int $InstanceID, bool $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Set the address with the ID InstanceID to value status

EXAMPLE

```
S7_WriteBit(12345, true); //Turn on device
```

S7_WriteByte

```
bool S7_WriteByte (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	0-255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Set the address with the ID **InstanceID** to **Value**

EXAMPLE

```
S7_WriteByte(12345, 123);
```

S7_WriteChar

Require: Symcon >= 4.4

```
bool S7_WriteChar (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-128 to 127

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function was named **S7_WriteShortInt** up to and including Version 4.3

Set the address with the ID **InstanceID** to **Value**

EXAMPLE

```
S7_WriteChar(12345, -123);
```

S7_WriteDWord

```
bool S7_WriteDWord (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	0-4294967295

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Set the address with the ID **InstanceID** to **Value**



On armhf systems, only the range up to 2147483647 can be used due to the limitation to 32-bit integers in PHP. armhf is supported until Symcon 10.0.

EXAMPLE

```
S7_WriteDWord(12345, 123);
```

S7_WriteInteger

```
bool S7_WriteInteger (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-2147483648 bis 2147483647

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Set the address with the ID **InstanceID** to **Value**

EXAMPLE

```
S7_WriteInteger(12345, -123);
```

S7_WriteReal

```
bool S7_WriteReal (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	64bit floating point value according to IEEE754

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Set the address with the ID **InstanceID** to **Value**

EXAMPLE

```
S7_WriteReal(12345, 23.5);
```

S7_WriteShort

Require: Symcon >= 4.4

```
bool S7_WriteShort (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	-32768 to 32767

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function was named **S7_WriteSmallInt** up to and including Version 4.3

Set the address with the ID **InstanceID** to **Value**

EXAMPLE

```
S7_WriteShort(12345, -123);
```

S7_WriteWord

```
bool S7_WriteWord (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	0-65535

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Set the address with the ID **InstanceID** to **Value**

EXAMPLE

```
S7_WriteWord(12345, 123);
```

PLC: Wago, Beckhoff, ABB

PLCs that support ModBus RTU/TCP can be integrated into IP-Symcon. A connection is established via ModBus RTU/TCP .



The following devices are supported by IP-Symcon:
Supported devices

For example
Wago 750-841 - The connection to IP-Symcon is made via the LAN interface.

DESCRIPTION

Programmable Logic Controllers (PLC) are independent electronic assemblies for controlling machines and systems. These work independently of a PC. The functions for control and regulation are stored in a program. IP-Symcon is used for visualization as well as the Human-Machine Interface (HMI), i.e. the interface between the operator and the machine.

SETUP IN IP-SYMCON

The Wago, Beckhoff and ABB PLCs can be controlled via the IP-Symcon ModBus RTU/TCP module. Since ModBus is the parent protocol for control, the names of the terminal device manufacturers are no longer explicitly mentioned in the control module. The documentation for this can be found here: ModBus RTU/TCP

CONVERT IEC 61131 TO MODBUS REGISTER

This tool can be used to quickly convert the WAGO (IEC 61131) addresses into the respective ModBus address.
%MW1000

Convert >>>

TIPS & TICKS

WAGO ModBus Register Mapping Table

Sync Remote

Require: Symcon >= 7.1



The Sync Remote module is a paid extension that can be purchased for any existing Symcon license. For each Symcon license, 1 remote system can be integrated free of charge. The extension can be purchased directly in the Shop.

The Sync Remote enables a server to synchronize with a remote client. The entire content of the client's object tree is integrated into the server's object tree. The server can then use the client's object tree as if it were part of its own object tree and can both read and modify the objects in it.
Switching mirrored variables (e.g. with SetValue or RequestAction), calls of module functions on mirrored instances as well as changes via IPS_ *functions (e.g. IPS_SetName, IPS_SetProperty, IPS_ApplyChanges, IPS_DeleteVariable or IPS_RunScript) are forwarded to the client and executed there. Reading functions (IPS_Get*, IPSIs*, IPS*\Exists, IPS_HasChildren, IPS_HasPermission) as well as the functions for converting the IDs (IPS_FindObjectID, IPS_FindRemoteID, IPS_FindServerID) on the other hand act locally on the mirrored object of the server. Exceptions are IPS_GetConfigurationForm, IPS_GetMediaContent, IPS_GetScriptContent and IPS_GetFlowScriptStatistic, which are queried from the client. The debug functions (IPS_EnableDebug, IPS_DisableDebug, IPS_EnableDebugFile, IPS_DisableDebugFile, IPS_SendDebug) are not available for mirrored instances and report an error.

SCOPE OF FUNCTIONS

Full access to the client from the server
Object tree of the client seamlessly integrated into that of the server
Server can read and modify client

SOFTWARE INSTALLATION

If the paid extension has been purchased for the server license, the module can be used there.

SET UP THE INSTANCES IN SYMCON

The 'Sync Remote' module can be found under "Add instance" using the quick filter.
Further information in the documentation of the instances.

CONFIGURATION

The Remote access must be activated on the client for synchronization. Sync Remote must be configured on the server.

NAME	DESCRIPTION
Enable synchronization of remote Symcon	The synchronization is performed if this switch is active
Host	The URL under which the client can be reached
Port	The port under which the client provides Symcon
Use SSL	If the switch is set, the connection to the client is encrypted via SSL
Verify Peer	Verify that the SSL certificate of the host is correct (only visible if Use SSL is active)
Verify Host	Verify that the host URL from the SSL certificate matches the host from the property (only visible if Use SSL is active)
Username	The user name, i.e. the license address, of the client
Password	The password for remote access to the client
Timeout	The timeout for requests to the client in milliseconds. If a request takes longer, the connection is marked as faulty and no synchronization is performed.
Strategy (Profile)	Selection of the strategy for synchronizing Variable profiles of the client **Only assign existing profiles**: No new profiles are created on the server or existing ones are modified. This means that variables are displayed on the basis of the server's profiles and may differ from those of the client **Create profiles if not existent**: Profiles of the client that do not exist on the server are also created on the server. This means that variables with previously unknown profiles are displayed as on the client. However, existing profiles are retained. This means that these variables can be displayed differently under certain circumstances **Create/Update matching profiles**: Create profiles of the client that are missing on the server and modify existing profiles of the server so that they match the client. This ensures that the display of variables corresponds to that of the client. However, this could unintentionally modify the display of variables outside the synchronized area.
Interval	This option specifies how often the server synchronizes with the client
Reduce logging for cyclic updates	If this switch is activated, logging in the message window is reduced.
Save data transfer costs by loading only diffs (uses more CPU on both sides)	If this switch is activated, the client only sends the changes since the last synchronization to the client. This reduces the amount of data that is transferred, but requires more computing power on both systems to calculate and integrate the changes (from Symcon 7.2).
Burst Interval when actions happen	If the server switches a synchronized variable, the synchronization is carried out more frequently for a short time so that the associated changes are visible more quickly. This option specifies how short the update interval is for this burst (from Symcon 7.2)
Number of Burst updates after an action	If the server switches a synchronized variable, this option specifies how many update requests are performed during the burst (from Symcon 7.2)

NAME	DESCRIPTION
"Update" button	When the button is pressed, synchronization is performed immediately



The systems should use the same version of Symcon. When upgrading the systems, it is recommended to update the server first and then the client. Otherwise there may be unwanted interactions.

Technische Alternative

Technische Alternative offers control of various controllers and system monitoring. A connection to IP-Symcon is established using CMI or BL-NET.

DESCRIPTION

IP-Symcon supports both the BL-NET (UVR1611) and the CMI variants of Technische Alternative.
CMI Installation
BL-NET Installation

CMI

The connection to IP-Symcon is established via LAN.

INTEGRATION IN IP-SYMCON

After a suitable CMI instance has been created, the IP address of the CMI must be entered. The device, node number and interval must also be specified.
If authentication is required, this must be entered in the parent instance of the WWW Reader.
The connection can be tested via "Update".
Variables are automatically created on first-time queries and updated at every interval.

BL-NET

The connection to IP-Symcon is made via LAN or USB using the "BOOTLOADER" (BL-NET). In any case, the DL (data line) must be used.



Reading out via the CAN bus is not possible.

INTEGRATION IN IP-SYMCON

After an instance has been created in IP-Symcon, the IP address of the BL-NET must be specified. If the USB version is to be used, a new parent instance must be created and the SerialPort must be selected when selecting it.
In order to display one or more variables in the Visualization, a profile (e.g. temperature) must be assigned to each of them in order to make it visually recognizable what type of value they are. As the UVR1611 controls and inputs are user defined, this cannot be done automatically by the system. More information about variable profiles can be found here:
Variable Profiles

If **two (2)** UVR1611 are to be connected, two instances must be created. Once with ID = 1 and once with ID = 2. However, the timer may only be activated in one of the two instances, since a request delivers both data packets and thus updates both instances.

TIPS & TRICKS

Since the BL-NET is not very resilient, as few applications as possible should access the device. Otherwise failures can occur during which the BL-NET cannot be reached.
The following entries in the log file are normal and not an error:

```
Client Socket | Socket: Connect.... #1
Client Socket | Socket: Connected
Client Socket | Socket: Disconnected
```

Device list

CMI

DEVICES	DESCRIPTION
CAN-BC2	Interface device for CAN bus devices
CAN-EZ2	Energy meter, heat meter
CAN-EZ3	Energy meter, heat meter
CAN-I/O45	RSM610/UVR16x2 Expansion unit for additional inputs/outputs
CAN-MTx2	RSM610/UVR16x2 Expansion unit with 4.3" touch display, one operating and display unit
RSM610	Freely programmable control and switching module. Possible extension of UVR1611/16x2
UVR1611	Universal control
UVR16x2	Freely programmable universal controller
UVR610	Freely programmable universal controller
UVR65*	Universal control
UVR67*	Universal control

BL-NET

DEVICES	DESCRIPTION
UVR1611	Universal control

Door Intercom

Require: Symcon >= 8.0



The door intercom module is a paid extension that can be purchased for any existing Symcon license in the Shop. The door intercom module can be tried out free of charge with any Symcon license.

FUNCTIONAL SCOPE

Support for Direct SIP and SIP with registration on PBX
Intercom with SIP-compatible door stations
Choice of audio/video, audio only or audio with external video (RTSP)
Forward video from door station or RTSP to visualization (web + app)
Audio incl. intercom from door station to visualization (web + app) and back
Also supported via the Connect service
Support for multiple parallel calls with different door stations
When it rings, a dialog is displayed and it can be accepted or rejected
The visualization can be used as a telephone that can be called
If only audio is available, an RTSP stream can be specified as an external video source
State variables (wait, ring, speak)
Variables can be configured with a value as an action
Notifications and calls are also displayed for open objects and in the settings.
A ringtone is played in the visualization when a call is received
Calls are displayed when notifications are opened

NECESSARY TECHNICAL PARAMETERS

SIP: Direct SIP or SIP on with registration on a PBX
Video: H264 (VoIP/SIP or RTSP as external source as with the streams)
Audio: G711 (PCMA, PCMU)

SETUP

Create a new instance called "Door Intercom". This must be in the visible area of the visualization so that the call is correctly routed to the visualization.
When the doorbell rings, "sip:symcon_12345@192.168.1.5:5060" must then be called, for example
When logging on to an external PBX, all incoming calls are forwarded to the visualization

DEVICE-SPECIFIC SETUP

2N

FANVIL
DoorBird
FRITZ!Box

CONFIGURATION PAGE

NAME	DESCRIPTION
Active	Activates the module
Mode	Direct SIP or SIP with registration
Domain	Only for SIP with registration
User name	The user name to be used for the login
AuthID	Only visible for SIP with registration
Password	Password can be left blank for less secure authentication
Tracks	Audio/Video, Audio only, Audio with external video
Media Stream	With external video, a stream can be selected here that is displayed during the call.
Actions	Actions can be executed directly on the call screen

ACTIONS

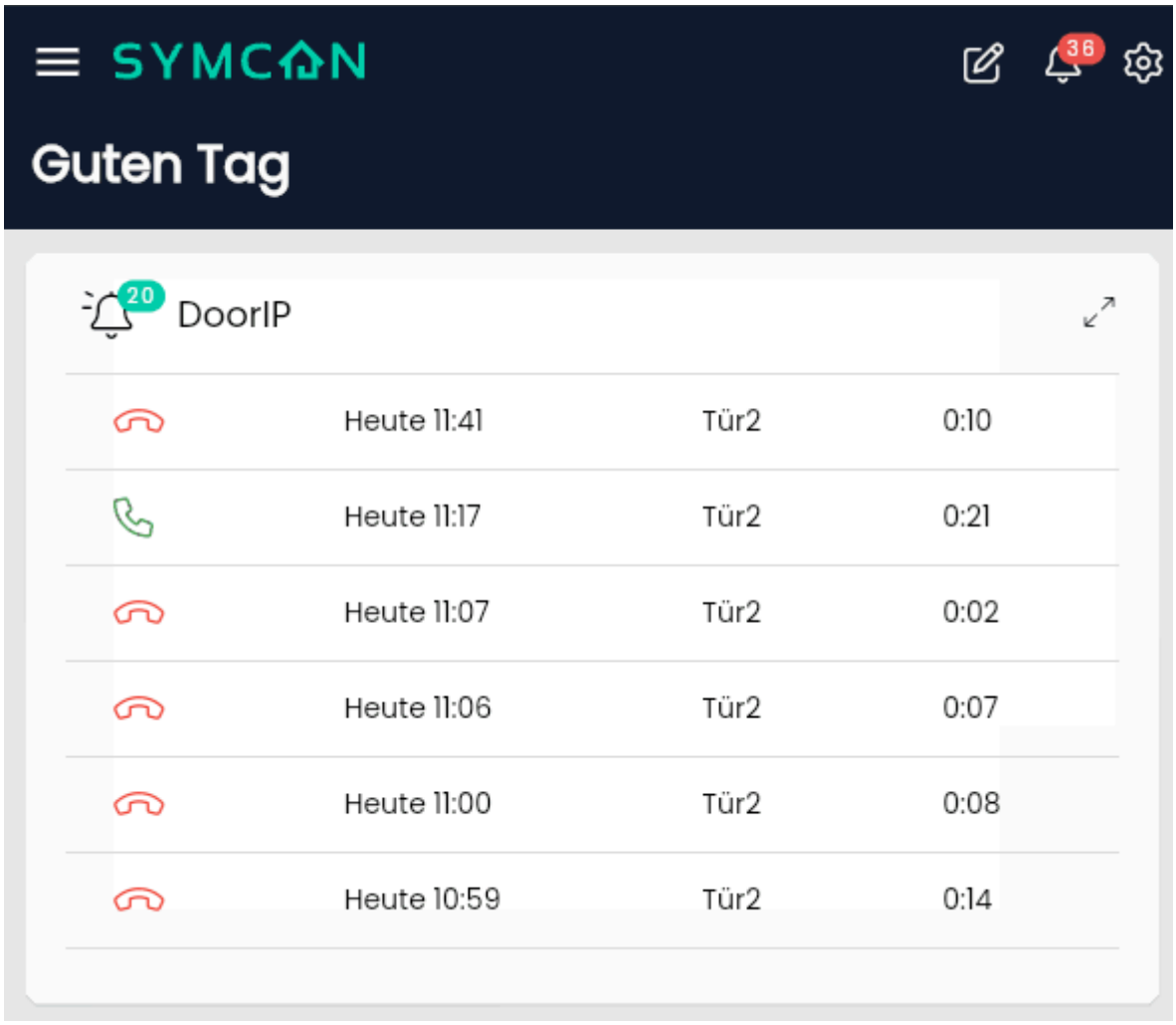
FIELD	DESCRIPTION
Name	The text that is displayed on the button in the visualization
Action	The variable to be switched or the action to be executed
Value	The value to which a selected variable should be set

STATUS VARIABLES

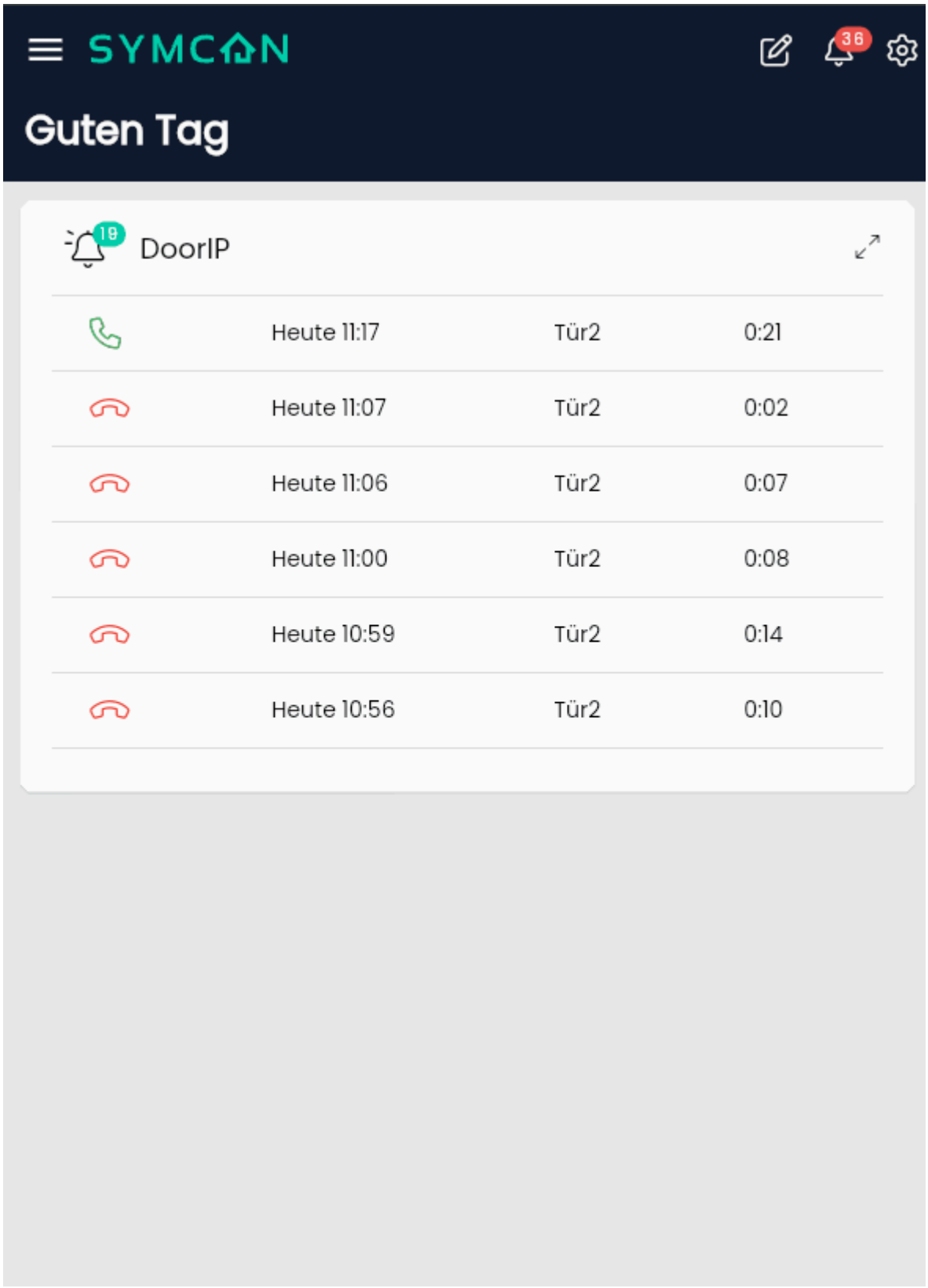
NAME	TYPE	DESCRIPTION
Call list	String	JSON-encoded list of recent calls that can be displayed in the visualization
Do not disturb	Boolean	If active, no dialog is displayed in full screen when a call is received. The call can still be answered via the tile
Doorbell	Integer	Doorbell status (0: Waiting, 1: Ringing, 2: Speaking)
Missed calls	Integer	Number of missed calls. Are displayed with in the tile

VISUALIZATION

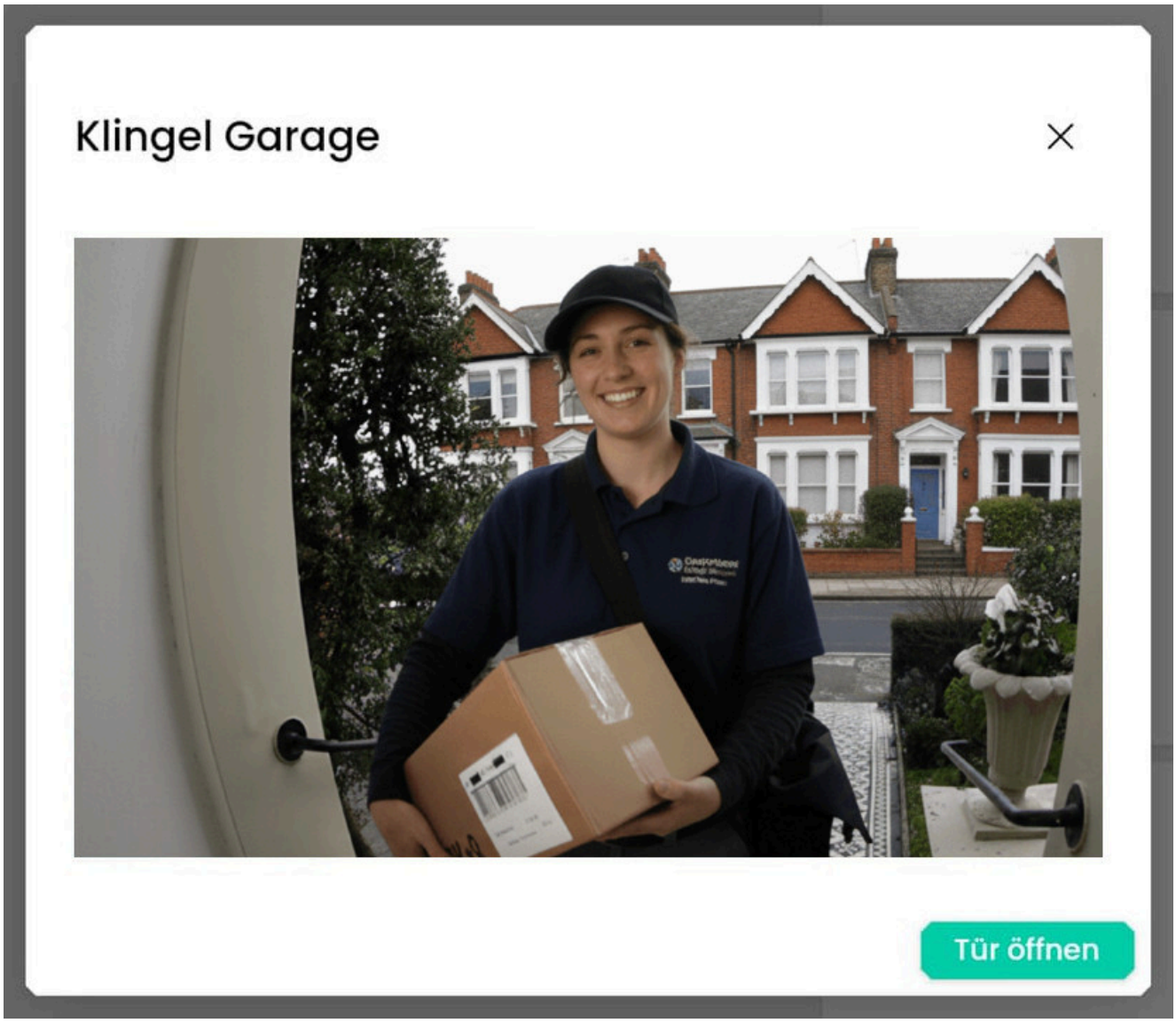
The door intercom system has its own display in the visualization.
A list of past calls is displayed in the tile.



When the Door Intercom instance is called, a dialog is displayed to accept or reject the call.



Once the connection has been established, the configured actions can be executed from the dialog.



Internally, the integration of the door stations has the code name: DoorIP

Device list

SUPPORTED COMPONENTS

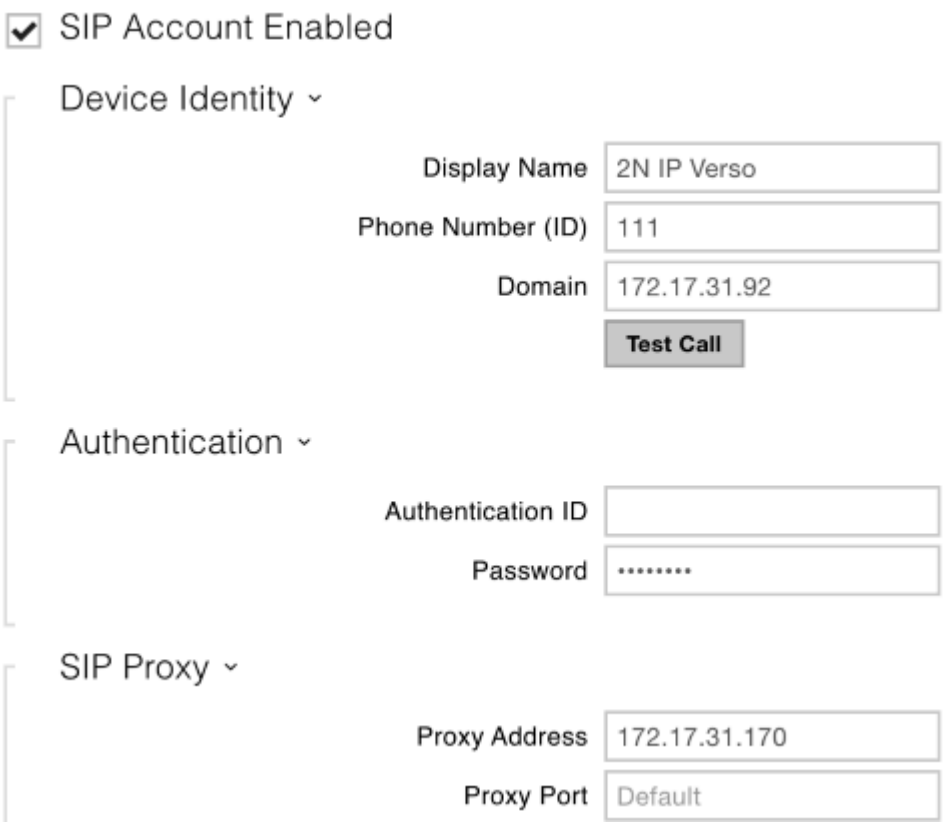
In general, all door stations that support SIP should be integrable. We guarantee functionality with the following components. The Support supported, should the setup of other devices cause difficulties.

SYSTEM	DESCRIPTION
2N IP Verso	
2N IP Style	
2N IP One	
FANVIL I10SV	
DoorBird	

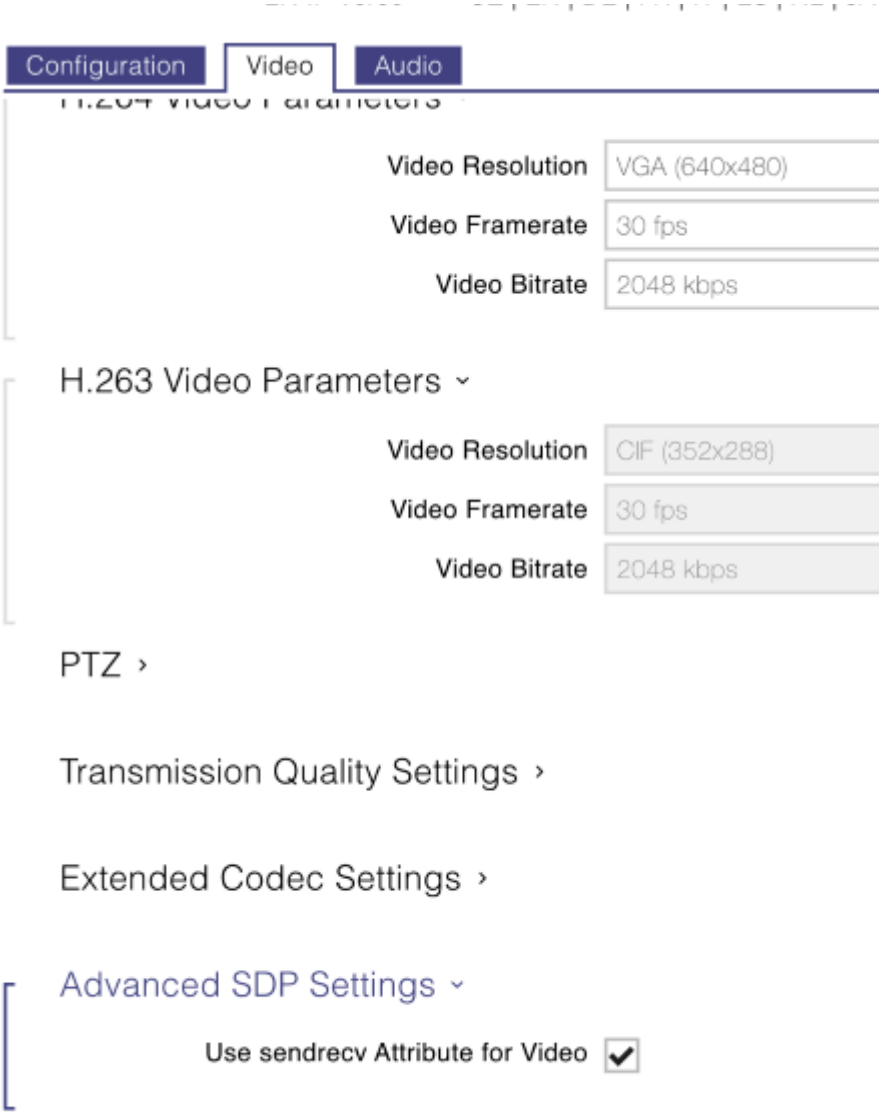
2N

SETTINGS IN THE 2N INTERFACE

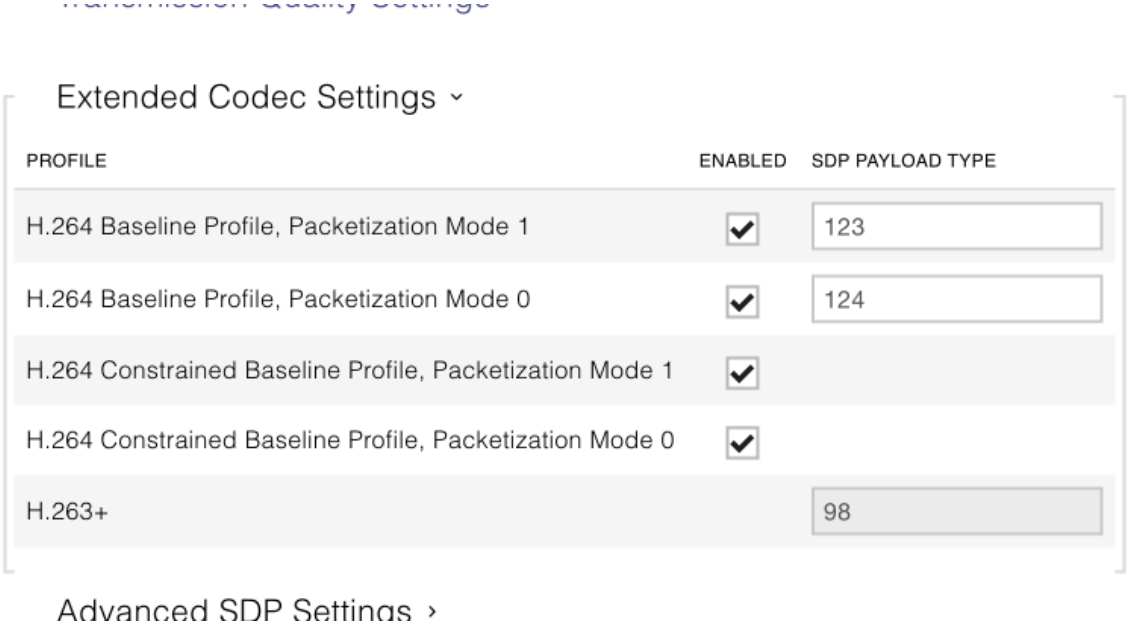
Domain: The IP address of the 2N
Proxy address: The IP address of the Symcon server
Display Name and Phone Number can be freely selected.

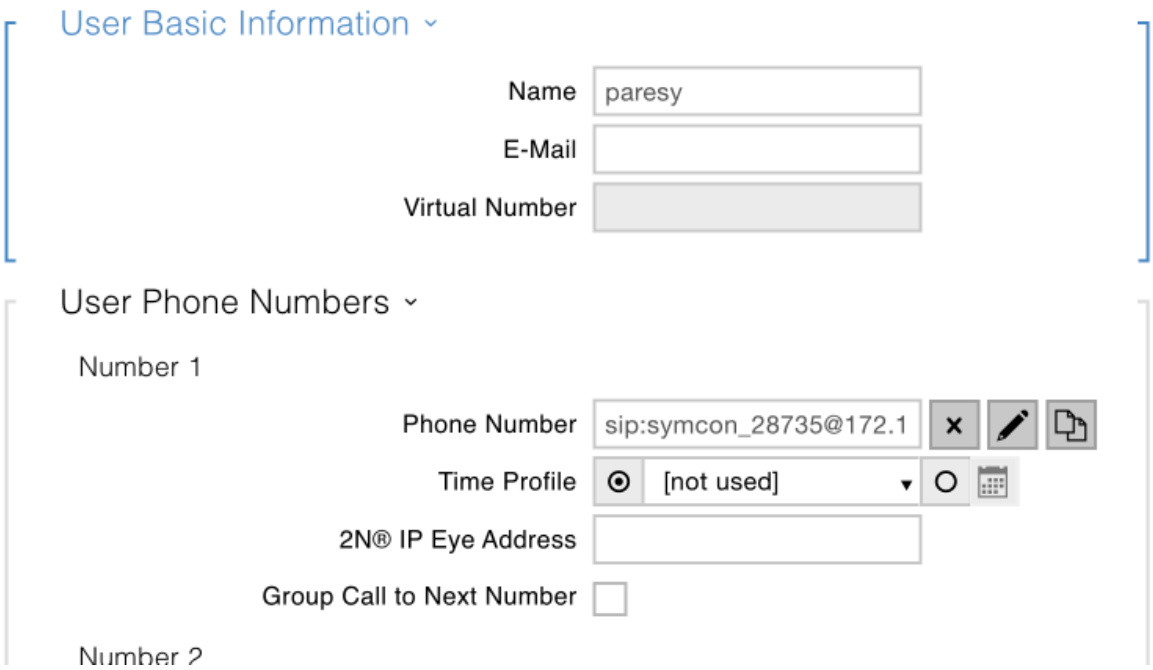


Use sendrecv Attribute for Video: Activated

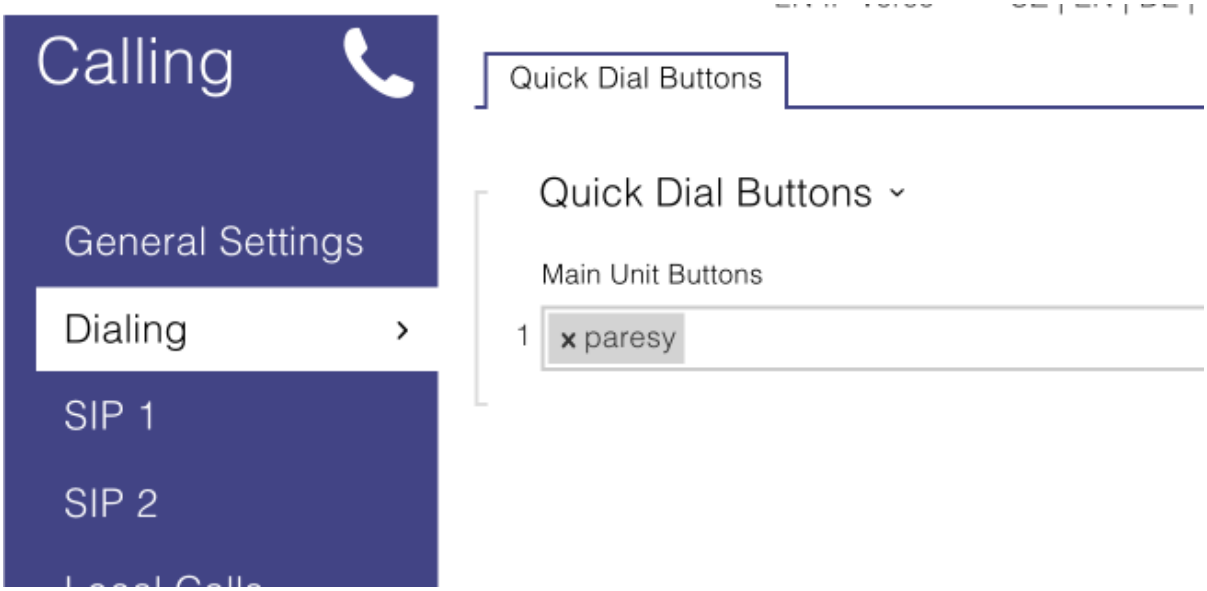


The codec settings must be set as follows





User Phone Numbers → Number 1:
UserNameFromIntercomInstance@IPfromSymconServer:5060
In the example: sip:symcon_28735@172.17.31.170
The previously configured user must be called when the bell is pressed.



DoorBird

SETTINGS IN THE DOORBIRD INTERFACE

Expert setting → SIP settings → ONLY enter the SIP user. In this case "symcon_28735"



No other settings must be set for Direct SIP to work correctly

Favorites → SIP numbers → Add → Name SYMCON-Direct → SIP address: UsernameFromDoorIntercomInstance@IPfromSymconServer. In this case symcon_28735@172.17.31.170
Key configuration → Settings → Key → Store the "SYMCON-Direct" favorites
If the button configuration item does not exist, the setting is located in Expert settings → Schedule for doorbell → SIP call

SETTINGS IN THE DOOR INTERCOM INSTANCE

Mode: Direct SIP
User name: symcon_28735
Password: *leave empty*
Tracks: Audio/Video

FANVIL

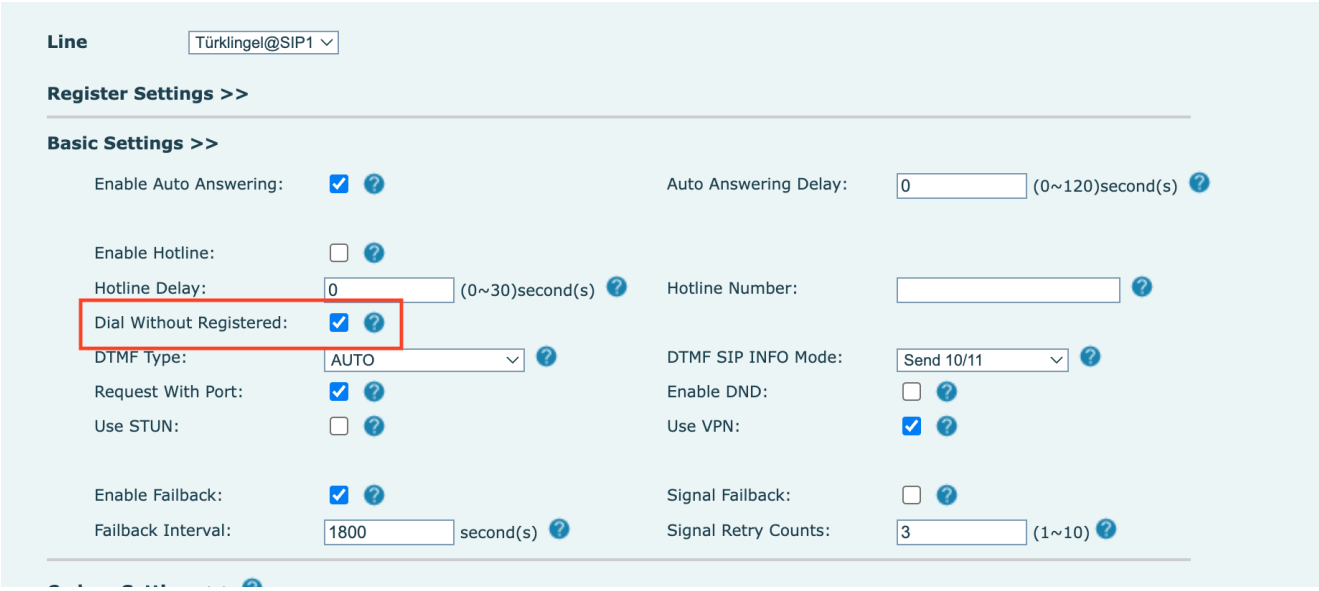
Update the firmware. Tested with 2.12.48.14

SETTINGS IN THE WEB INTERFACE

Username/Display: Can be freely selected
Server Address: The IP address of the Symcon server



Dial Without Registered: Activated



FUNCTION KEY SETTINGS

Type: Memory Key
Name: Can be freely selected
Value: User name in the Door Intercom instance

Function Key Settings >>

Key	Type	Name	Value			Subtype	Line	Media
DSS Key 1	Memory Key	Symcon	symcon_56855	+	-	Speed Dial	Türklingel@SIP	DEFAULT
DSS Key 2	None			+	-	None	AUTO	DEFAULT

Apply

Programmable Key Settings >>

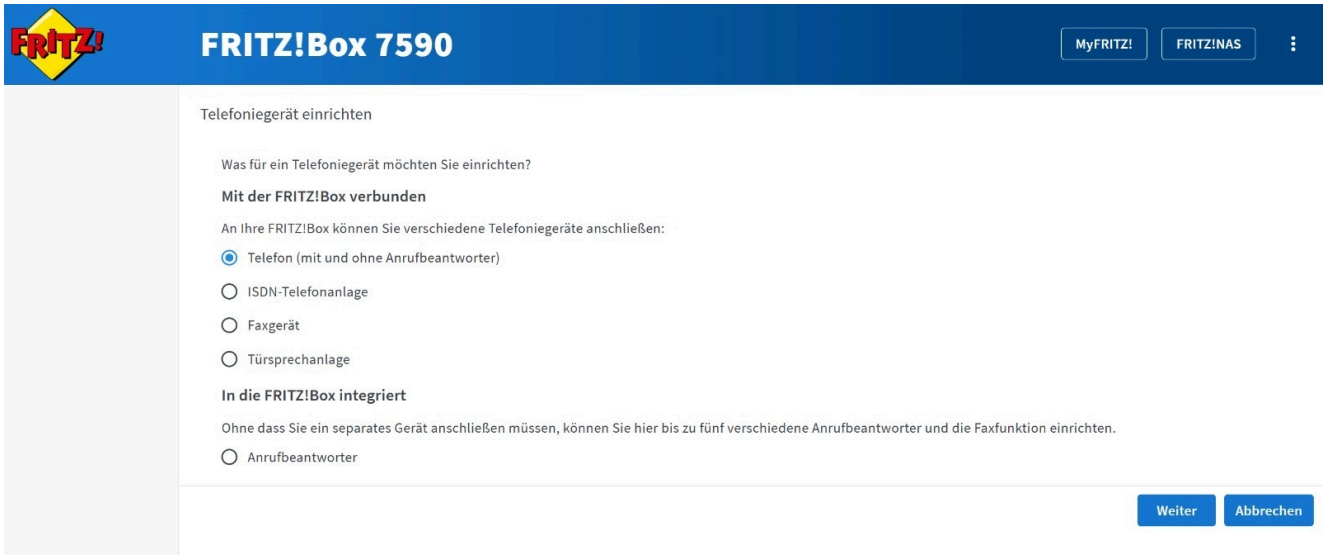
Advanced Settings >>

FRITZ!Box

Telephones registered in the Fritzbox can call Symcon. Only audio transmission is supported.

SETTINGS IN THE FRITZ!BOX

A new telephony device with the following settings must be added:

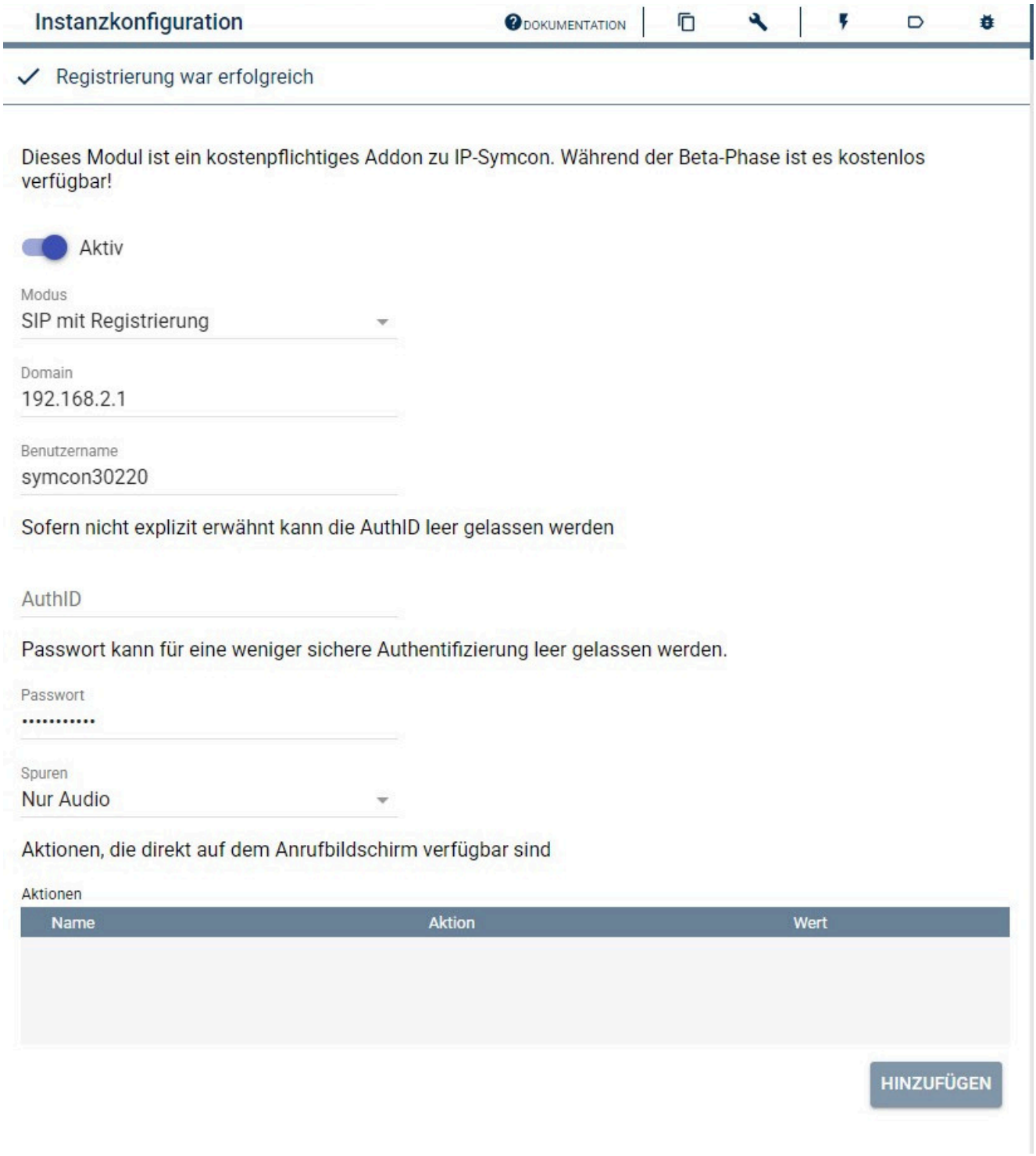




The user name corresponds to that in the Door Intercom instance.



SETTINGS IN THE DOOR INTERCOM INSTANCE


Instanzkonfiguration
DOKUMENTATION
Registrierung war erfolgreich
Dieses Modul ist ein kostenpflichtiges Addon zu IP-Symcon. Während der Beta-Phase ist es kostenlos verfügbar!
Aktiv
Modus
SIP mit Registrierung
Domain
192.168.2.1
Benutzername
symcon30220
Sofern nicht explizit erwähnt kann die AuthID leer gelassen werden
AuthID
Passwort kann für eine weniger sichere Authentifizierung leer gelassen werden.
Passwort
Spuren
Nur Audio
Aktionen, die direkt auf dem Anrufbildschirm verfügbar sind
Aktionen
Name
Aktion
Wert
HINZUFÜGEN

DoorIP_Accept

Require: Symcon >= 8.0

```
void DoorIP_Accept (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance of the device to be switched

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

Accepts the current call to the instance with the ID InstanceID.

EXAMPLE

```
DoorIP_Accept(12345);
```

DoorIP_Decline

Require: Symcon >= 8.0

```
void DoorIP_Accept (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance of the device to be switched

RETURNS

If the command could be executed successfully, the result is **TRUE**, otherwise **FALSE**.

DESCRIPTION

Rejects the current call to the instance with the ID **InstanceID**.

EXAMPLE

```
DoorIP_Decline(12345);
```

DoorIP_SendDTMF

Require: Symcon >= 8.0

```
void DoorIP_SendDTMF (int $InstanceID, string $Digits)
```

PARAMETERS

InstanceID	ID of the instance of the device to be switched
Digits	Characters to be sent as DTMF  Only these characters are allowed: 0-9, *, #

RETURNS

If the command could be executed successfully, the result is **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the characters **Digits** to the current connection on the door intercom instance with the ID **InstanceID**.

EXAMPLE

```
DoorIP_SendDTMF(12345, '*123#');
```

Voice over IP

The VoIP module can set up/receive telephone connections, process DTMF signals, buffer connection-related data and play sounds.



The following devices are supported by IP-Symcon:
Supported devices

REQUIREMENTS

An active portal for telephony is required. This can be a Fritzbox or a service like Sipgate.

INTEGRATION IN IP-SYMCON

For integration, a "VoIP" device can be added via Add object in the object tree.
Domain and user name for the telephone service must be specified in the instance configuration.
If required, AuthID and password must also be entered.



If the computer has several network cards or -interfaces (e.g. also virtual adapters such as VPN, Docker, Hyper-V) it may be necessary to define the IP address via which the VoIP connection is established. There is also the appropriate special switch "VoIPInterface"



To manage what should happen when a connection is active, a processing script must be added.

PROCESSING SCRIPT

The processing script determines the behavior for each incoming connection. The Systemvariables are also important for this.

EXAMPLES

SCRIPT FOR INCOMING CALLS (PROCESSING SCRIPT)

```
if($_IPS['SENDER'] == "VoIP") {
// Only incoming calls are to be processed
// $_IPS["INSTANCE"] is available since IP-Symcon 5.4
if(VoIP_GetConnection($_IPS["INSTANCE"], $_IPS["CONNECTION"])
["Direction"] == 1 /* Outbound */) {
return;
}
switch($_IPS["EVENT"]) {
case "Incoming":
IPS_LogMessage("VoIP", "An incoming call");
break;
case "Connect":
IPS_LogMessage("VoIP", "A connection was established");
break;

case "Disconnect":
IPS_LogMessage("VoIP", "A connection was terminated");
break;
case "DTMF":
IPS_LogMessage("VoIP", "A DTMF signal was received");

switch($_IPS["DATA"]) {
case '1':
case '2':
case '3':
case '4':
case '5':
case '6':
IPS_LogMessage("VoIP", "One of the keys 1 to 6 was pressed");
break;
case '#':
IPS_LogMessage("VoIP", "The # key was pressed");
break;
default:
IPS_LogMessage("VoIP", "The ". $_IPS["DATA"] ." button was pressed");
break;
}
break;
case "PlayFinish":
IPS_LogMessage("VoIP", "A sound file was played");
break;
default:
IPS_LogMessage("VoIP", "An unknown event was triggered");
break;
}
}
```

OUTBOUND CALL SCRIPT WITH TTS MODULE "AWS POLLY" FROM THE MODULE STORE

```
if($_IPS['SENDER'] == "Execute") {
$id = VoIP_Connect(12345, "0451305005xx");
//Wait a maximum of 10 seconds for someone to pick up
for($i = 0; $i &lt; 10; $i++) {
IPS_Sleep(1000);
$c = VoIP_GetConnection(12345, $id);
if($c['Connected']) {
// VoIP_Playwave() only supports WAV in the format: 16 Bit, 8000 Hz,
Mono.
VoIP_PlayWave(12345, $id, TTSAWSPOLLY_GenerateFile(23456, "IP-Symcon
wishes you a wonderful day"));
return;
}
}
//Hang up if nobody picks up
VoIP_Disconnect(12345, $id);
}
```

EXAMPLE SETUP ON THE FRITZBOX

The VoIP module can be used with the help of the FritzBox. A separate unused telephone number in the FritzBox is recommended for this and the following steps must be kept in mind.

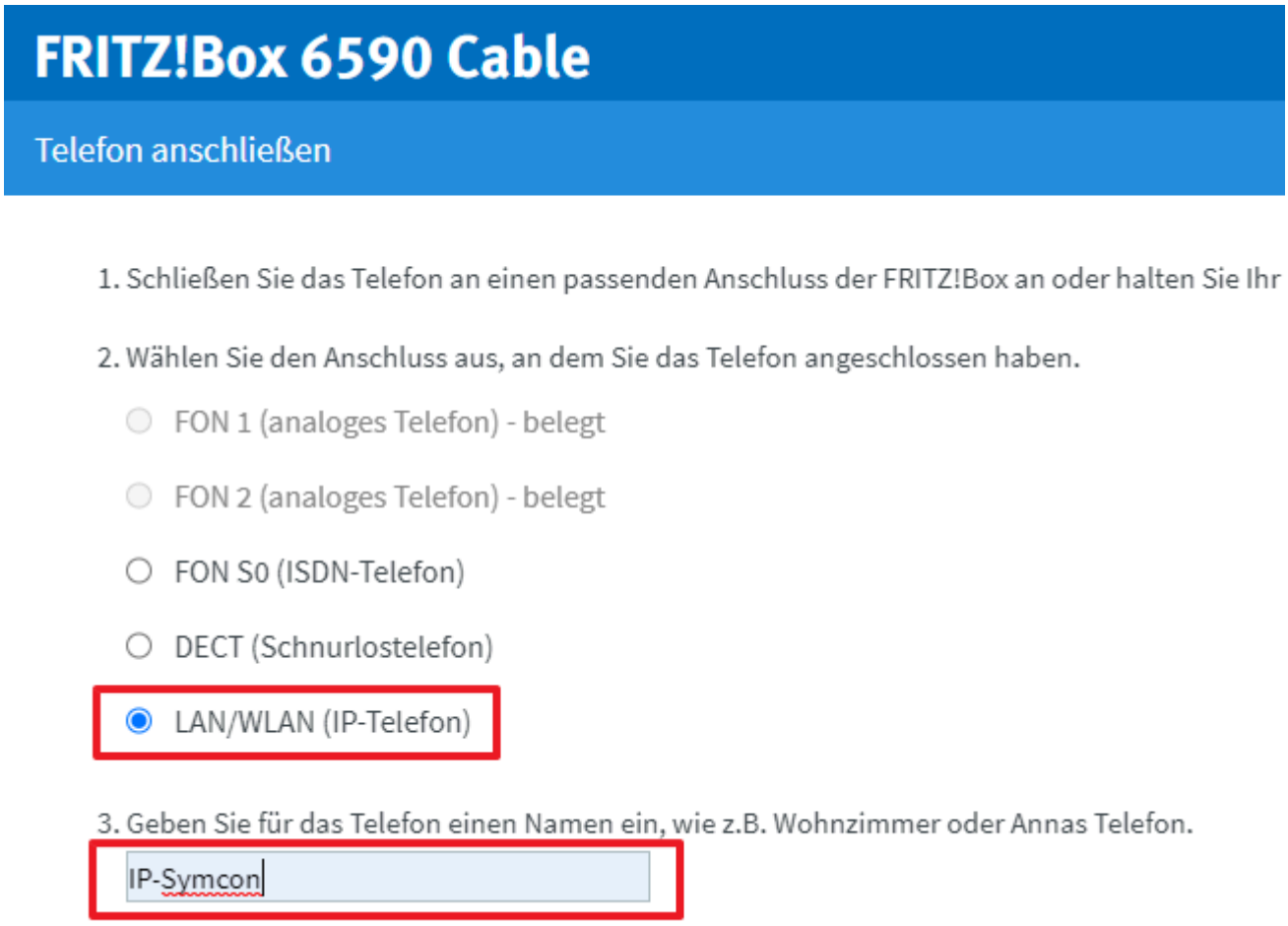


If necessary, a new number must be set up under "Telephony" -> "Own number" -> "New number"

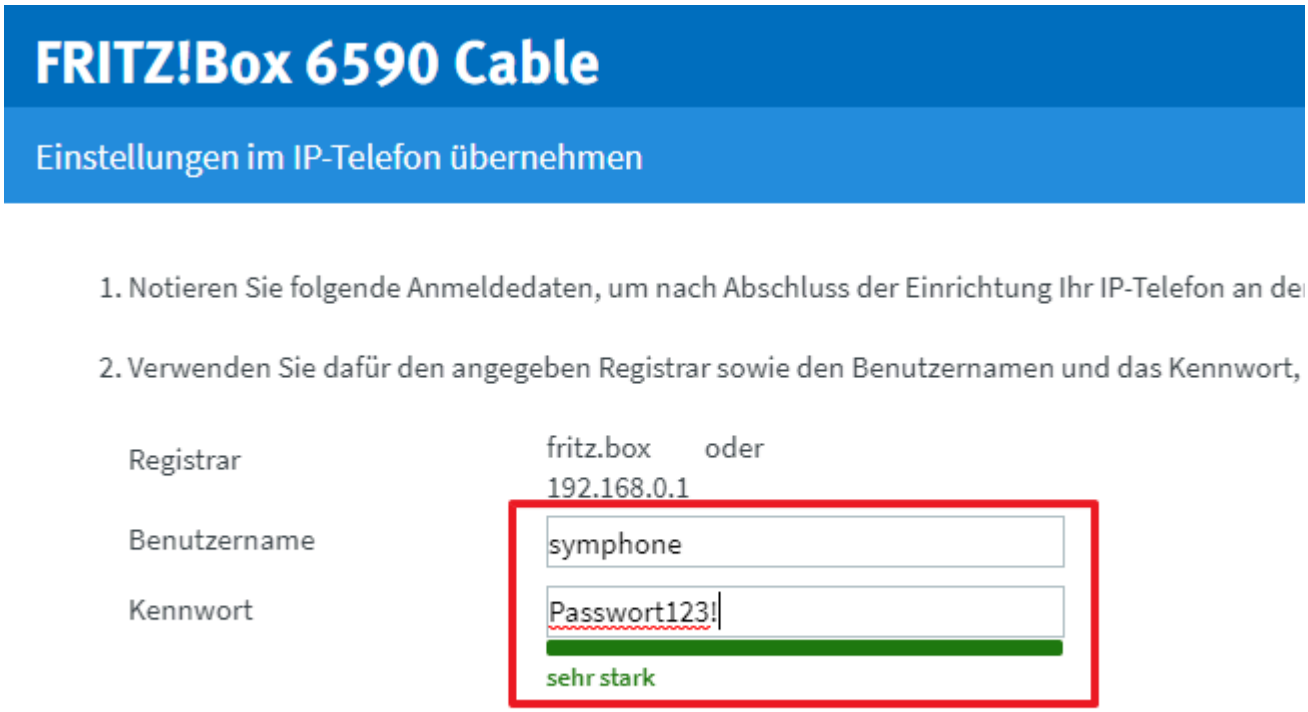
As a first step, a new telephony device must be added to the FritzBox under "Telephony" -> "Telephony Devices".
This must be added as a new "phone".



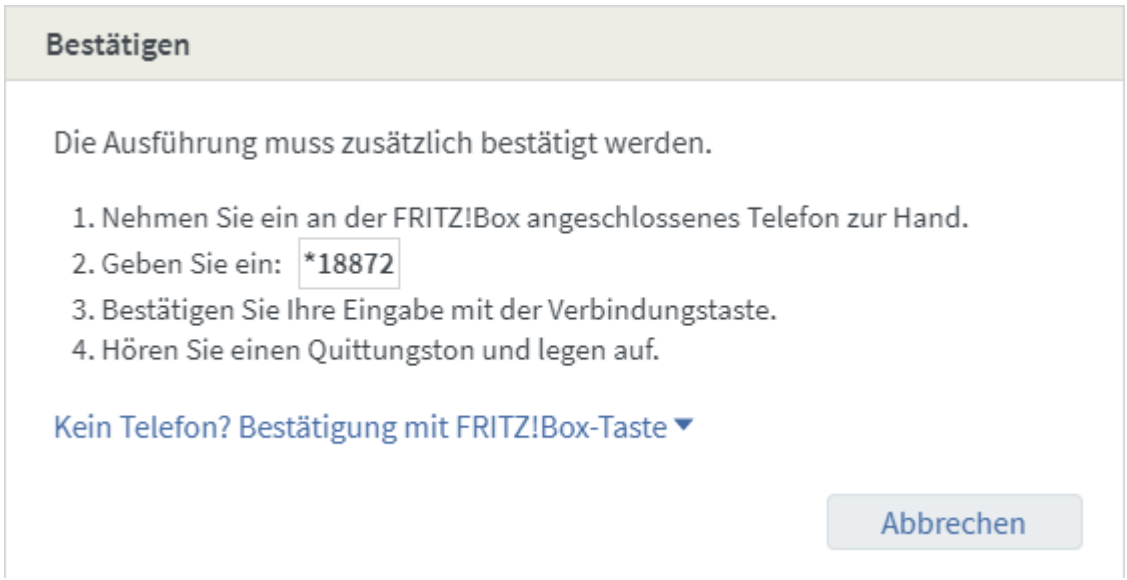
Since IP-Symcon acts as an IP telephone, this and then a separate telephone number must be selected. A descriptive name should be chosen for the telephony device.

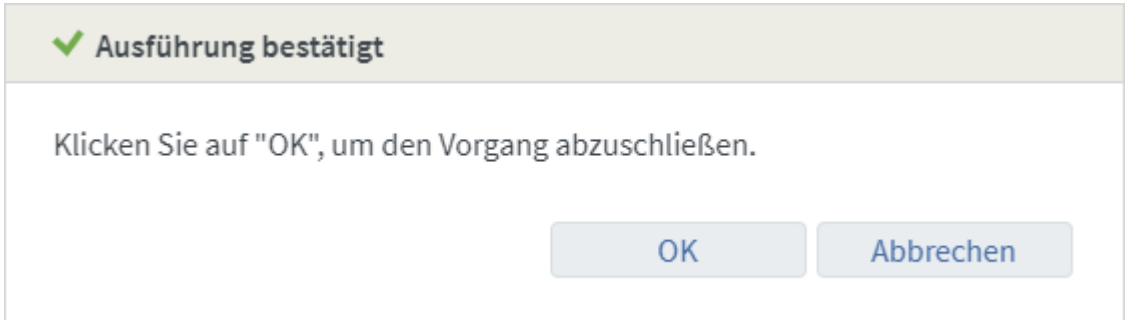


Then a user name and password must be specified, these are later entered into the VoIP in IP-Symcon.

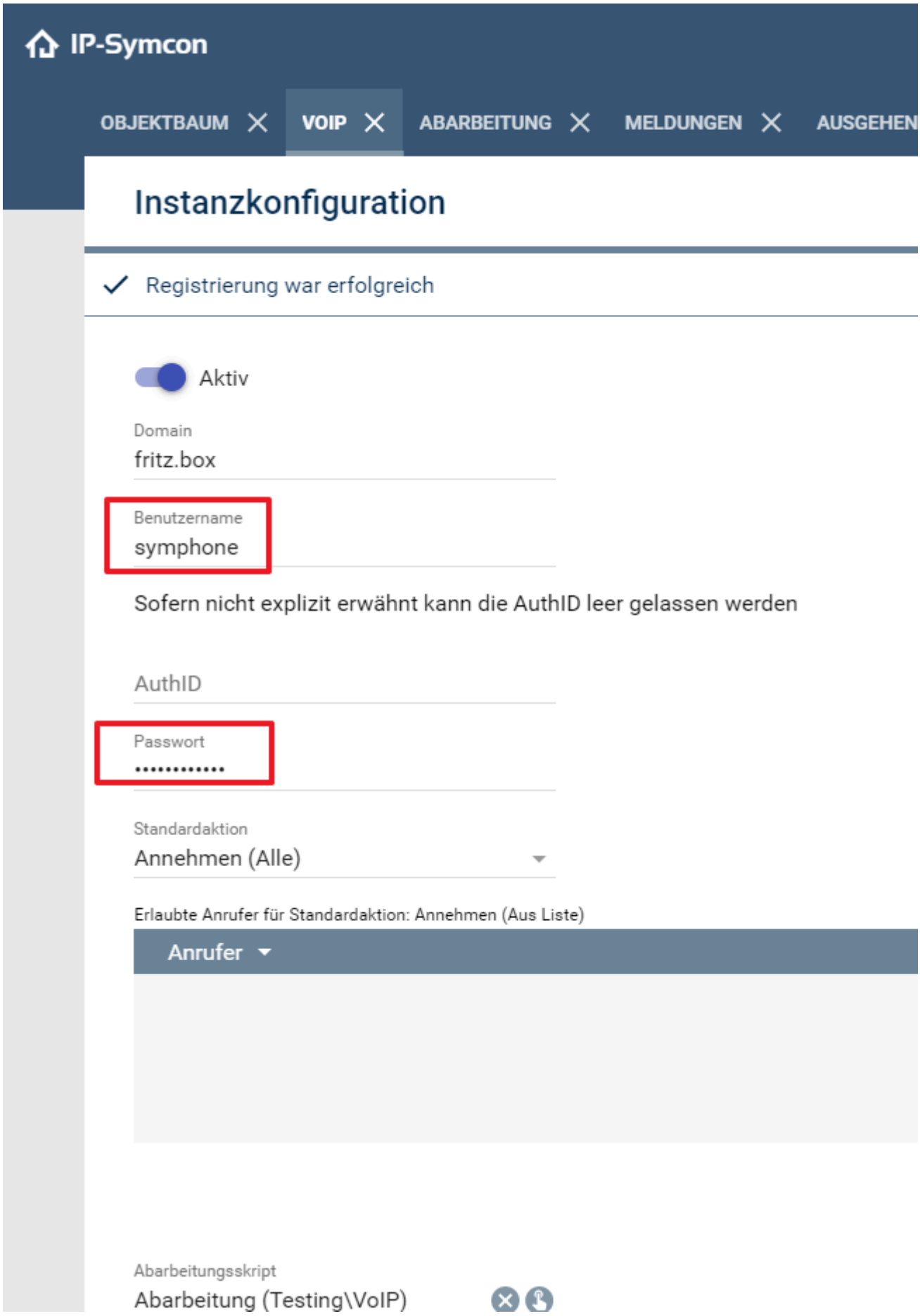


The telephony device to be set up must be confirmed and its setup then completed.





Now the telephony device can be taken over in IP-Symcon within the VoIP module.
If everything went well, the status "Registration was successful" is displayed.



A test call can be made with the Processing-Script set up. In the Messages the following messages should appear when the number keys on the phone are pressed.

OBJEKTBAUM ✕ VOIP ✕ ABARBEITUNG ✕ MELDUNGEN ✕ AUSGEHEND

Meldungen

Zeitpunkt	Absender	Nachricht
20.07.2020, 16:16:11	VoIP	Ein eingehender Anruf
20.07.2020, 16:16:11	VoIP	Es wurde eine Verbindung aufgebaut
20.07.2020, 16:16:15	VoIP	Es wurde ein DTMF Signal empfangen
20.07.2020, 16:16:15	VoIP	Es wurde eine der Tasten 1 bis 6 gedrückt
20.07.2020, 16:16:16	VoIP	Es wurde ein DTMF Signal empfangen
20.07.2020, 16:16:16	VoIP	Es wurde eine der Tasten 1 bis 6 gedrückt
20.07.2020, 16:16:17	VoIP	Es wurde ein DTMF Signal empfangen
20.07.2020, 16:16:17	VoIP	Es wurde eine der Tasten 1 bis 6 gedrückt
20.07.2020, 16:16:18	VoIP	Es wurde ein DTMF Signal empfangen
20.07.2020, 16:16:18	VoIP	Es wurde die Taste 7 gedrückt
20.07.2020, 16:16:19	VoIP	Es wurde ein DTMF Signal empfangen
20.07.2020, 16:16:19	VoIP	Es wurde die Taste 8 gedrückt
20.07.2020, 16:16:20	VoIP	Es wurde ein DTMF Signal empfangen
20.07.2020, 16:16:20	VoIP	Es wurde die Taste 9 gedrückt
20.07.2020, 16:16:20	VoIP	Es wurde ein DTMF Signal empfangen
20.07.2020, 16:16:20	VoIP	Es wurde die Taste 9 gedrückt
20.07.2020, 16:16:27	VoIP	Es wurde eine Verbindung beendet

Device list

SUPPORTED COMPONENTS

An excerpt of supported components.
- Fritzbox
- Sipgate



These are only the tested devices. Other devices may also work.

VoIP_AcceptCall

Require: Symcon >= 5.4

```
bool VoIP_AcceptCall (int $InstanceID, int $ConnectionID)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

For the VoIP instance with ID InstanceID, accepts the connection with ID ConnectionID.

EXAMPLE

```
// Accepts the connection with ID 3 of the instance with ID 12345
VoIP_AcceptCall(12345, 3)
```

VoIP_Connect

Require: Symcon >= 5.2

```
int VoIP_Connect (int $InstanceID, string $Number)
```

PARAMETERS

InstanceID	ID of the device to be switched
Number	Phone number to connect to

RETURNS

Returns the ID of the connection

DESCRIPTION

Establishes a connection with the number **Number** at the VoIP instance with the ID **InstanceID**.

EXAMPLE

```
// Establishes a connection with the phone number 0451305005xx
VoIP_Connect(12345, "0451305005xx");
```

VoIP_Disconnect

Require: Symcon >= 5.2

```
bool VoIP_Disconnect (int $InstanceID, int $ConnectionID)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Disconnects the VoIP instance with ID **InstanceID** from the connection with ID **ConnectionID**.

EXAMPLE

```
// Disconnects with ID 3 of the instance with ID 12345
VoIP_Disconnect(12345, 3);
```

VoIP_GetConnection

Require: Symcon >= 5.2

```
array VoIP_GetConnection (int $InstanceID, int $ConnectionID)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched

RETURNS

Array with the following information

FIELD	DATA TYPE	DESCRIPTION
ID	integer	ConnectionID
TimeStamp	integer	Time stamp of the call
Number	string	Number with which a connection was established
Direction	integer	Call direction: 0 = Incoming call, 1 = Outgoing call
Connected	boolean	Connection status: FALSE = not connected, TRUE = connected
Disconnected	boolean	Disconnection status: FALSE = not disconnected, TRUE = disconnected

DESCRIPTION

Returns the information of the connection with ID **ConnectionID** of the VoIP instance with ID **InstanceID**.
---return---
Array with the following information

FIELD	DATA TYPE	DESCRIPTION
ID	integer	ConnectionID
TimeStamp	integer	Time stamp of the c

EXAMPLE

```
// Returns information of the connection with ID 3.
print_r(VoIP_GetConnection(12345, 3));

// Sample output:
/*
Array
(
[ID] => 3
[TimeStamp] => 1566915689
[Number] => 045130500511
[Data] =>
[Direction] => 1
[Connected] => 1
[Disconnected] =>
)
*/
```

VoIP_GetData

Require: Symcon >= 5.2

```
string VoIP_GetData (int $InstanceID, int $ConnectionID)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched

RETURNS

Content of the connection data as a string

DESCRIPTION

Returns the content of the data of the connection with ID **ConnectionID** of the VoIP instance with ID **InstanceID**.

EXAMPLE

```
// Returns the content of the data of the connection with ID 3.
VoIP_SetData(12345, 3, "Hello World");
print_r(VoIP_GetData(12345, 3));

// Sample output
/*
Hello World
*/
```

VoIP_PlayWave

Require: Symcon >= 5.2

```
bool VoIP_PlayWave (int $InstanceID, int $ConnectionID, string
$Filename)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched
Filename	Path and name of the file to be played

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Plays the sound file **Filename** on the connection with the ID **ConnectionID** on the VoIP instance with the ID **InstanceID**.



Exclusively data in WAV format: 16-bit, 8000 Hz, mono is supported.



It does not wait for the sound file to finish playing. This is handled by the $_IPS['EVENT'] == "PlayFinish".



Dynamic sound files can also be created using the TTS Module or AWS Polly from the Module Store.

EXAMPLE

```
// Plays the sound file "welcome.wav" on the connection with ID 3
```

```
VoIP_PlayWave(12345, 3, IPS_GetKernelDir() . "/media/welcome.wav");
```

VoIP_RejectCall

Require: Symcon >= 5.4

```
bool VoIP_RejectCall (int $InstanceID, int $ConnectionID)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Rejects the connection with ID **ConnectionID** for the VoIP instance with ID **InstanceID**.

EXAMPLE

```
// Rejects the connection with the ID 3 of the instance with the ID
12345
VoIP_RejectCall(12345, 3)
```

VoIP_SendDTMF

Require: Symcon >= 5.3

```
bool VoIP_SendDTMF (int $InstanceID, int $ConnectionID, string $DTMF)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched
DTMF	Characters to be sent as DTMF

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the characters **DTMF** on the connection with the ID **ConnectionID** on the VoIP instance with the ID **InstanceID**.



Only these characters are allowed: 0-9, *, #

EXAMPLE

```
// Sends the DTMF signals for "*123#" on the connection with ID 3
VoIP_SendDTMF(12345, 3, "*123#");
```

VoIP_SetData

Require: Symcon >= 5.2

```
bool VoIP_SetData (int $InstanceID, int $ConnectionID, string $Data)
```

PARAMETERS

InstanceID	ID of the device to be switched
ConnectionID	ID of the connection to be switched
Data	Data for which connection is to be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the data **Data** of the connection with the ID **ConnectionID** on the VoIP instance with the ID **InstanceID**.

EXAMPLE

```
// Sets the content of the data of the connection with ID 3 to "Hello
World"
VoIP_SetData(12345, 3, "Hello World");
```

Weishaupt

Require: Symcon >= 7.0
Weishaupt offers various heat pumps. These can be read out via ModBus TCP. A connection with Symcon is possible via a MobBus gateway from Weishaupt.

List of supported components

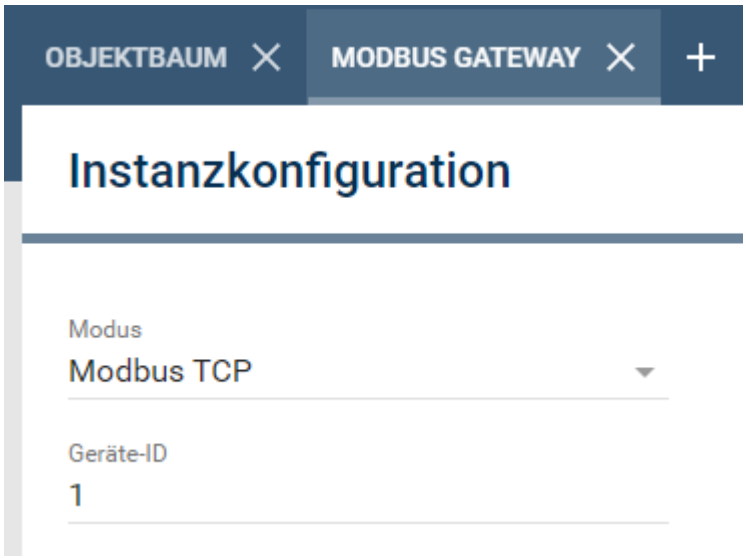
INSTALLATION

To be able to use Weishaupt heat pumps in Symcon, there must be a connection to the heat pump via Ethernet or Wifi.

SYMCON INTEGRATION

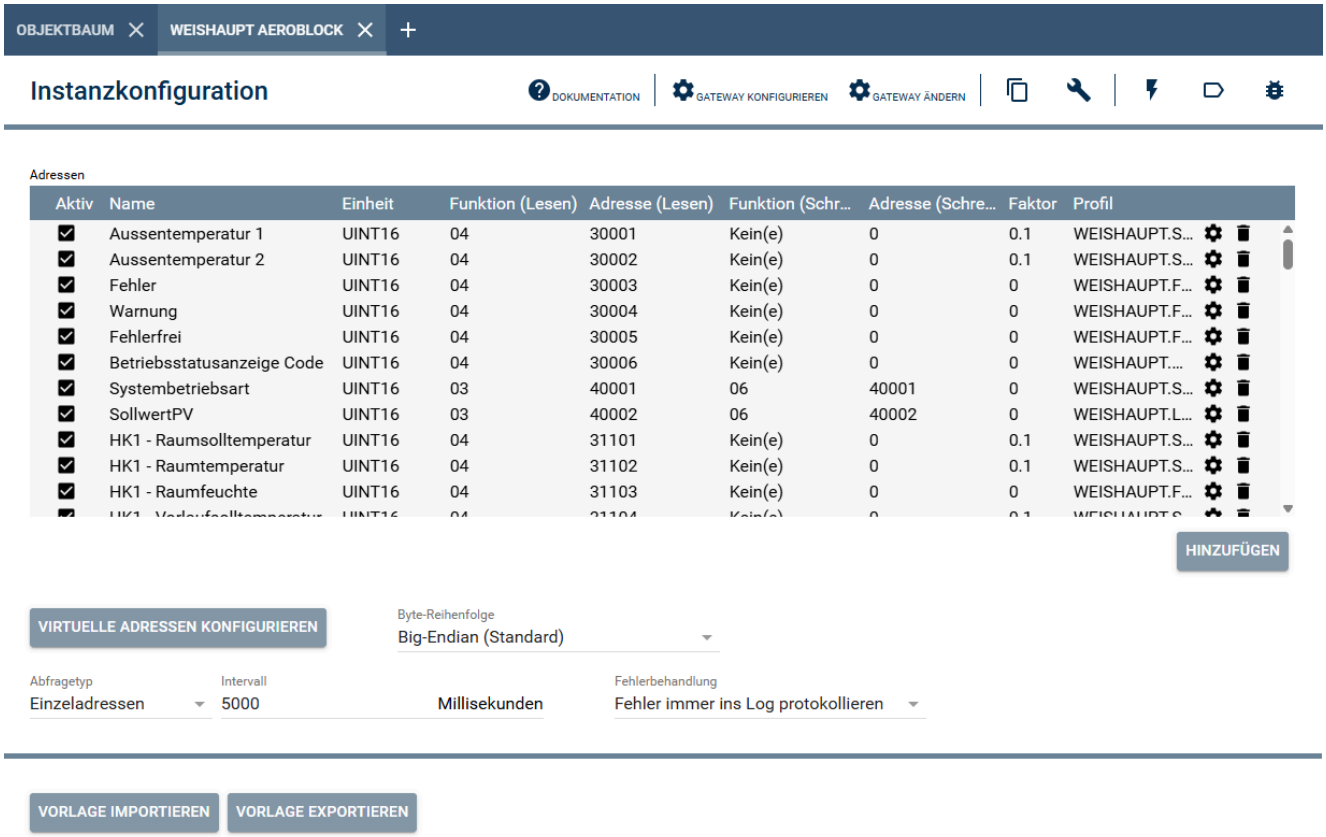
First, a "ModBus Device" instance must be added within the Symcon object tree. The IP address of the heat pump must be entered in the following dialog. The port is 502 by default. Both pieces of information can be viewed on the web interface of the heat pump.





The ModBus template for Weishaupt heat pumps can then be downloaded. This contains the entire configuration of the ModBus device. After downloading, the "WeishauptModBusx_vx.json" can be loaded via "Import". All ModBus addresses of the standard Weishaupt heat pump are then set up.

Templates in the Device overview



ADDING ADDRESSES

If further addresses are to be added, this can be done via "Add".
Depending on the design of the wallbox, individual addresses may need to be deactivated/activated. This can be controlled via the Active column.

PROTOCOL DESCRIPTION



Protocol description in the Device overview

Device List

Require: Symcon >= 7.0

SUPPORTED GATEWAYS

PRODUCT	DESCRIPTION
Modbus TCP (WWP)	IP Gateway for ModBus TCP

SUPPORTED DEVICES

Protocol description of all data points.

PRODUCT	DESCRIPTION	MODBUS TEMPLATE
Aeroblock (WAB)	Air/water heat pump	Download
Biblock (WBB)	Air/water heat pump	Download
Splitblock (WSB)	Air/water heat pump	Download
Geoblock (WGB)	Brine/water heat pump	Download

WinLIRC

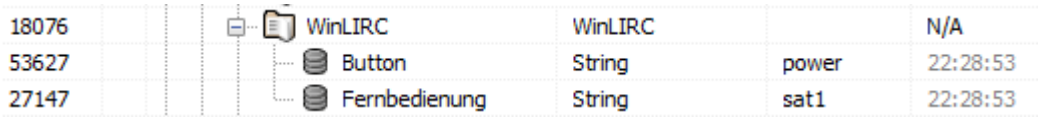
Using infrared signals allows an inexpensive and easy integration of existing audio and video equipment into home automation. Moreover, e.g. light scenarios using the infrared remote control can be activated or blinds can be lowered. For this, in principle, only one transmitter and receiver diode is necessary - as you want as a finished device or for crafting ...
Find out more about WinLIRC:
winlirc.sourceforge.net
For those who like to pick up once again the soldering iron, find more information here:
lirc.org/receivers.html
lirc.org/transmitters.html

INSTALLATION

To receive/send IR signals via IP Symcon, you need a LIRC compatible device.
WinLIRC Applikation must be installed and running
Download: winlirc.sourceforge.net/de/
Remote control/buttons must be trained in the WinLIRC application
Create WinLIRC module in IP-Symcon.
Host: localhost
Port: 8765

USE IN IP SYMCON

The WinLIRC module providess itself in the object tree as follows:



A successful installation can be seen at the open socket.



To output IR commands from IP-Symcon, the WinLIRC_SendOnce function is used.

```
$id = 18076 /*[Media IR\WinLIRC]*/;
// Turn on ZDF
WinLIRC_SendOnce($id, "humax", "power");
IPS_Sleep(1000);
WinLIRC_SendOnce($id, "humax", "power");
IPS_Sleep(200);
WinLIRC_SendOnce($id, "sat", "power");
// here is still place for lighting commands...
```

WinLIRC_SendOnce

```
bool WinLIRC_SendOnce (int $InstanceID, string $RemoteControl, string $Button)
```

PARAMETERS

InstanceID	ID of the device to be switched
RemoteControl	Name of the remote control, which is/ was registered in the remote database.
Button	Name of the remote control function, which is/ was registered in the remote database.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a remote command on the instance with ID **InstanceID**. The values remote control and button specify, which command should be sent. These must be set up accordingly in the client software (WinLIRC/ IRTrans Tray).

EXAMPLE

```
WinLIRC_SendOnce(37279,"soundmaster", "power");
```

Wireless M-Bus

Wireless M-Bus is a protocol standardized by the OMS Group for recording consumption data. A LAN, serial or USB gateway is required to connect to IP-Symcon.



The following devices are supported by IP-Symcon:
Supported devices

CONTENT

Installation
Integration in IP-Symcon
Configurator
Configuration

INSTALLATION

The **Wireless M-Bus LAN Gateway** is connected to the PC via LAN. DHCP and port 5000 are set up by default. If an individual IP address is required, the IP-Symcon "Network Configuration Tool" is needed. This is available for Download. A simple Description for configuring the gateway is available. The gateway can then be accessed via the configured IP address and port.

INTEGRATION IN IP-SYMCON

The LAN gateway can be integrated via the device search. To do this, "Wireless M-Bus Discovery" must be selected as the system. The Discovery instance then offers the creation of a Wireless M-Bus Configurator. Once the configurator has been created, all received devices that are integrated are shown.

CONFIGURATOR

The configurator displays all received devices in the public information. Information such as manufacturer, device type, encryption and the last received date are displayed in the respective columns. The received raw data can be viewed in the Data column so that it can be sent to Symcon Support if necessary.

Devices

Name	ID	Manufactu...	Version	Medium	Status	Encryption	RSSI	Data	InstanceID
Wireless M-Bus Device (76102435, DME, Heat)	76102435	DME	0	Heat	112		N/A		#59419
Wireless M-Bus Device (02355964, EFE, Heat)	02355964	EFE	0	Heat	0	OMS (Mode 5)	-43dbm		None
Wireless M-Bus Device (20119827, DWZ, Water)	20119827	DWZ	2	Water	0	OMS (Mode 5)	-40dbm		None
Wireless M-Bus Device (20844457, KAM, Cold Water)	20844457	KAM	29	Cold Water	16	BSI (Mode 7)	-43dbm		None
Wireless M-Bus Device (22160253, ITW, Water)	22160253	ITW	0	Water	0	OMS (Mode 5)	-43dbm		None
Wireless M-Bus Device (23160294, ITW, Heat)	23160294	ITW	0	Heat	19	OMS (Mode 5)	-37dbm		None
Wireless M-Bus Device (42069120, DME, Gas)	42069120	DME	48	Gas	179	OMS (Mode 5)	-44dbm		None
Wireless M-Bus Device (43152155, EFE, Heat)	43152155	EFE	0	Heat	0	OMS (Mode 5)	-86dbm		None
Wireless M-Bus Device (43152169, EFE, Heat)	43152169	EFE	0	Heat	0	OMS (Mode 5)	-67dbm		None
Wireless M-Bus Device (43152173, EFE, Heat)	43152173	EFE	0	Heat	0	OMS (Mode 5)	-83dbm		None
Wireless M-Bus Device (43152177, EFE, Heat)	43152177	EFE	0	Heat	0	OMS (Mode 5)	-73dbm		None
Wireless M-Bus Device (43152178, EFE, Heat)	43152178	EFE	0	Heat	0	OMS (Mode 5)	-66dbm		None
Wireless M-Bus Device (43152186, EFE, Heat)	43152186	EFE	0	Heat	0	OMS (Mode 5)	-83dbm		None
Wireless M-Bus Device (71927603, DME, Water)	71927603	DME	118	Water	112	BSI (Mode 7)	-45dbm		None
Wireless M-Bus Device (83014314, KAM, Heat)	83014314	KAM	64	Heat	0	AES CTR	-38dbm		None

New instances are created in the object tree in the main category. These created instances can then be renamed accordingly and sorted elsewhere. It is also possible to call up the respective instance configuration via "Configure" in the configurator.

CONFIGURATION

The device key can be entered on the configuration page of the device. This must consist of 32 characters and may only contain capital letters/numbers without spaces. Alternatively, key lists can also be imported into the Wireless M-Bus Gateway instance, which are then used directly by the instance.

CONFIGURATION

Key lists can be imported into the Wireless M-Bus Gateway instance so that they are available to the Symcon system more quickly and automatically. The following formats are supported:

```
012345678;00000000000000000000000000000000 (Address;Key)
012345678;TESTTEST;00000000000000000000000000000000 (Any other columns)
SOX;012345678;00000000000000000000000000000000 (Any other columns)
012345678;00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (Spaces
may appear anywhere in the key)
012345678,00000000000000000000000000000000 (Comma and semicolon are
permitted as separators)
0SOX012345678;00000000000000000000000000000000 (Address in DIN format)
```

Support for the OMS SecProfile A format is planned.

Device list

SUPPORTED GATEWAYS

The following wireless M-Bus gateways are compatible:

MANUFACTURER	DESCRIPTION
SymBox with Wireless M-Bus extension Order now	Serial integrated module
Wireless M-Bus LAN Gateway Order now	IP Gateway
Würth Elektronik 2605056083001	METIS I, USB, AMB8465-M, integrated antenna
Würth Elektronik 2607056283001	METIS II, USB, AMB8665-M, external antenna
IMST iU891A-XL	New model with external antenna
IMST iM871-A	Old model
WEPTECH SWAN2	Over HTTP/HTTPS
WEPTECH SWAN3	Over HTTP/HTTPS

SUPPORTED COMPONENTS

IP-Symcon supports the following encryption modes according to DIN EN 13757-7
- Mode 0 (no encryption)
- Mode 5 (OMS)
- Mode 7 (BSI)

IP-Symcon supports the following encryption modes according to DIN EN 13757-4
- AES-128-CTR

IP-Symcon supports the frame format A/B and the receive mode C/T. In addition, the somewhat more specialized Compact Frames, which are often used by Kamstrup devices, are supported.

WMRS200

WMRS200 is a weather station system and uses radio-based sensor components. A connection to IP-Symcon is possible via a USB gateway.



The following devices are supported by IP-Symcon:
Supported components

INSTALLATION

A properly connected USB communication-hub is required for operation with IP-Symcon.

INTEGRATION IN IP-SYMCON

In the object tree, WMRS can be searched for via "+" -> "Add instance" or by right-clicking -> "Add object" -> Instance in the dialog that appears. The individual devices are available for selection.
Gateway and I/O instances are created automatically and all that needs to be done is selecting the correct USB connection in the gateway and the device type in the instance as well as entering the appropriate device ID. The device ID can be entered via the "Search" function within the instance configuration.

Device list

SUPPORTED GATEWAYS

GATEWAYS	MANUFACTURER	CONNECTION TYPE	DESCRIPTION
USB communication hub	Oregon Scientific	Serial/ USB	USB interface from the weather station bundle

SUPPORTED COMPONENTS

COMPONENT	MANUFACTURER	CONNECTION TYPE	DESCRIPTION
Wind sensor	Oregon Scientific	Wireless	From bundle of the weather station
Rain sensor	Oregon Scientific	Wireless	From bundle of the weather station
Air pressure sensor	Oregon Scientific	Wireless	From bundle of the weather station
Temperature/ Humidity	Oregon Scientific	Wireless	From bundle of the weather station
THGR800	Oregon Scientific	Wireless	Thermo-Hygro (3-channel) - Temperature/ Humidity
THGR810	Oregon Scientific	Wireless	Thermo-Hygro (10-channel) - Temperature/ Humidity
UVN800	Oregon Scientific	Wireless	Pyranometer - Remote-UV-Sensor

W&T

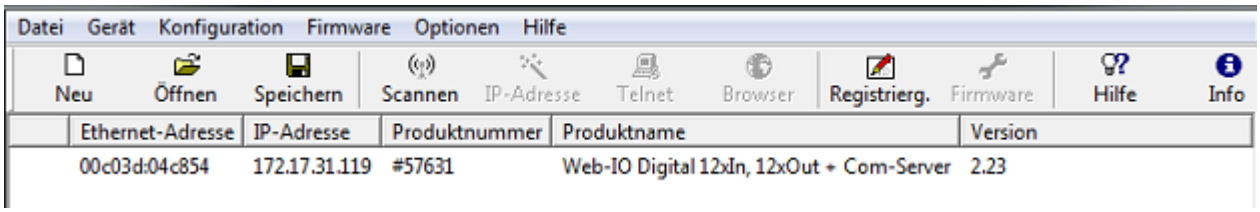
The W&T Web-IO Digital Boxes allow access via a variety of protocols and applications via a TCP/IP Ethernet 10/100BaseT interface.

INSTALLATION

A properly connected Web-IO and a DHCP server are required for operation with IP-Symcon. (The DHCP server automatically assigns the addresses for the network participants)
Initial configuration options and setting options (e.g. reassignment of the IP address) are possible using the **Wutility-Software**. If this is not available, it can be downloaded from www.wut.de . After installing the software, the Web-IO can be recognized and used.

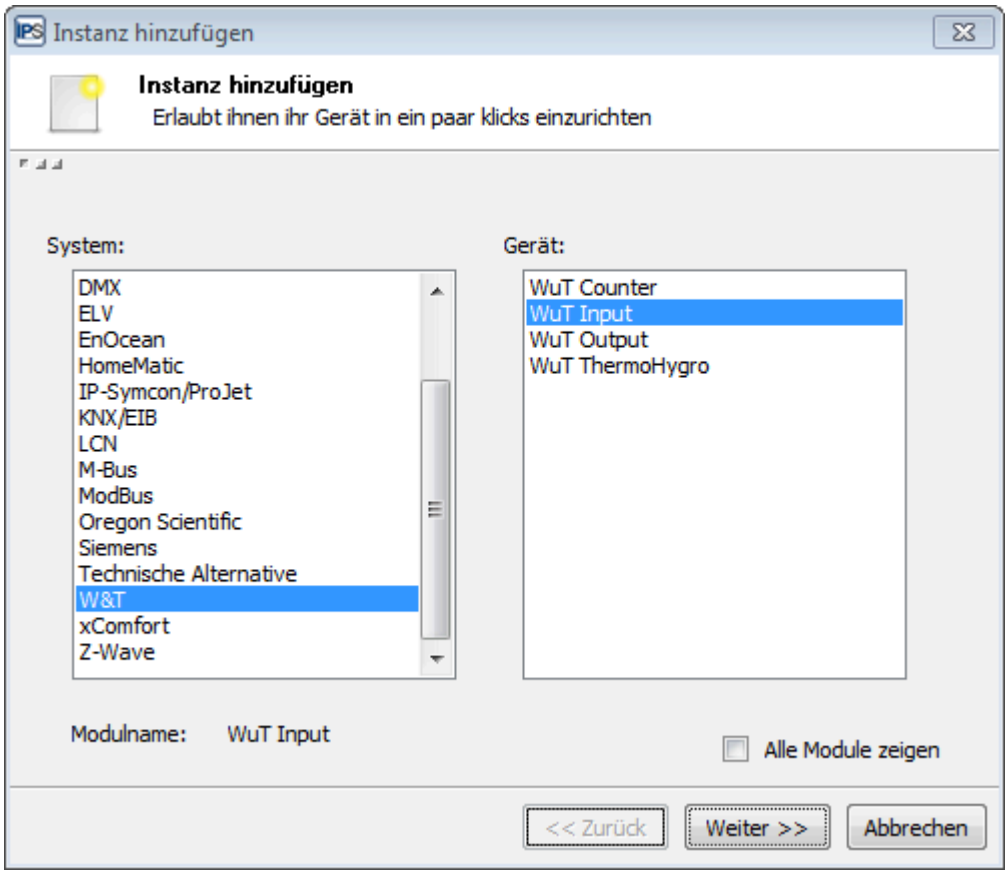


If no DHCP server is available, the IP address can also be configured with the Wutility-Software

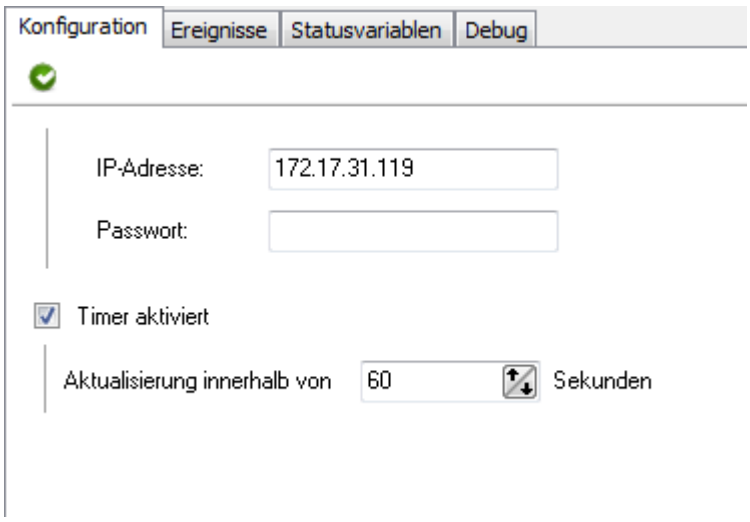


The configuration web page of the Web-IO can be called up in the browser by entering the IP. Initial settings, especially for test purposes, are possible here.

INTEGRATION IN IP-SYMCON



A function corresponding to the connected devices must be selected in IP-Symcon from counter, input and output. If several channels of the device are to be integrated, the step in the instructions can be repeated as often as necessary. Now, “Next” must be clicked on twice and “OK” once. For example, the configuration of the "WuT Output" opens. Settings for the channel of the device are possible here. The exact assignment of the channels can be found in the instructions for the W&T device.
The IP address of the Web-IO must then be entered on the configuration page. This can be opened by clicking on the cogwheel (bottom left).



A password is optional. The factory settings do not include a password.

The timer must be activated so that IP-Symcon receives new data from the Web-IO within a manually entered time and a change becomes visible.
(The described configuration of the Wut Gateway is only necessary for the first installation. This step can be skipped when adding more devices.)

WUT_SwitchMode

```
bool WUT_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the channel with ID InstanceID to value Status.

EXAMPLE

```
WUT_SwitchMode(12345, true); //Turn on channel
```

XBee

XBee is a module based on the ZigBee radio standard. It allows to serve as a radio bridge for hardwired systems.

INSTALLATION

To connect to IP-Symcon, an XBee coordinator (gateway) is required, which is connected via serial or serial USB.

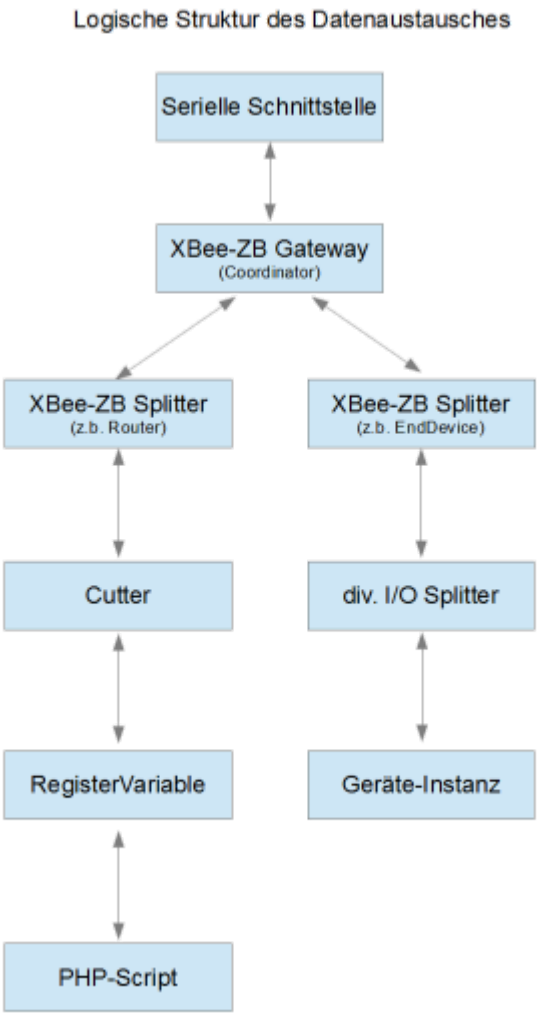
INTEGRATION IN IP-SYMCON

The XBee gateway (coordinator) is integrated into IP-Symcon via serial port.
An XBee Splitter instance must be created for each end device. This only contains the device ID for communication within XBee.
For the end-device gateways, the XBee splitter instances serve as a replacement parent instance. (See "Example object tree")

EXAMPLE

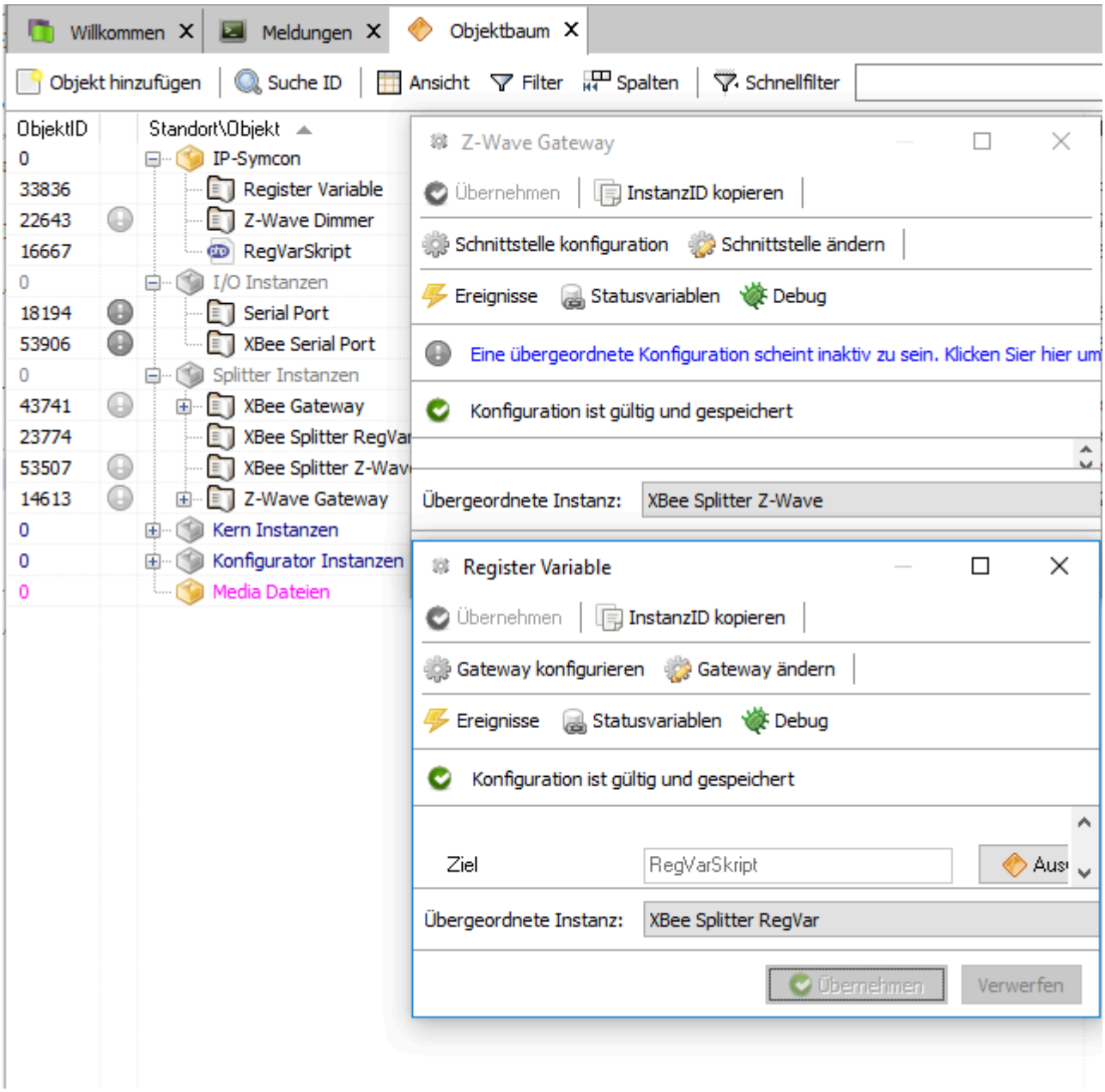
DATA FLOW

For a better understanding the abstract representation of the data flow can be seen here.



EXAMPLE OBJECT TREE

In this example, a Z-Wave gateway and a register variable are connected to a coordinator. XBee uses the DeviceID to ensure that the data from the register variable is made available in the associated script as System variables, as well as that the Z-Wave device data via the Z-Wave Gateway (parent instance: XBee Splitter Z-Wave) is updated.



In the contemporary example, this results in the following topology:

```
Serial Port -> XBee Gateway -> XBee Splitter Z-Wave -> Z-Wave Gateway -
> Z-Wave Dimmer
|
-> XBee Splitter RegVar -> Register Variable -> RegVarScript (System
variable)
```

XBee_SendBuffer

```
bool XBee_SendBuffer (int $InstanceID, int $TargetDevice, string
$Buffer)
```

PARAMETERS

InstanceID	ID of the device to query
TargetDevice	ID of the target device
Buffer	Data buffer to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the DataString **Buffer** to the target device **TargetDevice** and the XBee gateway with the ID **InstanceID**.

EXAMPLE

```
XBee_SendBuffer(12345, 2, "Any DataString");
```

XBee_SendCommand

```
bool XBee_SendCommand (int $InstanceID, string $Command)
```

PARAMETERS

InstanceID	ID of the device to be modified
Command	Command to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the command **Command** to the XBee splitter with the ID **InstanceID**.

EXAMPLE

```
XBee_SendCommand(12345, "Any CommandString");
```

Eaton xComfort

Eaton xComfort is a bidirectional radio-based system. To enable a setup in Symcon the configuration file of the installation or the programming interface (CRSZ-00/01) is required. The connection to Symcon is established with the communication interface CKOZ-00/03 or an Eaton USB or LAN gateway.



The following devices are supported by Symcon:
Supported devices

INSTALLATION

For connection to Symcon, Eaton (formerly Moeller) uses the **xComfort communication interface (CKOZ-00/03)**.
For programming the system in Eaton Comfort mode, the Eaton **xComfort programming interface (CRSZ-00/01)** is used.
Alternatively, it is also possible to connect an Eaton USB gateway or a LAN gateway for connection to Symcon.
The Eaton xComfort wireless system can be used to automatically control the monitoring of windows, air conditioning and lighting systems as well as many other building functions. The system can be retrofitted without the need to lay new cables. It is a bidirectional system, i.e. all sensors and actuators (switches) can communicate with each other, as they each contain a transmitter and a receiver unit.

VIDEO TUTORIAL FOR SETUP

PORT

In order to be able to connect the Eaton Xcomfort system with Symcon, a corresponding gateway is required. In this example this is the communication interface CKOZ-00/03. The programming interface CRZ-00/01 for the Comfort mode is not suitable for this, but is also needed, because it is needed for programming the system in Comfort mode from Eaton.

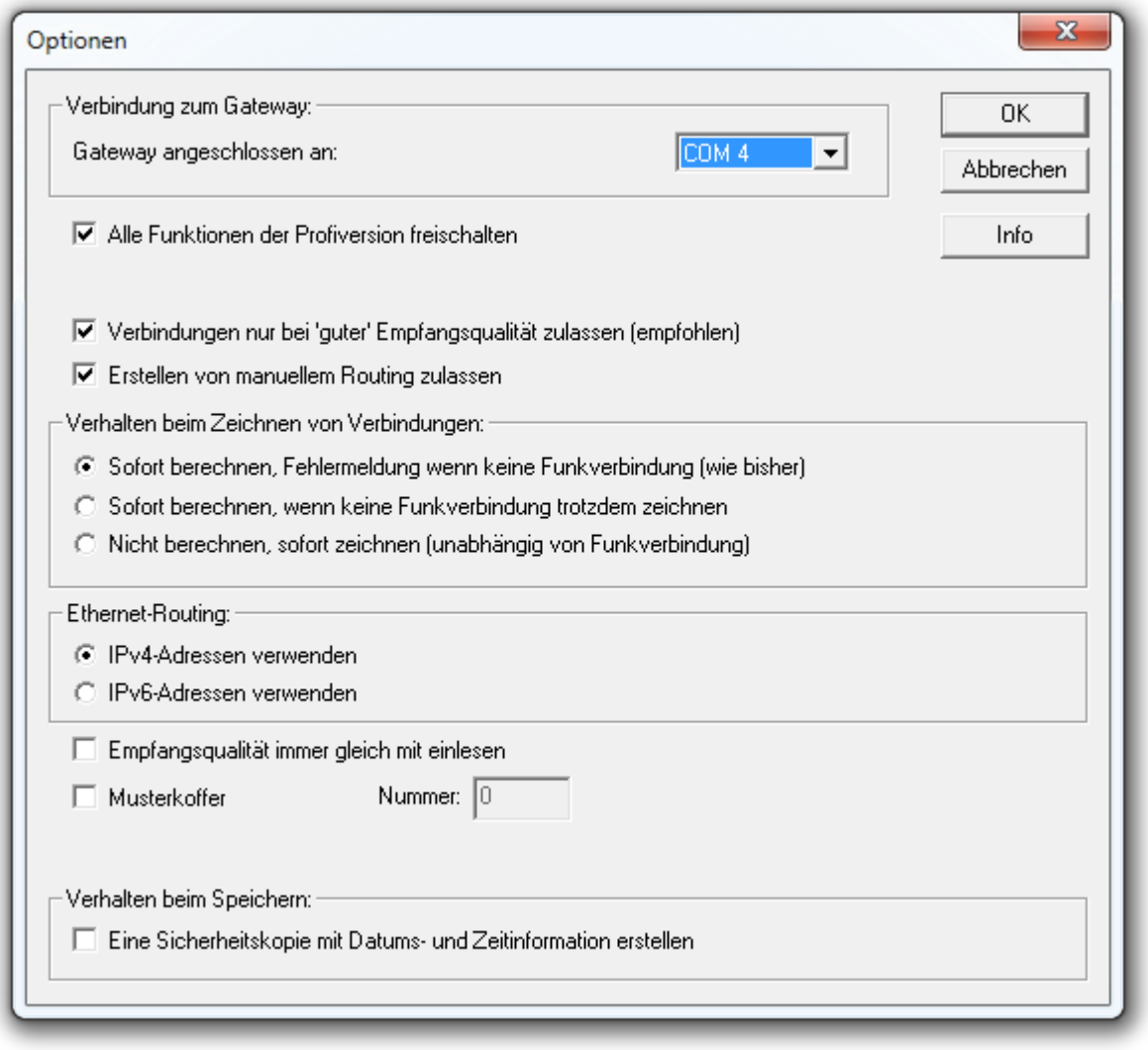


When using the RS232 port, please note that additional power must be supplied via the USB port. This is possible via a suitable adapter (230V->USB)

The programming interface CRZ-00/01 can also be connected via the USB interface (optional cable: RS232->USB) or via the RS232 port.

INSTALLATION

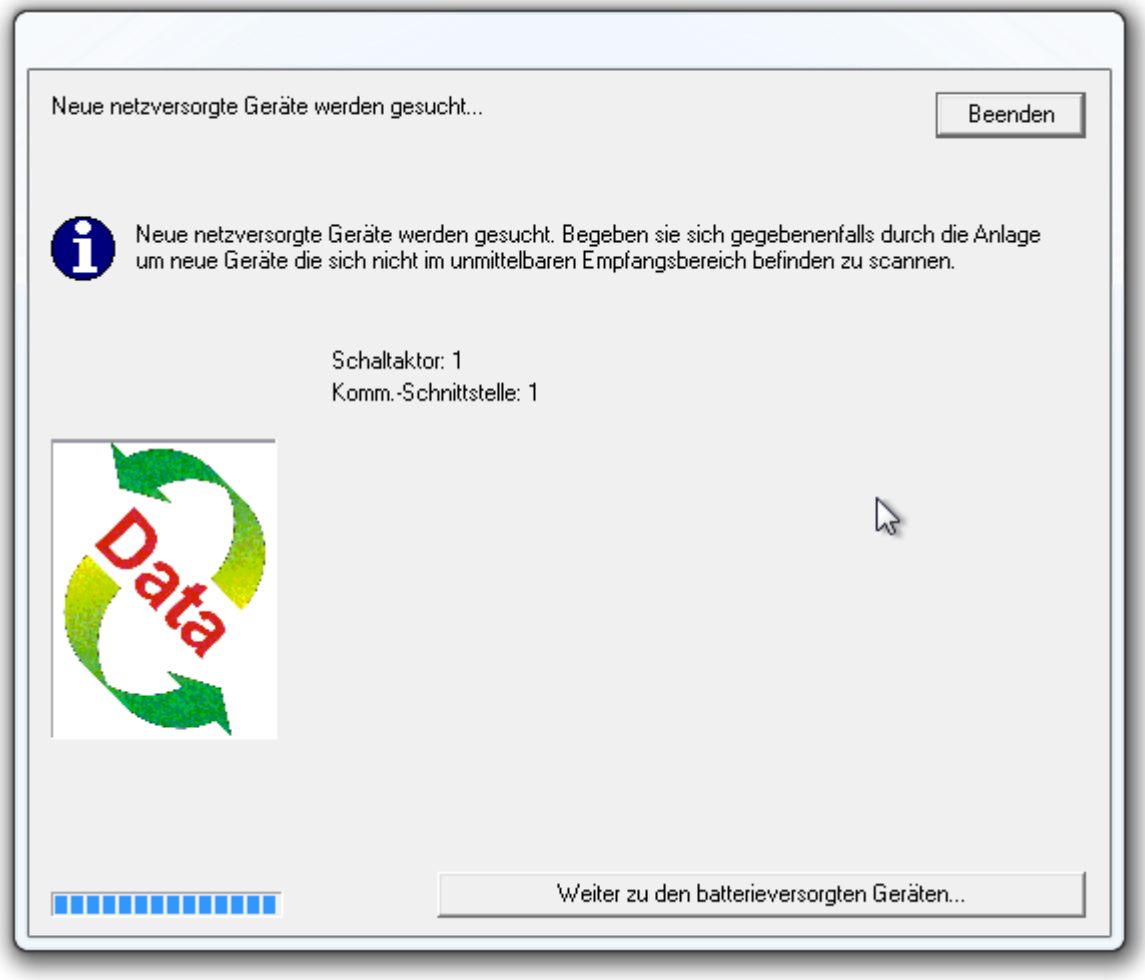
After the correct connection of the device(s), "Control Panel -> Hardware and Sounds -> Devices and Printers" must be opened. There you have to look for "Prolific USB-to-Serial Comm Port". The used port (e.g.: COM4) can be found in the brackets behind it, this should be noted or remembered.
The Eaton Firmware / MRF Software must be downloaded and installed. After starting the software, click on "Edit -> Options", select the correct port (e.g. COM4) and activate the appropriate options.



In the upper tab you can now click on "Gateway -> Read in". The system can now be programmed in comfort mode.

ADD SECONDARY DEVICES

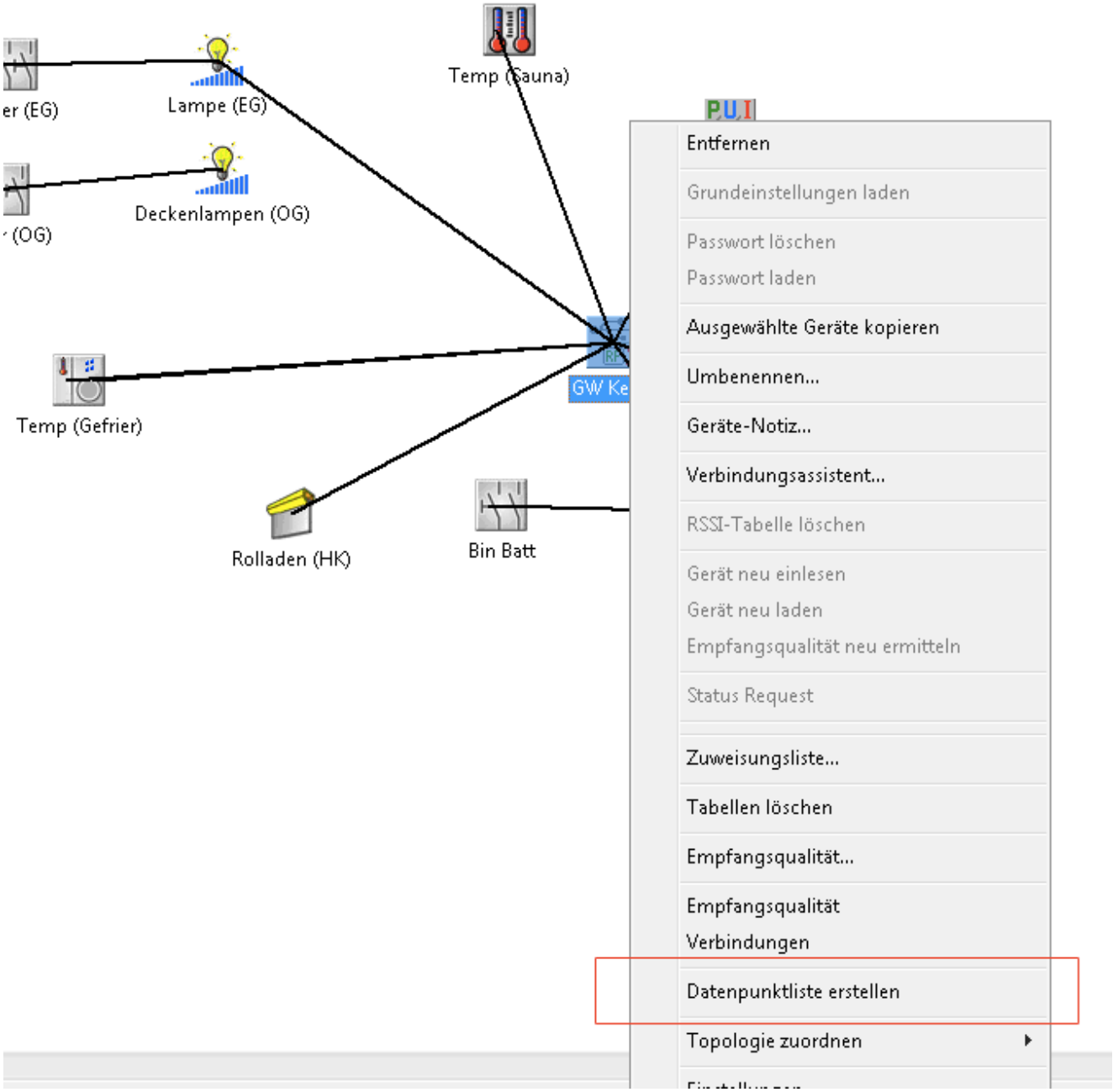
With the Eaton firmware, secondary devices from the Eaton company (e.g. intermediate plugs, room controllers, remote control, etc.) can now be integrated into the system.
This can be done in the upper tab via "Actions -> 2x Read".
Now the system will first search for the mains powered devices. If necessary, you have to go near the secondary devices for scanning, if they are not in the immediate reception area.



When all mains-powered devices have been read in, "Continue to battery-powered devices" must be clicked. The battery-powered devices must now be activated so that a connection to the gateway can be established. After this has been done, "Exit" must be clicked. Now a password entry is requested. It should be entered as **password "0000"** and confirmed.
In the upper tab "Actions" you should then click on "Read reception quality..." and then again on "Read".

If devices with question marks are still displayed, but all secondary devices are already assigned to another symbol, they can be deleted. This can be done by right-clicking on the symbol to be deleted and then left-clicking on "Remove". The following dialog must be confirmed with "Yes" and then "No".
Renaming is also possible with a click on the right mouse button and then on "Rename"

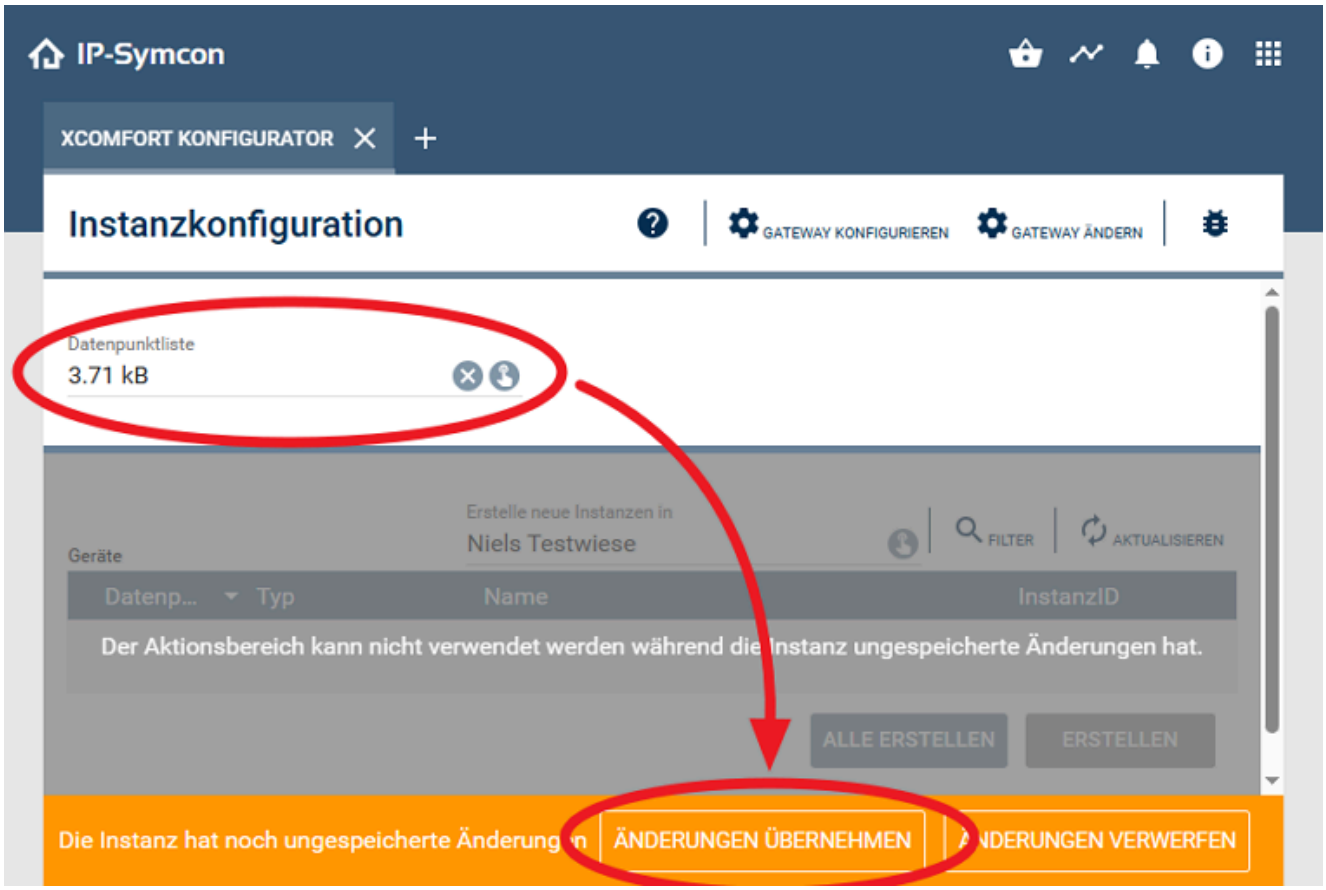
Now the devices are displayed in the Eaton firmware. Now the pencil symbol must be clicked on and the devices connected to the gateway via virtual pencil. After that you have to click on "Load changes" under "Actions".
So that Symcon later knows what is on data point 1 or on data point 2, for example, the right mouse button must be clicked on the gateway and then the left mouse button must be clicked on "Create data point list". This must be saved. Finally, the entire project must be saved. The Eaton firmware is no longer required.

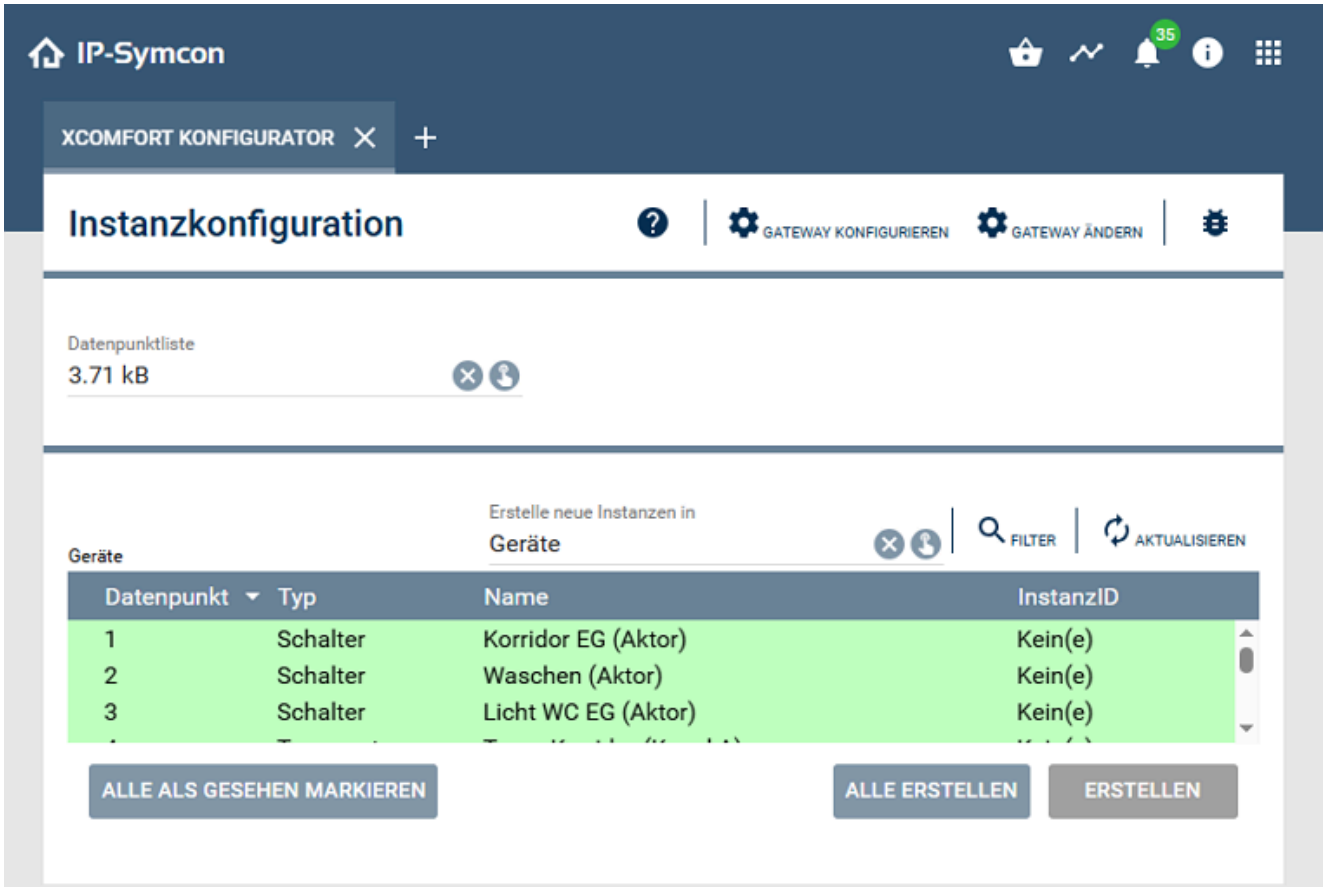


INTEGRATION IN SYMCON

In the management console, an ‘xComfort Configurator’ can be created in the object tree via ‘Add object -> Instance’. After selecting the mode appropriate for the gateway used, the connection to the gateway is configured.
Depending on the mode, an IP address must be entered, the Comport selected, or the appropriate USB device chosen, and the I/O activated. Finally, after confirming the dialogue, the configurator will open. Use the ‘Data point list’ file selection to upload the previously saved data point lists. Once the changes have been applied, the devices can be created.

To receive routable feedback, the data point list must have been loaded in the configurator. This is the only way to assign a serial number to a data point number





TIPS & TRICKS

New versions of the firmware/MRF software can be downloaded here:
Eaton Downloads

The gateway is compatible with all common USB->LAN converters (e.g. from Silex, Lantronix).
Status variables are not routed by the Eaton Xcomfort system. This means that the gateway must be positioned in a place that all relevant devices are within range.
Several gateways can be used simultaneously in Symcon and commands can also be forwarded via scripts. So it is common to place one gateway per floor.
Only from RF version **9.2** (approx. July/2008) the "Acknowledges" sent on the protocol level are evaluated by the USB gateway. In the previous versions it is not recognizable in the system whether a command has been successfully transmitted to a device. The RF version is a hardware upgrade and must not be confused with the FW version, which can be updated via software. (Concerns only PC Manager CKOZ-00/03)
The new thermostat requires an ECI LAN gateway or a USB gateway with at least firmware 2.6 and RF version 9.2. If the device is too old, the following error messages may appear in the message window: "Invalid RF-firmware revision"
The port for the ClientSocket configuration from the ECI is 7153

Device list

SUPPORTED INTERFACES

PRODUCT	DESCRIPTION
CCIA-01/01	Eaton ECI WLAN Interface
CCIA-02/01	Eaton ECI LAN Interface
CCIA-03/01	Eaton ECI LAN Interface Power over Ethernet
CKOZ-00/03	Communication interface (USB/RS232)
CKOZ-00/11	Communication interface (USB/RS232)
CKOZ-00/14	USB communication stick

SUPPORTED COMPONENTS

PRODUCT	DESCRIPTION
CAAE*	Analog actuator
CAEE*	Analog input
CBEU*	Binary input
CBMA*	Motion detector
CDAE*	Dimming actuator
CDAE*	Switch actuator
CDAP*	Switch/Dim actuator
CDAU*	Dimming actuator
CDWA* (since version 5.4)	Window/door contact
CEMU*	Energy measurement sensor
CHAP*	Heating actuator
CHAU*	Heating actuator
CHAZ*	Heating actuator for electric panels
CHAZ*	Heating actuator 12-fold
CHSZ*	Remote controls
CHVZ*	Radiator valve
CIZE*	Pulse counting input
CJAU*	Blind actuator
CJAE*	Blind actuator plug-in type
CRCA*	Room controller
CRMA*	Room-Manager

PRODUCT	DESCRIPTION
CROU*	Router
CSAE*	Switch actuator
CSAP*	Switch actuator
CSAU*	Switch actuator
CSEZ*	Window contact (via binary input)
CSEZ*	Humidity sensor (via analog input)
CSEZ*	Brightness sensor (via analog input)
CSEZ*	Air quality sensor (via analog input)
CSEZ*	Presence detector (via binary input)
CSEZ*	Smoke detector (via binary input)
CSEZ*	Temperature sensor
CSEZ*	Water leakage sensor (via binary sensor)
CSGZ*	Signal generator (via switching actuator)
CTAA*	Pushbutton
CTEU*	Temperature input

MXC_DimBrighter

```
bool MXC_DimBrighter (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts the dimming for the device with ID **InstanceID** at a higher intensity level. The process can be terminated with the command MXC_DimStop. If the maximum level is reached, the process will stop automatically. The status message is sent from the device, when the operation was terminated. There are no messages sent between.

EXAMPLE

```
MXC_DimBrigther(12345); //Dim brighter
IPS_Sleep(1000): //Wait 1 sec
MXC_DimStop(12345);
```

MXC_DimDarker

```
bool MXC_DimDarker (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts the dimming for the device with ID **InstanceID** at a lower intensity level. The process can be terminated with the command MXC_DimStop. If the maximum level is reached, the process will stop automatically. The status message is sent from the device, when the operation was terminated. There are no messages sent between.

EXAMPLE

```
MXC_DimDarker(12345); //Dim darker
IPS_Sleep(1000);
MXC_DimStop(12345);
```

MXC_DimSet

```
bool MXC_DimSet (int $InstanceID, int $Intensity)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	Value from 0-100 (in %)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** to value **Intensity**

EXAMPLE

```
MXC_DimSet(12345, 50); //Dimming to 50%
```

MXC_DimStop

```
bool MXC_DimStop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with ID **InstanceID** to value **Status**

EXAMPLE

```
MXC_DimStop(12345, true); //Turn on device
```

MXC_RequestStatus

```
bool MXC_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Requests a retransmission of the status messages for device with ID **InstanceID**.



Since status messages are not routed in xComfort system, it is possible that the command completed successfully, but no new status is received. This command does not query the status, but requires only a new status message.



This function loads the radio traffic considerably.
It should be used only to upgrade critical equipment for the IP Symcon start. A renewed queries at runtime is not required because xComfort forward all status changes directly to IP Symcon.

EXAMPLE

```
MXC_RequestStatus(12345); //Request a status report
```

MXC_SendBoolean

```
bool MXC_SendBoolean (int $InstanceID, bool $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	True for On, False for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a boolean value to the instance with ID **InstanceID**, which is connected to a room manager.

EXAMPLE

```
MXC_SendBoolean(12345, true);
```

MXC_SendFloat

```
bool MXC_SendFloat (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Number with decimal places

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a float to the instance with ID **InstanceID**, which is connected to a room manager.

EXAMPLE

```
MXC_SendFloat(12345, 35.34); //Send 35.34 as value
```

MXC_SendInteger

```
bool MXC_SendInteger (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Number without decimal places

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a boolean value to the instance with ID **InstanceID**, which is connected to a room manager.

EXAMPLE

```
MXC_SendInteger(12345, 40); //Send value 40
```

MXC_SetTemperature

```
bool MXC_SetTemperature (int $InstanceID, float $Temperature)
```

PARAMETERS

InstanceID	ID of the device to be switched
Temperature	Temperature in °C

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the new target temperature of the control valve with the ID **InstanceID** to the new **Temperature**.

EXAMPLE

```
MXC_SetTemperature(12345, 21.5); //Set to 21.5 ° C
```

MXC_ShutterMoveDown

```
bool MXC_ShutterMoveDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter with the ID **InstanceID** downwards until MXC_ShutterStop is running or the end position is reached.



Note that the running time limitation on the MRF Moeller is programmed. IP Symcon can not influence this limitation.

EXAMPLE

```
MXC_ShutterMoveDown(12345); //Move downwards
```

MXC_ShutterMoveUp

```
bool MXC_ShutterMoveUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter with the ID **InstanceID** upwards until MXC_ShutterStop is running or the end position is reached.



Note that the running time limitation on the MRF Moeller is programmed. IP Symcon can not influence this limitation.

EXAMPLE

```
MXC_ShutterMoveUp(12345); //Move upwards
```

MXC_ShutterStepDown

```
bool MXC_ShutterStepDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutters with the ID **InstanceID** one step down.

EXAMPLE

```
MXC_ShutterStepDown(12345); //Move one step down
```

MXC_ShutterStepUp

```
bool MXC_ShutterStepUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutters with the ID **InstanceID** one step up.

EXAMPLE

```
MXC_ShutterStepUp(12345); //Move one setp up
```

MXC_ShutterStop

```
bool MXC_ShutterStop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Stops a current movement of the shutter.

EXAMPLE

```
MXC_ShutterStop(12345); //Stop
```

MXC_SwitchMode

```
bool MXC_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device to be switched
Status	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the device with ID InstanceID to value Status

EXAMPLE

```
MXC_SwitchMode(12345, true); //Turn on device
```

Z-Wave

Z-Wave is a wireless system used by several manufacturers. Either a LAN or USB gateway is required to connect to IP-Symcon.



The following devices are supported by IP-Symcon:
Supported devices

INTEGRATION IN IP-SYMCON

When using the LAN gateway, this can be integrated via the Device Search. For this purpose, "Z-Wave Discovery" must be selected as the system. The Discovery instance then offers to create a Z-Wave configurator. After the configurator has been created, the individual devices can be integrated there as described below.
If a Z-Wave Configurator is to be created manually, "Z-Wave Configurator" can be searched for in the "Add instance" dialog.
After setting up a Z-Wave configurator, the next step is to teach-in devices.



Care must be taken to select the matching gateway , the matching client socket (default port: 5000) or Serial Port is set up and configured

This is done within the configurator by pressing the "Add device" key and then pressing the teach-in key of the device at least three times in quick succession.
IP-Symcon automatically finds the device and shows this green in the list.
When this is selected and the "Create" button is pressed, IP-Symcon creates an instance of the device, which can then be configured.
The configuration window of the instance can be opened via "Configure" in the Z-Wave Configurator or in the object tree by double-clicking on the instance.
The properties (Command classes see device list) of the device can be displayed under "Show" after "Loading". Status variables are added automatically and can be checked in the object tree.



To grant IP-Symcon access to certain functions (e.g. Z-Wave device database) from the outside it may be necessary to set them up in the firewall. Further information can be found under Firewall.

SETUP VIDEO-TUTORIA

SPECIAL EQUIPMENT - SETUP

link to forum post

Device list

SUPPORTED GATEWAYS

PRODUCT	DESCRIPTION
Symcon Z-Wave-Plus LAN Gateway | Product information	IP Gateway
300ZW-US-W	TrickleStar USB Stick
AEO_USB	USB stick from Aeon Labs
ZW090	USB Z-Stick from Aeotec
RaZberry	Z-Wave Transceiver for Rasperrry PI
VIS_ZU1401	Vision USB stick
ZME_Serial	Serial adapter plate
ZME_UZB1	USB stick
ZME_ZSTICKC	Z-Wave.Me USB stick

DEVICES

All listed command classes are supported by IP-Symcon. In addition, we offer a list of devices which have been successfully tested by us or our customers.
On request we will be happy to check other devices that are not listed for their compatibility.

SUPPORTED COMMAND CLASSES

Each Z-Wave device has different Command classes. These classes indicate which functions are supported by the respective device. In IP-Symcon, all functions are available with the following listed command classes.

COMMAND CLASS	CLASS VERSION	IP-SYMCON VERSION
ALARM	----------	Replaced by NOTIFICATION
ASSOCIATION	2	
ASSOCIATION_GRP_INFO	2	since 5.1
BASIC	2	
BASIC_WINDOW_COVERING	1	
BATTERY	1	
CENTRAL_SCENE	1	since 4.1
CLOCK	1	
COLOR	1	since 4.1
CONFIGURATION	1	

COMMAND CLASS	CLASS VERSION	IP-SYMCON VERSION
CRC_16	1	
DOOR_LOCK	2	since 4.1
LOCK	2	
MANUFACTURER_SPECIFIC	2	
METER	5	
MULTI_CHANNEL	4	
MULTI_CHANNEL_ASSOCIATION	3	since 5.1
MULTI_CMD	1	
MULTI_INSTANCE	----------	Replaced by MULTI_CHANNEL
NOTIFICATION	8	since 4.3
PROTECTION	2	
SCENE_ACTIVATION	1	since 4.1
SECURITY	1	since 4.1
SENSOR_ALARM	1	
SENSOR_BINARY	2	
SENSOR_MULTILEVEL	11	
SWITCH_ALL	1	since 4.1
SWITCH_BINARY	2	
SWITCH_MULTILEVEL	4	
SWITCH_SHUTTER	1	
THERMOSTAT_FAN_MODE	1	
THERMOSTAT_FAN_STATE	1	
THERMOSTAT_MODE	1	
THERMOSTAT_OPERATING_STATE	1	
THERMOSTAT_SETPOINT	1	
TIME	2	since 4.1
USER_CODE	1	since 4.1
Unknown	1	
VERSION	2	
WAKE_UP	2	
ZWAVEPLUS_INFO	2	since 5.1

TESTED COMPONENTS

sorted by alphabet

ACT

PRODUCT	DESCRIPTION
ZRM230	Wall Switch/Transmitter (2-gang)
ZTM230	RF Wall Transmitter (2-gang)

AEON LABS

PRODUCT	DESCRIPTION
AEO_DSC20-EU	Door/Window Sensor
AEO_DSB54	Recessed Door Sensor
AEO_DSD31	Siren
AEO_EXTENDER	Extender
AEO-HEM2, HEM3	Clamp Meter 40
AEO_KFOB	Key Fob
AEO-MEI	Micro Dimmer
AEO-MES	Micro Switch
AEO-MSEI	Micro Smart Energy Dimmer
AEO-MSES	Micro Smart Energy Switch
AEO_MULTISENS	Multisensor
AEO-SES3	Smart Energy Switch 3
AEO_ZW088	Key Fob Gen 5
AEO_ZW089	Recessed Doorsensor
DSD31	Outlet Plugable Siren

AEOTEC

PRODUCT	DESCRIPTION
ZW078	Heavy Duty Smart Switch
ZW080	Siren 5
ZW095	Home Energy Meter
ZW096	Smart Switch 6
ZW100	MultiSensor 6
ZW111	Nano Dimmer
ZW112	Door/Window Sensor 6
ZW116	Nano Switch with Power Metering
ZW117	Range Extender 6
ZW120	Door/Window Sensor Gen5 with Tamper Sensor
ZW121	LED Strip
ZW122	Water Sensor 6

PRODUCT	DESCRIPTION
ZW129	WallMote
ZW130	WallMote Quad
ZW132	Dual Nano Switch with Power Metering
ZW139	Nano Switch
ZW140	Dual Nano Switch
ZW141	Nano Shutter
ZW158	WallSwipe
ZW160	Water Sensor 6 Dock
ZW162	Doorbell 6
ZW166	Button
ZW164	Siren 6
ZW175	Smart Switch 7
ZW189	Range Extender 7
ZWA001	LED Bulb 6 Multi-White
ZWA002	LED Bulb 6 Multi-Colour
ZWA003	NanoMote Quad
ZWA004	NanoMote One
ZWA005	TriSensor
ZWA006	Smart Boost Timer Switch
ZWA008	Door/Window Sensor 7

DANFOSS

PRODUCT	DESCRIPTION
DAN_LC-13 (014G0013)	Thermostat Radiator Valve
DAN_LIVC_RAK (014G0012)	Thermostat Radiator Valve

EVERSPRING

PRODUCT	DESCRIPTION
EVR_AD142	Dimmer Plug
EVR_AN145	Screw-In Lamp Holder
EVR_AN157	On/Off Plug
EVR_AN158	On/Off Plug with power meter
EVR_HSM02	Door/Window Detector
EVR_HSP02	Motion Detector
EVR_SE812	Siren

PRODUCT	DESCRIPTION
EVR_SF812	Smoke Detector with Alarm
EVR_SP103	Motion Sensor
EVR_SP814	Motion Detector
EVR_ST812	Flood Sensor
EVR_ST814	Temperature & Humidity Sensor

FIBARO

PRODUCT	DESCRIPTION
FGBS-001	Universal Binary Sensor
FGD-211	Universal Dimmer
FGFS-101	Flood Sensor
FGK 101-107 Door/Window Sensor 2.5	Door/Window Sensor
FGK-001	Door/Window Sensor
FGMS-001	Motion Sensor
FGRGB-101	RGBW Controller
FGRGBWM-441	RGBW Controller
FGRM-222	Roller Shutter 2
FGS-211	1x3KW Relay
FGS-221	2x1.5KW Relay
FGSS-001	Smoke Sensor
FGSS-101	Smoke Sensor
FGWPE/F-101	Wall Plug

NORTHQ

PRODUCT	DESCRIPTION
Energy Guard Electrical Meter	Meter (incomplete support)
Energy Guard Gas Meter	Gas Meter (incomplete support)

PHILIO

PRODUCT	DESCRIPTION
In Wall Dual Relay (1 Way)	230VAC powered static controller with binary switch capability
In Wall Dual Relay(1 Way)	In-wall switch module
In Wall Dual Relay(1 Way)	In-wall dual relay switch module
In Wall Single Relay(1 way)	In-wall switch module

PRODUCT	DESCRIPTION
PAN04	2 x 1.5Kw Relay with Power Measurement
PAN06	2 x 1.5Kw Relay
PAN08	In-Wall Roller Shutter Controller
PAN11-1	Plugin Switch with meter functionality PAN11-1
PAT02-1A	Flood multisensor (flood, temperature and humidity)
PSE02	Wireless multiple sound siren
PSG01	Smoke Sensor
PSM01	3-in-1 Sensor (Door, Light, Temp)
PSM02	4-in-1 Sensor (PIR, Door, Light, Temp)
PSM02-1	PIR, door/window, temperature and illumination Sensor
PSP01	3-in-1 Sensor (PIR, Light, Temp)
PSR03	Remote
PST02	Slim Multisensor

POLYCONTROL

PRODUCT	DESCRIPTION
Danalock Circle	Z-Wave Door Lock
Danalock Square	Z-Wave Door Lock

POPP

PRODUCT	DESCRIPTION
POP_004308	Smoke Detector and Temp Sensor
POP_123610	Plug-in Switch
POP_123658	Plug-in Switch with Meter
POP_123665-58	Wall Plug Switch/Meter
POP_123xxx	On/Off Plug
POP_123xxx	Dimmer Plug

VISION

PRODUCT	DESCRIPTION
PIR Sensor	PIR Motion Sensor
Plugin ON/OFF	Light Relay
Power Monitor	Power Monitor
VIS_ZD2102EU 2013	Door Window Sensor with external digital input

PRODUCT	DESCRIPTION
VIS_ZD3102EU 2013	Motion detector with Temperature sensor and temper protection
VIS_ZG8101	Garage Door Detector
VIS_ZL7432	In-Wall Dual Relay Switch
VIS_ZM1601	Siren (Battery)
VIS_ZM1602	Siren (Mains)
VIS_ZS5102	Shock & Vibration Sensor
VIS_ZS6101	Smoke Detector
VIS_ZS6301	CO Detector

ZIPATO

PRODUCT	DESCRIPTION
Mini Keypad RFID/Z-Wave	Mini Keypad with RFID support
RGBW bulb	RGBW Bulb

Z-WAVE.ME

PRODUCT	DESCRIPTION
Mini remote control KFOB	Mini remote control 4 buttons Z-Wave KFOB, Version 2 Z-WAVE.ME

OTHER

MANUFACTURER	PRODUCT	DESCRIPTION
Express Control	EZMotion	Motion detector, luminosity and temperature sensor



The following are the currently definitely unsupported devices

NON-SUPPORTED COMPONENTS

sorted by alphabet

MERTEN

PRODUCT	DESCRIPTION
All	Almost all Merten products cannot be addressed via the normal Z-Wave standard

ZW_Basic

```
bool ZW_Basic (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	ID of the device
Value	0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Turns the basic function of the specified device with InstanceID **InstanceID**



The basic function can be determined in most devices via parameters. Please find further information in the guidance of your specific device.

EXAMPLE

```
ZW_Basic(12345, 255); //z.B. Turn on
```

ZW_ColorCW

Require: Symcon >= 5.4

```
bool ZW_ColorCW (int $InstanceID, int $ColdWhite)
```

PARAMETERS

InstanceID	Device ID
ColdWhite	ColdWhite intensity from 0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with the ID **InstanceID** in the cold white channel to the value **ColdWhite**.

EXAMPLE

```
// Set ColdWhite to 50%
ZW_ColorCW(12345, 128);
```

ZW_ColorRGB

Require: Symcon >= 5.4

```
bool ZW_ColorRGB (int $InstanceID, int $Red, int $Green, int $Blue)
```

PARAMETERS

InstanceID	Device ID
Red	Red Intensity from 0..255
Green	Green Intensity from 0..255
Blue	Blue Intensity from 0..255

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the device with the ID InstanceID to the color value combination RGB.

EXAMPLE

```
// Set dark red
ZW_ColorRGB(12345, 255, 0, 0);
```

ZW_ColorRGBWW

Require: Symcon >= 4.1

```
bool ZW_ColorRGBWW (int $InstanceID, int $Red, int $Green, int $Blue,
int $WarmWhite, int $ColdWhite)
```

PARAMETERS

InstanceID	Device ID
Red	Red intensity from 0..255
Green	Green intensity from 0..255
Blue	Blue intensity from 0..255
WarmWhite	WarmWhite Intensity from 0..255
ColdWhite	ColdWhite intensity from 0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the device with the ID **InstanceID** to the color value combination **RGBWW**.



The warm white component is only effective if all other color and cold white components are set to 0.



Since IP-Symcon 5.4 it is checked whether the device even supports RGB, warm white and cold white

EXAMPLE

```
// Set dark red
ZW_ColorRGBWW(12345, 255, 0, 0, 0, 50);
```

```
// Set warm white
ZW_ColorRGBWW(12345, 0, 0, 0, 255, 0);
```

ZW_ColorWW

Require: Symcon >= 4.1

```
bool ZW_ColorWW (int $InstanceID, int $WarmWhite)
```

PARAMETERS

InstanceID	Device ID
WarmWhite	WarmWhite Intensity from 0..255

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the device with the ID InstanceID in the warm white channel to the value WarmWhite.

EXAMPLE

```
// Set WarmWhite to 50%
ZW_ColorWW(12345, 128);
```

ZW_DimDown

```
bool ZW_DimDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Starts dimming down the device with ID InstanceID.

EXAMPLE

```
//The device with the ID 12345 starts to dim down
ZW_DimDown(12345);
```

ZW_DimDownEx

```
bool ZW_DimDownEx (int $InstanceID, int $Duration)
```

PARAMETERS

InstanceID	ID of the device to be switched
Duration	Running time until dimming down completely (0..255) (see table)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts dimming down the device with ID **InstanceID** in duration **Duration**.

DURATION	DESCRIPTION
0	Immediately
1 - 127	1 - 127 Seconds
128 - 254	1 - 127 Minutes
255	Device default for duration

EXAMPLE

```
//The device with ID 12345 begins to dim down over a runtime of 10
minutes
ZW_DimDownEx(12345, 138);

//The device with the ID 12345 begins to dim down over a period of 5
seconds
ZW_DimDownEx(12345, 5);
```

ZW_DimSet

```
bool ZW_DimSet (int $InstanceID, int $Intensity)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	Value from 0-100 (in %)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** to value **Intensity**

EXAMPLE

```
ZW_DimSet(12345, 50); //Dimming to 50%
```

ZW_DimSetEx

```
bool ZW_DimSetEx (int $InstanceID, int $Intensity, int $Duration)
```

PARAMETERS

InstanceID	ID of the device to be switched
Intensity	Value from 0-100 (in %)
Duration	Runtime until the intensity is reached (0..255) (see table)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Dims the device with ID **InstanceID** to value **Intensity** in runtime **Duration**.

DURATION	DESCRIPTION
0	Immediately
1 - 127	1 - 127 Seconds
128 - 254	1 - 127 Minutes
255	Device default for duration

EXAMPLE

```
//Dims the device with ID 12345 to 50% in 10 minutes
ZW_DimSetEx(12345, 50, 138);

//Dims the device with ID 12345 to 30% in 5 seconds
ZW_DimSetEx(12345, 30, 5);
```

ZW_DimStop

```
bool ZW_DimStop (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Stops dimming the device with ID **InstanceID**

EXAMPLE

```
ZW_DimStop(12345);
```

ZW_DimUp

```
bool ZW_DimUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Starts dimming up the device with ID InstanceID.

EXAMPLE

```
//The device with ID 12345 starts to dim up
ZW_DimUp(12345);
```

ZW_DimUpEx

```
bool ZW_DimUpEx (int $InstanceID, int $Duration)
```

PARAMETERS

InstanceID	ID of the device to be switched
Duration	Runtime until dimming up completely (0..255) (see table)

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Starts dimming up the device with ID InstanceID in runtime Duration.

DURATION	DESCRIPTION
0	Immediately
1 - 127	1 - 127 Seconds
128 - 254	1 - 127 Minutes
255	Device default for duration

EXAMPLE

```
//The device with the ID 12345 starts to dim up over a runtime of 10
minutes
ZW_DimUpEx(12345, 138);

//The device with ID 12345 begins to dim up over a runtime of 5 seconds
ZW_DimUpEx(12345, 5);
```

ZW_DoorLockOperation

Require: Symcon >= 4.1

```
bool ZW_DoorLockOperation (int $InstanceID, int $Mode)
```

PARAMETERS

InstanceID	ID of the device to be switched
Mode	Mode to set the Z-Wave device to

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the Door Lock Operation Mode of the device with ID **InstanceID** to mode **Mode**.

MODUS	VALUE
Unsecured	0
Unsecured with timeout	1
Unsecured (inside)	16
Unsecured (inside) with timeout	17
Unsecured (outside)	32
Unsecured (outside) with timeout	33
Secured	255

EXAMPLE

```
//e.g. Switches the door lock operation mode to unsecured with a
timeout
ZW_DoorLockOperation(12345,1);
```

ZW_LockMode

```
bool ZW_LockMode (int $InstanceID, int $Value)
```

PARAMETERS

InstanceID	Device ID
Value	0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the LockMode for the device with the InstanceID **InstanceID**

EXAMPLE

```
ZW_LockMode(12345, 0);
```

ZW_MeterReset

```
bool ZW_MeterReset (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Resets the meter from the device with InstanceID **InstanceID**

EXAMPLE

```
ZW_MeterReset(12345);
```

ZW_Optimize

Require: Symcon >= 4.1

```
bool ZW_Optimize (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts the radio link optimization process for the device with InstanceID **InstanceID**



The optimization process is automatically executed cyclically for all devices integrated in IP-Symcon.

EXAMPLE

```
ZW_Optimize(12345);
```

ZW_ProtectionSet

```
bool ZW_ProtectionSet (int $InstanceID, int $Mode)
```

PARAMETERS

InstanceID	Device ID
Mode	0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the Protection and its Mode **Mode** from the specified device with InstanceID **InstanceID**

MODE	INTEGER-VALUE
Off	0
Limited	1
On	2

EXAMPLE

```
ZW_ProtectionSet(12345, 1);
```

ZW_RequestStatus

```
bool ZW_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the status from the device with the ID **InstanceID**. Only power-operated devices can be queried directly by command, because battery-operated devices do not permanently listen to the queries.



This function loads the radio traffic considerably.
It should be used only to upgrade critical equipment for the IP-Symcon start. A renewed queries at runtime is not required, because Z-Wave sends normally all status changes directly to IP Symcon. Older devices which refuse to do so, should be replaced with newer equipment.

EXAMPLE

```
ZW_RequestStatus(12345);
```

ZW_ShutterMoveDown

```
bool ZW_ShutterMoveDown (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter with the ID **InstanceID** downwards until MXC_ShutterStop is running or the end position is reached.



Note that some blind control from Z-Wave must be adressed over ZW_DimSet like the dimmer.

EXAMPLE

```
ZW_ShutterMoveDown(12345); //Move downwards
```

ZW_ShutterMoveUp

```
bool ZW_ShutterMoveUp (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter with the ID **InstanceID** upwards until MXC_ShutterStop is running or the end position is reached.



Note that some blind control from Z-Wave must be adressed over ZW_DimSet like the dimmer.

EXAMPLE

```
ZW_ShutterMoveUp(12345); //Move upwards
```

ZW_ShutterStop

```
bool ZW_ShutterStop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Stops a current movement of the shutter.



Note that some blind control from Z-Wave must be adressed over ZW_DimSet like the dimmer.

EXAMPLE

```
ZW_ShutterStop(12345); //Stop
```

ZW_SwitchAllMode

Require: Symcon >= 4.1

```
bool ZW_SwitchAllMode (int $InstanceID, int $Mode)
```

PARAMETERS

InstanceID	ID of the device to be switched
Mode	Mode to set the Z-Wave device to

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the instance with ID **InstanceID** to Mode **Mode**.

MODE	VALUE
All On	0
Only Off	1
Only On	2
All Off	255

EXAMPLE

```
//Mode is set to Only On
ZW_SwitchAllMode(12345, 2);
```

ZW_SwitchMode

```
bool ZW_SwitchMode (int $InstanceID, bool $Status)
```

PARAMETERS

InstanceID	ID of the device
Status	TRUE for on, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Switches the device with ID InstanceID to value Status

EXAMPLE

```
ZW_SwitchMode(12345, true); //Turn on device
```

ZW_Test

Require: Symcon >= 4.1

```
bool ZW_Test (int $InstanceID)
```

PARAMETERS

InstanceID	Device ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Tests the connection to the device with InstanceID **InstanceID**

EXAMPLE

```
ZW_Test(12345);
```

ZW_ThermostatFanModeSet

```
bool ZW_ThermostatFanModeSet (int $InstanceID, int $FanMode)
```

PARAMETERS

InstanceID	Device ID
FanMode	0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the FanMode **FanMode** from the specified device with the InstanceID **InstanceID**

FANMODE	INTEGER-VALUE
AutoLow	0
OnLow	1
AutoHigh	2
OnHigh	3

EXAMPLE

```
ZW_ThermostatFanModeSet(12345,1); //e.g. Switches the FanMode to OnLow
```

ZW_ThermostatModeSet

```
bool ZW_ThermostatModeSet (int $InstanceID, int $Mode)
```

PARAMETERS

InstanceID	Device ID
Mode	0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the Mode **Mode** from the specified device with the InstanceID **InstanceID**

MODE	INTEGER-VALUE
Off	0
Heat	1
Cool	2
Auto	3
Auxiliary	4
Resume	5
FanOnly	6
Furnace	7
DryAir	8
MoistAir	9
AutoChangeover	10

EXAMPLE

```
ZW_ThermostatModeSet(12345, 1); //e.g., Sets the mode to Heat
```

ZW_ThermostatSetPointSet

```
bool ZW_ThermostatSetPointSet (int $InstanceID, int $SetPoint, float
$Value)
```

PARAMETERS

InstanceID	Device ID
SetPoint	0..255
Value	0..255

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the PointSet **PointSet** of the specified device with the InstanceID **InstanceID**

SETPOINT	INTEGER-VALUE
Invalid	0
Heating	1
Cooling	2
Furnace	3
DryAir	4
MoistAir	5
AutoChangeover	6

EXAMPLE

```
ZW_ThermostatSetPointSet(12345, 1, 20); //e.g. switch on
```

Zevvy

Require: Symcon >= 7.0
This module sends the data of selected variables to a Zevvy account, which then appears under "Settings" -> "Device list" with the ID of the variable.

FUNCTIONAL SCOPE

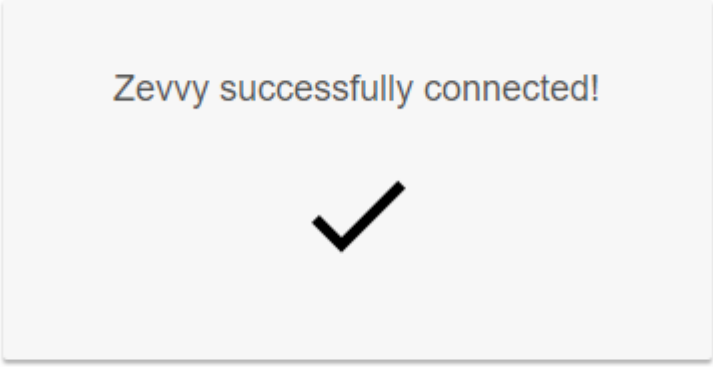
Sending the values

SETTING UP THE INSTANCES IN IP-SYMCON

The 'Zevvy' module can be found under 'Add instance' using the quick filter.
Further information on adding instances can be found in the Instances documentation
Before the variables can be set up, the Register button must be clicked. A new tab will now open in the browser with the Zevvy login screen.



After entering the login data, the following dialog is displayed if successful.



You can now return to IP-Symcon. A token should now be present in the Zevvy instance.
Configuration page:

NAME	DESCRIPTION
Device list	List with the variable IDs that are to be sent.
Interval	Interval for automatically sending the data
Transmit data	Button which sends the data manually
Registration	Links the Zevvy account to IP-Symcon using OAuth

ZY_SendMeasurements

Require: Symcon >= 7.0

```
bool ZY_SendMeasurements (int $InstanzID)
```

PARAMETERS

InstanzID	Id of the Zevvy instance

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

Sends the data of the set up variables to a Zevvy account.

EXAMPLE

```
ZY_SendMeasurements(12345);
```

Logic

ACTIVE LIST

Overview of all active variables in the Visualization

AL_SwitchOff	Switches all active variables in the list to inactive.

PRESENCE SIMULATION

Simulate presence via logged values

AS_SetSimulation	De-/Activate the presence simulation

IMAGE ARCHIVE

The module copies an image when triggered by a selected variable.

BA_AddImage	Copies the current source image to the "Images" category of the "ImageArchive" module with the InstanceID.

COUNTDOWN

CSV ZIP EXPORT

Exports aggregated values of a variable as a CSV file in a ZIP archive

CSV_DeleteZip	Removes the generated file.
CSV_Export	Creates a zip archive based on the given parameters.
CSV_SendMail	Sends a mail through an SMTP instance with a generated zip file

DUMMY MODULE

Placeholder module for display as an instance in the visualization

EGG TIMER

An egg timer that runs for a certain amount of time and can be rewound at any time. Ideal for use in events and scripts.

GROUP CONTROL

With the help of the 'Group Control', variables can be switched together in groups.

JSON DECODER

Decodes JSON and makes the content available in variables

JSON EXPORTER

Exports any variable in a user-defined structure via an HTTP endpoint.

LOGIC GATE

Links variables by logical operations

COMPUTATION MODULE

Modules, which enables various calculations within IP-Symcon

Computation Module	Calculation of different auxiliary values based on a group of variables, e.g. buzzer or average.
RM_Update	Updates the calculated values
Converter	Converts a variable using a self-defined formula.
UMR_Calculate	Calculates the return of the formula for the Value and returns it.
ConvertMultiBoundaries	Converts a variable within set limits using a self-defined formula.
UMG_Calculate	Calculates the return of the formula for the value __Value__ and returns it.
ValueRangeScale	Scaling of a value within a value range
VRC_Scale	Scales the input variable value

RGBMULTIPLEXER

Connects individual R, G, B channels to a variable, which can be controlled with the color wheel.

RGBM_RequestStatus	Determines the value of the individual R, G, B channels and sets the color variable.
RGBM_SetRGB	Sends the R, G, B values to the individual channels and updates the color variable

DRAGPOINTER

A variable that remains at the highest or lowest value until reset.

SZ_Reset	Resets the drag pointer

GAME COLLECTION

A collection of simple games which can be played via the Visualization.

Rock Paper Scissors	A way to play rock, paper, scissors on the Visualization.
Number Guessing	A little game where a random number has to be guessed.

ZR_Generate	Generates a new number that can be guessed

SCENE CONTROL

Stores the state of any number of actuators and makes them available via Visualization/PHP function.

SZS_CallScene	Calls the scene and sets it associated variables
SZS_GetActiveScene	Returns the number of the scene that is currently active.
SZS_SaveScene	Saves the values of the variables present in the list in the corresponding scene.

DEW POINT TEMPERATURE CALCULATION

Calculation of the dew point temperature

STAIRCASE LIGHT CONTROLS

When activated, the light in the staircase is switched on and is switched off again shortly after leaving the staircase.

THL_SetActive	Activates or deactivates the staircase lighting control.
THL_Start	Activates the light in the staircase and starts the timer, which deactivates the light again.
THL_Stop	Deactivates the light in the staircase and the timer.

RENAMER

Enables renaming of selected objects via Visualization

RAIN CENTRAL

Fetches the current rain radar image from Wetteronline and evaluates the color pixels of the rain amount.

UWZ_RequestInfo	Calculates the variable rain value of the rain central module

VARIABLE COMPARISON

The VariablenVergleich module allows you to create a point cloud from two variables and compare them using linear regression.

LR_Download	Regenerates the graph and outputs an address
LR_GenerateChart	Regenerates the chart and returns it as SVG and PNG.
LR_UpdateChart	Regenerates the diagram

VIRTUELLE DEVICES

Modules for simulating various devices

E-Car (Simulation)	Simulates the charging process of an electric car
Heater (Simulation)	Simulates a heater
Light (Simulation)	Simulates a light
Mediaplayer (Simulation)	Simulates a MediaPlayer
PV-System (Simulation)	Simulation of a PV system
Shutter (Simulation)	Simulates a roller shutter
Thermostat (Simulation)	Simulates a thermostat
Consumption/Costs	Calculates the costs according to consumption
Counter (Simulation)	Simulates an adjustable number of counters.

RANDOM LIGHTING

The module enables one to switch the color value of lamps randomly between different colors.

ZB_ChangeLight	Selects random colors and sets the selected variables to them

Active List

Require: Symcon >= 5.0
The 'Active List' shows all active variables in the Visualization and offers the possibility to switch them off simultaneously.
To do this, they must have been added to the list on the configuration page beforehand.

FUNCTION SCOPE

Displays all active variables in the Visualization and allows switching them off.

SOFTWARE INSTALLATION

Use the Module Store to install the Active List module.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Active List' module can be found using the quick filter.
More information in the Instances documentation.

CONFIGURATION PAGE

NAME	DESCRIPTION
Turn Off Action	Creates an action to turn off the variable simultaneously
Variables	A list of variables whose status is checked.

Variables are considered active when ...
the value of an integer or float variable is greater than the minimum value. If the variable has a .reversed profile it is considered active if the value is less than the maximum value.
the value of a boolean variable is true. If the variable has a .reversed profile, false is the active state.
the value of a string variable is not empty.

Accordingly, variables are considered inactive if ...
the value of an integer or float variable is the minimum value. If the variable has a .Reversed profile, it is considered inactive if the value is the maximum value.
the value of a boolean is false. If the variable has a .reversed profile, true is the inactive state.
the value of a string variable is empty.

STATUSVARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Disable	Script	Disables all variables that are still active.

VISUALIZATION

All active variables will be displayed on the Visualization. With a click on "Switch off" all displayed variables are switched to inactive.

AL_SwitchOff

Require: Symcon >= 5.0

```
bool AL_SwitchOff (int $InstanzID)
```

PARAMETERS

InstanzID	ID des zu schaltenden Geräts

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Switches all active variables in the list to inactive.

EXAMPLE

```
AL_SwitchOff(12345);
```

Presence Simulation

Require: Symcon >= 5.0
Simulates the presence of persons in the household. For this purpose, the module randomly obtains the daily data from one of the last 4 identical weekdays. If there are not enough switching operations logged on any of these 4 days, one of the last 30 days is selected at random.
If also within these 30 days no valid day data record is available, no simulation is possible. If no simulation is possible, this is displayed as a message in the "Simulation source (day)" string variable.

FUNCTION SCOPE

Switching of selected actuators/variables via logged values.
Adjustability of the required average switching operations before a tag is allowed as a source for simulation.
Switching on/off via Visualization button or script function.
Display which day is used for simulation.
Automatic update when day changes.

SOFTWARE INSTALLATION

Via the Module Store install the module 'Presence Simulation'.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Presence Simulation' module can be found using the quick filter.
More information in the Instances documentation.
All variables to be switched must be added to the variables list in the instance configuration.

CONFIGURATION PAGE

NAME	DESCRIPTION
Variables	A list of variables to be used for the presence simulation. The added variables need an action and must be logged.
Minimum number	This describes the average minimum number of variable switches of all selected variables that must be present.

STATUSVARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

NAME	TYPE	DESCRIPTION
Simulation active	Boolean	Indicates whether the simulation is activated or not. True = Enabled; False = Disabled;
Simulation Source (Day)	String	The string contains the date after which the simulation data was selected.
Simulation Preview	String	Shows a table which gives an overview of the future switching operations.

VISUALIZATION

Via the Visualization the simulation can be de-/activated.
The information which tag is used for simulation is also displayed.
If there is not enough or invalid data, this is also displayed here. A list with all selected variables is displayed, which contains the current and next value, as well as the time of the circuit.

AS_SetSimulation

Require: Symcon >= 5.0

```
bool AS_SetSimulation (int $InstanceID, bool $SetActive)
```

PARAMETERS

InstanceID	ID of the presence simulation
SetActive	Enables/Disables Presence Simulation

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

SetActive enables (true) or disables (false) the presence simulation with InstanceID InstanceID.

EXAMPLE

```
AS_SetSimulation(12345, true);
```

Image Archive

Require: Symcon >= 4.2
The module copies an image when triggered by a selected variable. It is adjustable how many images should be saved.

FUNCTION SCOPE

Selectable source image and selectable trigger variable
Adjustable number of images to be saved
When a trigger variable is selected, an event is created that reacts to its update
The event can be fully personalized except for the trigger variable
Images get the time of triggering as name
Images are displayed chronologically from old -> new in the category images
Automatic deletion of the oldest images, when the maximum number of images is reached

SOFTWARE INSTALLATION

Install the 'Image-Archive' module via the Module Store.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Image-Archive' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE

NAME	DESCRIPTION
Image	Source image to be copied
Number	Maximum number of images to be saved
Trigger Variable	Variable at whose change the source image should be copied to the archive

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Images	Category	The source images are copied into here

NAME	TYPE	DESCRIPTION
AddImage	Event	Triggered by the trigger variable and triggers the copying of the source image

VISUALIZATION

The Visualization is used to display the variable. No further control or separate display is integrated.
To display the images in the 'Bildarchiv' category, a content divider can be added in the Visualization or a link of the category can be made.
Attention: It is not useful to link directly the images from the archive, because the updates change the ID's.

BA_AddImage

Require: Symcon >= 4.2

```
bool BA_AddImage (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Copies the current source image to the "Images" category of the 'Image Archive' module with InstanceID **InstanceID**. The function does not return any value.

EXAMPLE

```
BA_AddImage(12345);
```

Countdown

Require: Symcon >= 5.0
The module allows to display a countdown to an event in the webfront.

FEATURE

Allows to display a countdown in the web front that is updated every minute
Setting a time and name of an event

SOFTWARE INSTALLATION

Using the Store module, install the 'Countdown' module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Countdown' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATIONPAGE

NAME	DESCRIPTION
Date and Time	Date and Time
Event	Name to be displayed in Visualization

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Event reached	Boolean	Indicates whether the event was reached
Time Remaining	String	Contains the remaining time

VISUALIZATION

Displays the countdown and a message when the event has occurred.

CSV ZIP Export

Require: Symcon >= 5.5
This module provides the possibility to export the aggregated values of a variable as a CSV file in a ZIP archive.

FUNCTION SCOPE

Exporting aggregated data of a variable or several variables
Export data to CSV file in a ZIP archive
Listing of all logged variables
Period of aggregation can be freely selected
Aggregation level can be selected
Cyclic creation and sending of an archive by email or ftp/ftps/sftp

SOFTWARE INSTALLATION

Install the 'CSVZipExport' module via the Module Store.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance", the 'CSV-ZIP-Export' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE

NAME	DESCRIPTION
Export Options	Option whether one or more variables per file should be exported, or a single variable used by MySQL should be exported
Selected variables	[Export options: One file per variable, One file with multiple variables] List of variables that can be set for the CSV column
Filter	[Export options: MySQL] Filters the selection of logged variables
Logged Variables	[Export options: MySQL] Selection of the variable to be exported
Create zip file	Checkbox whether the file should be compressed into a zip archive
Decimal separator	Select whether point or comma should be used as decimal separator
Start of aggregation	Start of aggregation period
End of aggregation	End of aggregation period
Decimal separator	Selection of decimal separator comma or point.
Aggregation Level	Level of Aggregation
Export	The aggregated data of the variable will be exported

NAME	DESCRIPTION
Cyclic send	
Enable cyclic send	Activates the cyclical sending of data
Interval	Interval at which data is sent. With “Weekly”, data is sent on Monday and with “Monthly”, data is sent on the 1st of the month at the selected time
Time	Time at which to send
Mail	
Send by mail	Checkbox whether the data should be sent by mail
SMPT instance	Select the e-mail instance
Send mail now	Sends a mail manually
FTP, FTPS, SFTP	
Send via SFTP/FTP/FTPS	Checkbox whether the data should be sent via SFTP, FTP or FTPS
Connection type	Connection whether to send via SFTP, FTP or FTPS
Host	IP address of the server
Port	Port
Username	Username for the SFTP connection
Password	Password for the SFTP connection

CSV_DeleteZip

Require: Symcon >= 5.5

```
bool CSV_DeleteZip (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Removes the generated file.

EXAMPLE

```
CSV_DeleteZip(12345);
```

CSV_Export

Require: Symcon >= 5.5

```
string CSV_Export (int $InstanceID, int $ArchiveVariable, int
$AggregationStage, int $AggregationStart, int $AggregationEnd)
```

PARAMETERS

InstanceID	ID of the Instance
ArchiveVariable	VariableID, which should be exported
AggregationStage	Aggregation level analogous to AC_GetAggregatedValues
AggregationStart	Date/time as Unix timestamp (0 = from beginning)
AggregationEnd	Datum/Zeit as Unix timestamp (0 = until now)

RETURNS

The relative path of the zip archive.

DESCRIPTION

Creates a zip archive based on the given parameters and returns the relative path of the archive.

EXAMPLE

```
CSV_Export(12345, 54321, 4, 2293574400, 3127161600);
```

CSV_SendMail

Require: Symcon >= 5.5

```
bool CSV_SendMail (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Sends a mail through an SMTP instance with a generated zip file

EXAMPLE

```
CSV_SendMail(12345);
```

Dummy Module

The dummy module is an instance, that does not provide its own function or status variables, but can be used as a placeholder only to get a presentation as a device in the visualization. So it is conceivable that you create several variables that belong to a device for which IP-Symcon provides no module. To represent these variables within an unit together in the visualization (e.g. Visualization) you can create the dummy instance and place the dummy variable in the instance.
In combination with the variable profiles and the variable actions you can create your own "instances", which have similar functionality in the function and automatic visualization of virgin instances.

Egg Timer

Require: Symcon >= 5.0
An egg timer that runs for a certain amount of time and can be rewound at any time. Ideal for use in events and scripts.

FUNCTION SCOPE

A simple egg timer
Can be rewound at any time
Runs for an adjustable time
Controllable via the Visualization
Easy integration into events and scripts via the active variable

SOFTWARE INSTALLATION

Install the 'Egg Timer' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Eggtimer' module is listed under the vendor '(device)'.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Update Interval	The interval at which the remaining time is updated (If the selected interval is greater than the remaining time, "Active" will not be set to false again until the interval has expired)

STATUSVARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Active	boolean	Determines whether the egg timer is running or not; Setting it again restarts the timer
Time in seconds	integer	The duration in seconds that the egg timer runs; After the time has elapsed, "Active" is set to false

NAME	TYPE	DESCRIPTION
Remaining	string	Displays the remaining time of the running egg timer
Canceled	integer	Indicates whether the Active variable was switched off again before the time expired and thus the timer was canceled.

VISUALIZATION

In the Visualization the running time of the egg timer can be set and enabled/disabled

Group Control

Require: Symcon >= 5.0
With the help of the 'Group Control', variables can be switched together in groups.

FUNCTION SCOPE

When one variable in the group is switched, all remaining variables in the group are switched as well
All variables that are added to a list can be switched with a separate variable at the same time

SOFTWARE INSTALLATION

Via the Module Store install the 'Group Control' module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Group Control' module can be found using the quick filter. - More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Variables	The variables present in this list belong to the group; All variables must be of the same type and have the same profile and an action

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Status	variant	Displays the status of the current group

VISUALIZATION

This displays the status variable that can switch the group.

JSON Decoder

Require: Symcon >= 6.0
The JSON Decoder is a module that breaks down JSON-coded content into its components and makes the elements available as variables.

INTEGRATION IN IP-SYMCON

The JSON decoder can be searched for and added as an instance using the "+" in the object tree.
By default, a HTTP Client is created and used as source. Within the configuration of the JSON decoder, a gateway that provides the text to be edited can be selected via "Change gateway".

EXAMPLE

As an example, the page filesamples.com/samples/code/json/sample4.json should be read and decoded using the JSON decoder.
To do this, the URL of the page must be entered in the HTTP client and then read in.
The page contains the following JSON.

```
{
"people" : [
{
"firstName": "Joe",
"lastName": "Jackson",
"gender": "male",
"age": 28,
"number": "7349282382"
},
{
"firstName": "James",
"lastName": "Smith",
"gender": "male",
"age": 32,
"number": "5678568567"
},
{
"firstName": "Emily",
"lastName": "Jones",
"gender": "female",
"age": 24,
"number": "456754675"
}
]
}
```

The JSON read in via the HTTP client is now processed with the JSON decoder and the following information is made available as variables in the object tree.

23811	JSON Decoder	JSON Decoder		
46224	people	Dummy Module		
40910	Index_0	Dummy Module		
10707	age	Integer	28	15:24:55
14117	firstName	String	Joe	15:24:55
14032	gender	String	male	15:24:55
41550	lastName	String	Jackson	15:24:55
18695	number	String	7349282382	15:24:55
22449	Index_1	Dummy Module		
59407	Index_2	Dummy Module		

JSON Exporter

Require: Symcon >= 5.5
The module offers the possibilities to represent the current values of variables as JSON export. The object structure can be defined in advance, so that e.g. an array of objects with ID, name, value and profile suffix is returned. Any number of objects can then be returned in this structure, with the individual fields linked to variables or defined as free text.

FUNCTION SCOPE

Individual structure of the objects can be defined
Output of current values in JSON format, returning an array of objects

SOFTWARE-INSTALLATION

Install the 'JSON Exporter' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance' the 'JSON Exporter' module is listed under the vendor '(core)'.

CONFIGURATION PAGE

NAME	DESCRIPTION
Ident	Selectable name
Configure structure	The data set to be provided can be configured via **Configure structure**. Each value is entered in the table **Variables to be exported**
Value	**User defined** Selection of what to select from the variable
Variables to export	List, where the variables can be listed and added according to the structure
Open in browser	Opens the JSON structure in a browser

Logic Gate

Require: Symcon >= 5.0
This module enables the determination of logical connection between several variables.
Here, the Boolean interpretation of the input variables is used.

FUNCTIONAL SCOPE

Determining the logical relationship of variable values
Supported operations:
- OR
- AND
- NOR
- NAND

Output is kept up to date and is updated as soon as one of the input variables changes

SOFTWARE INSTALLATION

Via the Module Store install the 'Logic Gate' module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add instance" the 'Logik Gate' module is listed under the manufacturer '(Other)'
At 'Calculation' select the desired operation
In the list 'Input' select the desired input variables
- 'Invert' can be activated to negate the input value of this variable
- If a non-Boolean variable is selected, it will be interpreted as a boolean value, e.g. 0 as false

STATUS VARIABLES

The status variable "output" contains the current result of the logical operation

VISUALIZATION

The current output is displayed.

Computation Module

Modules, which enables various calculations within IP-Symcon

Computation Module	Calculation of different auxiliary values based on a group of variables, e.g. buzzer or average.
RM_Update	Updates the calculated values
Converter	Converts a variable using a self-defined formula.
UMR_Calculate	Calculates the return of the formula for the Value and returns it.
ConvertMultiBoundaries	Converts a variable within set limits using a self-defined formula.
UMG_Calculate	Calculates the return of the formula for the value __Value__ and returns it.
ValueRangeScale	Scaling of a value within a value range
VRC_Scale	Scales the input variable value

Computation Module

Require: Symcon >= 4.3
This module can perform various auxiliary calculations on a set of variables, for example, the sum or the average.

FUNCTION SCOPE

Calculations of various values based on a set of variables:
- Sum
- Minimum
- Maximum
- Average
- Number of variables

Output of the calculated values in variables
Updating of the values as soon as one of the variables changes

SOFTWARE INSTALLATION

Via the Module Store install the 'Computation Module'.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Computation Modulel' can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Calculation	Selection of the performed calculation(s)
Variables	VariableIDs of the variables on which the calculation(s) will be performed; for all calculations except Number, all variables must be of type Float or Integer

_POSSIBLE*CALCULATIONS:*

NAME	DESCRIPTION
All	All calculations presented in this table are performed
Sum	The sum of all selected variables is stored in the status variable Sum
Minimum	The minimum value of the selected variables is stored in the Minimum status variable
Maximum	The maximum value of the selected variables is stored in the status variable Maximum

NAME	DESCRIPTION
Average	The average of the selected variables is stored in the status variable Average
Number	The number of selected variables is stored in the status variable Number

STATUS VARIABLES

The status variables are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

A corresponding status variable is created for each calculation.

NAME	TYPE	DESCRIPTION
Sum	Float	The sum of all selected variables
Minimum	Float	The minimum value of the selected variables
Maximum	Float	The maximum value of the selected variables
Average	Float	The average of the selected variables
Number	Float	The number of selected variables

RM_Update

Require: Symcon >= 4.3

```
bool RM_Update (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Updates the calculated values

EXAMPLE

```
RM_Update(12345);
```

Converter

Require: Symcon >= 4.3

FUNCTION SCOPE

Calculates a value from a selected source variable using a set-up formula.
If the source variable is changed, the value is automatically recalculated.

SOFTWARE INSTALLATION

Install the 'Converter' module via the Module Store.

SETTING_UP_INSTANCES_IN IP-SYMCON

Under "Add Instance" the 'Converter' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	Source variable to be used for calculation.
Formula	Formula where calculation is to be used.
Value	Test value to test the formula
Calculate	Calculates the value using the test value

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Value	Float	Contains the value calculated using the set up formula.

UMR_Calculate

Require: Symcon >= 4.3

```
float UMR_Calculate (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the Instance
Value	Value to be inserted in the formula

RETURNS

Converted Value **Value**

DESCRIPTION

Calculates the return of the formula for the Value **Value** and returns it.

EXAMPLE

```
//Function in the configuration: $Value/10\n$output =
UMR_Calculate(12345, 50);\nvar_dump($output) //Output is 5
```

ConvertMultiBoundaries

Require: Symcon >= 4.3

FUNCTION SCOPE

Calculates a value from a selected source variable using set-up formulas.
Which formula is to be used is decided via configurable limit values.
If the source variable is changed, the value is automatically recalculated.
If no calculation/limits apply, the original value is entered.

SOFTWARE INSTALLATION

Install the 'ConvertMultiBoundaries' module via the Module Store.

SET UP INSTANCES IN IP-SYMCON)

Under "Add Instance" the 'ConvertMultiBoundaries' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	Source variable to be used for calculation.
Formula 1-10	Formula to be applied to the source variable.
Limit 0-10	Limit values between which the set up formula is used.
Value	Test Value to test the formula
Calculate	Calculates the value based on the test value

The value of the source variable can be implemented inside the formula with "$Value".
Example:

```
10*$Value+20
```

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Value	Float	Contains the value calculated using the set up formula.

UMG_Calculate

Require: Symcon >= 4.3

```
float UMG_Calculate (int $InstanceID, float $Value)
```

PARAMETERS

InstanceID	ID of the Instance
Value	In die Formel einzusetzender Wert

RETURNS

Result of the calculation.

DESCRIPTION

Calculates the return of the formula for the value **Value** and returns it.

EXAMPLE

```
//Funktion in der Konfiguration: $Value/10\n$ausgabe =
UMG_Calculate(12345, 50);\nvar_dump($ausgabe) //Ausgabe ist 5
```

ValueRangeScale

Require: Symcon >= 4.3

FUNCTIONAL SCOPE

Input from input value range
Input from output value range
Conversion from an input value
Calculation on change of value of the input value

SOFTWARE INSTALLATION

Via the Module Store install the 'ValueRangeScale' module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'ValueRangeScaling' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Input variable	Variable which serves as input value
Minimum input value	The lower value limit of the input
Maximum input value	The upper value limit of the input
Minimum output value	The lower value limit of the output
Maximum output value	The upper value limit of the output

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Output	float	Output value of the calculation

VRC_Scale

Require: Symcon >= 4.3

```
bool VRC_Scale (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Scales the value of the input variable according to the set limits with a linear function.

EXAMPLE

```
VRC_Scale(12345)
```

RGBMultiplexer

Require: Symcon >= 4.2

FUNCTION SCOPE

Provides a color wheel that controls three individual R, G, B channels in the background
If the R, G, B channels change the value the new state is also transferred to the color wheel

SOFTWARE INSTALLATION

Install the RGB Multiplexer module via the Module Store.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance" the 'RGB Multiplexer' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Variable (R)	Variable for the Red Channel
Variable (G)	Variable for the Green Channel
Variable (B)	Variable for the Blue Channel
Button "Set RGB"	Sends the RGB value to the channels

STATUSVARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Color	Integer	Contains value matching color wheel

VISUALIZATION

The Visualization is used to display the variables. The color can be set via the color wheel.

RGBM_RequestStatus

Require: Symcon >= 4.2

```
bool RGBM_RequestStatus (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Determines the value of the individual R, G, B channels and sets the color variable. This function does not have to be called. This happens automatically when changing one of the variables of the R, G, B channels.

EXAMPLE

```
RGBM_RequestStatus(12345);
```

RGBM_SetRGB

Require: Symcon >= 4.2

```
bool RGBM_SetRGB (int $InstanceID, int $Red, int $Green, int $Blue)
```

PARAMETERS

InstanceID	ID of the Instance
Red	Value of the red color channel
Green	Value of the green color channel
Blue	Value of the blue color channel

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Sends the R, G, B values to the individual channels and updates the color variable

EXAMPLE

```
RGBM_SetRGB(12345, 255, 255, 255);
```

DragPointer

Require: Symcon >= 6.0
A variable that remains at the highest or lowest value until reset.

FUNCTION SCOPE

Variable that shows the highest/lowest value since the last reset
Reset via timer or manually possible

SOFTWARE INSTALLATION

Install the 'drag pointer' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Drag Pointer' module can be found using the quick filter.
More information about adding instances in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Target variable	Variable whose values are read out
Options	**Maximum** Specifies whether the drag pointer should display the highest or lowest value
Interval	Time in seconds until the drag pointer is reset to the value of the target variable.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Drag pointer	float	Displays the highest/lowest value since last reset
Reset	boolean	Association to reset the drag pointer

PROFILES

NAME	TYPE
SZ.Reset	bool

VISUALIZATION

Display of drag pointer, as well as reset of drag pointer

SZ_Reset

Require: Symcon >= 6.0

```
bool SZ_Reset (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the drag pointer instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Resets the drag pointer.

EXAMPLE

```
SZ_Reset(12345);
```

Game Collection

A collection of simple games which can be played via the Visualization.

Rock Paper Scissors	A way to play rock, paper, scissors on the Visualization.
Number Guessing	A little game where a random number has to be guessed.

ZR_Generate	Generates a new number that can be guessed

Rock Paper Scissors

Require: Symcon >= 5.0

FUNCTION SCOPE

Rock, Paper, Scissors in IP-Symcon

SOFTWARE INSTALLATION

Via the Module Store install the game collection module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'RockPaperSissors' module can be easily found using the quick search.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Your choice	Integer	Player's choice.
Computer chose	Integer	Shows the choice of the computer.
Result	String	Shows the result.

PROFILE:

NAME	TYPE
SSP.Choice	Integer

VISUALIZATION

This is where the game is played.

Number Guessing

Require: Symcon >= 5.0

RANGE OF FUNCTIONS

A little game where a random number has to be guessed.

SOFTWARE INSTALLATION

Via the Module Store install the game collection module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Number Guessing' module can be easily found using the quick search.

CONFIGURATION PAGE:

NAME	TYPE	DESCRIPTION
Minimum of the generated number	Integer	The minimum of the number that will be generated.
Maximum of the generated number	Integer	Maximum of the generated number.
Number of allowed attempts	Integer	The number of attempts the player has to guess the number.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Moves left	Integer	Shows the number of remaining attempts.
Your number is	String	Shows if the guess is greater or less than the number to guess.
Your Hint	Integer	The input field for the player's hint.

VISUALIZATION

This is where the game is played.

ZR_Generate

Require: Symcon >= 5.0

```
bool ZR_Generate (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Generates a new number that can be guessed.

EXAMPLE

```
ZR_Generate(12345);
```

Scene Control

Require: Symcon >= 5.0
The scene control saves values of variables stored in a list in scenes and can call them up again from the Visualization and mobile apps at the press of a button.
The variables to be switched must be added to the "Variables" list in the instance configuration. Once all the variables required for a scene have been added and set to the desired value, they can be added to the corresponding scene using the "Save" button (in the Visualization). Now the scene can be called up at any time with the corresponding "Execute" button. The variables are set to the previously saved values.

FUNCTION SCOPE

Allows variables stored in a list to be saved and recalled via scenes.
The currently active scene is displayed
Display and operation via Visualization and mobile apps
When a variable is replaced, scenes already created are retained and switch the new device

SOFTWARE INSTALLATION

Install the 'Scene Controll' module via the Module Store.

SET UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Scene Controll' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Scenes	Number of scenes provided.
Variables	List of variables to be switched.

STATUS VARIABLES AND PROFILES

The status variables are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

The scenes are numbered 1,2..n in ascending order.

NAME	TYPE	DESCRIPTION
Active scene	String	Displays the name of the currently active scene. If the values of the variables can be assigned to a scene, this is also displayed. If the values of the variables do not correspond to a scene, 'Unknown' is displayed
Scene 1..n	Integer	For display in the Visualization and the mobile apps. Calls "Save" or "Call".

PROFILE:

NAME	TYPE
SZS.SceneControl	Integer

VISUALIZATION

Via the Visualization the current values of the listed target variables can be stored in a Scene. Via "Execute" already saved scenes can be called.

SZS_CallScene

Require: Symcon >= 5.0

```
bool SZS_CallScene (int $InstanceID, int $SceneNumber)
```

PARAMETERS

InstanceID	ID of the Instance
SceneNumber	Number of the Scene

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

Gets the values of the scene with number **SceneNumber** stored in the scene control module with InstanceID **InstanceID** and sets the associated variables.

EXAMPLE

```
SZS_CallScene(12345, 1);
```

SZS_GetActiveScene

Require: Symcon >= 5.0

```
int SZS_GetActiveScene (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

Number of the active scene. If an unknown scene is active, 0 is returned.

DESCRIPTION

Returns the number of the scene that is currently active.

EXAMPLE

```
SZS_GetActiveScene(12345);
```

SZS_SaveScene

Require: Symcon >= 5.0

```
bool SZS_SaveScene (int $InstanceID, int $SceneNumber)
```

PARAMETERS

InstanceID	ID of Instance
SceneNumber	Number of the Scene

RETURNS

Konnte der Befehl erfolgreich ausgeführt werden, liefert er als Ergebnis TRUE, andernfalls FALSE.

DESCRIPTION

Saves the values of the variables present in the list in the corresponding scene.

EXAMPLE

```
SZS_SaveScene(12345, 1);
```

Dew Point Temperature Calculation

Calculation of the dew point with the relative humidity and room temperature.
The calculation is based on the formulas from the page Wetterochs.

FUNCTION SCOPE

Calculation of dew point temperature
Calculation of the temperature above which the risk of mold growth increases

SOFTWARE INSTALLATION

Install the 'DewPointCalculation' module via the Module Store.

SET UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'DewPointCalculation' module can be found using the quick filter.
More information about adding instances in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Room temperature	Temperature prevailing in the room in °C
Humidity	Relative humidity, which prevails

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
dew point	float	temperature, when falling below, excess water vapor will turn into water
temperature for risk of mold	float	temperature, when falling below this increases the risk of mold growth
Alarm for mold risk	boolean	Alarm, which is triggered when the temperature falls below the temperature for mold risk. It will be turned off when the room temperature is 1°C above the mold risk temperature.

Staircase Light Controls

Require: Symcon >= 5.0
After a trigger is activated, the light in the staircase turns on. If the trigger is activated repeatedly, the light remains on and the timer is reset. Only when no further triggering takes place for a predefined time will the light be switched off.

FUNCTION SCOPE

Selection of input and output variables in a list.
Selection of the duration before the light is switched off.
Specification of brightness depending on a night mode variable
.Reversed profiles are supported for both input and output variables
Possibility to display the remaining time before the light turns off.

SOFTWARE INSTALLATION

Use the Module Store to install the 'Staircase Light Controls' module.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Staircase Light Controls' module can be found using the quick filter.
More information about adding instances in the instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Input Sensors	List of input sensors, which should activate the light, e.g. motion sensors or push buttons - The light will be activated as soon as a variable is set to active. Variables with a value that is not false, 0, or "" are considered active here. If the variable has a .Reversed profile, the mentioned values are considered active.
Output variables	List of variables that are active, i.e. set to their maximum value and represent the light. If a variable has a .Reversed profile it will be switched to its minimum value. Variables of the type String are not switched. The variables are switched active when a sensor from the input sensor list is triggered.
Duration	After the selected duration elapses without another input sensor being triggered, the light will be deactivated.
Resend action	If an unreliable radio system is used, it may be necessary to resend the action at each pulse. Normally this option should remain deactivated, because continuous sending of the action for radio actuators may use up the duty cycle.
Display remaining time	If active, the remaining time until switch off is displayed in a variable.

NAME	DESCRIPTION
Update interval	The interval in which the "Remaining time" variable is updated.
Night/Day Mode	Allows to switch the selected variables to different values based on the time of day, or the ambient brightness

NIGHT/DAY MODE - NIGHT/DAY VARAIBLE

NAME	DESCRIPTION
Day/Night	A variable that indicates whether it is night or day.
Inverted	Specifies whether the value of the night mode variable should be inverted. This is necessary if the actual day variable is to be used by the location instance. This is FALSE if it is dark.
Brightness (night mode)	Specifies the brightness in percent that should be switched at night.
Brightness (Day Mode)	Specifies the brightness in percent to switch to during the day.

NIGHT/DAY-MODE-AMBIENT-BRIGHTNESS-VARIABLE

NAME	DESCRIPTION
Ambient Brightness	The variable used as ambient brightness.
Ambient Brightness Threshold	The threshold at which to switch between day and night values.
Brightness (night mode)	Specifies the brightness in percent to be switched when the brightness falls below.
Brightness (Day mode)	Specifies the brightness in percent that is to be switched when the brightness is exceeded.

STATUS VARIABLES

STATUSVARIABLEN

NAME	TYPE	DESCRIPTION
Staircase control active	Boolean	The variable indicates whether the staircase control is active
Remaining time	String	If "Show remaining time" is active, the remaining time until switch off is displayed here

THL_SetActive

Require: Symcon >= 5.0

```
bool THL_SetActive (int $InstanceID, bool $Value)
```

PARAMETERS

InstanceID	ID of the Instance
Value	TODO

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Activates or deactivates the staircase lighting control. If the staircase lighting was switched on by the control and the control is deactivated, the current control process is still completed. However, the timer is not reset when the input sensor is triggered again. The staircase lighting will therefore be switched off after the timer has expired despite the deactivated control.

EXAMPLE

```
THL_SetActive(12345, true);
```

THL_Start

Require: Symcon >= 5.0

```
bool THL_Start (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Activates the light in the staircase and starts the timer, which deactivates the light again. The timer is reset if it is called up again.

EXAMPLE

```
THL_Start(12345);
```

THL_Stop

Require: Symcon >= 5.0

```
bool THL_Stop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Deactivates the light in the staircase and the timer.

EXAMPLE

```
THL_Stop(12345);
```

Renamer

Require: Symcon >= 4.4
Enables renaming objects from the Visualization.

FUNCTION SCOPE

Allows renaming multiple objects from the Visualization.
Name changes of objects that are not made by the module are not passed to the status variables.

SOFTWARE INSTALLATION

Via the Module Store install the 'Renamer' module.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add instance" the 'Renamer' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Objects	A list to which all objects must be added that should be able to be renamed in the Visualization.

STATUS VARIABLES

STATUS VARIABLES

A status variable of type string is created for each object selected in the instance configuration. The name of the variable is the current location of the object which is renamed.

VISUALIZATION

Here the previously selected objects can be renamed.

Rain Central

Require: Symcon >= 4.2
The module calculates a weighted rain value in a selected area of the radar image.

FUNCTION SCOPE

Calculates instantaneous rainfall in a self-determined area of the rain radar.
Selection between statewide or nationwide radar images.
The radar image and the calculation takes place automatically every 15 min.

SOFTWARE INSTALLATION

Install the module 'Rain Central' via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'RainCentral' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Area	Selection of state or nationwide map section
X	X-position from the center of the rectangle
Y	Y-position from the center of the rectangle
Radius	Edge length of the rectangle in pixels

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Rain Value	Integer	Weighted value of the evaluated rain amount. 8-level weighting: Light rain = light blue = 1 -> ... -> Heavy rain = dark blue = 8
Radar image	Media	Radar image of the selected map section

VISUALIZATION

The Visualization is used to display the variables and the radar image. No further control or separate display is integrated. The radar image also shows the evaluated section of the car.

UWZ_RequestInfo

Require: Symcon >= 4.2

```
bool UWZ_RequestInfo (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Calculates the variable rain value of the rain central module with InstanceID InstanceID.

EXAMPLE

```
UWZ_RequestInfo(12345);
```

Variable Comparison

Require: Symcon >= 6.0
This module was sponsored by IB-Jaetzel.
The module creates a graph of point clouds based on one or more pairs of variables and a line through the point cloud using simple linear regression. The diagram can be output as SVG or PNG.

FEATURE

Create a diagram with one or more pairs of values
Output of the diagram as SVG or PNG
Output of slope, y-axis intercept, function and coefficient of determination of the graph

SOFTWARE INSTALLATION

Via the Module Store install the 'VariableComparison' module.

SET UP INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'VariableComparison' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

Values:

NAME	DESCRIPTION
X value	Variable selection for the X axis
Y value	Variable selection for the Y axis
point	Color selection for the point macro
Line	Color selection for the line
Aggregation Level	Level of how detailed the data is fetched from the archive

Diagram settings:

NAME	DESCRIPTION
Axes small steps	Specifies in which steps small markings should be
Axes large steps	Specifies in which steps the marking with labeling is
Width	Specifies how wide the diagram is
Height	Indicates how high the diagram is
Y - Min	Indicates the minimum value on the Y-axis

NAME	DESCRIPTION
Y - Max	Indicates the maximum value on the Y-axis
X - Min	Indicates the minimum value on the X-axis
X - Max	Indicates the maximum value on the X-axis
Digram Format	Specifies whether the diagram should be output as SVG or PNG

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Function	string	Specifies the mathematical function of the graph
b	Float	Describes the y-axis intercept of the graph
m	Float	Describes the slope of the graph
Coefficient of determination	Float	Describes how exactly the graph fits the point cloud
Start Date	Integer	Date from which the point cloud should start
End Date	Integer	Date to which the point cloud goes
Chart	String / Media	Returns the chart as PNG or SVG depending on the setting

LR_Download

Require: Symcon >= 6.0

```
void LR_Download (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

No return.

DESCRIPTION

Regenerates the diagram and outputs an address. A button in the configuration form displays the diagram in the browser.

EXAMPLE

```
LR_Download(12345);
```

LR_GenerateChart

Require: Symcon >= 6.0

```
array LR_GenerateChart (int $InstanceID)
```

PARAMETERS

InstanceID	Id of the Instance

RETURNS

SVG and PNG diagram

DESCRIPTION

Regenerates the diagram and returns it as SVG and PNG.

EXAMPLE

```
LR_GenerateChart(12345);
```

LR_UpdateChart

Require: Symcon >= 6.0

```
void LR_UpdateChart (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

No return.

DESCRIPTION

Regenerates the diagram. Sets it to the selected format and displays it in the configuration form.

EXAMPLE

```
LR_UpdateChart(12345);
```

Virtuelle Devices

The virtual devices simulate various functions of different devices.

E-Car (Simulation)	Simulates the charging process of an electric car
Heater (Simulation)	Simulates a heater
Light (Simulation)	Simulates a light
Mediaplayer (Simulation)	Simulates a MediaPlayer
PV-System (Simulation)	Simulation of a PV system
Shutter (Simulation)	Simulates a roller shutter
Thermostat (Simulation)	Simulates a thermostat
Consumption/Costs	Calculates the costs according to consumption
Counter (Simulation)	Simulates an adjustable number of counters.

E-Car (Simulation)

Require: Symcon >= 6.0
The electric car simulates the charging process of an electric car.

RANGE OF FUNCTIONS

Simulates the charging process via a wallbox
Simulates a charging process with specified power in a simplified manner

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance' the 'E-Auto (Simulation)' module can be found using the quick filter.
Further information on adding instances in the Documentation of instances

CONFIGURATION PAGE:

NAME	DESCRIPTION
Mode	Options: 'Simple', in which only the power is specified. 'Wallbox', with extensive configuration to simulate the charging process at a wallbox
Variance	Specification of how much the consumption should fluctuate
Interval	Time between updates

Mode: Simple

NAME	DESCRIPTION
Power	Maximum power needed

Mode: Wallbox

NAME	DESCRIPTION
Phases	How many phases the wallbox charges over
Split the phases into individual variables (simulates the Alfen Wallbox)	Split the phases into individual variables
Minimum current (per phase)	Minimum current per phase
Maximum current (per phase)	Maximum current per phase

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Status	Boolean	Staus, whether the loading process is currently running
Intensity	Integer	Available in Simpel mode and switches the intensity of consumption
Consumption	Float	Maximum charging power
SoC (current)	Integer	Current charge level
Power (target)	Float	Switches the consumption of the charging process
Current	Float	Available in wallbox mode and switches the current amperage of the phases. Can be divided into several variables

Heater (Simulation)

Simulates a heater.

RANGE OF FUNCTIONS

Simulates a heater.

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

The 'Heater (simulation)' module can be found under 'Add instance' using the quick filter.
Further information on adding instances can be found in the Documentation of instances
Configuration page:

NAME	DESCRIPTION
Power	Specification of how much the heater should consume
Variance	Specification of how much the consumption should fluctuate
Interval	Time interval of the updates

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Status	Boolean	Switches the heater on/off
Intensity	Integer	Switches the intensity of the heater
Consumption	Float	Displays the current consumption

Light (Simulation)

Simulates a light.

RANGE OF FUNCTIONS

Simulates a light

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Light (simulation)' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Status	Boolean	Switches the light on/off.
Brightness	Integer	Switches the brightness of the light
Color	Integer	Switches the color of the light
Color temperature	Integer	Switches the color temperature of the light

VISUALIZATION

Initially displays the light tile with several options in the tile visualization.

Mediaplayer (Simulation)

Simulates a media player.

RANGE OF FUNCTIONS

Simulates a media player

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Mediaplayer (Simulation)' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Progress	Float	Shows the progress of the song
Artist	String	Displays the artist of the current song
Volume	Integer	Displays the volume
Song	String	Displays the current song
Mute	Boolean	Indicates whether the output is deactivated/activated
Playback	Integer	Displays the current playback status
Playlist	String	JSON encoded string with the next songs and their information
Repeat	Integer	Indicates whether the playlist, song or nothing is repeated
Shuffle	Boolean	Switches the random playback of the songs on/off
Cover	Image	Shows the cover of the song

PV-System (Simulation)

Simulation of a PV system

RANGE OF FUNCTIONS

Simulation of a PV system

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'PV system (simulation)' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Power	Specification of how much the PV system should consume

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Generation	Float	Indicates how much the PV system generates

PROFILES

NAME	TYPE
WattSlider.10	float

Shutter (Simulation)

Simulates a roller shutter.

RANGE OF FUNCTIONS

Simulates a roller shutter

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Rollo (simulation)' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Blind	Integer	Height of the blind
Lamella	Integer	Height of slats

VISUALIZATION

The shutter tile is displayed in the tile visualization.

Thermostat (Simulation)

Simulates a thermostat.

RANGE OF FUNCTIONS

Simulates a thermostat

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Thermostat (simulation)' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Create logged values for the current temperature	
hour value	table with the values to be logged

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Set temperature	Float	Variable for setting the temperature
Actual temperature	Float	Current temperature

VISUALIZATION

The heating tile is displayed in the tile visualization.

Consumption/Costs

Calculates the costs with consumption in standby mode.

RANGE OF FUNCTIONS

Calculates the costs of the given consumption

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Consumption/costs' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Standby consumption	Float	Consumption in standby mode
Costs per kWh	Float	Costs per kWh
Total cost per year	Float	Total cost per year

PROFILES

NAME	TYPE
WattSlider	Float
EuroCent	Float

Counter (Simulation)

Simulates an adjustable number of counters.

RANGE OF FUNCTIONS

Simulates several meters

SOFTWARE INSTALLATION

Install the 'Virtual Devices' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Counter (simulation)' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Number of counters	Number of counters to be simulated.
Minimum value per hour	Minimum value to be added
Maximum value per hour	Maximum value to be added
Maximum distance between two subsequent values	Maximum distance between two values
Use a different profile for each variable	Each counter receives a different profile
Create graph	Creates a multigraph

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Counter x	Float	A counter variable. x stands for the number of the counter
Graph	Multigraph	A multigraph of the counters

Random Lighting

Require: Symcon >= 5.0
The module allows to switch the color value of lamps randomly between different colors.

FUNCTION SCOPE

Color variables can be added to a list
A list of colors can be selected from which can be expanded
When the module is deactivated, the selected color variables are set to the value before activation
Setting of the interval in which the colors should be changed

SOFTWARE INSTALLATION

Using the Module Store install the 'Random Lighting' module.

SET UP INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Random Lighting' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Color Variables	List of color variables to be switched
Colors	Colors from which one is randomly selected for switching
Change Interval	Interval between each switching in seconds
Simultaneous switching	defines if all color variables are set to the same color or to different ones

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Active	Boolean	Switches the module on or off

ZB_ChangeLight

Require: Symcon >= 5.0

```
bool ZB_ChangeLight (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Selects random colors and sets the selected variables to them.

EXAMPLE

```
ZB_ChangeLight(12345);
```

Energy

WORK EFFICIENCY

Efficiency calculation

ARZ_Calculation	Calculates the efficiency

OPERATING HOURS COUNTER

Determines and displays the operating time of a device

BSZ_Calculate	The operating hours variable is set to the calculated value

ENERGY DASHBOARD

An overview with several diagrams

ENERGY MANAGER

Module, which switches the consumers on and off when energy is available.

ENERGY DISTRIBUTION

ENERGY COUNTER

Calculates the consumption

Energy Counter Pulse	Converts meter values to instantaneous consumption.
EZI_Update	Updates the calculated values of the 'Energy Counter Pulse'-module with the InstanceID InstanceID.
Energy Counter Power	Calculates instantaneous and cumulative power consumption from current/power.
EZS_Update	Updates the calculated values of the Energy Counter Power module with the InstanceID.

POWER BILLING MODULE

The module provides a cost statement similar to the annual statement from the energy provider.

SAM_UpdateCalculations	Calculates all values

POWER PRICE

Reads out the current/predicted electricity prices from aWATTar, Tibber or Epex Spot (via ENTSO-E).

SPX_Update	Updates the data

CONSUMPTION PER CATEGORY

Shows the consumption in percent in category.

VIK_CalculationConsumption	Recalculates the categories

CONSUMPTION WITHIN TIMESPAN

Calculates the consumption between a start and end date

VIZ_Calculate	Recalculates consumption between start and end dates

CONSUMPTION BEHAVIOUR

The consumption behavior module displays the expected consumption.

VBV_UpdateCalculation	The function updates the predicted values

CALCULATED COUNTER

Adds positive variable changes according to adjustable rules with changes of a main variable and adds this to the value of a variable.

VM_Update	Berechnet die Steigung der sekundären Messstellen und addiert dieses mit dem Ergebnis

VIRTUAL COUNTER

Input mask for manually read meter readings.

VZ_WriteNewCounterValue	After checking, sets the current counter reading

READING (DAY)

The meter reading can be displayed for a selected date

ZST_Calculate	Recalculates the value for the selected date

METER OVERFLOW

Meters which can overflow can be displayed as meters.

Work Efficiency

Require: Symcon >= 6.3
The efficiency is calculated.

FUNCTION SCOPE

The efficiency for the past 365 days
The efficiency for the past 30 days

SOFTWARE-INSTALLATION

Using the Module Store, install the 'work efficiency' module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Work efficiency' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE

NAME	DESCRIPTION
Thermal energy (kWh)	Variable logged as counter
Electrical Energy (kWh)	Variable which is logged as counter

STATUS VARIABLES

The status variables are created automatically. Deleting individual ones can lead to malfunctions.

NAME	TYPE	DESCRIPTION
SPF Month	Float	Stands for the sessional performance facto on month
SPF Year	Float	Stands for the sessional performance factor on year

ARZ_Calculation

Require: Symcon >= 6.3

```
bool ARZ_Calculation (int $InstanceID)
```

PARAMETERS

InstanceID	Instance ID

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Calculates the efficiency

EXAMPLE

```
ARZ_Calculation(12345)
```

Operating Hours Counter

Require: Symcon >= 5.2
With the help of the operating hours counter module, the operating time of a device can be determined and displayed.

FUNCTION SCOPE

Displaying the hours a device is active
Adding the device by a Boolean type variable
Displaying the costs for the current period, the last period and prediction for the end of the period
Possibility to calculate the costs with a dynamic price

SOFTWARE INSTALLATION

Through the Module Store install the 'Operating Hours Counter' module.

SETTING UP INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Operating Hours Counter' module can be found using the quick filter.

More information about adding instances in the Instances documentation.

CONFIGURATION PAGE

NAME	DESCRIPTION
Active	Determines if the invoice is updated based on the set interval
Source	The Boolean type variable that indicates the activity status of a device, where true is considered active. To calculate the operating hours this variable must be logged
Level	The level specifies the start of the period that is considered (start of day, week, month, year) as well as the aggregation level
Update interval	The interval in minutes in which the operating time is recalculated
Cost calculation	Defines if the cost calculation will be executed
Price type	Option whether the price is fixed or dynamically calculated by a variable
Price	The price which is calculated per operating hour
Calculate	Calculates the operating time with all specified parameters

If the option "Dynamic" is set in the price type, a logged variable is required which contains the cost price.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLEN

NAME	TYPE	DESCRIPTION
Operating hours	float	The calculated operating hours of the source variable in the selected period
Cost of the current period	float	The calculated cost of the current period
Cost of the last period	float	The calculated cost of the last completed period
Prediction at the end of the period	float	Prediction of the costs at the end of the current period

PROFILE

NAME	TYPE
BSZ.OperatingHours	float

VISUALIZATION

The Visualization displays the operating hours.

BSZ_Calculate

Require: Symcon >= 5.2

```
void BSZ_Calculate (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

No return.

DESCRIPTION

The operating hours variable is set to the calculated value.

EXAMPLE

```
BSZ_Calculate(12345);
```

Energy Dashboard

Require: Symcon >= 8.1
The energy dashboard serves as an overview of several meters with several diagrams.



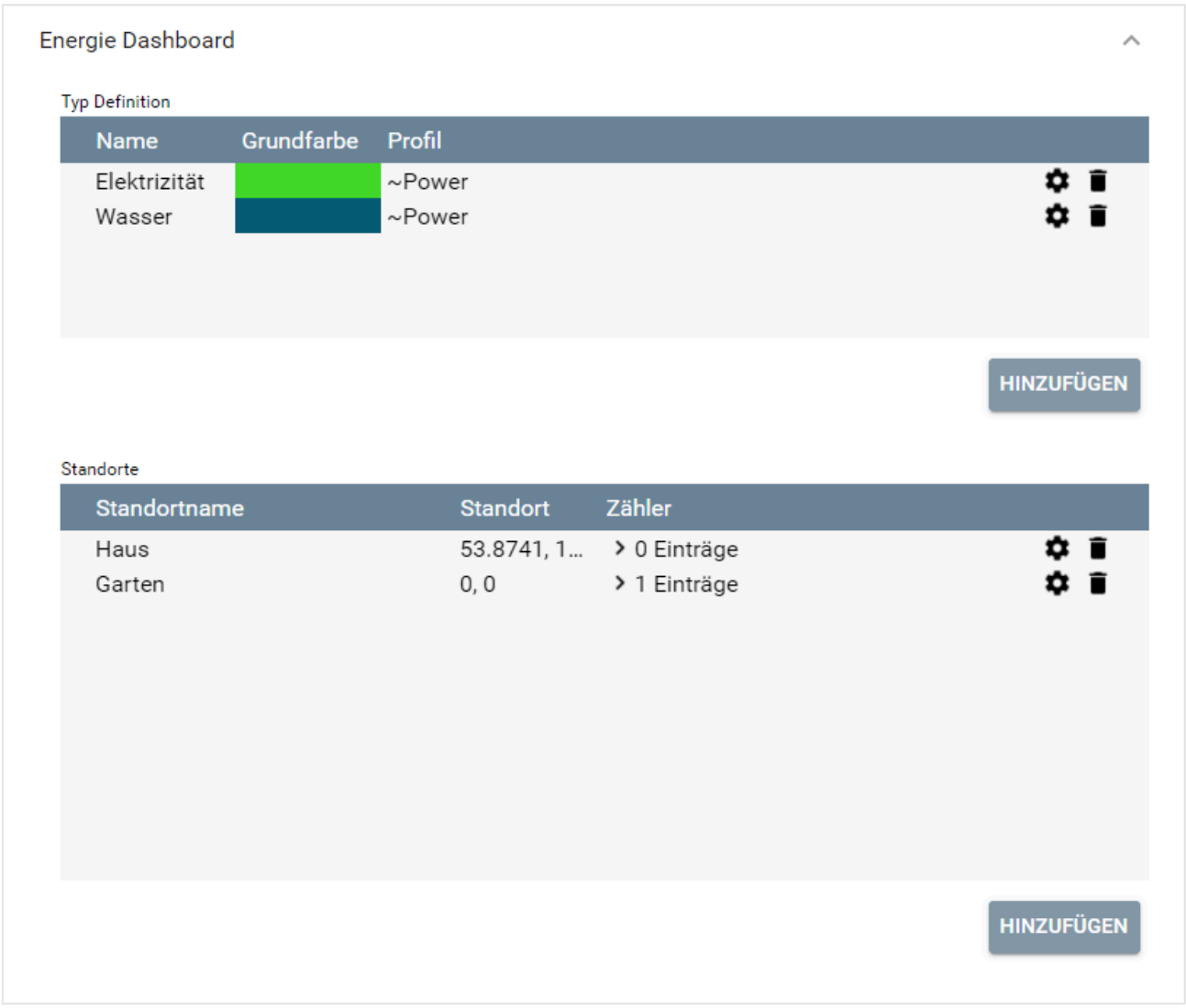
The Energy Dashboard is a paid extension that can be purchased for any existing Symcon license. The extension can be purchased directly in the Shop.



The enterprise features are explained below. The features for all users are still under development.

SETTING UP THE ENERGY DASHBOARD

To use the energy dashboard, it must first be set up. To do this, the instance of a 'tile visualisation' is called up in the console. Here you should ensure that the checkbox with the label 'Show energy dashboard' is activated in the 'Appbar' section. This will display the lightning bolt and thus the access in the visualisation.
The types and areas can now be defined in the 'Energy dashboard' section.

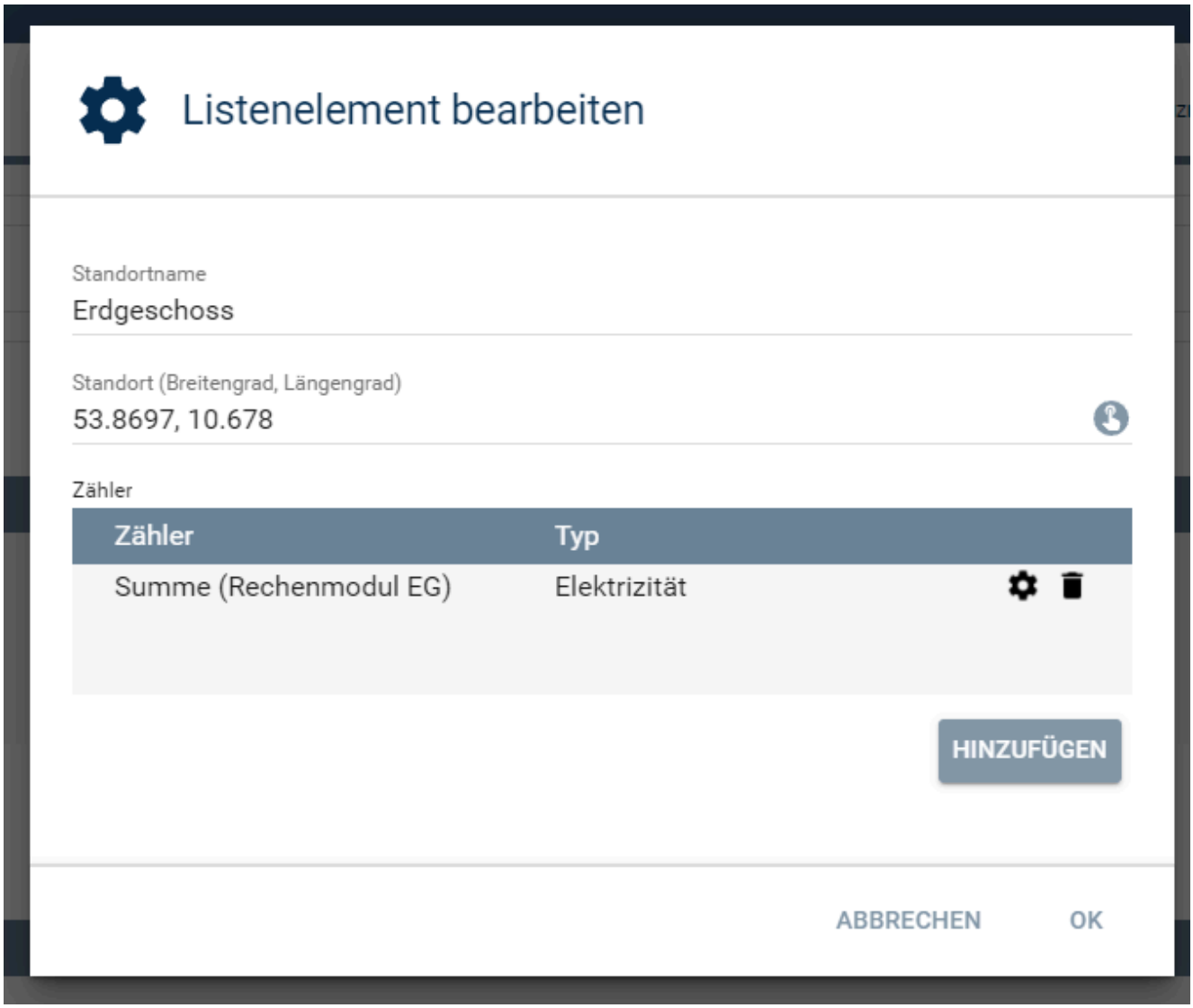


DEFINITION OF TYPES

Initially, the types electricity, water and gas are defined.

NAME	TYPE	DESCRIPTION
Name	Free text	The name of the type, which can be freely specified by the user.
Basic colour	Colour selection	Selection of a basic colour for the diagrams. All areas of the same type are displayed in a variation of the colour in the diagrams.
Profile	Profile selection	Selection of a profile. This allows the values of the diagrams to be formatted with a suffix/prefix.

CONFIGURATION OF AREAS



The areas are set up as a tree. Click on Add to create an area at the top level.
Click on an area and then click on Add to create an area below the selected area.
Each area has the following configurations:

NAME	TYPE	DESCRIPTION
Area name	Free text	Freely assignable name for the area. This will be displayed later in the energy dashboard.
Location	Coordinates selection	Optional selection of the coordinates of the area.
Meters	List	Configuration of the relevant meters

CONFIGURATION OF THE COUNTERS

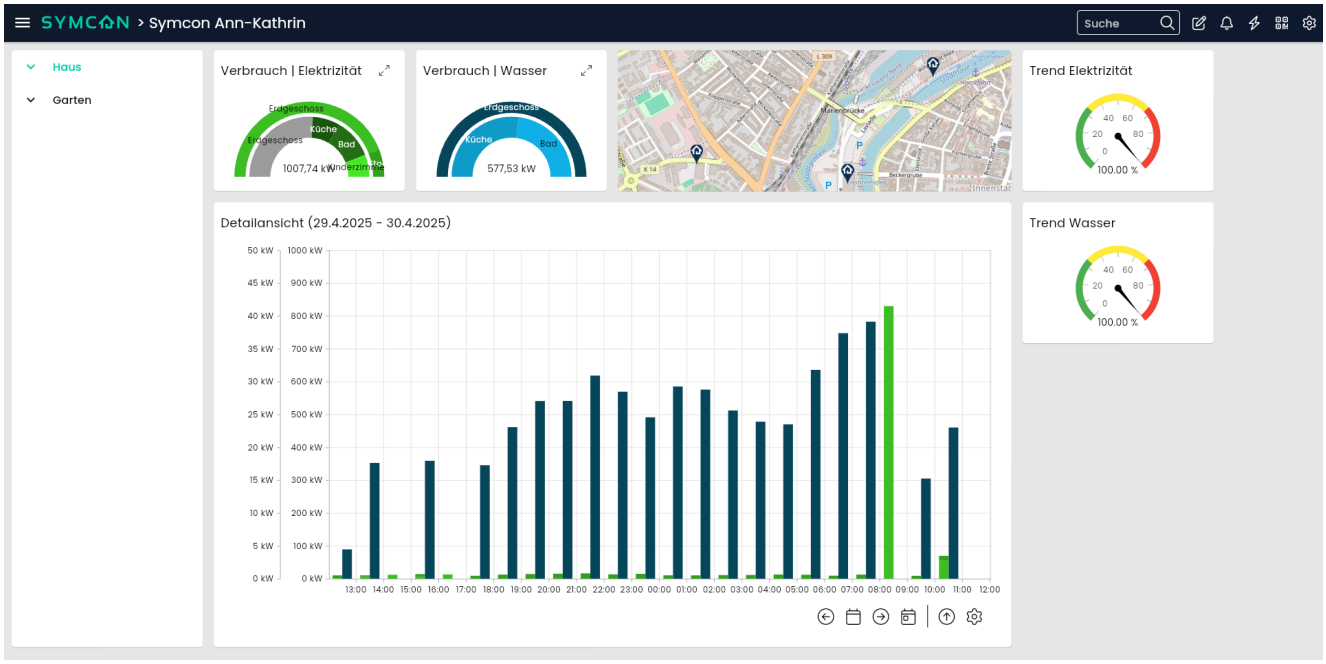
The counters can be added optionally.

NAME	TYPE	DESCRIPTION
Counter	Variable selection	Selection of the counter variable that provides the aggregated values for this area.
Type	Type selection	The type of counter based on the previously defined types

UTILISATION OF THE ENERGY DASHBOARD

If the energy dashboard has been activated, a lightning bolt icon appears in the app bar of the visualisation. Click on it to open the energy dashboard.

THE ENERGY DASHBOARD



TREE

The tree reflects the configuration of the areas by name.
The first area is initially selected. If there is an arrow next to the area, the area can be expanded, allowing you to access subordinate areas.
Clicking on the various areas updates the values of the diagrams and the map.

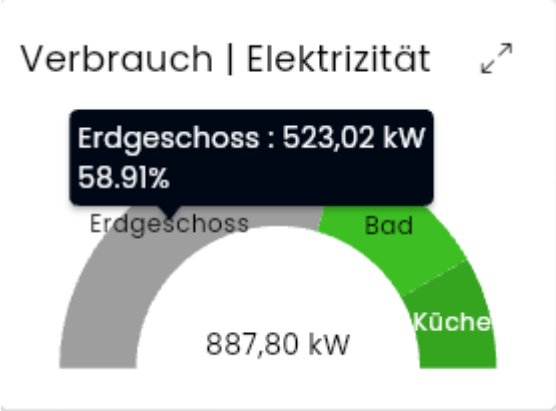
DIAGRAMS

When calling up the energy dashboard, the current day is selected as the time period and the time span hour.

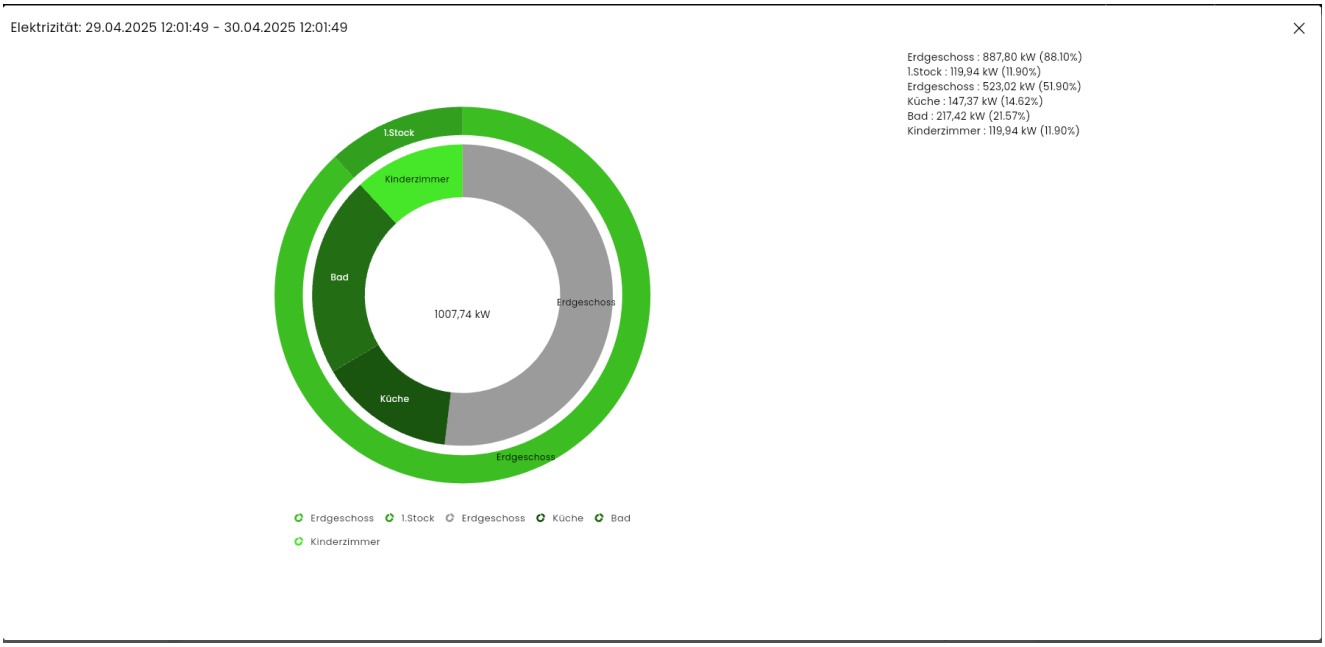
Detail view

The selected time period is displayed in the detailed view.

Consumption diagram



The consumption diagram shows the selected period with the time span of the day. The outer doughnut shows the selected counter, if necessary, and the counters directly below it. The inner doughnut is displayed if there is still a level below the selected area.



Click on the maximise button to enlarge the doughnut and display a detailed list of the entries.

Conditional display of counters

Counters are displayed conditionally in the detailed view and the doughnut chart.

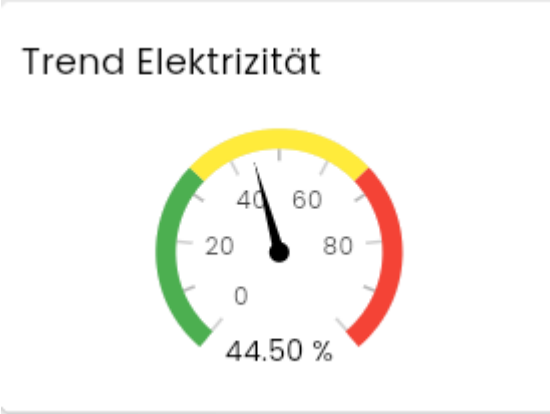
If the selected area has a counter and no areas below it, this is displayed.

If the selected area has no counter and no areas below it, it is not displayed.

If the selected area has areas below it, the area counters below it are displayed.

If the area has a counter and its total does not match the subordinate area counters, the difference is displayed in grey in addition to the subordinate area counters.

Trend diagram



The trend diagram shows the percentage relationship between the top area and the selected area.

MAP



The map lists the configured coordinates. If no coordinates are configured for the selected area, the subordinate areas are searched for coordinates. This means that there can also be several markers on the map. If no area is found, a text is displayed instead of the map. OpenStreetMap is used for the map.
Translated with DeepL.com (free version)

Energy Manager

Require: Symcon >= 6.3
The module automatically switches variables when the source energy falls below a certain level.

FUNCTIONAL SCOPE

Switching a device on when sufficient energy is available
Switching a device off when there is no longer sufficient energy available
Overnight charging of devices to ensure that an electric vehicle or energy storage is charged in the morning
Cheap charging to automatically activate devices when electricity prices are low
Restriction of devices in accordance with §14a EnWG and §9 EEG
Management of energy storages to compensate for deficits
Various calculation options

SOFTWARE INSTALLATION

Via the Module Store install the 'Energy Manager' module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Energy Manager' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Mode	Selection of calculation option **Absolute**: Devices are switched on until the source energy is reached and switched off when the power exceeds the source energy **Relative**: Devices are switched on when the source energy is above zero and switched off when the source energy falls below zero
Hysteresis	Tolerance at which switching does not take place
Constellation for Surplus/Deficit	Selection of whether surplus and deficit are specified via a shared variable or two separate variables (only for Relative mode, since Symcon 8.1)
Available power (W)	Reference variable that contains the absolute available power for the Energy Manager (only for absolute mode)
Surplus (W)	Reference variable containing the current surplus (only for Relative mode)
Deficit (W)	Reference variable containing the current deficit (only for Relative mode and configuration "Two variables, one per function", since Symcon 8.1)

NAME	DESCRIPTION
Invert	Selection of whether available power or surplus is positive or negative
Update Mode	**With Timer**: The status is updated at a fixed interval **On Source Change**: The status is always updated when Available power or Surplus is updated
Interval	Update interval of status (only for update mode With Timer)
Energy prices	Reference variable that contains the current electricity prices, provided by the module Power Price or another module that uses its format. This variable is only required for night-time charging or night-time runtime. If it is not selected, night-time charging or night-time runtime cannot be used.
Consumers	List of all consumers, see Consumers
Energy Storages	List of all energy storages, see Energy Storages
Electric Vehicles	List of all electric vehicles, see Electric vehicles
Configuration for §14a and §9	Area to configure restrictions according to §14a EnWG and §9 EEG, see Configuration for §14a and §9 (since Symcon 8.1)

CONSUMERS

NAME	DESCRIPTION
Consumer	The variable to switch the consumer. If the variable is of type integer or float, the Energy Manager attempts to scale it if necessary. With Boolean, the consumer can only be switched on and off.
Maximum Usage	Consumption in watts when the consumer is completely switched on.
Current Usage (Optional)	Variable with the current usage of the device in watts. If none is set, the consumption is calculated based on the Consumer variable and the Maximum Usage (since Symcon 8.1)
Name (Optional)	The displayed name of the consumer. If no name is specified, the name of the consumer variable is used.
Condition (Optional)	If a condition is specified, the consumer is only activated if the condition defined here is fulfilled.
Hint for blocked by condition (Optional)	If the condition is not fulfilled, this text is displayed in the visualization. If no hint is defined, a standard text is displayed.
Minimum Runtime	If the consumer is activated, it runs for at least the time defined here in seconds. The Energy Manager can only deactivate the consumer again once the minimum runtime has expired after the initial activation. If the consumer is scalable, it can be scaled appropriately if there is not enough energy available, but it is never deactivated completely.
Follow-up Time	If there is no longer enough energy available for the consumer, it is only deactivated if the energy is not available for the duration defined here in seconds. In this way, short gaps in the available energy can be bridged. If the consumer is scalable, it can be scaled appropriately if there is not enough energy available, but is never deactivated completely.

NAME	DESCRIPTION
If minimum runtime/follow-up time is active but condition is not fulfilled...	Define the behavior of the device if the condition is no longer met during the minimum runtime or follow-up time **...stop immediately**: The device is deactivated at the next status update even though the time is still running **... only stop once minimum runtime/follow-up time has elapsed**: The current time continues to run. The device is only deactivated after it has expired. (only if both a condition is set and minimum or follow-up time are positive, since Symcon 8.1)

ENERGY STORAGES

NAME	DESCRIPTION
Template	Selection of a template via which the technical parameters of the device can be entered automatically. If "User-defined" is selected, all parameters are released and can be adjusted manually, otherwise they are locked and set by the template (since Symcon 8.1)
Instance	Selection of the instance matching the selected template. After selecting an instance, all parameters of the template are entered in the configuration. In some templates, specific parameters remain editable, as they are not explicitly contained in the template. In this case, corresponding information appears. (only if the template is not set to "User-defined", since Symcon 8.1)
Energy storage controllable?	Specify whether the energy storage device has variables for switching **Controlled directly**: The variables can be switched directly. If the Energy Manager wants to make adjustments, the corresponding variables are switched **Controlled indirectly by omitting energy**: The variables cannot be switched, but the current status can be read. If the Energy Manager wants to make changes to the energy storage, this is done indirectly, for example by deliberately releasing energy and not consuming it, so that the energy storage's automatic system uses it for charging (since Symcon 8.1)
Constellation for Charge/Discharge	Selection of whether charge and discharge are controlled via a shared variable or two separate variables (only if energy storage is controllable, since Symcon 8.1)
Charge Variable	The variable to switch the charging of the energy storage. If the variable is of type integer or float, the Energy Manager attempts to scale it as required. With Boolean, charging can only be activated or deactivated.
Discharge Variable	The variable to switch the discharging of the energy storage device. If the variable is of type Integer or Float, the Energy Manager attempts to scale it as required. With Boolean, discharging can only be activated or deactivated. (only if energy storage is controllable and configuration "Two variables, one per function", since Symcon 8.1)

NAME	DESCRIPTION
Invert	Specify whether discharging is represented as a positive or negative variable value. Charging is represented accordingly by the opposite sign. (only if energy storage is controllable and configuration "One combined variable for both", since Symcon 8.1)
Control Discharge to avoid purchasing energy	If this switch is set, the Energy Manager activates the discharge of this energy storage if there is a deficit and the energy storage still has charge (only if energy storage is controllable, since Symcon 8.1)
Variable for switching between charge/discharge (Optional)	If the device is not only controlled via the variable for charge/discharge, but also has a status variable that must be switched to a specific value for charging or discharging, it can be specified here. (only if energy storage is controllable, since Symcon 8.1)
Value for charging	Value to which the Variable for switching between charge/discharge must be switched for charging (only if Variable for charging/discharging is set, since Symcon 8.1)
Value for discharging	Value to which the Variable for switching between charge/discharge must be switched for discharging (only if Variable for charging/discharging is set, since Symcon 8.1)
Level Variable	Variable that contains the current charge level. The minimum value of the variable is interpreted as completely empty and the maximum value as completely full. This means that both percentage and absolute representations can be used.
Maximum Usage	Usage in watts when the energy storage is fully charging.
Current Charge/Feed-In (Optional)	Variable with the current charge or feed-in of the energy storage. This variable is optional for controllable energy storage devices. If it has not been set, the current value is calculated based on the charge/discharge variable and the maximum usage (since Symcon 8.1)
Invert Current Charge/Feed-In?	Specifies whether charging is positive and feed-in is negative or vice versa (only if Current Charge/Feed-In is set, since Symcon 8.1)
Start charging when below	If the energy storage charge level falls below this percentage value, the energy storage is charged with available energy until it reaches the "Stop charging when above" charge level.
Stop charging when above	If the charge level reaches this percentage value, the energy storage is no longer charged until it falls below the "Start charging when below" charge level again.
Continue charging when every other device is done	If this switch is set, the energy storage is charged regardless of the "Start charging below" and "Stop charging above" settings if there are no other consumers and any excess energy would otherwise remain unused.
Name (Optional)	The displayed name of the energy storage. If no name is specified, the name of the instance provided for the template is used. If no instance is selected either, the name of the charge variable is used. If no charge variable is set due to the energy storage not being controllable, the name of the Current Charge/Feed-In variable is used.

NAME	DESCRIPTION
Capacity (Optional, required for overnight charge)	The capacity of the energy storage in kWh. This value is only required for overnight charge. If it is not set, overnight charge cannot be used for the energy storage.
Condition (Optional)	If a condition is specified, the consumer is only activated if the condition defined here is fulfilled.
Note for blocking by condition)Optional)	If the condition is not fulfilled, this text is displayed in the visualization. If no note is defined, a standard text is displayed.
Minimum Runtime	If the energy storage is charging, it runs for at least the time defined here in seconds. The Energy Manager can only deactivate the energy storage again once the minimum runtime has expired after the initial activation. If the energy storage is scalable, it can be scaled appropriately if there is not enough energy available, but it is never deactivated completely.
Follow-up Time	If there is no longer enough energy available for the energy storage, it is only deactivated if the energy is not available for the duration defined here in seconds. In this way, short gaps in the available energy can be bridged. If the energy storage is scalable, it can be scaled appropriately if there is not enough energy available, but is never deactivated completely.
If minimum runtime/follow-up time is active but condition is not fulfilled...	Define the behavior of the device if the condition is no longer met during the minimum runtime or follow-up time **...stop immediately**: The device is deactivated at the next status update even though the time is still running **... only stop once minimum runtime/follow-up time has elapsed**: The current time continues to run. The device is only deactivated after it has expired. (only if both a condition is set and minimum or follow-up time are positive, since Symcon 8.1)

ELECTRIC VEHICLES

NAME	DESCRIPTION
Template	Selection of a template via which the technical parameters of the device can be entered automatically. If "User-defined" is selected, all parameters are released and can be adjusted manually, otherwise they are locked and set by the template (since Symcon 8.1)
Instance	Selection of the instance matching the selected template. After selecting an instance, all parameters of the template are entered in the configuration. In some templates, specific parameters remain editable, as they are not explicitly contained in the template. In this case, corresponding information appears. (only if the template is not set to "User-defined", since Symcon 8.1)
Power (Target)	The variable to switch the target power of the electric vehicle in watts.
Unit for Power	Specify whether the target power is specified in watts or amperes (since Symcon 8.1)
On/Off Variable (Optional)	If the device is not only switched via the target value, but also has a variable that switches the device On or Off, it can be specified here. (since Symcon 8.1)

NAME	DESCRIPTION
Phases	Selection of how many phases the electric vehicle supports.
Set power	Select whether the power is switched per phase or in total (only if at least 2-phase, since Symcon 8.1)
Supports switching between 1-phased and multi-phased charging	The switch indicates whether the electric vehicle supports switching the phases from the above selection to one phase and back. (only if at least 2-phase)
Min. charging current per phase	Minimum charging current per phase in amperes
Max. charging current per phase Charging current per phase	Maximum charging current per phase in amperes
Current Usage (Optional)	Variable with the current usage of the device in watts. If none is set, the consumption is assumed to be equal to the target power (since Symcon 8.1)
Name (Optional)	The displayed name of the electric vehicle. If no name is specified, the name of the instance provided for the template is used. If no instance is selected either, the name of the power variable is used.
Delay	If the target power of the electric vehicle is adjusted by the Energy Manager, no further switching operations are carried out for the duration specified here in seconds.
State of Charge	Variable that contains the current state of charge. The minimum value of the variable is interpreted as completely empty and the maximum value as completely full. This means that both percentage and absolute representations can be used. The variable is only required for night charging. If it is not set, overnight charging is not possible.
Capacity	The capacity of the electric vehicle in kWh. This value is only required for overnight charging. If it is not set, night charging cannot be used for the vehicle.
Energy per km	The energy in Wh that the electric vehicle consumes per kilometer. This value is only required for overnight charging. If it is not set, overnight charging cannot be used for the vehicle. (only if target for night charging should be specified in kilometers)
Condition (Optional)	If a condition is specified, the electric vehicle is only charged if the condition defined here is fulfilled.
Note for blocking by condition (Optional)	If the condition is not fulfilled, this text is displayed in the visualization. If no note is defined, a standard text is displayed.
Minimum Runtime	If the electric vehicle is charging, it runs for at least the time defined here in seconds. The Energy Manager can only deactivate the electric vehicle again once the minimum runtime has expired after the initial activation. If the electric vehicle is scalable, it can be scaled appropriately if there is not enough energy available, but it is never deactivated completely.
Follow-up Time	If there is no longer enough energy available for the electric vehicle, it is only deactivated if the energy is not available for the duration defined here in seconds. In this way, short gaps in the available energy can be bridged. If the electric vehicle is scalable, it can be scaled appropriately if there is not enough energy available, but is never deactivated completely.

NAME	DESCRIPTION
If minimum runtime/follow-up time is active but condition is not fulfilled...	Define the behavior of the device if the condition is no longer met during the minimum runtime or follow-up time **...stop immediately**: The device is deactivated at the next status update even though the time is still running **... only stop once minimum runtime/follow-up time has elapsed**: The current time continues to run. The device is only deactivated after it has expired. (only if both a condition is set and minimum or follow-up time are positive, since Symcon 8.1)

CONFIGURATION FOR §14A AND §9 (SINCE SYMCON 8.1)

NAME	DESCRIPTION
Limitation Variable (§14a)	Selection of a variable of type Boolean. If this is set, usage is limited in accordance with §14a EnWG. While the variable is active, the entire behavior of the Energy Manager is also saved in a media file
Variable for Maximum Feed-In (§9)	Selection of a variable of type Integer or Float. If this is set, it specifies the maximum feed-in in accordance with §9 EEG. If a restriction exists, the Energy Manager will restrict the feed-in to remain compliant with §9 EEG.
Energy production	List of all energy producers, see Energy production
Simulate restrictions according to §14a	If this switch is set, a restriction is applied in accordance with §14a EnWG, regardless of the actual value of the variable for restriction
Current consumption	Listing of all consumers and the relevant parameters according to §14a EnWg to verify that the Energy Manager is behaving correctly

ENERGY PRODUCTION (SINCE SYMCON 8.1)

NAME	DESCRIPTION
Energy Production Variable	Variable that specifies the current energy production of the device in watts
Restriction Variable	Variable via which the maximum feed-in of the device can be restricted in watts
Maximum Production	Maximum production of the device in watts

STATUS VARIABLES AND PROFILES

The status variables are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

Here, the term "device" covers consumers, energy storages and electric vehicles.

NAME	TYPE	DESCRIPTION
Planned usage	Float	The current consumption of all entered devices calculated by the Energy Manager
PMin (§14a)	Float	The calculated value PMin for applying the restriction in the sense of §14a EnWG (only if Variable for restriction (§14a) is set, since Symcon 8.1)
Protocol for restrictions (§14a/ §9)	Media - Document	Protocol with all operations while a restriction by §14a EnWG is active as well as all restrictions made according to §9 EEG (since Symcon 8.1)

All other status variables are created per device. If there are 3 devices in the module, 12 to 27 variables are created depending on the configuration.

NAME	TYPE	DESCRIPTION
Priority	Integer	Order in which the devices are viewed; lower priority values are preferentially activated
Status	Link	Link to the status of the device, i.e. "Consumer" for consumers, "Charge Variable" for energy storages and "Power (Target)" for electric vehicles
Mode	Integer	If "Automatic", the device is managed by the Energy Manager, otherwise it can be overridden with this
Current Consumption	Link	Linking to the variable with the current consumption of the device (only if variable for the device was set, since Symcon 8.1)
Charging	Link	Linking to the variable with the current charge level of the energy storage device (only if variable for the energy storage device was set, since Symcon 8.1)
Operating Mode	Integer	Specifies whether the energy storage device is currently charging, discharging or passive (only for energy storage devices, since Symcon 8.1)
Condition	String	Current status of the condition of the device
Locked	Boolean	If this variable is set, the device is locked due to the minimum runtime or run-on time and must not be deactivated. The variable is only created if the device has a minimum runtime or run-on time.
Locked until	Integer	If the device is locked, this variable contains the time when this lock will be removed. The variable is only created if the device has a minimum runtime or follow-up time.
Overnight Charge/Overnight Runtime	Boolean	Activates overnight charging (for energy storage and electric vehicle) or overnight runtime (for consumers). This variable is only created if the device supports overnight charging or overnight runtime.
Overnight Charge: Range	Float	Desired range after overnight charging for the electric vehicle. This variable is only created for electric vehicles that support overnight charging and has its goal for overnight charge defined in km.

NAME	TYPE	DESCRIPTION
Overnight Charge: Range available at	Integer	Time at which the desired range should be available the next day. This variable is only created for electric vehicles that support overnight charging and has its goal for overnight charge defined in km.
Overnight Charge: Charge	Float	Desired percentage charge level for the energy storage after overnight charging. This variable is only created for energy storages or electric vehicles that support overnight charging. In addition, an electric vehicle must have its goal defined in percentage.
Overnight Charge: Charge available at	Integer	Time at which the desired charge should be available the next day. This variable is only created for energy storages or electric vehicles that support overnight charging. In addition, an electric vehicle must have its goal defined in percentage.
Overnight Runtime: Duration	Float	Duration in seconds that the consumer should be active overnight. This variable is only created for loads that support overnight runtime.
Overnight Runtime: Done until	Integer	Time at which the desired overnight runtime should be completed. This variable is only created for consumers that support night-time runtime
Cheap Charge/Runtime: Active	Boolean	Activates Cheap Charge (for energy storage devices and electric vehicles) or Cheap Runtime (for consumers). This variable is only created if a variable for energy prices is set. Non-switchable energy storage devices do not support this function. (since Symcon 8.1)
Cheap Charge/Runtime: Maximum Price per kWh	Float	Specification of the maximum price in cents up to which Cheap Charge/Runtime should be activated (since Symcon 8.1)

PROFILE

NAME	TYPE
EO.Mode	Integer
EO.Priority	Integer
EO.Range	Float
EO.Percentage	Float
EO.Duration	Float
EO.Cents	Float
EO.BatteryMode	Float

Associations EO.Mode

NAME	DESCRIPTION
Active	The variable is switched to active and is excluded from consideration

NAME	DESCRIPTION
Inactive	The variable is switched off and is excluded from consideration
Automatic	The variable is switched according to priority and energy.

FUNCTIONALITY

OPTIMIZATION

During operation of the Energy Manager, the energy is optimized in individual steps so that the energy produced is used as sensibly as possible. Depending on the configuration, steps are taken at a fixed time interval or when the available power is updated. Only one appliance is switched in each step so that differences between entered and actual consumption can be detected and managed at an early stage. In the Energy Manager, each device can be assigned a priority by the user.
In each step, the system initially checks whether there is a "gap" in the activated devices, i.e. whether devices with a lower priority are active although devices with a higher priority are still inactive and enough energy would be available to activate them. In this case, a device with a lower priority is deactivated in order to release the energy for devices with a higher priority in the next step.
Otherwise, the current surplus is determined. If this is positive, a device with the highest priority is determined, which can absorb the surplus. The device is then activated or scaled accordingly.
If the surplus is negative, a device with the lowest priority is deactivated or scaled down to compensate for the missing energy.

OVERNIGHT CHARGE/OVERNIGHT RUNTIME

Overnight charge or overnight runtime is considered as soon as the day ends according to the Location Control. If overnight charge or overnight runtime is activated for devices, the Energy Manager calculates how long the corresponding device must be activated. This duration is compared with the energy prices from the current time to the corresponding target time and determines at which times this runtime can be used most favorably. If such a cheapest price is available at an optimization step, the device is activated at maximum scaling, otherwise it is deactivated.

CHEAP CHARGE/CHEAP RUNTIME

Cheap Charge or Cheap Runtime can be used to activate a device as soon as the electricity price falls below a user-defined value. If the electricity price is sufficiently low during an update, one of the affected devices is primarily activated, regardless of its set priority and the current surplus. Only when all corresponding devices are already active will an update cycle proceed as usual.

RESTRICTION ACCORDING TO §14A ENWG

If the restriction according to §14a becomes active, an update is carried out immediately. This occurs in addition to the usual update at intervals or when the source variable is updated. If the restriction is active during an update, the permissible maximum consumption

"PMin" for energy storage systems and electric vehicles is calculated. If the energy storage systems and electric vehicles are currently consuming more, they are immediately regulated down. Devices that are not locked by minimum runtime or follow-up time are preferentially deactivated. However, if this is not sufficient, devices with active minimum runtime or follow-up time will also be deactivated. For devices that are not in automatic mode, the value before the restriction is saved and restored as soon as the restriction ends. If an attempt is made to manually activate energy storage systems or electric vehicles above PMin, the switching operations are reversed. However, they are saved so that they can be switched to the desired value when the restriction ends. For more information on §14a EnWG, see here.

RESTRICTION ACCORDING TO §9 EEG

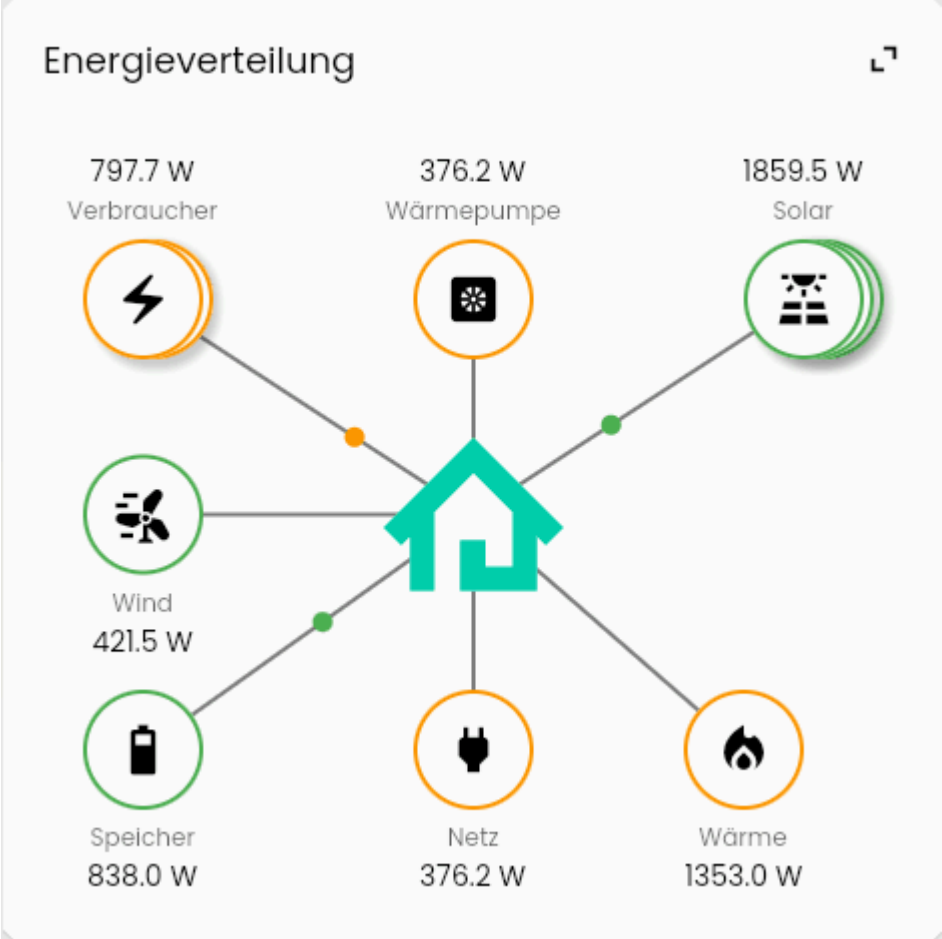
If a restriction according to §9 is present, i.e., below 100%, the energy producers are restricted accordingly after each update step. For this, the maximum permissible energy production is calculated. This is composed of the permissible feed-in, i.e., the restriction value multiplied by the maximum total production, and the currently self-consumed energy, which is calculated across all devices. If the current restriction of the energy producers is above the permissible restriction, it is reduced. Preference is given to applying a restriction to devices that are not producing anyway. If this is not sufficient, energy producers that are currently producing energy are restricted. For more information on §9 EEG, see here.

VISUALIZATION

The Energy Manager has its own presentation in the visualization.

Energy Distribution

Require: Symcon >= 7.0
The energy distribution provides an overview of the energy generated/consumed and fed in/referred.



RANGE OF FUNCTIONS

Graphical representation of the energy flow between different devices
Division of devices according to producer and consumer
Special displays for grid and storage
Calculation of total consumption, total generation, consumption and surplus
Grouping of several appliances of the same type

SET UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Energy distribution' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation

CONFIGURATION PAGE

All selected variables should offer the values in the same unit to ensure error-free display and function.

NAME	DESCRIPTION
Calculate summarized values	If activated, status variables are created that calculate generation, consumption, reference and surplus values based on the added variables.

CONFIGURATION LIST

NAME	DESCRIPTION
Type	The selection of the type determines whether the device is treated as a producer or consumer and offers corresponding further configuration options
Name	The name used for the display. If this remains empty, the variable name is used.
Group	All devices of the same type are grouped together in the visualization.
Additional information	A variable whose value can be displayed in brackets after the actual variable value. For example, the percentage charge level of an e-vehicle or a meter reading
Calculation type	Either the sum or the mean value is calculated from the values of the additional information variables in a group. (only if 'Grouping' is activated)

TYPE CONFIGURATION

Wind/Solar

NAME	DESCRIPTION
Generation	The variable that specifies the generated energy.

Consumer/Wallbox/Heat/Heat Pump

NAME	DESCRIPTION
Consumer	The variable that specifies the energy consumed.

Grid

NAME	DESCRIPTION
Configuration type	Determines whether the value results from one or two variables.
Supply / Surplus	If the value of the variable is positive, it is counted as supply. If it is negative, it is counted as surplus. (only for configuration type 'Single variable')

NAME	DESCRIPTION
Inverted	Inverts the behavior of the individual variable. (only for 'Configuration type' 'Single variable')
Supply	The variable that specifies the supply. (only for 'Configuration type' 'Two variables')
Surplus	The variable that specifies the surplus energy. (only for 'Configuration type' 'Two variables')

Storage

NAME	DESCRIPTION
Configuration type	Determines whether the value results from one or two variables.
Charge / Discharge	If the value of the variable is positive, it is counted as charging. If it is negative, it is counted as discharging. (only for configuration type 'Single variable')
Inverted	Inverts the behavior of the individual variable. (only for 'Configuration type' 'Single variable')
Charge	The variable that specifies the charged energy. (only for 'Configuration type' 'Two variables')
Discharge	The variable that specifies the discharged energy. (only for 'Configuration type' 'Two variables')

LAYOUT

The house in the middle is fixed as the central point of energy exchange. The objects around it are arranged according to their position in the list above. The positioning is as follows:

Top left
Top right
Bottom left
Bottom right
Top center
Bottom center
Left center
Right center

STATUS VARIABLES

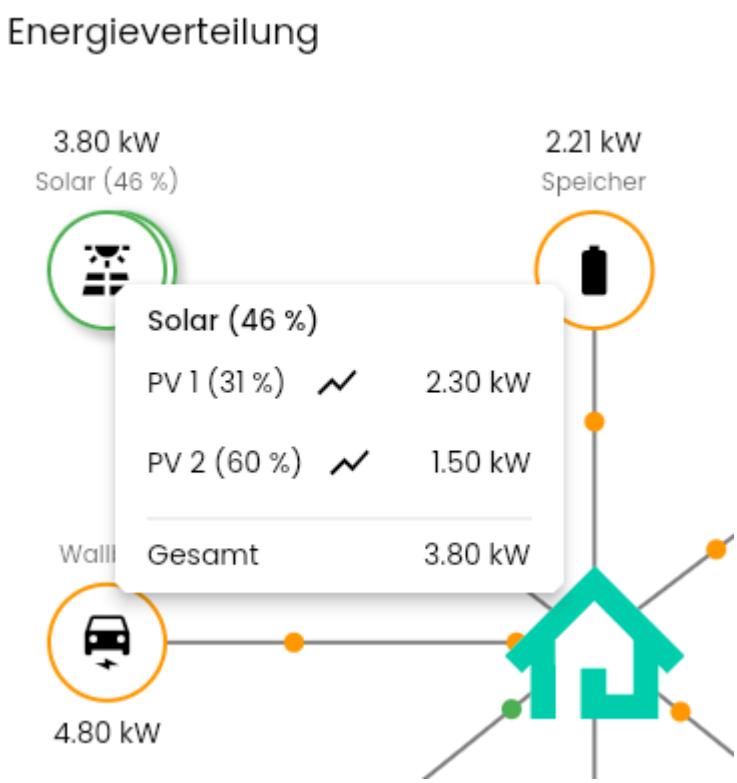
The status variables are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

The status variables are only created if the corresponding setting has been activated on the configuration page

NAME	TYPE	DESCRIPTION
Consumption (total)	Float	The sum of the consumption of the type Consumer, Wallbox, Heat pump and Heat.
Generation (Total)	Float	The sum of solar and wind generation.
Draw	Float	The difference between energy consumed and energy generated.
Surplus	Float	Difference between generated energy and consumed energy.

VISUALIZATION



Each device and each group can be clicked to display a breakdown of the individual values. Once the selected variables have been logged, the corresponding diagram can be opened directly.
If required, a resource-saving mode can be activated by clicking on the house in the middle, which replaces the moving points with static arrows.

The energy distribution does not have its own presentation in the WebFront

Energy Counter

Calculates the consumption

Energy Counter Pulse	Converts meter values to instantaneous consumption.
EZI_Update	Updates the calculated values of the 'Energy Counter Pulse'-module with the InstanceID InstanceID.
Energy Counter Power	Calculates instantaneous and cumulative power consumption from current/power.
EZS_Update	Updates the calculated values of the Energy Counter Power module with the InstanceID.

Energy Counter Pulse

Require: Symcon >= 4.2
The module calculates the instantaneous and cumulative power consumption via an electricity meter (e.g. S0 connection).

FUNCTION SCOPE

Calculates the instantaneous power consumption in watts and the total power consumption in kWh.
Adjustability of the pulses of the counting device.
Adjustability of interval frequency in seconds for recalculation of consumption

SOFTWARE INSTALLATION

Install the 'Energy Counter Pulse' module via the Module Store.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Energy Counter Pulse' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	Source variable (pulse counter) to be used for the calculation.
Pulses	How many pulses the device sends per kilowatt. This must be taken from the respective operating manual of the device.
Interval	In which seconds interval the device should recalculate automatically. (Note: Intervals that are set too short (faster than incoming pulses), can lead to strong fluctuations in the indication of the current consumption. It has no influence on the calculated total consumption. => Recommendation: min. 300 seconds)

STATUSVARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Current	Float	Specification in W

NAME	TYPE	DESCRIPTION
Counter	Float	Specification in kWh
Last Value (Temporary)	Float	Auxiliary variable for last value. Required for difference calculation.

EZI_Update

Require: Symcon >= 4.2

```
bool EZI_Update (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Updates the calculated values of the 'Energy Counter Pulse'-module with the InstanceID InstanceID.

EXAMPLE

```
EZI_Update(12345);
```

Energy Counter Power

Require: Symcon >= 4.2
The module calculates the instantaneous and cumulative power consumption via an electricity meter (current or power).

FUNCTION SCOPE

Calculates instantaneous power consumption in watts and total power consumption in kWh.
Adjustability of type and source of the counting device.
Adjustability of the interval frequency in seconds to recalculate the consumption if the source variable has not changed.

SOFTWARE INSTALLATION

Install the 'Energy Counter Power' module via the Module Store.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Energy Counter Power' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Type	Is the source variable of type power (W) or current (A).
Source	Source variable to be used for the calculation.
Voltage	Voltage (V) to be used for conversion.
Interval	At which seconds interval the calculation should be automatically recalculated if the source variable has not changed.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	DESCRIPTION
Type	Is the source variable of type power (W) or current (A).
Source	Source variable to be used for the calculation.

NAME	DESCRIPTION
Voltage	Voltage (V) to be used for conversion.
Interval	At which seconds interval the calculation should be automatically recalculated if the source variable has not changed.

EZS_Update

Require: Symcon >= 4.2

```
bool EZS_Update (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Updates the calculated values of the Energy Counter Power module with the InstanceID InstanceID.

EXAMPLE

```
EZS_Update(12345);
```

Power Billing Module

Require: Symcon >= 5.0
The module provides a cost statement similar to the annual statement from the energy provider.

SCOPE OF FUNCTIONS

Generates a cost statement using some key data.

SOFTWARE INSTALLATION

Via the Module Store install the 'Power Billing' Module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Power Billing Module' can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	The variable of the main meter, which is logged as meter
Base price	The base price
Energy price	The energy price
Meter reading date	Date of last meter reading
Last meter reading	The meter reading on the last meter reading date
Planned consumption	The planned electricity consumption/year

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Energy price	Float	Energy price including basic price
Days until next meter reading	Integer	Days until next meter reading (1 year starting from last meter reading)

NAME	TYPE	DESCRIPTION
Days since last meter reading	Integer	Days since last meter reading
Meter Reading(Target)	Float	The target value of today's meter reading based on the planned consumption
Planned Consumption/Day	Float	The planned consumption per day based on the planned annual consumption
Average consumption of the last 30 days	Float	The average value of the consumption of the last 30 days
Deviation	Float	The deviation of the actual meter reading from the target value
Credit/Repayment	Float	The amount of repayment or credit calculated by the deviation

PROFILE:

NAME	TYPE
SAM.EuroRating	Float
SAM.PowerPrice	Float
SAM.Calendar	Integer

VISUALIZATION

All important calculated values are displayed here.

SAM_UpdateCalculations

Require: Symcon >= 5.0

```
bool SAM_UpdateCalculations (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Calculates all values displayed in the webfront based on the data on the configuration page.

EXAMPLE

```
SAM_UpdateCalculations(12345);
```

Power price

Require: Symcon >= 7.1
Reads out the current/predicted electricity prices from aWATTar, Tibber or Epex Spot (via ENTSO-E).

RANGE OF FUNCTIONS

Reads out electricity prices from various providers
Visual history of market data
Manual input of taxes, surcharge and base price

SOFTWARE INSTALLATION

Install the 'Electricity price' module via the Module Store.
Alternatively, add the following URL via the Module Control

SET UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Electricity price' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Provider	Electricity provider from which the data is to be obtained
Postal code	Postal code to determine electricity prices (only for provider "Tibber")
ENTSO-E Security Token	Security token from ENTSO-E, which is used for the query. A token can be requested free of charge. The process is explained here: Instructions (only for provider "EPEX Spot (via ENTSO-E)")
Market	Selection of country (not for provider "Tibber")
Base price	Base price of the electricity (not for provider "Tibber")
Surcharge	Percentage surcharge on the market price (not for provider "Tibber")
Tax rate	Tax rate to be added to the electricity (not for provider "Tibber")

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Market data	String	Data to display the chart
Current price	Float	Current electricity price

PROFILES

NAME	TYPE
Cent	Float

VISUALIZATION

In the tile visualization, the module offers a bar chart that shows the course of the electricity price.

SPX_Update

Require: Symcon >= 7.1

```
bool SPX_Update (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

Updates the data.

EXAMPLE

```
SPX_Update(12345)
```

Consumption per Category

Require: Symcon >= 6.3
Calculates consumption in percent per freely selectable category according to specified start and end dates. The calculation is based on daily aggregation.

FUNCTION SCOPE

Selection of multiple variables
Free text for category

SOFTWARE INSTALLATION

Install the 'Consumption per category' module via the Module Store.

INSTANCE SETUP IN IP-SYMCON

Under 'Add instance' the 'Consumption per category' module can be found using the quick filter.
More information about adding instances in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Interval	Interval in which the calculation is executed. 0 = Disabled
Sources	List of variables and categories

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Category	Float	A variable is created per category, in which the percentage consumption is displayed. The category name is internally reduced to A-Z, a-z, 0-9 and _ (underscore). This means that the category "Test" and "Test €" will also be categorized to "Test".
Start time	Integer	Date from when the calculation should start
End time	Integer	Date until when the calculation should go

VIK_CalculationConsumption

Require: Symcon >= 6.3

```
bool VIK_CalculateConsumption (int $InstanceID)
```

PARAMETERS

InstanceID	Instance Id

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Recalculates the percentage of the category.

EXAMPLE

```
VIK_CalculateConsumption(12345);
```

Consumption within Timespan

Require: Symcon >= 4.2

FUNCTION SCOPE

Calculates consumption for a time span based on the aggregation of the selected source variable.
When the time span is changed, the consumption is recalculated
Different detail lines for start and end date
Adjustable interval for updating the calculation

SOFTWARE INSTALLATION

Install the module 'Consumption within Timespan' via the Module Store.

SET UP INSTANCES IN IP-SYMCON

Under "Add instance" the 'Consumption within Timespan' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	Source variable to be used for consumption
Level of Detail	Determines how precisely the start and end time can be set (date, time, date/time)
Use interval to update	If active, an interval will be used to update the calculation
Interval	The interval in minutes at which the calculation will be updated



If the date is selected in level of detail, the start date with the time 00:00 and the end date with the time 23:59:59 is used.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Start	Integer	Start date/start time for consumption (seconds are ignored)
End	Integer	End date/end time for consumption (seconds are ignored)
Consumption	Float	Consumption between start and end date

VIZ_Calculate

Require: Symcon >= 4.2

```
bool VIZ_Calculate (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Recalculates consumption between start and end dates.

EXAMPLE

```
VIZ_Calculate(12345);
```

Consumption Behaviour

Require: Symcon >= 6.0
The module calculates the probable consumption for the period based on an outdoor temperature variable and a meter variable using linear regression. The more values are available for the outdoor temperature and the meter, the more accurately the expected consumption can be determined.

FUNCTIONAL SCOPE

Calculates a probable consumption from two variables

SOFTWARE INSTALLATION

Via the Module Store install the 'Consumption Behaviour' module.

SET UP INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Consumption Behaviour' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Variable for outdoor temperature	Logged variable for outdoor temperature
Variable for counter	Logged variable for counter
Period	Period for which the calculation is to be performed
Limit	Maximum number of records to be used for the regression. 0 = No limit
Interval	Time interval of the timer in which the variable should be calculated again
Calculate	Button to recalculate the variable

For the counter variable, there must be at least 2 records for 3 consecutive periods for a calculation to take place. For period day, the hourly aggregation of the variables is used. For all other periods the daily aggregation of the variables is used.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Expectation of current period	float	Displays the expected consumption based on the outdoor temperature of the current period
Expectation of the last period	float	Displays the expected consumption based on the outdoor temperature of the last period
Extrapolation of the current period	float	Displays the extrapolated consumption of the current period
Extrapolation of the last period	float	Displays the extrapolated consumption of the last period
Value of the current period	float	Displays the current consumption of the current period
Value of the last period	float	Displays the consumption of the last period
Percent of the current period	float	Shows how far the extrapolated consumption differs from the expected value in percent
Percent of the last period	float	Shows how far the extrapolated consumption differs from the expected value in percent
Coefficient of determination of the current period	float	Accuracy of the expectation calculation of the current period
Coefficient of determination of the last period	float	Accuracy of the expectation calculation of the last period

MATHEMATICAL FORMULAS

Expectation is calculated using simple linear regression. Mathematically Explained. Here the sum of the positive result is calculated from the calculation m *temperature average (depending on the period)* b. m and b are determined by the linear regression.
The extrapolation again consists of the average value of the period multiplied by the period length.

VBV_UpdateCalculation

Require: Symcon >= 5.0

```
bool VBV_UpdateCalculation (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

The function updates the predicted values.

EXAMPLE

```
VBV_UpdateCalculation(12345);
```

Calculated Counter

Require: Symcon >= 5.5
Calculates positive variable changes with changes of a main variable according to adjustable rules and adds this to the value of a variable.
The "Smart Energy Box" module was developed in cooperation between Symcon GmbH, Stark Elektronik GmbH and Biberach University of Applied Sciences (IGE Institute) as part of the state project EnMa HAW: Concept for automation-supported energy management at non-university universities in Baden-Württemberg.

FUNCTIONAL SCOPE

Selection of a logged variable as primary measuring point
Selection of any number of variables as secondary measuring points
Adjustable whether the changes of the secondary measuring points should be added or subtracted to the changes of the primary measuring points

SOFTWARE INSTALLATION

Install the 'Calculated Counter' module via the Module Store.

SET UP INSTANCES IN IP-SYMCON

Under 'Add instance' the 'Calculated Counter' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Primary measuring point	Selection of a logged variable
Secondary measuring point	Multiple selection of variables
Operation	Add/Subtract Option whether to add or subtract the positive change of the secondary measurement point to the primary measurement point
Variable	Selection of a logged variable as secondary measuring point

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Result	Float	Calculated Result

VM_Update

Require: Symcon >= 5.5

```
bool VM_Update (int $InstanceID, float $PrimaryDelta)
```

PARAMETERS

InstanceID	ID of the Instance
PrimaryDelta	Positive Veränderung der Hauptmessstelle

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

The slope of the secondary measuring points is added or subtracted depending on the selected option. The result is added together with the passed value to the value of the 'Result' variable.

EXAMPLE

```
VM_Update(12345, 5.3);
```

Virtual Counter

Require: Symcon >= 6.0
The module provides an input mask for manually read meter readings. It accepts the manual entry, checks it for plausibility and transfers the value to a counter variable. To check for plausibility, a limit value can be set.

FUNCTIONAL SCOPE

Check for plausibility
Display of the last entered counter reading
Input mask for new meter reading

SOFTWARE INSTALLATION

Install the 'Virtual Counter' module via the Module Store.

SET_UP_THE_INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Virtual Counter' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Limit	Number by which a new counter value may increase at most. If the limit value is 0, it will not be checked
Require confirmation to accept	If enabled, a script must be executed to trigger the plausibility check
Activate Logging	Button, which activates the logging of the counter variable. It will be hidden if logging is enabled.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Current counter reading	Float	Displays the last entered counter reading
New counter reading	String	Input variable for new counter reading

NAME	TYPE	DESCRIPTION
Set safe new counter reading: Enter new counter reading	Boolean	Appears if the new counter reading exceeds the current one by the limit value. The value can still be accepted with confirmation

PROFILE

NAME	TYPE
VZ.Confirm	Boolean
VZ.NewCounterReading	String

SCRIPTS

NAME	DESCRIPTION
Set Meter Reading	Executes the plausibility check and pushes the value into the vaiable "Current Meter Reading" if successful

WEBFRONT

Via the Visualization the new counter value can be entered and the script can be executed, furthermore the last accepted counter value is displayed.

VZ_WriteNewCounterValue

Require: Symcon >= 6.0

```
void VZ_WriteNewCounterValue (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

The function does not return any value.

DESCRIPTION

Checks whether the value in variable "New counter reading" is negative, the "New counter reading" is less than the "Current counter reading" or the "New counter reading" exceeds the "Current counter reading" with the "Limit value". If these do not apply, the new value is set in "Current counter reading".

EXAMPLE

```
VZ_WriteNewCounterValue(12345);
```

Reading (Day)

Require: Symcon >= 4.2
The module allows to select a date and then displays the respective (First/Last) counter value of that day.

FUNCTION SCOPE

When the date is changed, the First/Last read counter value of the logged variable is output, depending on the value selection.

SOFTWARE INSTALLATION

Via the Module Store install the module 'Reading (Day)'.

SET UP INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Reading (Day)' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	Source variable to be used as data source.
Value	First/Last value of the day.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Counter value	integer/float	Value for the selected date.

VISUALIZATION

The Visualization is used to display the variable. A date can be selected for which the counter reading (first/last of the day) will be displayed.

ZST_Calculate

Require: Symcon >= 4.2

```
bool ZST_Calculate (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Recalculates the value for the selected date.

EXAMPLE

```
ZST_Calculate(12345);
```

Meter Overflow

Require: Symcon >= 4.2
The module represents overflowing counters as continuous counters.

FUNCTION SCOPE

Calculates the total value of a variable and counts it up even though the device has an overflow.

SOFTWARE INSTALLATION

Install the 'Meter Overflow' module via the Module Store.

SET UP INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Meter Overflow' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	Source variable to be used for the calculation.
Maximum value	From which value an overflow takes place. The maximum value which the device will count.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Counter	Float	Continuously incrementing value.

Visualizations

Tile Visualization	Functions to control the tile visualization on the client devices

VISU_OpenObject	opens a specific object in the Tile Visualization.
VISU_PostNotification	sends a push notification to the tile visualization
VISU_PostNotificationEx	sends a push notification to the tile visualization

WebFront Visualization	Functions for controlling the WebFront on the client computers

WFC_AudioNotification	sends an audio message to the WebFront
WFC_OpenCategory	Opens a category as a popup in the WebFront
WFC_PushNotification	sends a push message to the mobile apps
WFC_Reload	reloads the WebFront on the client computer
WFC_SendNotification	sends a small message to the WebFront
WFC_SendPopup	sends a large message to the WebFront
WFC_SwitchPage	switches the tab in the WebFront

Tile Visualization

Require: Symcon >= 7.0



A general tutorial can be found in the section Tile Visualization.

FUNCTION	DESCRIPTION
Send Notification	Push notifications, can be sent to all connected devices and assigned a notification type.
Send advanced notification	Push notifications, can be sent to all connected devices and be assigned a custom icon and sound.

VISU_OpenObject

Require: Symcon >= 8.2

```
void VISU_OpenObject (int $InstanzID, int $ObjektID, string $TokenList)
```

PARAMETERS

InstanzID	ID of the Tile Visualization
ObjektID	ID of the object to be displayed in the visualization.
TokenList	List of tokens of the devices on which the object can be opened. Can be left blank to open the object on all devices.



The TokenList parameter is not yet supported and should be passed as an empty string.

RETURNS

If the command could be executed successfully, it returns **TRUE** as the result, otherwise **FALSE**.

DESCRIPTION

Shows the object with the ID **ObjektID** in all open instances of the visualization with the ID **InstanzID**. If there are device tokens in the **TokenList**, the display of the object is restricted to these devices. If the selected object is a category, this is opened. All other objects are displayed as a maximized tile in full screen.

EXAMPLE

```
VISU_OpenObject(12345, 54321, '');
```

VISU_PostNotification

Require: Symcon >= 7.0

```
mixed VISU_PostNotification (int $InstanceID, string $Title, string
$Text, string $Type, int $TargetID)
```

PARAMETERS

InstanceID	ID of the tile visualization
Title	Title of the message (maximum 32 characters). Can also be empty.
Text	message text (up to 256 characters).
Type	Currently, the default sounds and icons of the device are used regardless of the type.
TargetID	The ID of the object that will be opened when tapping the message

RETURNS

If the command could be executed successfully, it returns the ID of the notification as the result, otherwise **FALSE**.

DESCRIPTION

Sends a push message with **Title**, **Text** to all devices assigned to the visualization with the ID **InstanceID**.



The TargetID object must be available within the specified tile visualization!



If messages should be sent to further groups (special case: also single devices), another page frame visualization must be created per group

Please note that push notifications are forwarded encrypted via the IP-Symcon server to Apple/Google, and this implies a certain delay time. If there are too many messages in a row, Apple/Google's services will automatically delay the deliveries, which IP-Symcon cannot influence

The use of this feature requires, as already described in the description of the visualization, a valid subscription. You can extend this subscription at any time via our Shop. In addition, the device must be registered in IP-Symcon. You can find more information about this under the item Notification Control

EXAMPLE

```
VISU_PostNotification(12345, 'Weather', 'It's raining soon', 'Info',
54321);
```

VISU_PostNotificationEx

Require: Symcon >= 7.0

```
mixed VISU_PostNotificationEx (int $InstanceID, string $Title, string $Text, string $Icon, string $Sound, int $TargetID)
```

PARAMETERS

InstanceID	ID of the tile visualization
Title	Title of the message (maximum 32 characters). Can also be empty.
Text	message text (up to 256 characters).
Icon	If not empty, then the icon is displayed. See Icons
Sound	Sound played when receiving the message. (alarm bell, boom, buzzer, connected, dark, digital, drums, duck, full, happy, horn, inception, kazoo, roll, siren, space, trickling, turn)
TargetID	The ID of the object that will be opened when tapping the message

RETURNS

If the command could be executed successfully, it returns the ID of the notification as the result, otherwise **FALSE**.

DESCRIPTION

Sends a push message with **Title**, **Text**, **Icon**, **Sound** to all devices assigned to the visualization with the ID **InstanceID**.



The TargetID object must be available within the specified tile visualization!



If messages should be sent to further groups (special case: also single devices), another page frame visualization must be created per group

Please note that push notifications are forwarded encrypted via the IP-Symcon server to Apple/Google, and this implies a certain delay time. If there are too many messages in a row, Apple/Google's services will automatically delay the deliveries, which IP-Symcon cannot influence

The use of this feature requires, as already described in the description of the visualization, a valid subscription. You can renew this at any time via our Shop. In addition, the device must be registered in IP-Symcon. You can find more information about this under the item Notification Control

EXAMPLE

```
VISU_PostNotificationEx(12345, 'Front Door', 'Movement detected',
'Alert', 'alarm', 54321);
```

WebFront Visualization

A general description can be found in the WebFront Visualization area

From IP-Symcon version 2.2 it is possible to control the WebFront from IP Symcon and send messages to the WebFront. The following functions are available for the user to influence the WebFront.

FUNCTION	DESCRIPTION
Open Popup	To view larger messages a pop can be opened which almost fills the whole screen and displays important messages that do not fit in a small notification.
Reload WebFront	Loading the WebFront completely new (like a reload of the page in the browser (F5)).
Select WebFront Tab	Selects the tab with the specified ID. This feature is useful e.g. to jump during a movement at the front door to door camera tab.
Send Audio Notification	Audio notifications can be sent to the WebFront, which are then played audibly.
Send Notification	Small reports can be sent which are displayed with an icon, header and text on the right side in WebFront. (Similarly Growl/Snarl)
Send Push Notification	Push notifications can be sent to the mobile apps (iOS/Android), which are displayed on the mobile device and signalled audibly.

WFC_AudioNotification

Require: Symcon >= 3.0

```
bool WFC_AudioNotification (int $InstanceID, string $Title, int
$MediaID)
```

PARAMETERS

InstanceID	ID of the WebFront visualization
Title	Audio message title
MediaID	ID of the media object, which is of type "Sound"

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends an audio message with **Title** to the WebFront assigned to the configurator with the ID **InstanceID**. The audio message appears as a small box in the upper right corner of the WebFront and is displayed for the playback duration.



Please note that not all browsers support all formats:
http://www.w3schools.com/tags/tag_audio.asp



To be able to play "media files" in the WebFront, these must be made available as links in the "visible" area of the WebFront. These can also be stored invisibly. If this is not done, the WebFront cannot access the file and an error, that the instance cannot be found, is displayed.



Under iOS this function has no effect, as Apple disables the playback of sound files in the WebBrowser!



Under Android Browser and Android Chrome this function has no effect, as Google prevents the playback of sound files in the WebBrowser!

Under Android Chrome, the automatic playback of sound files can be activated. To do this, "Disable motion request for media playback" has to be enabled under about:flags and then the browser has to be restarted.

EXAMPLE

```
WFC_AudioNotification(12345, 'Gong!', 55541); //55541 is the ID of the
media object in IP-Symcon
```

WFC_OpenCategory

Require: Symcon >= 4.1

```
bool WFC_OpenCategory (int $InstanceID, int $CategoryID)
```

PARAMETERS

InstanceID	ID of the WebFront visualization
CategoryID	Category ID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Opens the category with the ID **CategoryID** as a popup. All sub-objects of the category are displayed in the popup.



For security reasons, the category to be opened must be available within the WebFront. If this is not the case, the category cannot be displayed.

EXAMPLE

```
WFC_OpenCategory(12345, 45678);
```

WFC_PushNotification

Require: Symcon >= 3.4

```
bool WFC_PushNotification (int $InstanceID, string $Title, string
$Text, string $Sound, int $TargetID)
```

PARAMETERS

InstanceID	ID of the WebFront visualization
Title	Title of the message (maximum 32 characters). Can also be empty.
Text	Text of the message (maximum 256 characters).
Sound	Evaluated in the apps since version 3.0.6
TargetID	It is possible to jump directly to an object when opening the push message. (since version 5.0)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a push message with **Title**, **Text** to all mobile devices associated with the configurator with ID **InstanceID**. Alternatively, one of the following sounds can be played:

alarm
bell
boom
buzzer
connected
dark
digital
drums
duck
full
happy
horn
inception
kazoo
roll
siren
space

trickling
turn



The TargetID object must be available within the specified WebFront configurator, otherwise an error message will appear! (TargetID is not inside the allowed security context)



If messages are to be sent to additional groups (special case: also individual mobile devices), another WebFront configurator must be created for each group.



Please note that push notifications are encrypted and forwarded to Apple/Google via the IP Symcon server, and this implies a certain delay time. If there are too many messages in a row, the Apple/Google services will automatically delay the deliveries, which IP-Symcon has no influence on.



The use of this feature requires a valid subscription, as already mentioned in the description of the mobile apps. This can be extended at any time via our Shop . In addition, the device must be registered in IP-Symcon. Further information about this can be found under the item Notification Control

EXAMPLE

```
WFC_PushNotification(12345, 'Warning', 'It will be raining soon!', '',
0 );
```

WFC_Reload

Require: Symcon >= 2.2

```
bool WFC_Reload (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the WebFront configurator

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Loads the WebFront that is reassigned to the configurator with the ID **InstanceID**.

EXAMPLE

```
WFC_Reload(12345);
```

WFC_SendNotification

Require: Symcon >= 2.2

```
bool WFC_SendNotification (int $InstanceID, string $Title, string
$Text, string $Icon, int $Timeout)
```

PARAMETERS

InstanceID	ID of the WebFront configurator
Title	Titel of the message
Text	Text of the message. Can also be empty.
Icon	If not empty, then the icon will be displayed. See Icons
Timeout	Time to be displayed in seconds. 0 = Only hide when the user clicks on the message.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a short message with **Title**, **Text** and **Icon** to the WebFront that is reassigned to the configurator with the ID **InstanceID**. The message appears as a small box in the upper right corner of the WebFront and is displayed for **Timeout** seconds.

EXAMPLE

```
WFC_SendNotification(12345, 'Test', 'This is a nice Test', 'Speaker',
4);
```

WFC_SendPopup

Require: Symcon >= 2.2

```
bool WFC_SendPopup (int $InstanceID, string $Title, string $Text)
```

PARAMETERS

InstanceID	ID of the WebFront configurator
Title	Title to be displayed
Text	Text to be displayed (HTML is filtered!)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a large message to the WebFront that is reassigned to the configurator with the ID **InstanceID** assigned. The content **Text** and **Titel** will appear.

EXAMPLE

```
WFC_SendPopup(12345, "Test", "A nice message");
```

WFC_SwitchPage

Require: Symcon >= 2.2

```
bool WFC_SwitchPage (int $InstanceID, string $PageName)
```

PARAMETERS

InstanceID	ID of the WebFront configurator
PageName	Name of tab (e.g. item1234)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Changes in WebFront, that is reassigned to the configurator with the ID **InstanceID**, to the tab with the ID **PageName**.

EXAMPLE

```
WFC_SwitchPage(12345, "item1234");
```

Voice assistents

Amazon Alexa	Module to connect Amazon Alexa
Expert Options	Expert Options of Amazon Alexa
Television	Television in Amazon Alexa
Generic Switch	Generic Switches in Amazon Alexa
Generic Slider	Generic Sliders in Amazon Alexa
Speaker	Speaker in Amazon Alexa
Speaker (Muteable)	Speaker in Amazon Alexa
Light (Dimmer)	Light in Amazon Alexa, that can be dimmed
Light (Expert)	Expert light in Amazon Alexa
Light (Color)	Colored Light in Amazon Alexa
Light (Switch)	Light in Amazon Alexa, that can be switched on or off
Mediaplayer	Mediaplayer in Amazon Alexa
Shutters	Shutters in Amazon Alexa
Lock	Locks in Amazon Alexa
Scenes	Scenes in Amazon Alexa
Scenes (deactivatable)	Deactivatable Scenes in Amazon Alexa
Temperature Sensor	Temperature Sensors in Amazon Alexa
Thermostat	Thermostats in Amazon Alexa
Google Assistant	Module to connect Google Assistant
Expert Options	Expert Options of Google Assistant
Generic Switch	Generic Switches in Google Assistant
Light (Dimmer)	Light in Google Assistant, that can be dimmed
Light (Expert)	Expert light in Google Assistant
Light (Color)	Colored Light in Google Assistant
Light (Switch)	Light in Google Assistant, that can be switched on or off
Shutter	Shutters in Google Assistant
Scenes	Scenes in Google Assistant
Scenes (deactivatable)	Deactivatable Scenes in Google Assistant
Thermostat	Thermostats in Google Assistant

Amazon Alexa

Require: Symcon >= 5.0

DESCRIPTION

The Amazon Alexa module enables the control of devices via Amazon Alexa. This includes the voice control via Echo Dots.

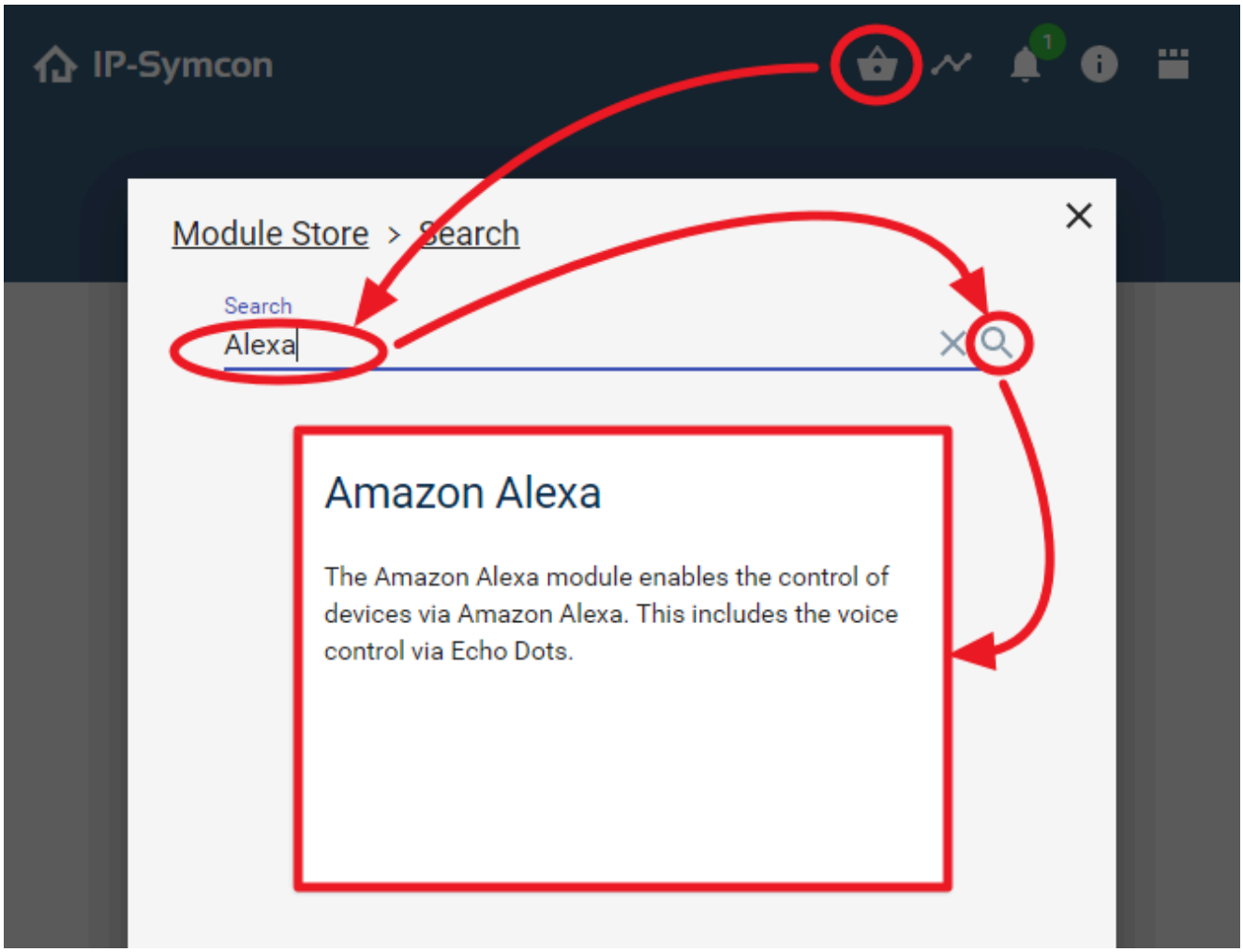
INTEGRATION IN IP-SYMCON

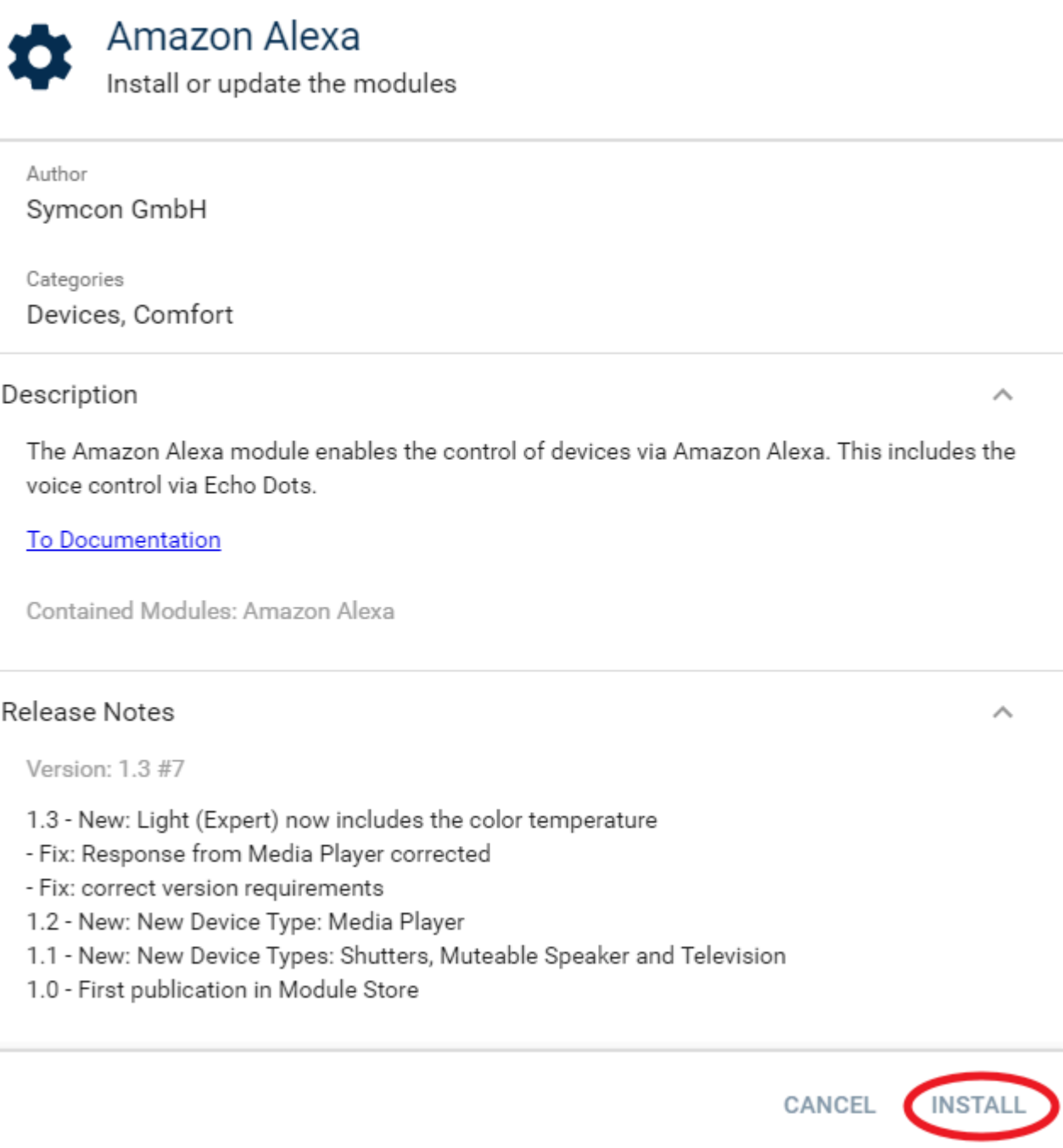


Using the Amazon Alexa module requires an active connection to the Connect Service, which in turn requires an active subscription.

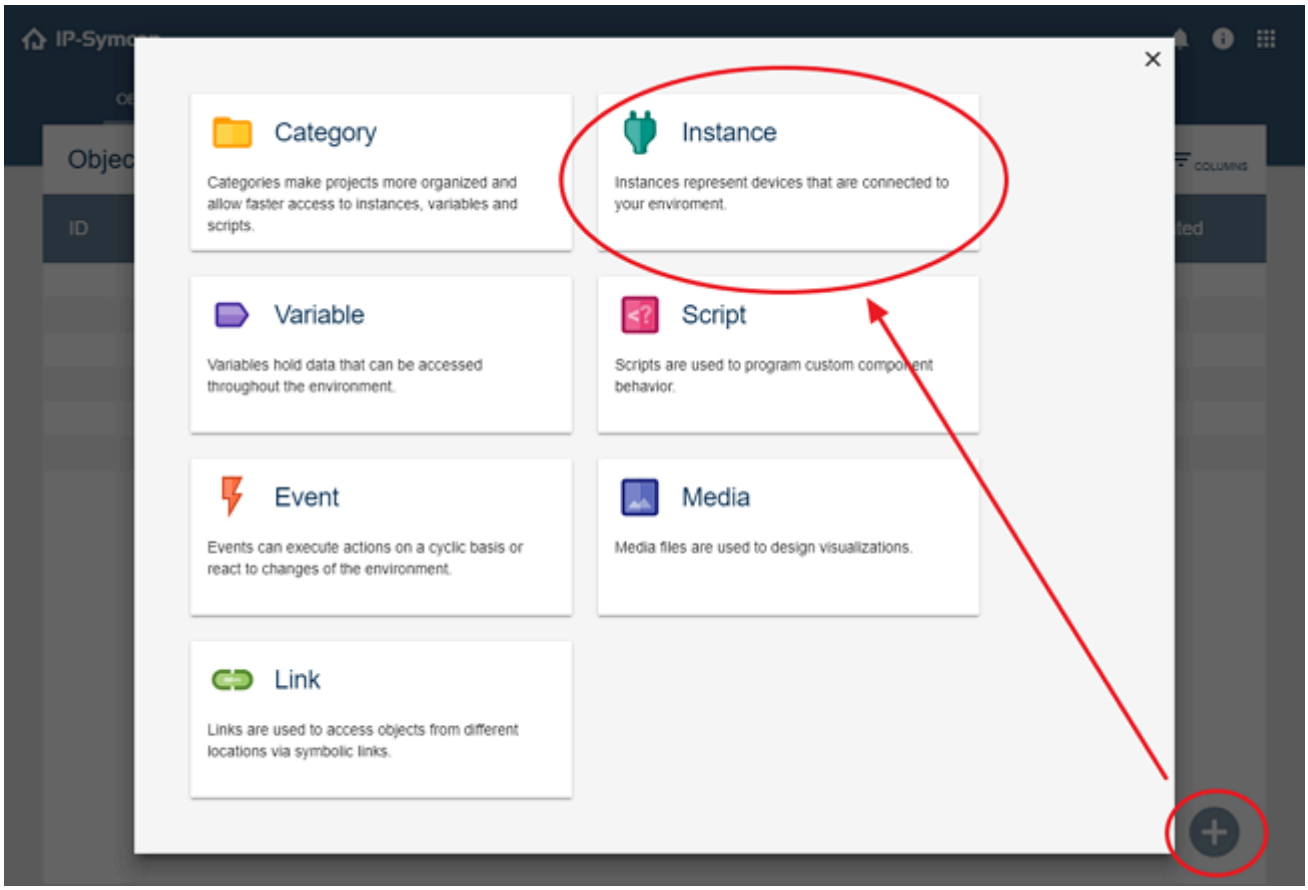
CREATE AMAZON ALEXA INSTANCE

Using the Amazon Alexa module requires its installation via the Module Store. First, the Module Store needs to be opened. It can be found in the upper right area. The Amazon Alexa module can be found by entering "Alexa" into the search field. When opening the found module, the installation of the module can be started in the next dialog by clicking "Install".

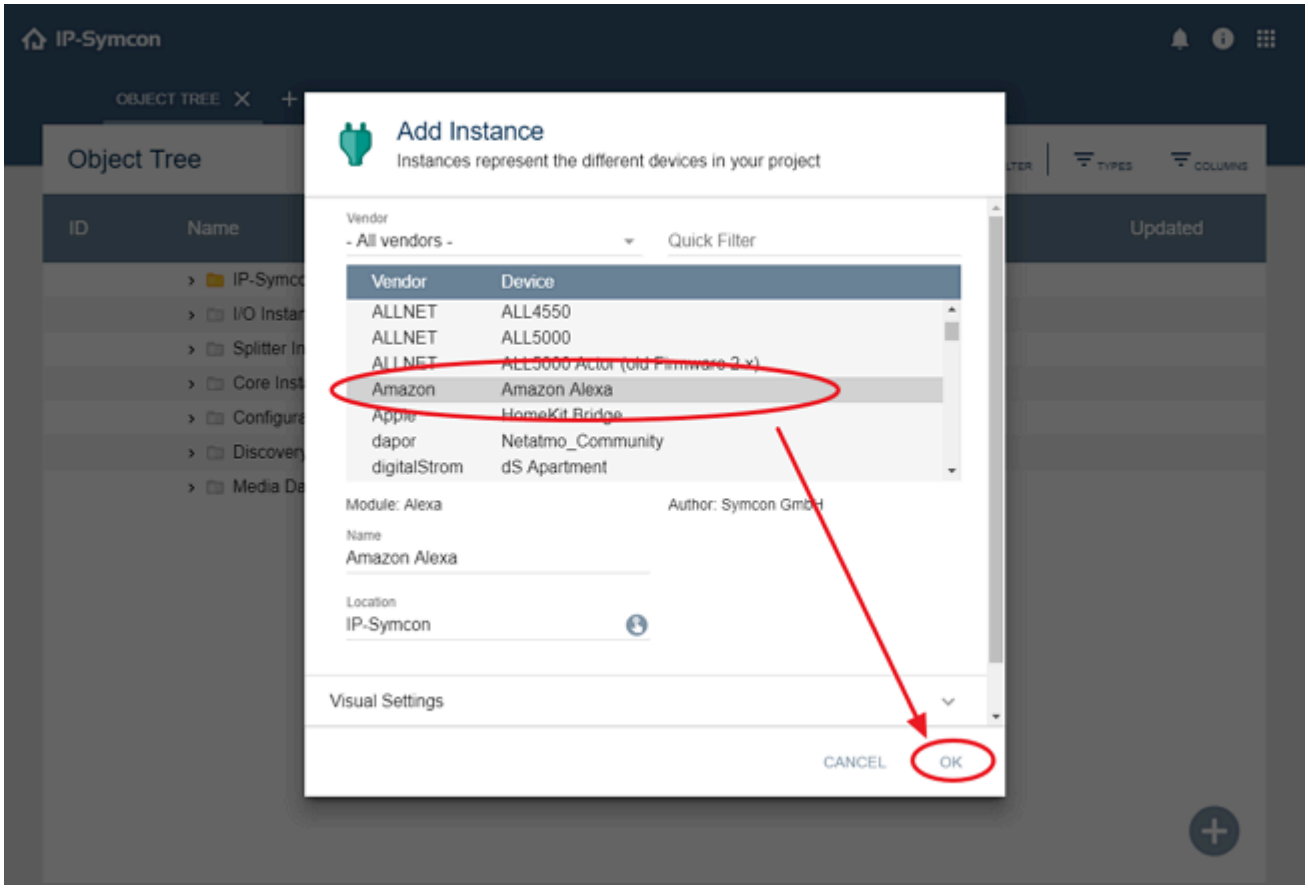

IP-Symcon
Module Store > Search
Search
Alexa
Amazon Alexa
The Amazon Alexa module enables the control of devices via Amazon Alexa. This includes the voice control via Echo Dots.



Afterwards, an add dialog for creating an Amazon Alexa instance opens.
If the instance should be created manually, an Instance of the Amazon Alexa module needs to be created. This is done by opening the Object Tree. Here, the add button "+" at the bottom right needs to be clicked and Instance selected.



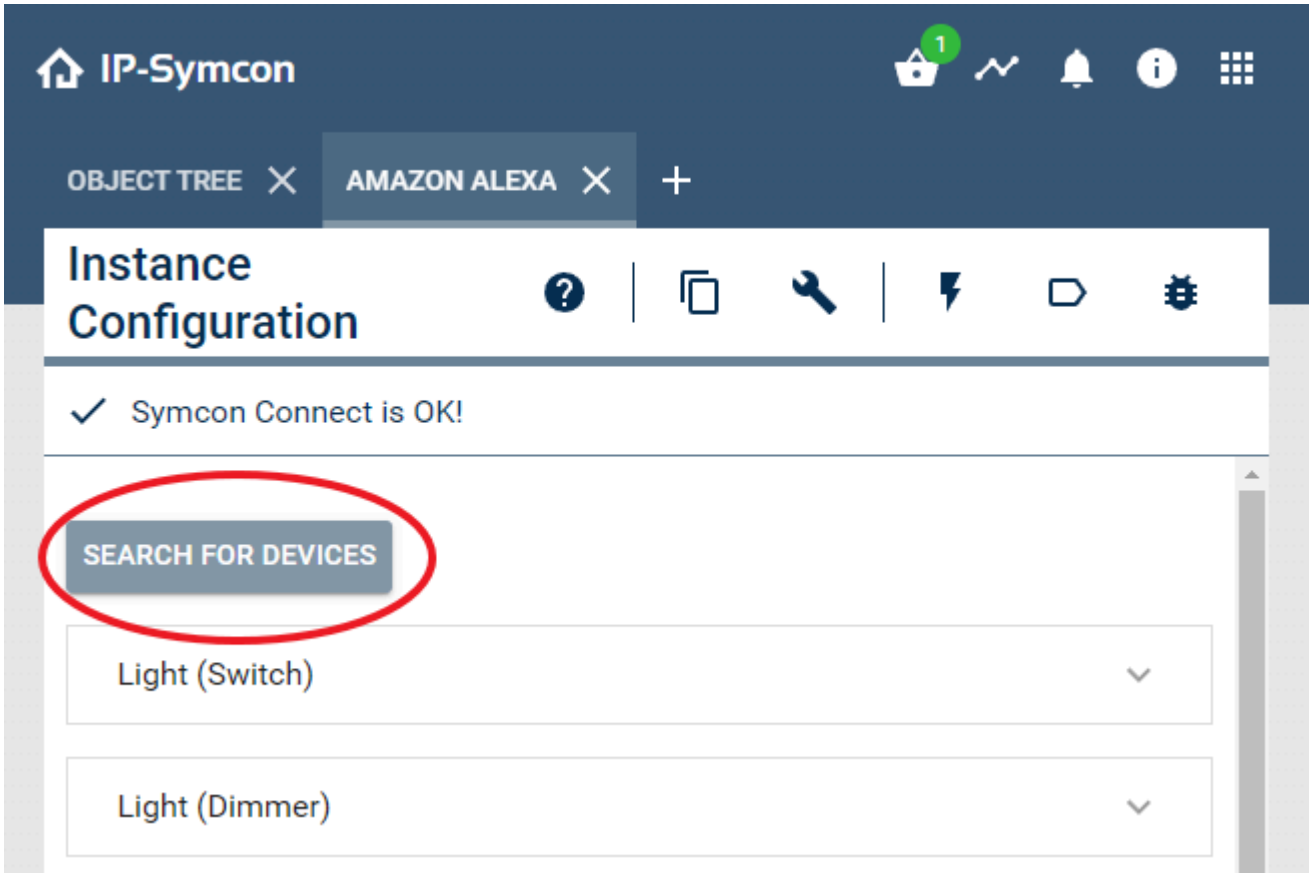
In the module list, select the device "Amazon Alexa" from the vendor "Amazon". The location should not be changed. The name can be chosen at will. Finally, confirm with "OK".



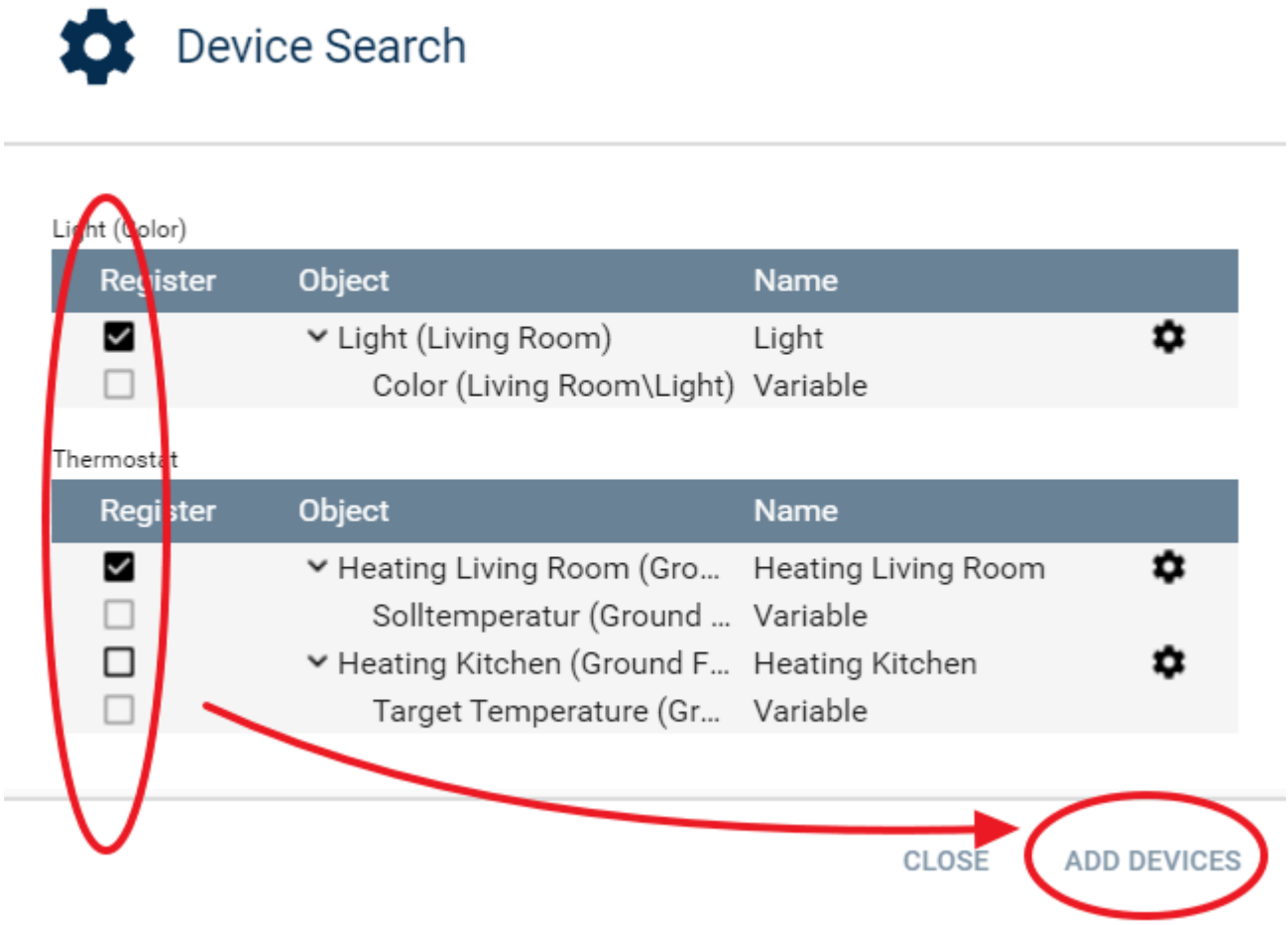
After the creation of the Amazon Alexa instance, it is opened automatically and can be configured.

CONFIGURE AMAZON ALEXA INSTANCE

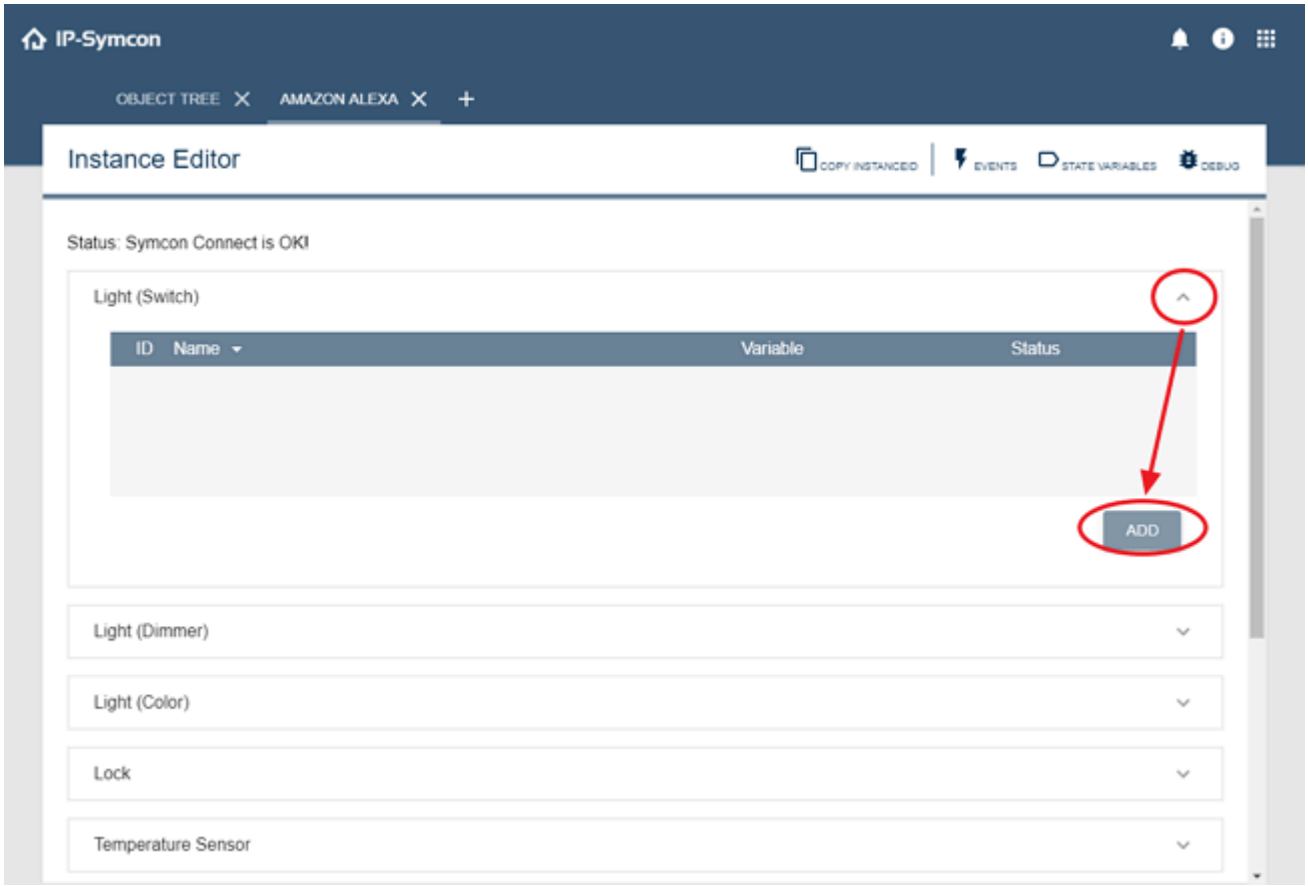
The module offers multiple device types that can be used. Especially the initial setup can be sped up significantly by using the Device Search. Later on, additional devices can easily be added manually.
The Device Search is started by clicking the button "Search for Devices" at the top of the instance configuration.



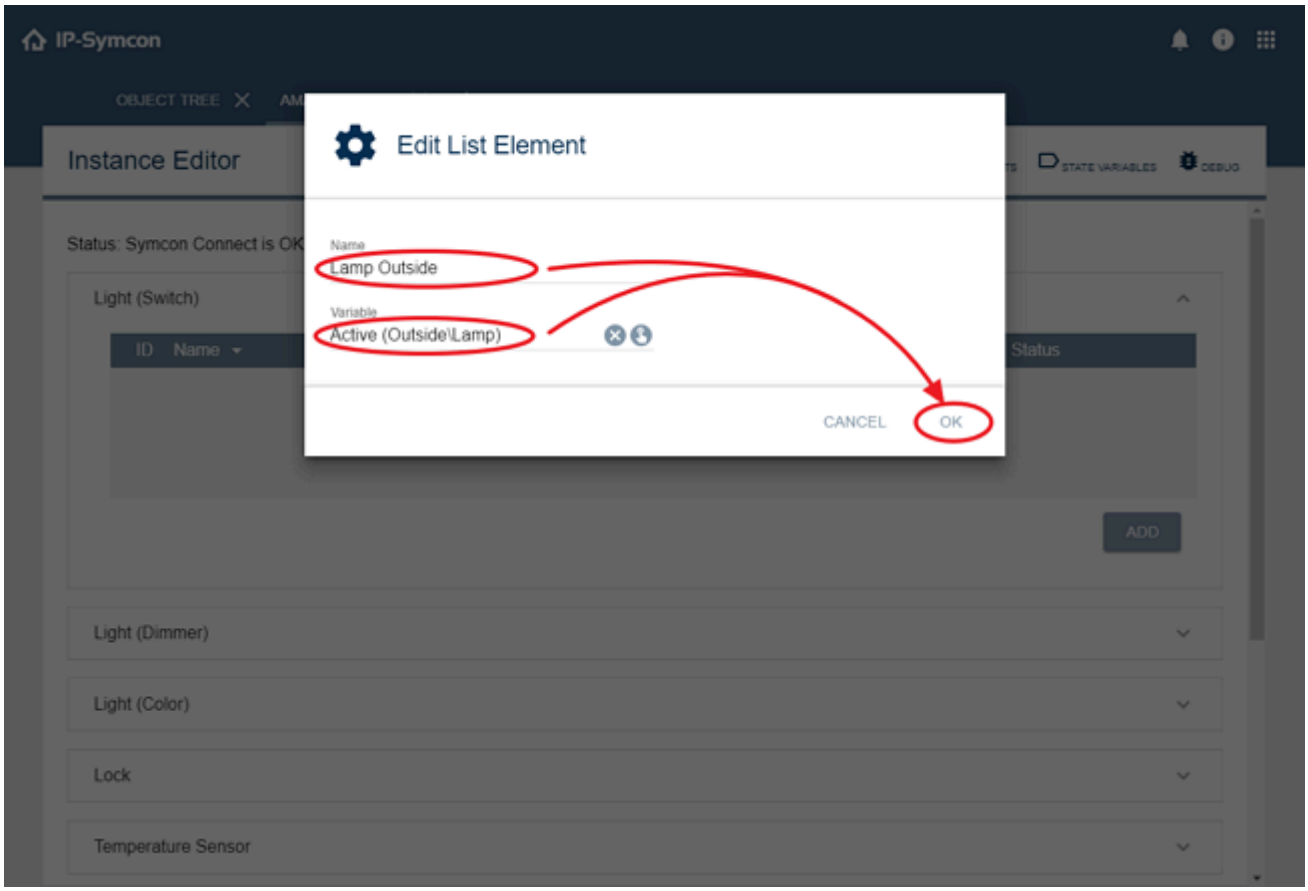
Then, a dialog is opened. After a loading time, found devices that are not yet implemented into Alexa are shown. These instances are seperated by device type. Below each instance, the status variables that are used upon implementation are shown. If an instance should be implemented, the check box at the beginning of the instances row must be checked. By default, the name of the instance is used. If it should be updated, the "Edit" button at the right of the list entry can be clicked. After all desired instances are selected and eventually their names updated, the selection is confirmed by clicking "Add Devices". Then, the selected devices are added to the respective lists in the instance configuration.



To prepare a device manually, expand its panel and create a new entry in its list by clicking "Add" below the list.



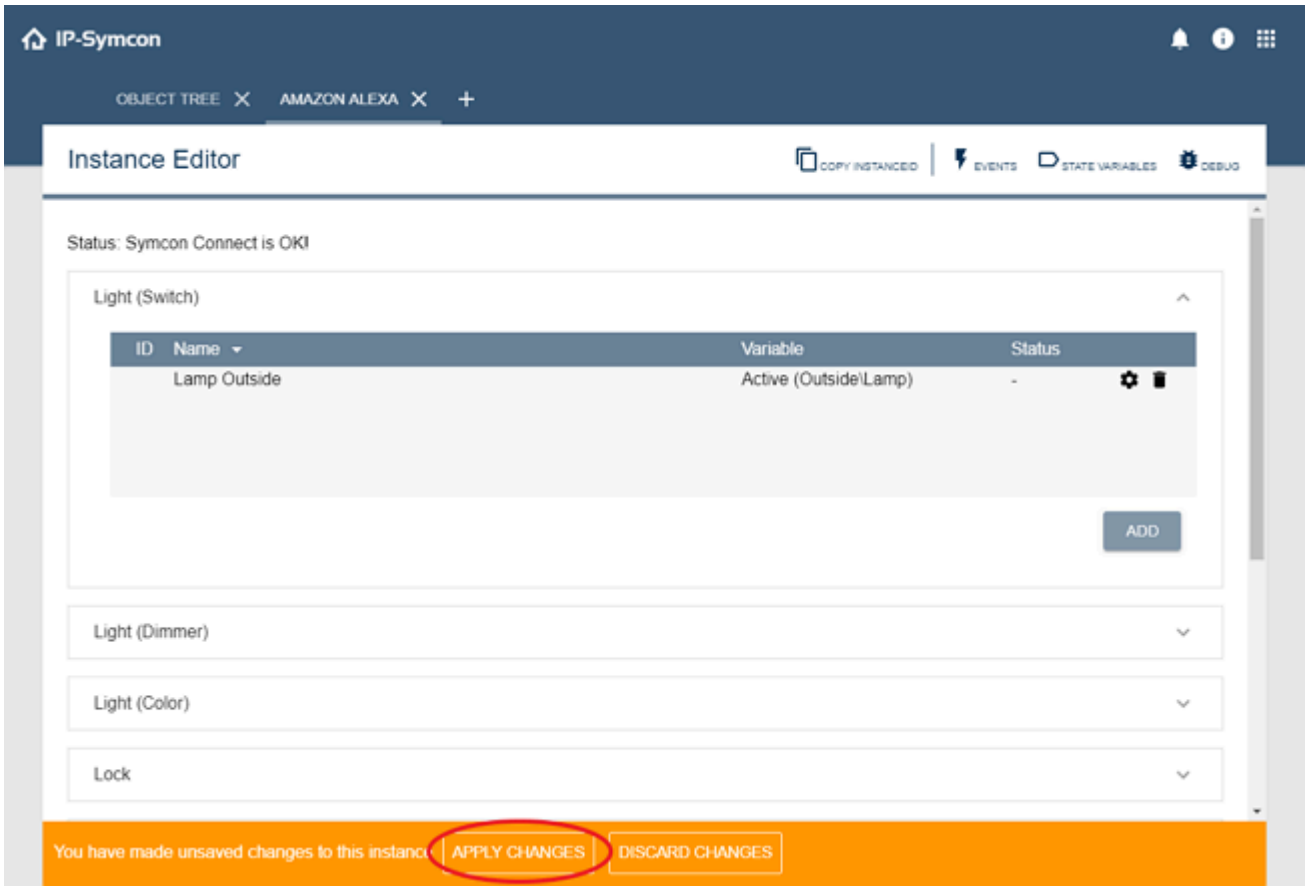
Each entry requires a name, under which the device will be known to Alexa. Additional parameters depend on the device type and can be found in the corresponding site of the documentation. After setting all parameters, the device is confirmed with "OK".



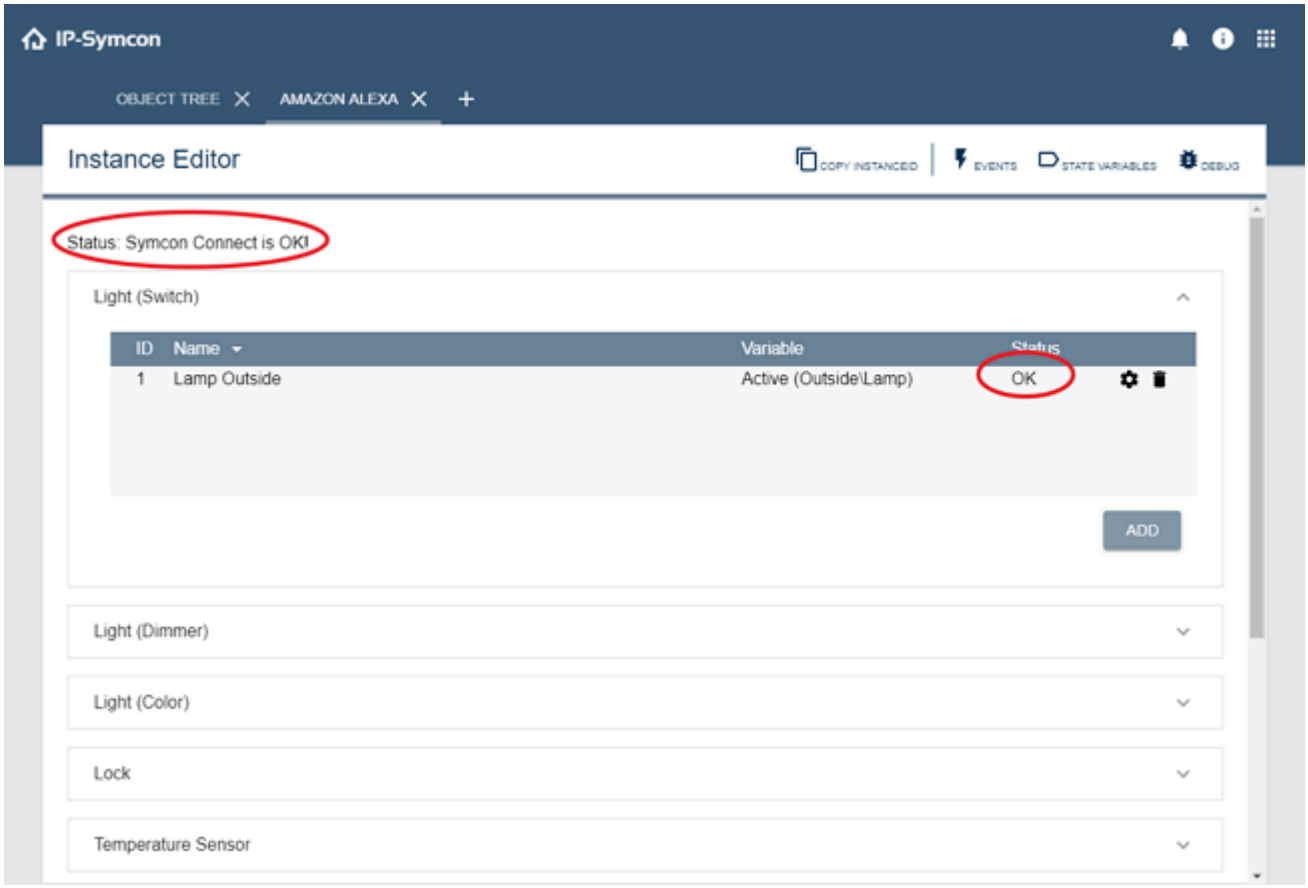
POSSIBLE DEVICE TYPES

Light (Switch)
Light (Dimmer)
Light (Color)
Light (Expert)
Lock
Temperature Sensor
Thermostat
Speaker
Speaker (Muteable)
Television
Mediaplayer
Shutters
Generic Switch
Generic Slider
Scenes
Scenes (deactivatable)

After all devices are configured, confirm the settings by clicking "Apply Changes".

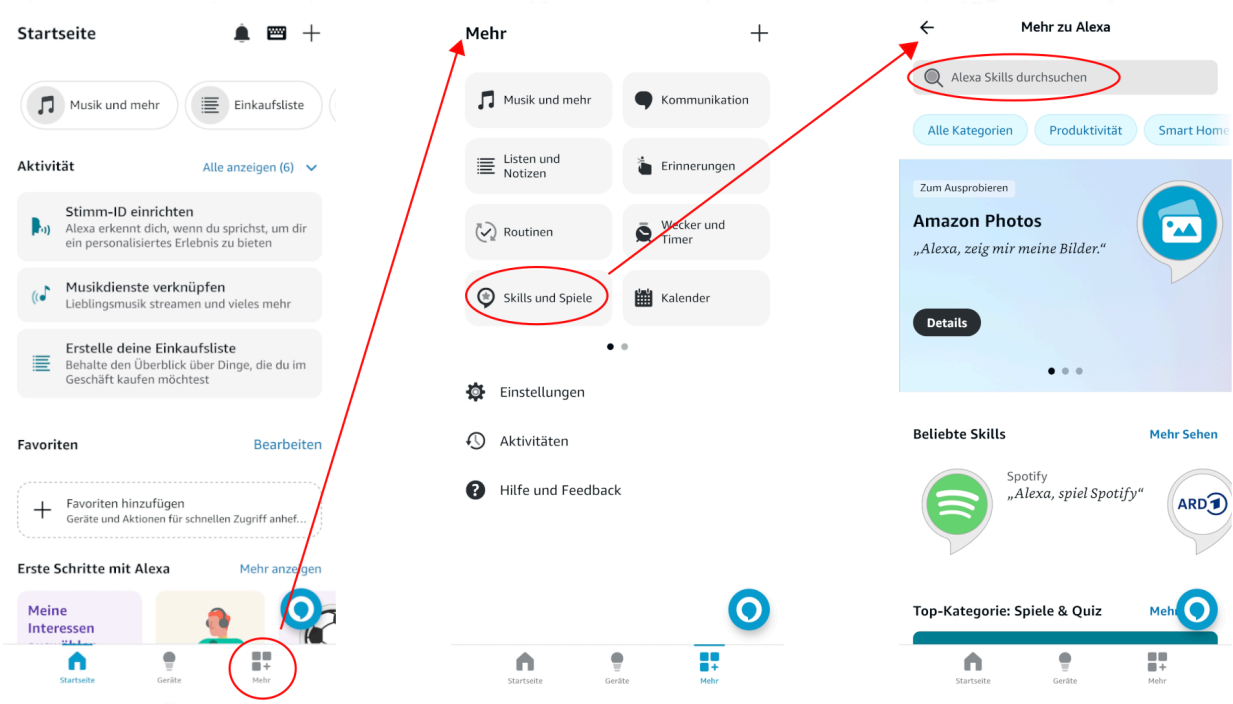


After applying, it is recommended to check that the column "Status" contains "OK" for all devices. If this is not the case, the column contains an error message that contains further information about the complication.
It should also be checked that the label at the top of the instance states "Status: Symcon Connect is OK". Otherwise, the Connect Service needs to be activated.

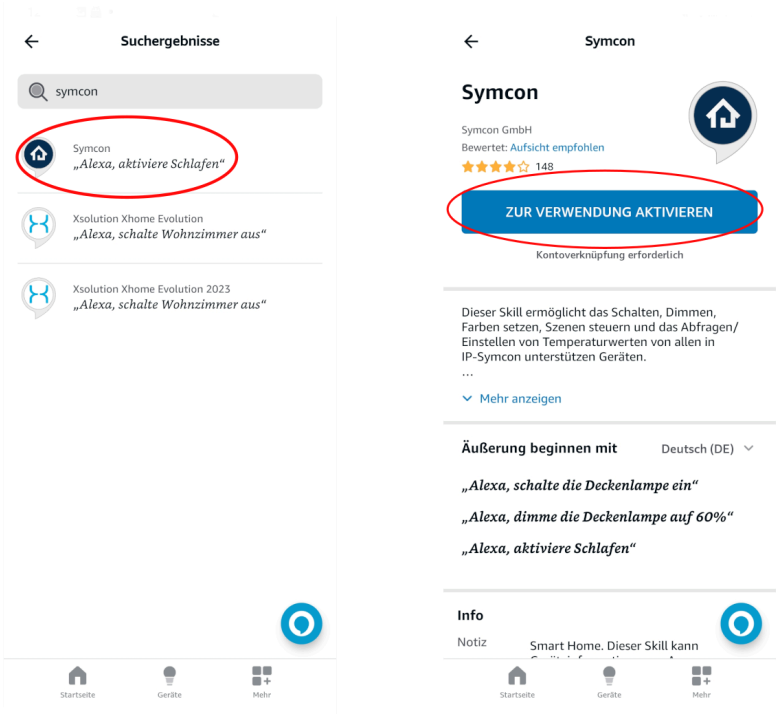


CONNECT IP-SYMCON WITH AMAZON ALEXA

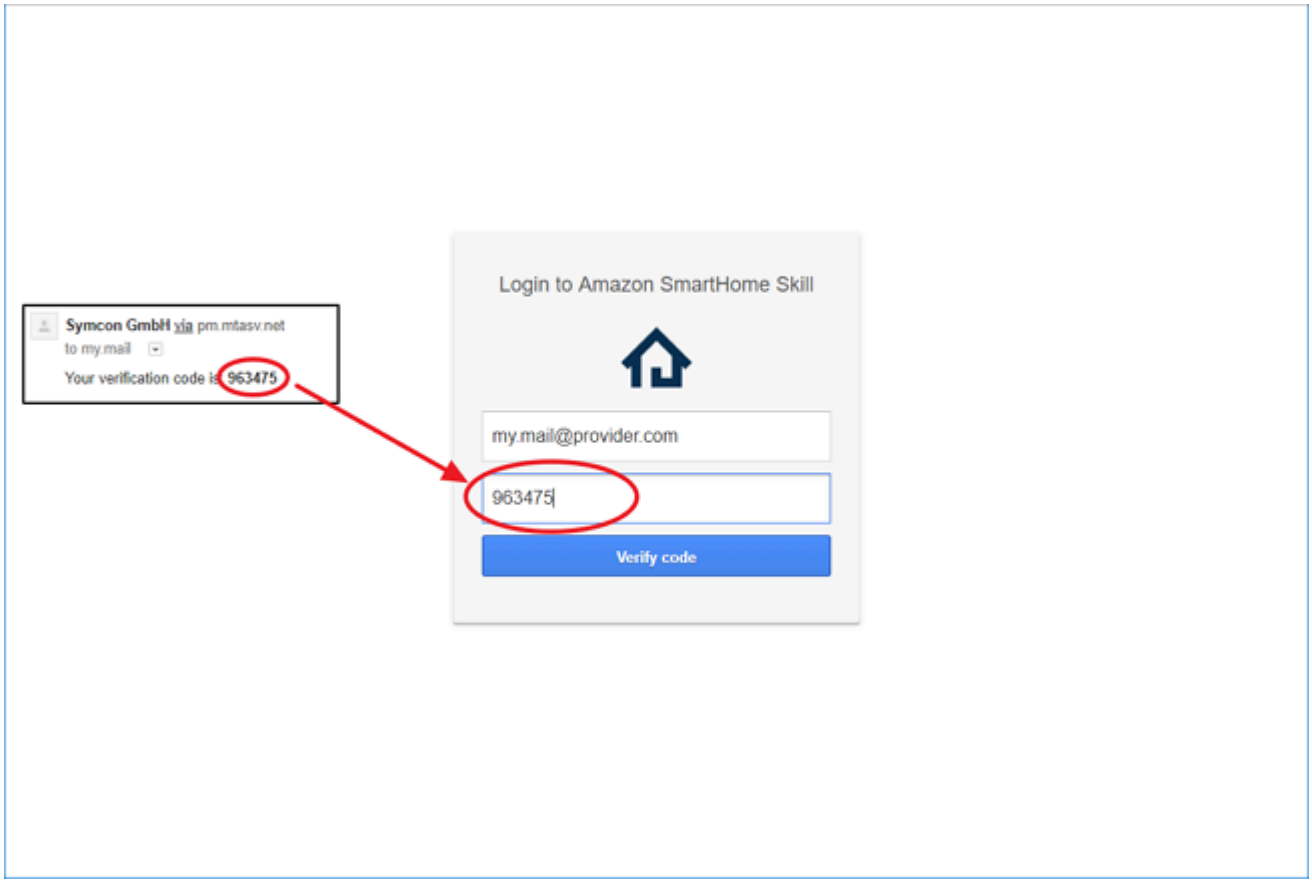
Finally, IP-Symcon must be linked to Amazon Alexa so that the Amazon Alexa instance can receive requests from Amazon Alexa. To do this, the Symcon skill for Amazon Alexa must first be installed.
The installation is done via the Amazon Alexa app.
From the start page of the app, go to "More" and click on "Skills and games".



Search for the skill "Symcon", click on the first entry "Symcon" and press the "Activate for use" button.



The browser should then open. In the field, enter the e-mail address to which the IP Symcon license used is registered and confirm with "Send verification code". An e-mail with a code will be sent to the specified e-mail address. Enter the code in this e-mail in the pop-up and confirm with "Verify code".



A message "Symcon has been successfully connected" should appear. Finally, the devices set up by Amazon Alexa must be searched for. This can be done, for example, by saying "Alexa, search for devices".

Expert Options

Require: Symcon >= 5.0



The Expert Options usually do not need to be adjusted. Handle these options carefully to ensure that the Amazon Alexa module works as intended.

OPTIONS

NAME	DESCRIPTION
Emulate Status	If Emulate Status is active, the module does not wait for the update of switched variables. Instead, it returns the new state based on the triggered value.
Show Expert Devices	If Show Expert Devices is active, expert devices are displayed. Expert devices are devices that are more complex in the configuration and require a deeper understanding of IP-Symcon.

Television

Require: Symcon >= 5.3



Television is an expert device and is only shown if the corresponding Expert Option is set.

DESCRIPTION

For devices of the type Television, diverse parameters be set. The device can be switched on and off, change the channel, set the volume, supports muting and can change the input. When a television supports only some of these properties, only the corresponding variables need to be defined.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Switch Variable	A triggerable variable of type Boolean, that switches the device on or off (optional)
Channel Variable	A triggerable variable of type Integer with a profile that has associations and step size 0. The associations need to be numbered consecutively and start with 0 or 1 (optional)
Volume Variable	A triggerable variable of type Integer or Float that controls the volume (optional)
Mute Variable	A triggerable variable of type Boolean, that controls the muting (optional)
Input Variable	A triggerable variable of type String, that controls the used input (optional)
Supported Inputs	The supported inputs for the Input Variable are defined in this list. If the Input Variable is set, at least one input needs to be selected

At least one of the optional variables needs to be defined



When optional variables are not set, Alexa recognizes that the device does not support the corresponding properties. If optional variables are defined later on, a new search for devices needs to be executed before the newly unlocked functions can be used.

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the Switch Variable to True or False	"Alexa, switch *<Name>* on."
Switch to channel	Switch the channel variable to the value whose association corresponds to the mentioned channel. For this process, upper and lower case and not differenciated and spaces are ignored	"Alexa, change the channel to PBS on *<Name>*."
Switch to next channel	Increase the Channel Variable by 1. If the variable is currently at the last channel, it is set to the first channel instead	"Alexa, next channel on *<Name>*."
Set volume	Switch the Volume Variable to the given value	"Alexa, switch the volume of *<Name>* to 40."
Mute or unmute	Switch the Mute Variable to True or False	"Alexa, mute *<Name>*."
Change input	Switch the Input Variable to the name of the input	"Alexa, change the input to DVD on *<Name>*"

DEVICE SEARCH

This device type is not supported by the device search.

Generic Switch

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Generic Switch can be switched On or Off. Within Amazon Alexa, the device is interpreted as a common switch and can be used for different devices.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Switch Variable	A triggerable variable of type Boolean

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the variable to True or False	"Alexa, switch *<Name>* on."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~Switch

Generic Slider

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Generic Slider can be set to are percentual value. Within Amazon Alexa, the device is interpreted as a common switch and can be used for different devices.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Slider Variable	A triggerable variable of type Integer or Float

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Switch the variable towards 100% or 0%	"Alexa, switch *<Name>* on."
Set Value	Switch the variable towards the given value	"Alexa, switch *<Name>* to 40%."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Slider Variable	~Intensity.100, ~Intensity.255, ~Intensity.1

Speaker

Require: Symcon >= 5.0

DESCRIPTION

For devices of the type Speaker, the volume can be set.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Volume Variable	A triggerable variable of type Integer or Float that controls the volume

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Set volume	Switch the variable to the given value	"Alexa, switch the volume of *<Name>* to 40."

DEVICE SEARCH

This device type is not supported by the device search.

Speaker (Muteable)

Require: Symcon >= 5.3

DESCRIPTION

For devices of the type Speaker, the volume can be set. In addition, the device can be muted and unmuted without loosing the configured volume.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Volume Variable	A triggerable variable of type Integer or Float that controls the volume (optional)
Mute Variable	A triggerable variable of type Boolean, that controls the muting (optional)

Either the Volume Variable or the Mute Variable needs to be defined



When optional variables are not set, Alexa recognizes that the device does not support the corresponding properties. If optional variables are defined later on, a new search for devices needs to be executed before the newly unlocked functions can be used.

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Set volume	Switch the Volume Variable to the given value	"Alexa, switch the volume of *<Name>* to 40."
Mute or unmute	Switch the Mute Variable to True or False	"Alexa, mute *<Name>*."

DEVICE SEARCH

This device type is not supported by the device search.

Light (Dimmer)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Dimmer) describe lamps that can be dimmed.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Brightness Variable	A triggerable variable of type Integer or Float that dims the light

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the variable to 100% or 0%	"Alexa, switch *<Name>* on."
Dim	Trigger the variable to the given value	"Alexa, dim *<Name>* to 40%."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Brightness Variable	~Intensity.100, ~Intensity.255, ~Intensity.1

Light (Expert)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Expert) describe lamps whose properties within multiple variables can be switched independently.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Switch Variable	A triggerable variable of type Boolean, that switches the light on or off
Brightness Variable	A triggerable variable of type Integer or Float that dims the light (optional)
Color Variable	A triggerable Variable of type Integer with the profile ~HexColor, that triggers the color of the light (optional)
Color Temperature Variable	A triggerable variable of type Integer or Float that contains the color temperature in Kelvin (optional)



If optional variables are not set, Alexa recognizes that the lamp does not have the corresponding properties. If optional variables are set later on, a search for devices is required to discover the changed properties.

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the Switch Variable to True or False	"Alexa, switch *<Name>* on."
Dim	Trigger the Brightness Variable to the given value	"Alexa, dim *<Name>* to 40%."
Switch Color	Trigger the Color Variable to the given color	"Alexa, switch *<Name>* to red."
Switch Color Temparature	Trigger the Color Temperature Variable to the corresponding value	"Alexa, switch *<Name>* to daylight."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search. To be detected as Expert light, at least the Switch Variable and another variable need to be detected.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~Switch
Brightness Variable	~Intensity.100, ~Intensity.255, ~Intensity.1
Color Variable	~HexColor
Color Temperature Variable	~TWColor

Light (Color)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Color) describe lamps, that can be switched to any color.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Color Variable	A triggerable Variable of type Integer with the profile ~HexColor, that triggers the color of the light

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the variable to white or black	"Alexa, switch *<Name>* on."
Dim	Trigger the brightness of the current color to the given value	"Alexa, dim *<Name>* to 40%."
Switch Color	Trigger the variable to the given color	"Alexa, switch *<Name>* to red."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Color Variable	~HexColor

Light (Switch)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Switch) describe lamps that can be switched on or off.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Switch Variable	A triggerable variable of type Boolean, that switches the light on or off

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Triggers the variable to True or False	"Alexa, switch *<Name>* on."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~Switch

Mediaplayer

Require: Symcon >= 5.4

DESCRIPTION

For devices of the type Mediaplayer, diverse parameters can be set. The device can be switched on and off, set the volume, supports muting and can control the playback. When a mediaplayer supports only some of these properties, only the corresponding variables need to be defined.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Switch Variable	A triggerable variable of type Boolean, that switches the device on or off (optional)
Volume Variable	A triggerable variable of type Integer or Float that controls the volume (optional)
Mute Variable	A triggerable variable of type Boolean, that controls the muting (optional)
Playback Variable	A triggerable variable of type Integer with the profile ~Playback or ~PlaybackPreviousNext that defines the state of the playback; Depending on the profile, Alexa only provides the corresponding actions, i.e., Stop, Play and Pause for ~Playback and additionally Previous and Next for ~PlaybackPreviousNext (optional)

At least one of the optional variables needs to be defined



When optional variables are not set, Alexa recognizes that the device does not support the corresponding properties. If optional variables are defined later on, a new search for devices needs to be executed before the newly unlocked functions can be used.

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the Switch Variable to True or False	"Alexa, switch *<Name>* on."
Set volume	Switch the Volume Variable to the given value	"Alexa, switch the volume of *<Name>* to 40."
Mute or unmute	Switch the Mute Variable to True or False	"Alexa, mute *<Name>*."

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Set playback	Switch the Playback Variable to the provided value	"Alexa, play *<Name>*"

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search. To be detected as Mediaplayer, at least the Playback Variable needs to be detected.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~Switch
Volume Variable	This variable is not detected via device search
Mute Variable	This variable is not detected via device search
Playback Variable	~Playback, ~PlaybackPreviousNext

Shutters

Require: Symcon >= 5.3

DESCRIPTION

Devices of the type Shutters can be opened and closed. If the device supports percentual opening, it can be controlled as well.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Shutter Variable	A triggerable variable of the type Integer or Float with the profile ~ShutterMoveStop or ~ShutterMoveStep. Alternatively, a variable that has a defined minimum and maximum value via profile can be used

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Open	When a ~ShutterMove profile is used, trigger the variable to "Open". Otherwise, the variable is triggered to the minimal value of the profile.	"Alexa, open *<Name>*"
Close	When a ~ShutterMove profile is used, trigger the variable to "Close". Otherwise, the variable is triggered to the maximum value of the profile.	"Alexa, close *<Name>*"
Open percentually	When a ~ShutterMove profile is used, trigger the variable to "Open" if the percentual value is bigger than 50% and to "Close" otherwise. Otherwise, the variable is triggered according to the percentage value. It should be mentioned, that the variable is triggered to 0% on a complete opening and similar computations are applied for other values as well. For example, opening by 30% will trigger the variable to 70%.	"Alexa, open *<Name>* by 30%"

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~ShutterMoveStop, ~ShutterMoveStep, ~Intensity.100, ~Intensity.255, ~Intensity.1

Lock

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Lock can be locked.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Lock Variable	A triggerable variable of the type Boolean that triggers the lock

POSSIBLE ACTIONS



Before a lock can be unlocked, unlocking needs to be enabled via the Alexa App.

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Lock	Trigger the variable to True	"Alexa, lock *<Name>*."
Unlock	Trigger the variable to False	"Alexa, unlock *<Name>*."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Lock Variable	~Lock, ~Lock.Reversed, ~Door, ~Door.Reversed

Scenes

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Scene can be activated and execute any kind of action.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Activate Action	An action that is executed on activation of the scene

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Activate Scene	Execute the action	"Alexa, activate *<Name>*."

DEVICE SEARCH

This device type is not supported by the device search.

Scenes (deactivatable)

Require: Symcon >= 5.0

DESCRIPTION

Devices a the type Scene (deactivatable) can be activated and deactivated and execute any kind of action in the process.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Activate Action	An action that is executed on activation of the scene
Deactivate Action	An action that is executed on deactivation of the scene

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Activate Scene	Execute the Activate Action	"Alexa, activate *<Name>*."
Deactivate Scene	Execute the Deactivate Action	"Alexa, deactivate *<Name>*."

DEVICE SEARCH

This device type is not supported by the device search.

Temperature Sensor

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Temperature Sensor measure temperatures.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Sensor Variable	A variable of type Float that contains the current temperature in °C

POSSIBLE ACTIONS



Currently, Alexa does not support direct requests to a temperature sensor. Thus, a group needs to be used to request the temperature via voice command. Once the temperature sensor is part of a group, the temperature of that group can be requested.

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Request temperature	Request the current temperature of the sensor	"Alexa, what's the temperature of *<group>*?"

DEVICE SEARCH

This device type is not supported by the device search.

Thermostat

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Thermostat can set temperatures.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Temperaturvariable	A triggerable variable of type Float that sets the temperature

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Set Temperature	Trigger the variable toward the given temperature.	"Alexa, switch *<Name>* to 35 °C."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Temperaturvariable	~Temperature, ~Temperature.Room

Google Assistant

Require: Symcon >= 5.0

DESCRIPTION

The Google Assistant module enables the control of devices via Google Assistant. This includes the voice control via Google Home.

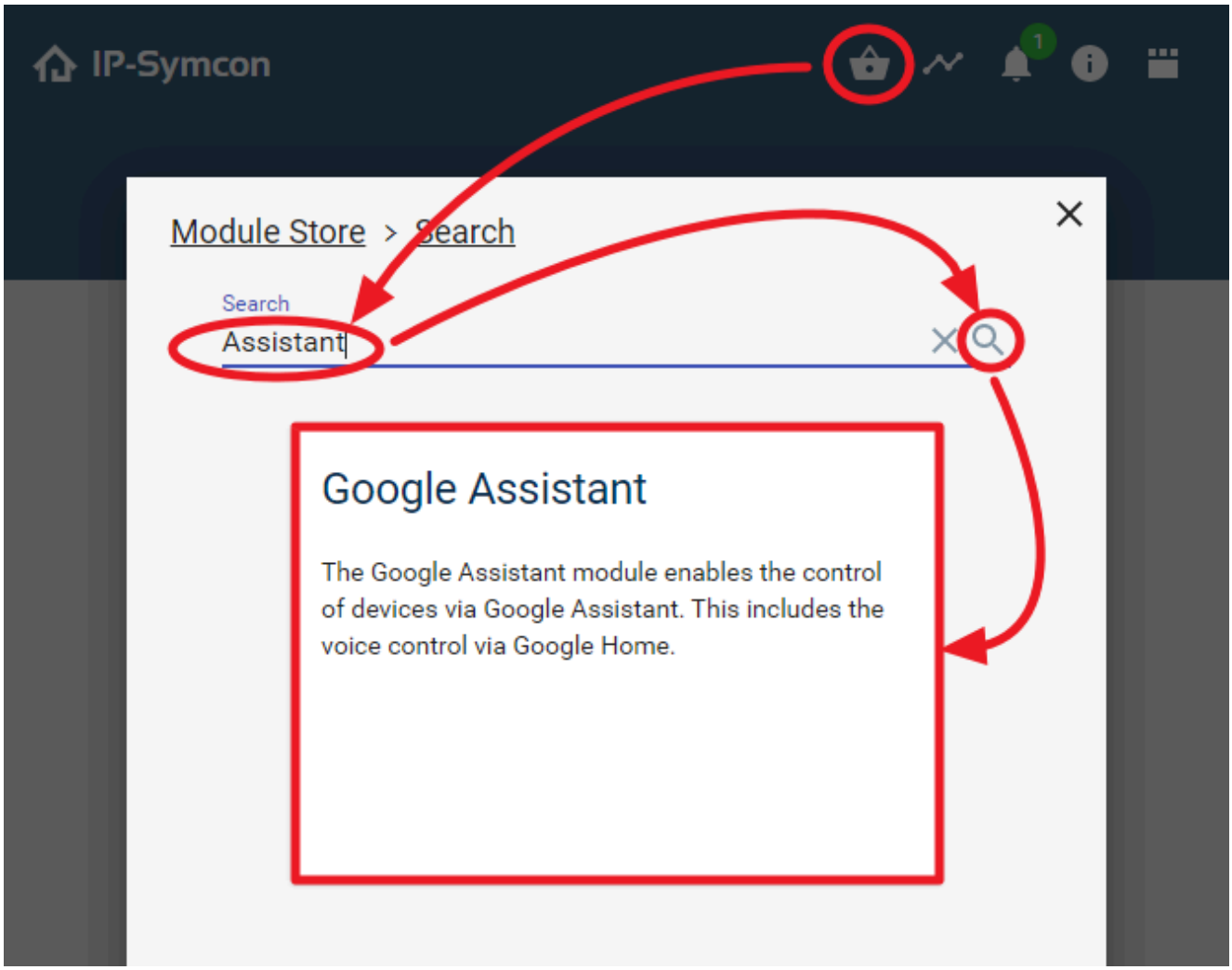
INCLUSION IN IP-SYMCON

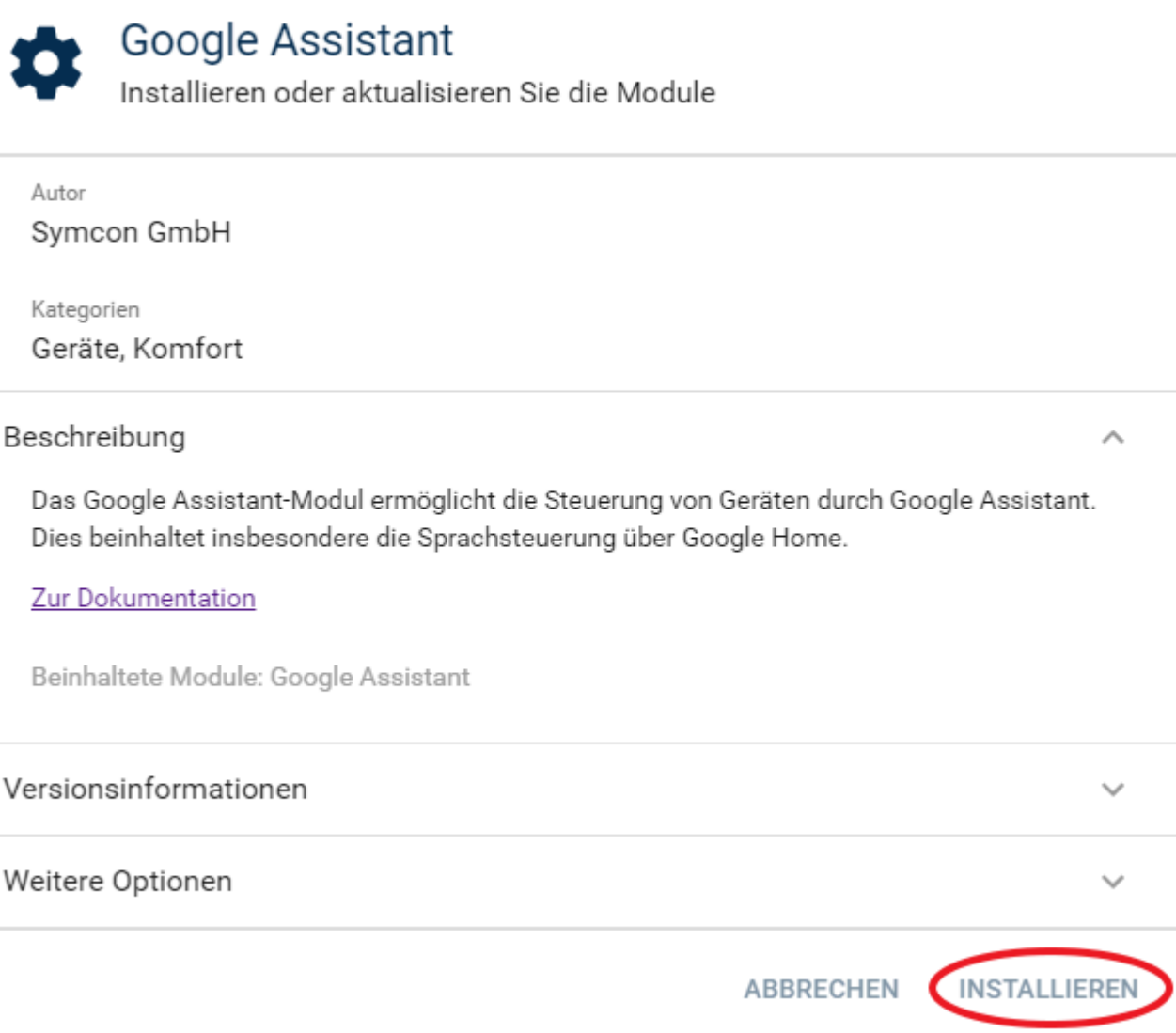


Using the Google Assistant module requires an active connection to the Connect Service, which in turn requires an active subscription.

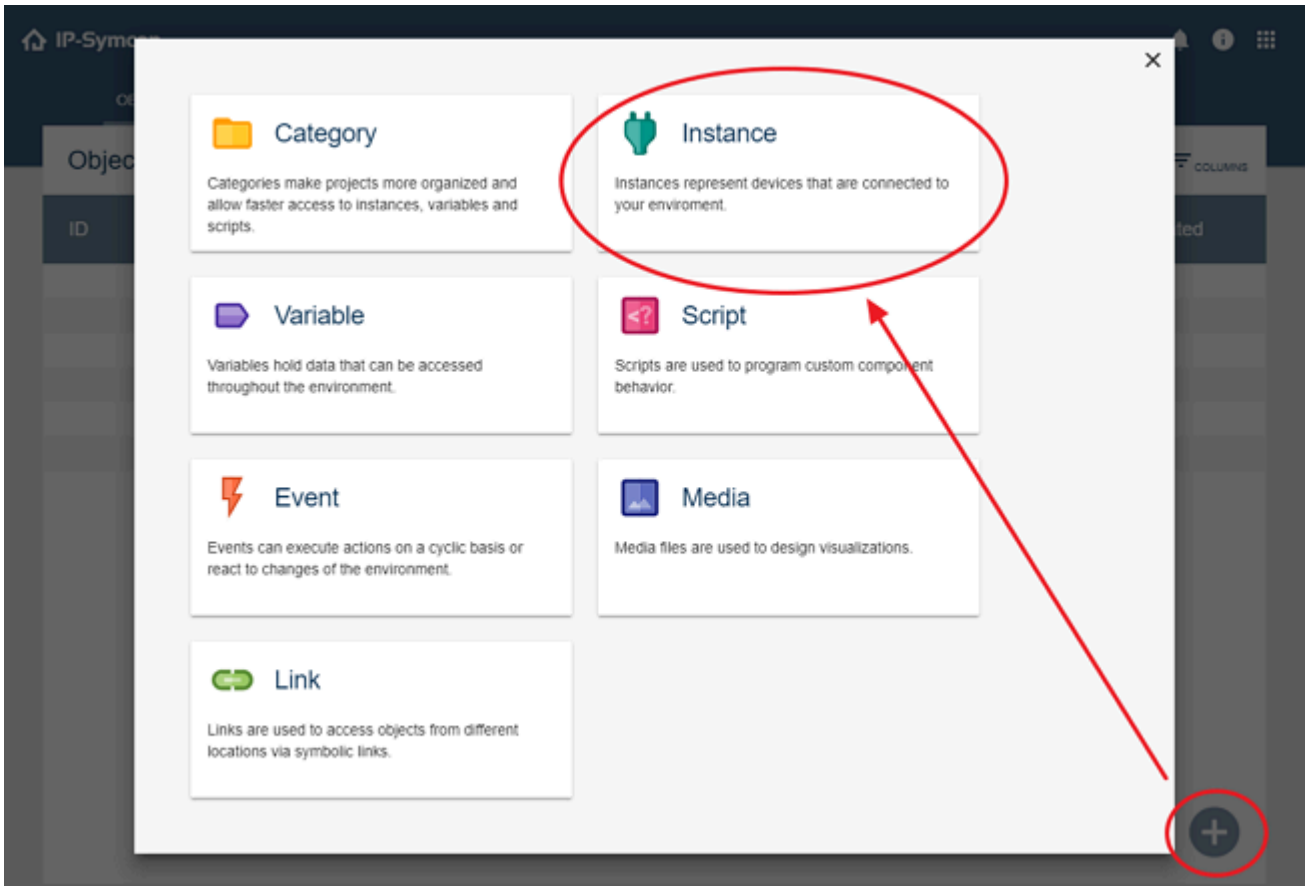
CREATE GOOGLE ASSISTANT INSTANCE

Using the Google Assistant module requires its installation via the Module Store. First, the Module Store needs to be opened. It can be found in the upper right area. The Google Assistant module can be found by entering "Assistant" into the search field. When opening the found module, the installation of the module can be started in the next dialog by clicking "Install".

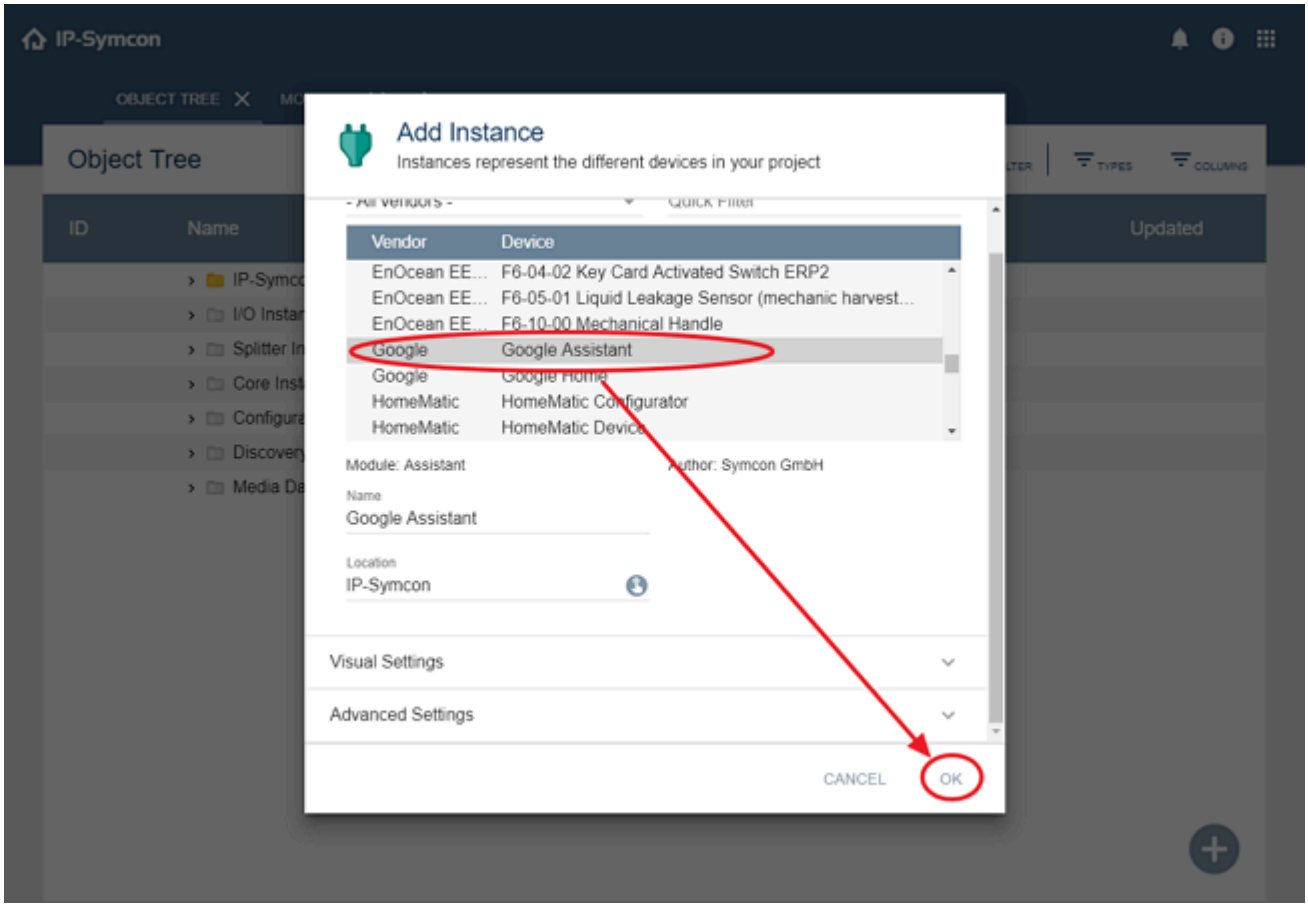

IP-Symcon
Module Store > Search
Search
Assistant
Google Assistant
The Google Assistant module enables the control of devices via Google Assistant. This includes the voice control via Google Home.



Afterwards, an add dialog for creating an Google Assistant instance opens.
If the instance should be added manually, an Instance of the Google Assistant module needs to be created. This is done by opening the Object Tree. Here, the add button "+" at the bottom right needs to be clicked and Instance selected.



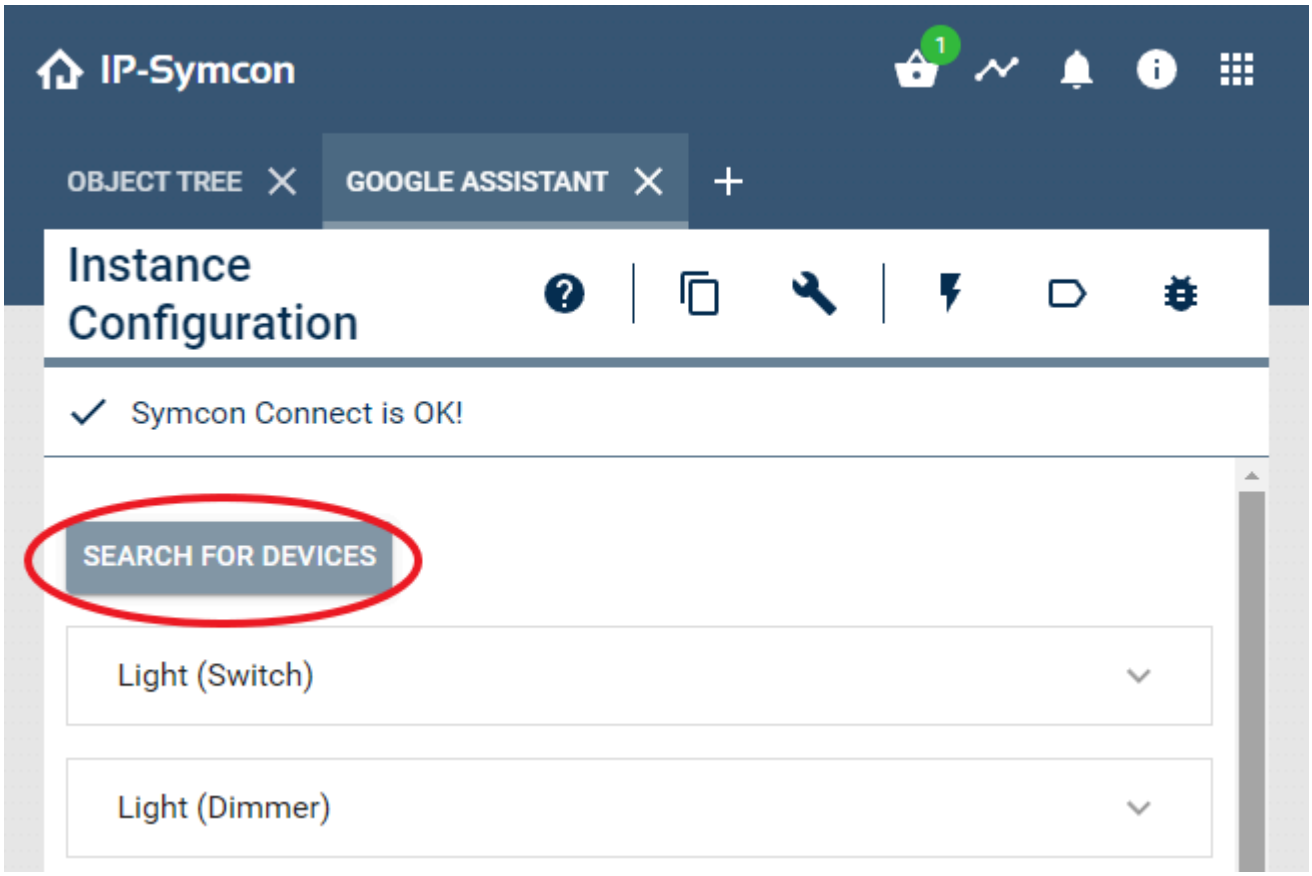
In the module list, select the device "Google Assistant" or "Google Home" from the vendor "Google". The location should not be changed. The name can be chosen at will. Finally, confirm with "OK".



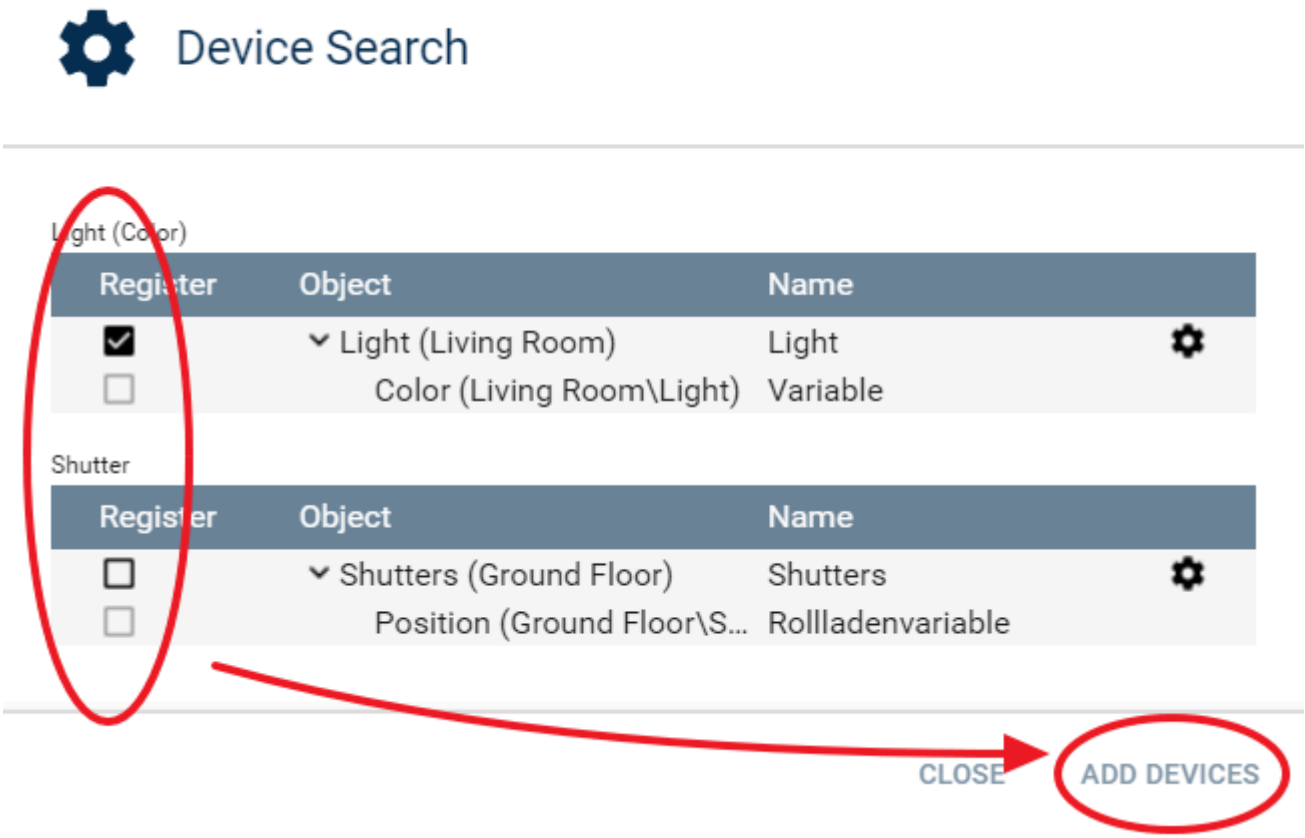
After the creation of the Google Assistant instances, it is opened automatically and can be configured.

CONFIGURE GOOGLE ASSISTANT INSTANCE

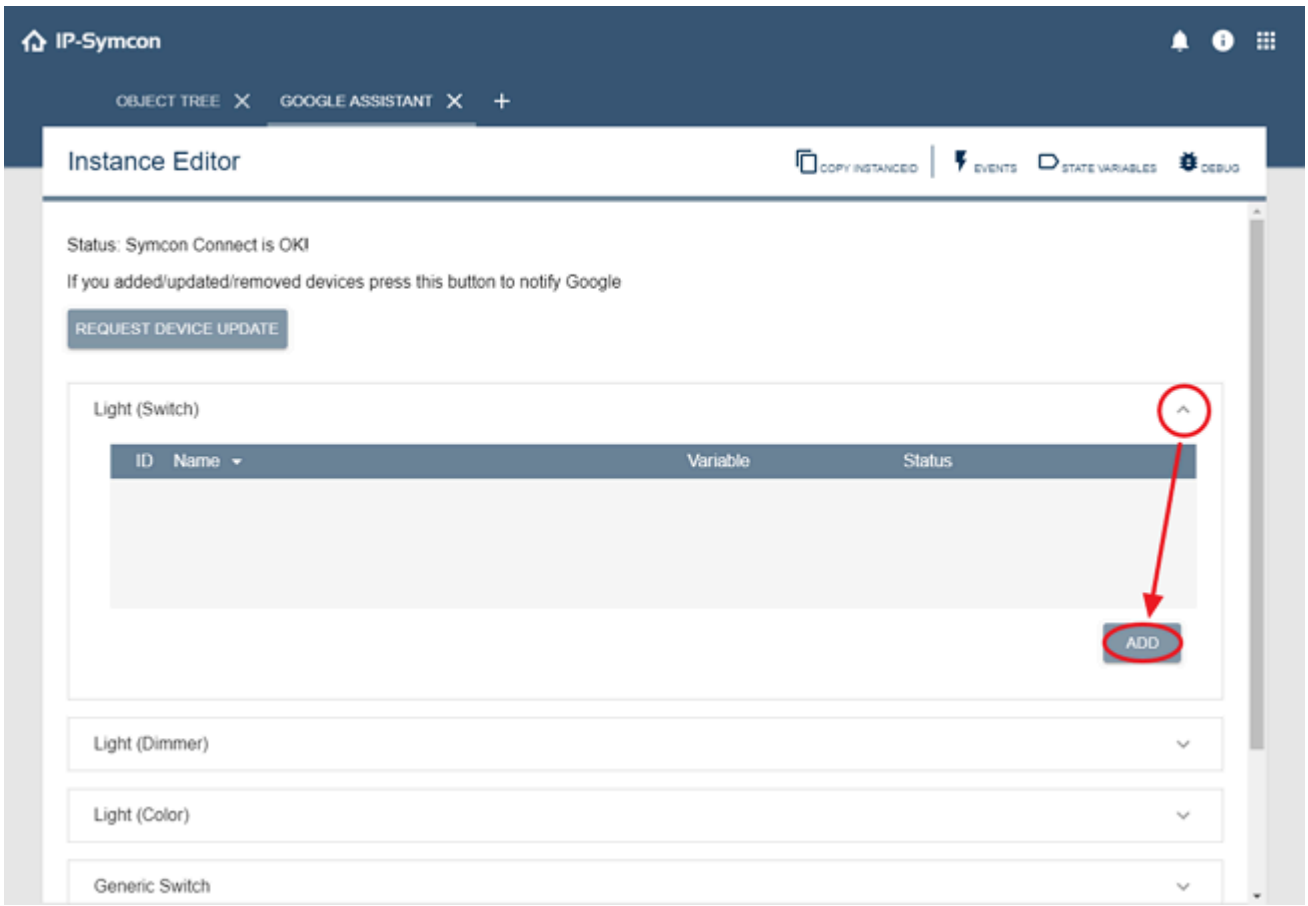
The module offers multiple device types that can be used. Especially the initial setup can be sped up significantly by using the Device Search. Later on, additional devices can easily be added manually.
The Device Search is started by clicking the button "Search for Devices" at the top of the instance configuration.



Then, a dialog is opened. After a loading time, found devices that are not yet implemented into Alexa are shown. These instances are seperated by device type. Below each instance, the status variables that are used upon implementation are shown. If an instance should be implemented, the check box at the beginning of the instances row must be checked. By default, the name of the instance is used. If it should be updated, the "Edit" button at the right of the list entry can be clicked. After all desired instances are selected and eventually their names updated, the selection is confirmed by clicking "Add Devices". Then, the selected devices are added to the respective lists in the instance configuration.

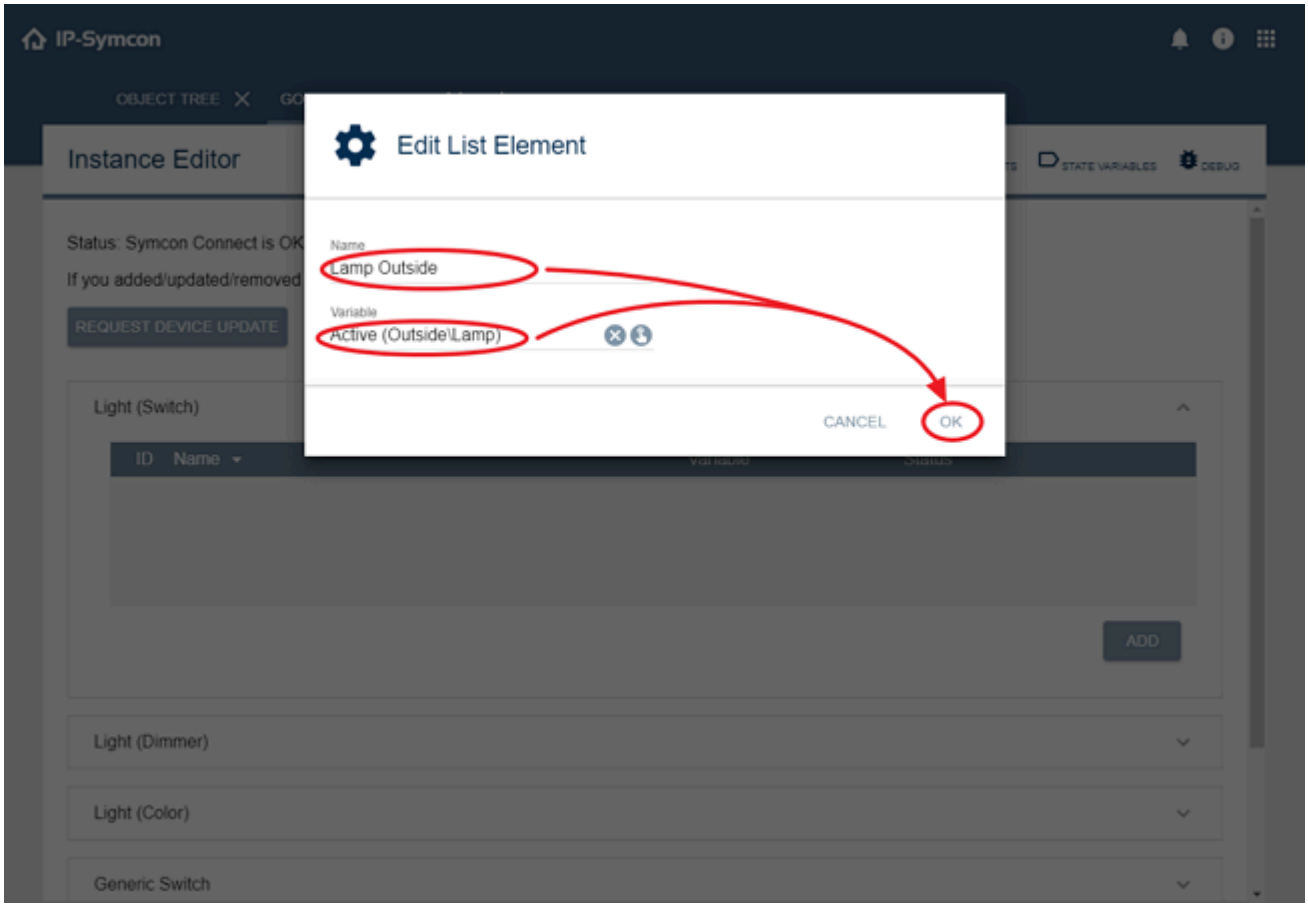


To prepare a device manually, expand its panel and create a new entry in its list by clicking "Add" below the list.



Each entry requires a name, under which the device will be known to Google Assistant. Additional parameters depend on the device type and can be found in the corresponding

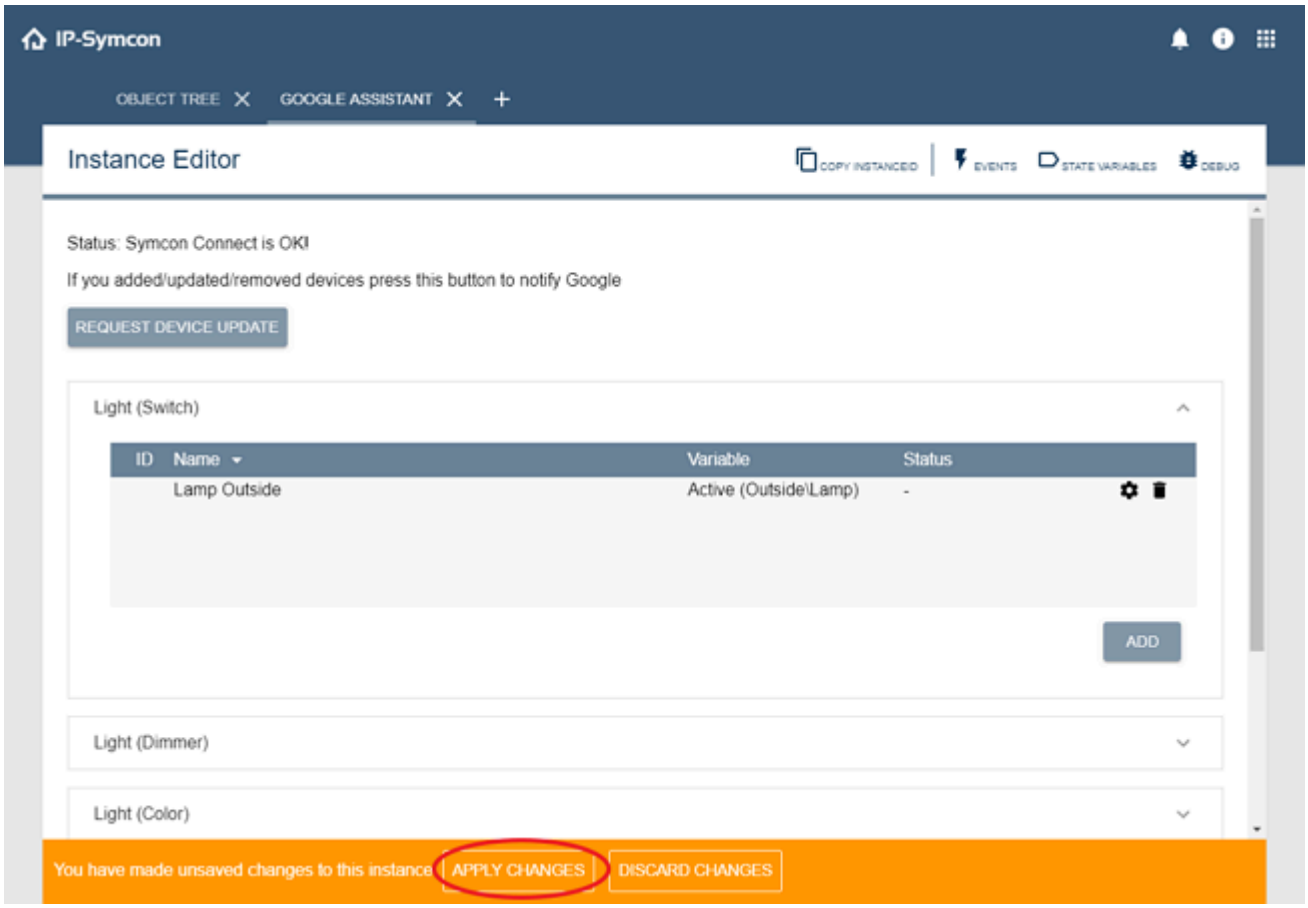
site of the documentation. After setting all parameters, confirm with "OK".



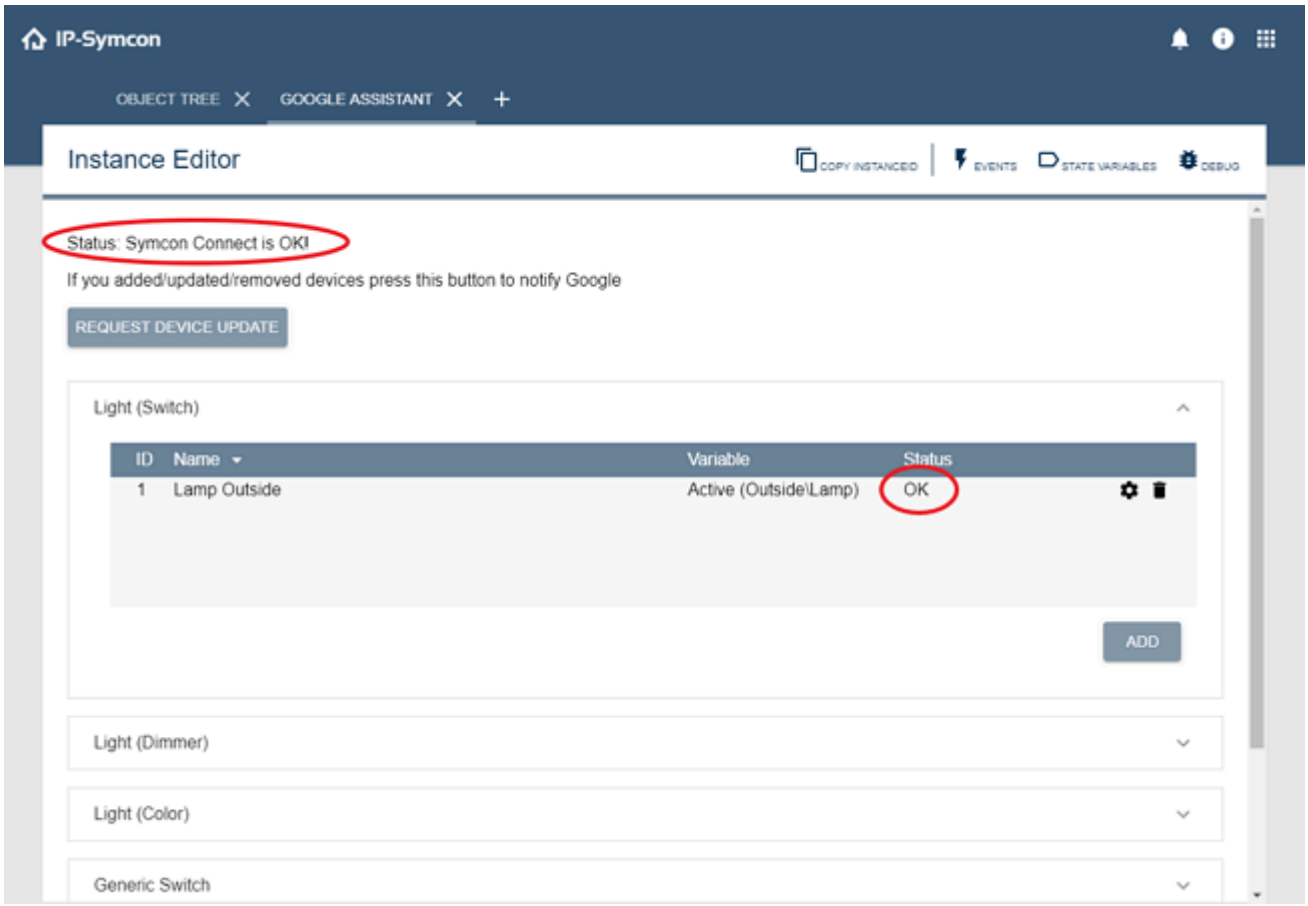
POSSIBLE DEVICE TYPES

Light (Switch)
Light (Dimmer)
Light (Color)
Light (Expert)
Thermostat
Shutter
Generic Switch
Scenes
Scenes (deactivatable)

After all devices are configured, confirm the settings by clicking "Apply Changes".

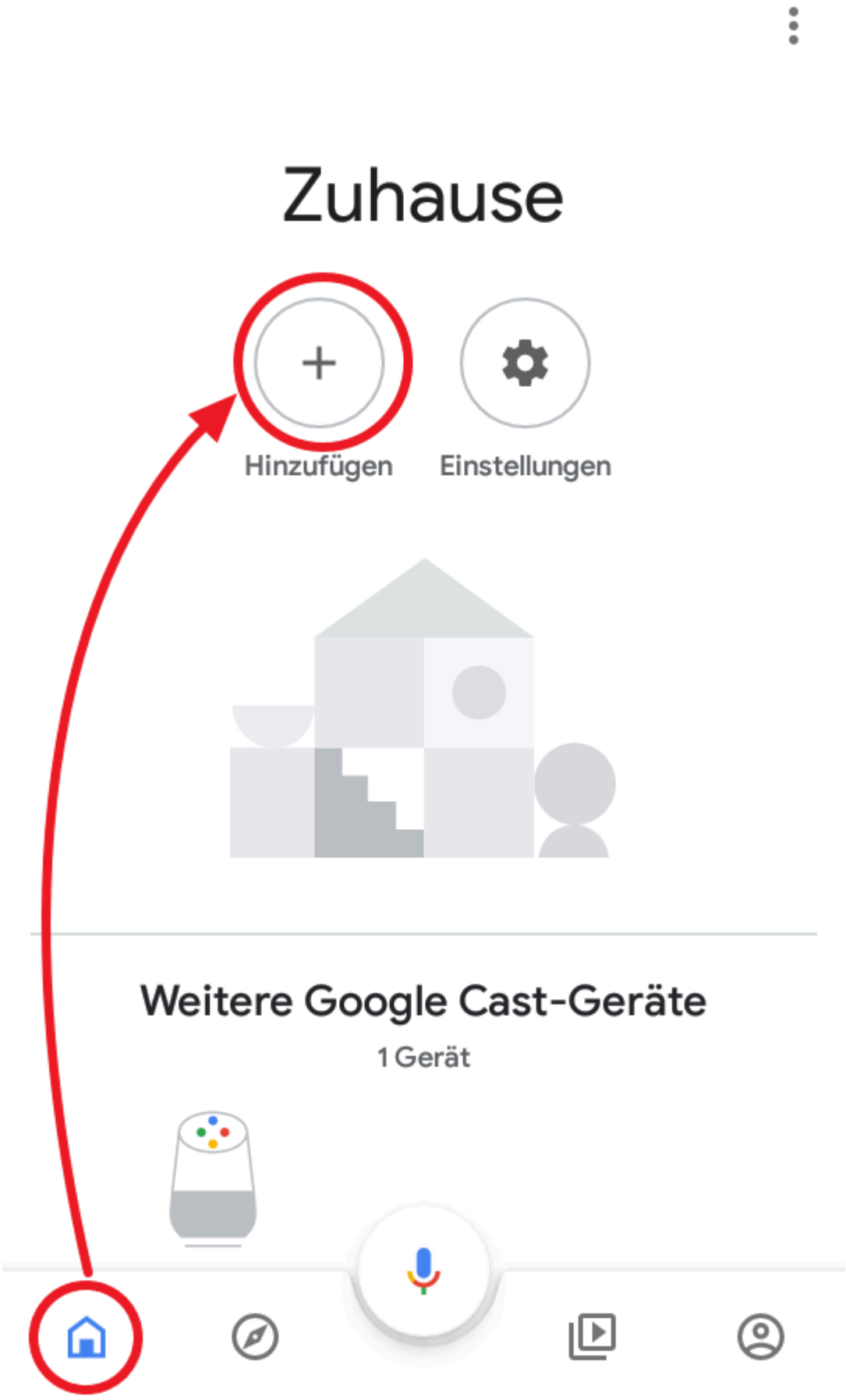


After applying, it is recommended to check that the column "Status" contains "OK" for all devices. If this is not the case, the column contains an error message that contains further information about the complication.
It should also be checked that the label at the top of the instance states "Status: Symcon Connect is OK". Otherwise, the Connect Service needs to be activated.

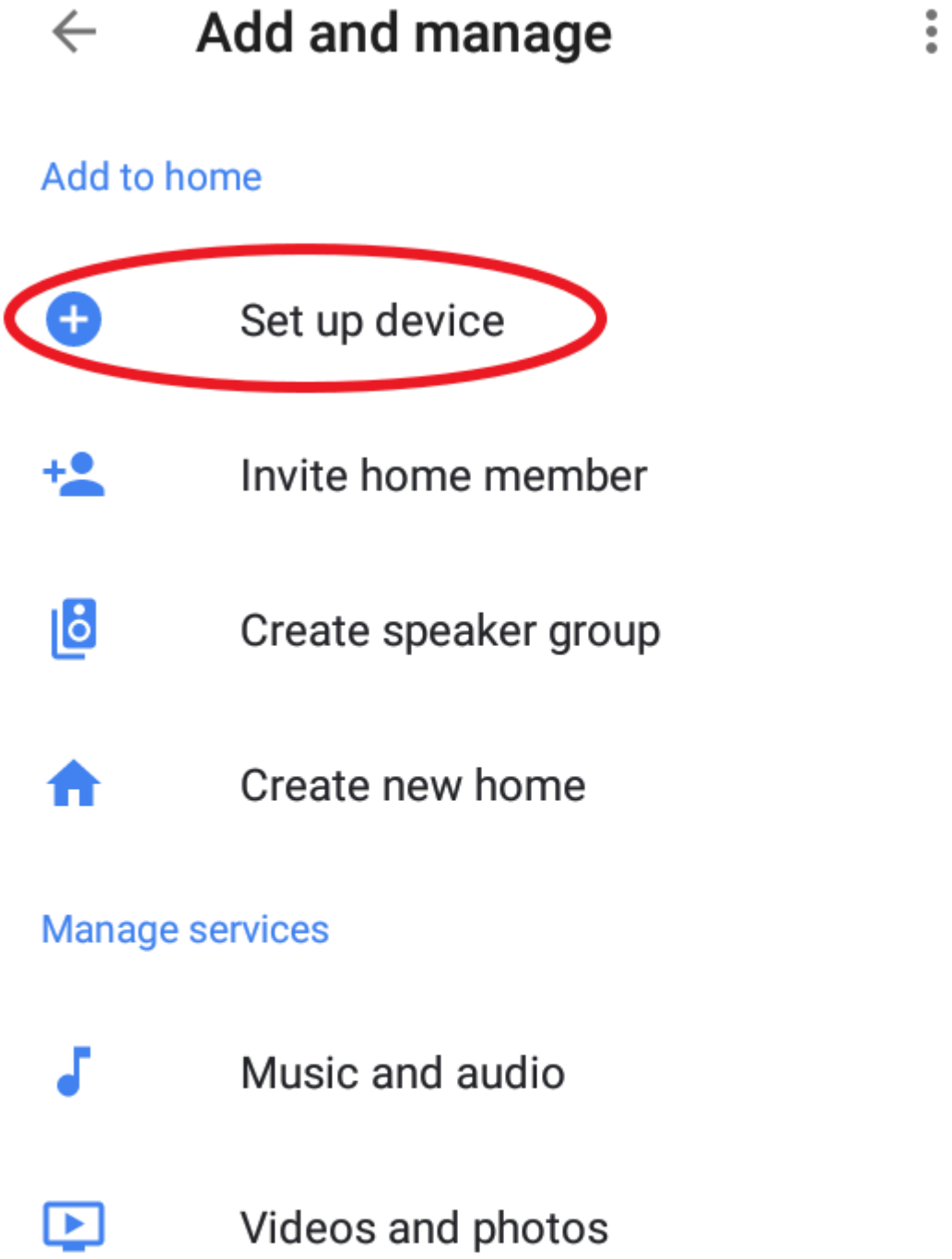


CONNECT IP-SYMCON TO GOOGLE ASSISTANT

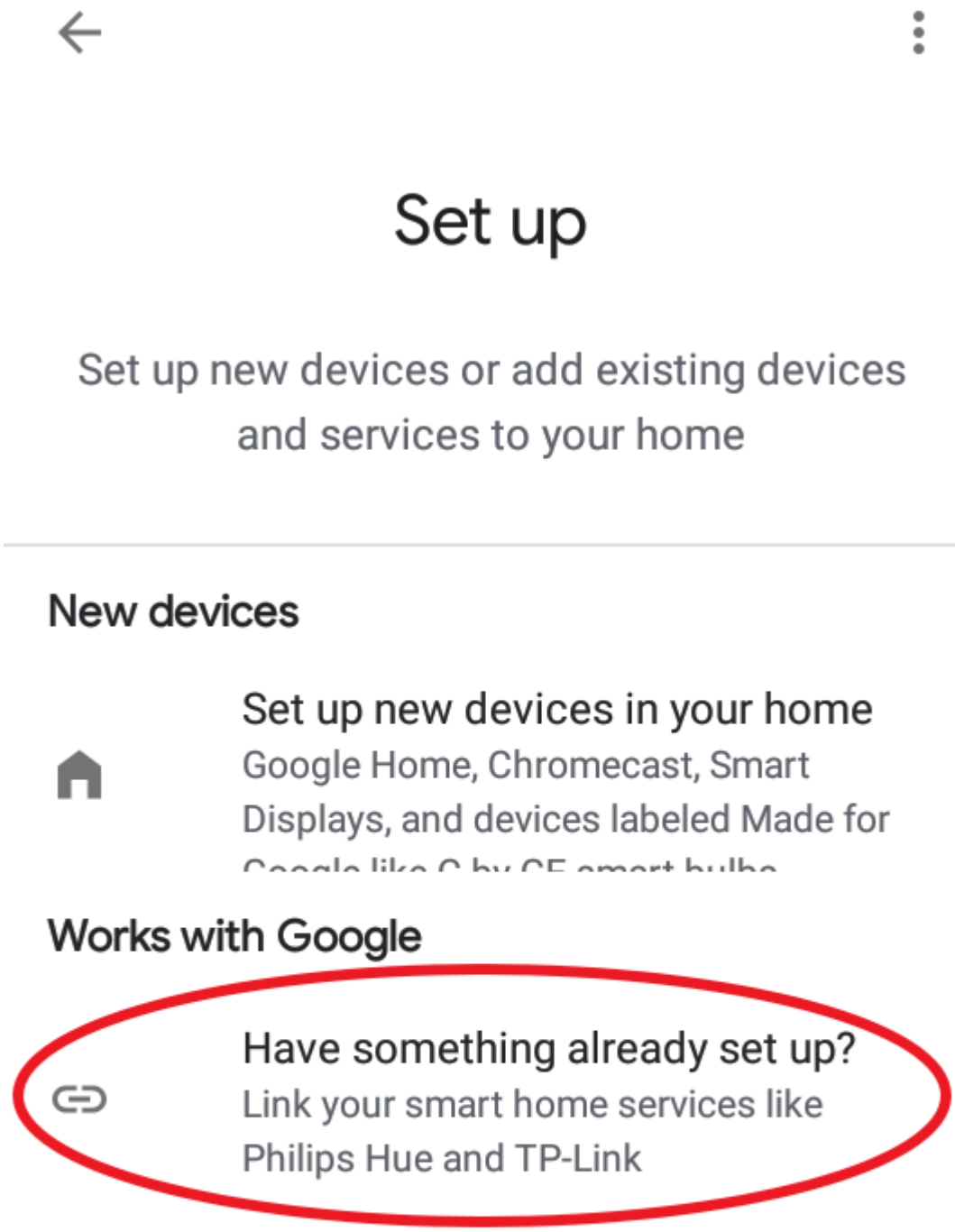
Finally, IP-Symcon needs to be connected to Google Assistant so the Google Assistant instance can receive requests from Google.
The installation is done with the Google Home app.
Within the app, select the home control tab at the bottom left and tap on "Add".



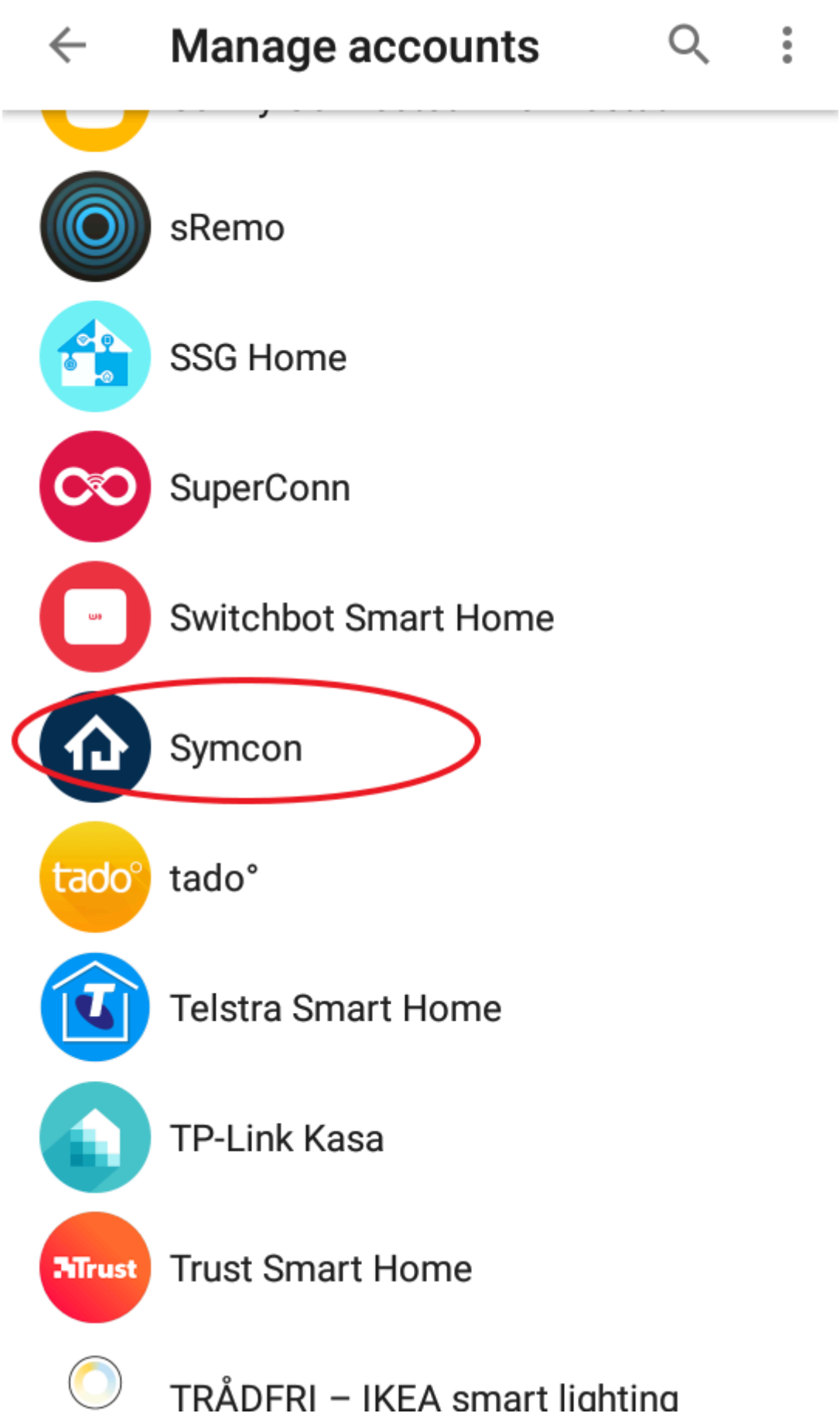
Tap "Set up device" in the menu.



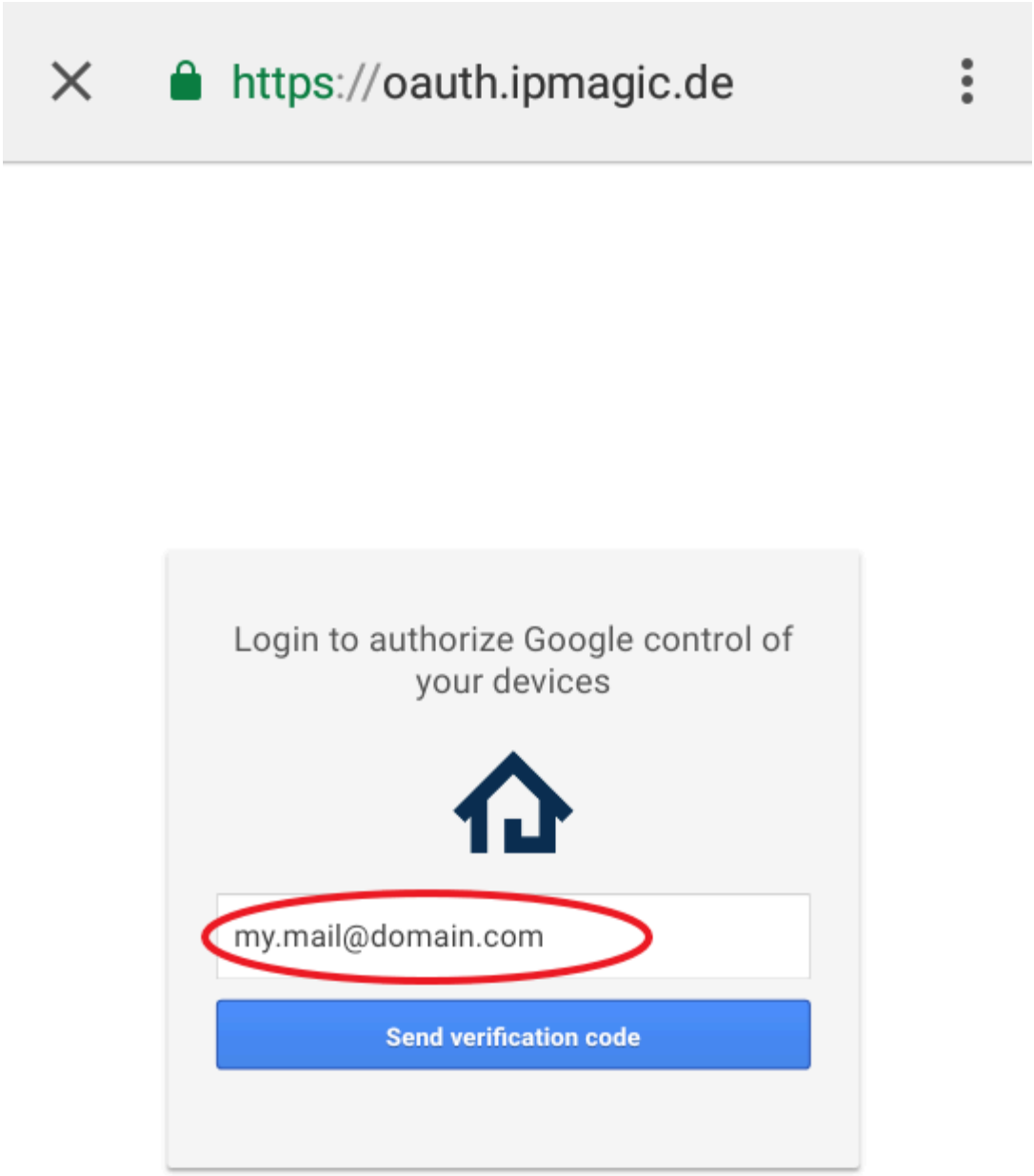
Tap "Have something already set up?".



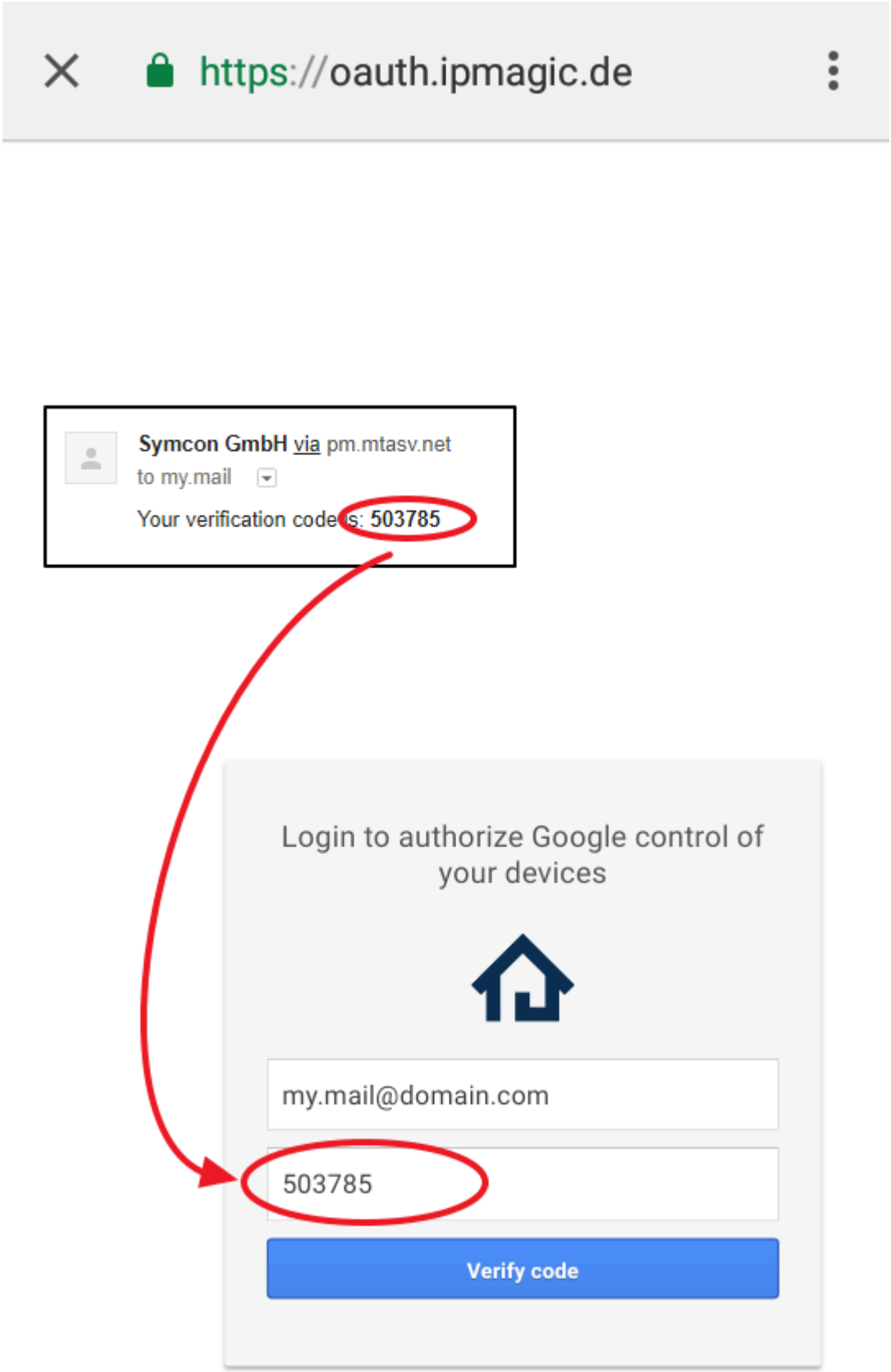
Tap "Symcon" in the displayed list.



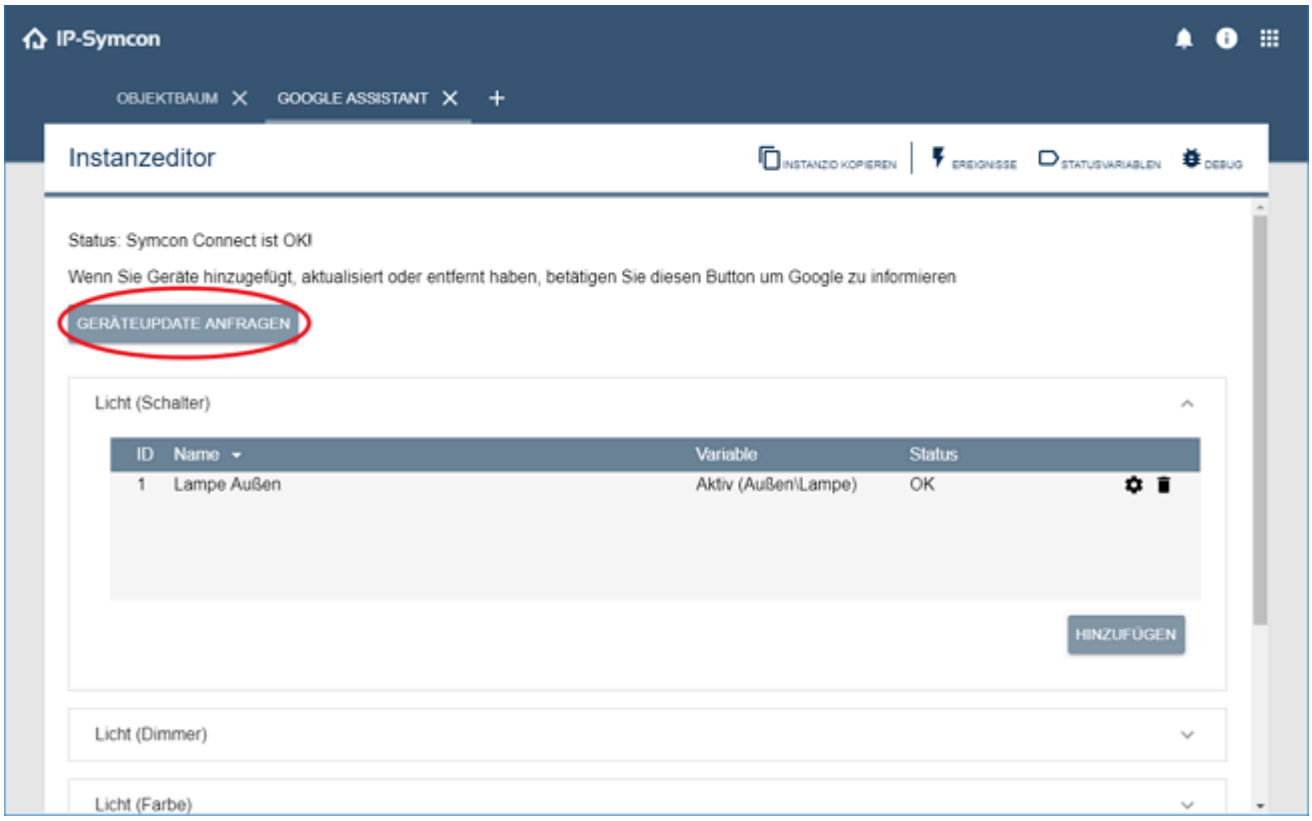
Enter the e-mail address, that the IP-Symcon is registered to, in the popup and confirm with "Send verification code".



An e-mail with a code is sent to the provided e-mail address. Enter the code from the e-mail into the popup and confirm with "Verify code".



After successful confirmation, the prepared devices appear in the Google Home app and can be used.


IP-Symcon
OBJEKTBAUM
GOOGLE ASSISTANT
Instanzeditor
INSTANZ KOPIEREN
EREIGNISSE
STATUSVARIABLEN
DEBUG
Status: Symcon Connect ist OK!
Wenn Sie Geräte hinzugefügt, aktualisiert oder entfernt haben, betätigen Sie diesen Button um Google zu informieren
GERÄTEUPDATE ANFRAGEN
Licht (Schalter)
ID
Name
Variable
Status
1
Lampe Außen
Aktiv (Außen\Lampe)
OK
HINZUFÜGEN
Licht (Dimmer)
Licht (Farbe)

Expert Options

Require: Symcon >= 5.0



The Expert Options usually do not need to be adjusted. Handle these options carefully to ensure that the Google Assistant module works as intended.

OPTIONS

NAME	DESCRIPTION
Emulate Status	If Emulate Status is active, the module does not wait for the update of switched variables. Instead, it returns the new state based on the triggered value.
Request device update	A device update can be requested from Google with this button, synchronizing the configured devices again with Google Assistant. This is usually done automatically when applying changes.

Generic Switch

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Generic Switch can be switched On or Off. Within Google Assistant, the device is interpreted as a common switch and can be used for different devices.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Google Assistant
Switch Variable	A triggerable variable of type Boolean

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the variable to True or False	"Ok Google, switch *<Name>* on."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~Switch

Light (Dimmer)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Dimmer) describe lamps that can be dimmed.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Google Assistant
Variable	A triggerable variable of type Integer or Float that dims the light

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the variable to 100% or 0%	"Ok Google, switch *<Name>* on."
Dim	Trigger the variable to the given value	"Ok Google, dim *<Name>* to 40%."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Variable	~Intensity.100, ~Intensity.255, ~Intensity.1

Light (Expert)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Expert) describe lamps whose properties within multiple variables can be switched independently.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Switch Variable	A triggerable variable of type Boolean, that switches the light on or off
Brightness Variable	A triggerable variable of type Integer or Float that dims the light (optional)
Color Variable	A triggerable Variable of type Integer with the profile ~HexColor, that triggers the color of the light (optional)



If optional variables are not set, Google recognizes that the lamp does not have the corresponding properties. If optional variables are set later on, a device update is required to discover the changed properties.

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the Switch Variable to True or False	"Ok Google, switch *<Name>* on."
Dim	Trigger the Brightness Variable to the given value	"Ok Google, dim *<Name>* to 40%."
Switch Color	Trigger the Color Variable to the given color	"Ok Google, switch *<Name>* to red."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search. To be detected as Expert light, at least the Switch Variable and another variable need to be detected.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~Switch
Brightness Variable	~Intensity.100, ~Intensity.255, ~Intensity.1
Color Variable	~HexColor

Light (Color)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Color) describe lamps, that can be switched to any color.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Google Assistant
Variable	A triggerable Variable of type Integer with the profile ~HexColor, that triggers the color of the light

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Trigger the variable to white or black	"Ok Google, switch *<Name>* on."
Dim	Trigger the brightness of the current color to the given value	"Ok Google, dim *<Name>* to 40%."
Switch Color	Trigger the variable to the given color	"Ok Google, switch *<Name>* to red."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Variable	~HexColor

Light (Switch)

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Light (Switch) describe lamps that can be switched on or off.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Google Assistant
Variable	A triggerable variable of type Boolean, that switches the light on or off

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Switch On or Off	Triggers the variable to True or False	"Ok Google, switch *<Name>* on."

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~Switch

Shutter

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Shutter can be opened and closed. If the device supports percentual opening, this is controllable as well.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Google Assistant
Shutter Variable	A triggerable variable of type Integer or Float, with the profile ~ShutterMoveStop or ~ShutterMoveStep. Alternatively, a variable can be used that has defined minimum and maximum values from its profile

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Open	Switch the variable to "Open" if it has a ~ShutterMove profile. Otherwise the variable is switched to the minimal value of the profile.	"Ok Google, open *<Name>*"
Close	Switch the variable to "Close" if it has a ~ShutterMove profile. Otherwise the variable is switched to the maximum value of the profile.	"Ok Google, close *<Name>*"
Percentual Opening	If the variable has a ~ShutterMove profile, switch it to "Open" if the percentual value is more than 50% and to "Close" otherwise. If the variable has another profile, the variable is switched according to the percentual value. It should be considered that the variable is switched to 0% if it is completely opened and other percentual values are applied analogously. For example, opening by 30% will result in 70% for the variable.	"Ok Google, open *<Name>* by 30%"

DEVICE SEARCH

An instance must have status variables with the fitting profiles to be detected by the device search.

VARIABLE	POSSIBLE PROFILES
Switch Variable	~ShutterMoveStop, ~ShutterMoveStep, ~Intensity.100, ~Intensity.255, ~Intensity.1

Scenes

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Scene can be activated and execute any kind of action.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Google Assistant
Action	An action that is executed on activation of the scene

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Activate Scene	Execute the action	"Ok Google, activate *<Name>*."

DEVICE SEARCH

This device type is not supported by the device search.

Scenes (deactivatable)

Require: Symcon >= 5.0

DESCRIPTION

Devices a the type Scene (deactivatable) can be activated and deactivated and execute any kind of action in the process.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Google Assistant
Activate Action	An action that is executed on activation of the scene
Deactivate Action	An action that is executed on deactivation of the scene

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Activate Scene	Execute the Activate Action	"Ok Google, activate *<Name>*."
Deactivate Scene	Execute the Deactivate Action	"Ok Google, deactivate *<Name>*."

DEVICE SEARCH

This device type is not supported by the device search.

Thermostat

Require: Symcon >= 5.0

DESCRIPTION

Devices of the type Thermostat can set temperatures.

PARAMETER

NAME	DESCRIPTION
Name	Name that is used to address the device via Amazon Alexa
Setpoint	A triggerable variable of type Float that sets the setpoint temperature in °C
Ambient Temperature	A variable of type Float that contains the ambient temperature in °C

POSSIBLE ACTIONS

ACTION	DESCRIPTION	POSSIBLE SENTENCE FOR ACTIVATION
Set Temperature	Trigger the variable toward the given temperature.	"Ok Google, set *<Name>* to 22 °C"

DEVICE SEARCH

This device type is not supported by the device search.

Notifications

ALERTING

Triggers an alarm when one of the sensor variables becomes active.

ARM_GetLastAlertID	Returns the ID of the variable that last triggered an alarm.
ARM_SetActive	Switches the alarming module with the InstanceID __InstanceID__ to the value __Value__ (true = On; false = Off).
ARM_SetAlert	Switches the alarm with the InstanceID __InstanceID__ to the value __Value__ (true = On; false = Off).

NOTIFICATION

Enables multi-level notification, with the level increasing after a defined time and resetting when acknowledged.

BN_IncreaseLevel	Increases the level and performs the action
BN_Reset	Resets the notification level and disables the chain
BN_SetNotifyLevel	Sets the notification to a specified level

ANNOUNCEMENT

The 'Announcement' Module provides the ability to play audio generated by AWS Polly through Sonos or Media Player.

DS_Play	Plays the text as an announcement.

DYNAMIC MAIL

Sends an e-mail with a dynamic text.

DM_SendMail	Use the subject and text to send an e-mail.

DONE NOTIFIER

Reports after a dead time, after a device has fallen below a certain consumption.

IMAP

Functions of the IMAP module for retrieving e-mails

IMAP_DeleteMail	deletes a mail with a specific e-mail(UID)

IMAP_GetCachedMails	The command returns an array containing information about the cached emails of an IMAP instance with ID __InstanceID__. Should the instance not exist, an alert is generated.
IMAP_GetMailEx	This command loads an email with the UID __UID__ from the IMAP instance with ID __InstanceID__ and returns an array containing the data of the e-mail. Should the instance or an e-mail with the given UID not exist, an alert is generated.

MEDIAPLAYER

Functions for controlling Media Player instances

WAC_AddFile	adds a music file to the playlist
WAC_ClearPlaylist	clears the playlist
WAC_GetPlaylistLength	returns the length of the playlist
WAC_GetPlaylistPosition	returns the current position in the playlist
WAC_Next	plays the next title of the playlist
WAC_Pause	pauses the playback
WAC_Play	plays the playlist
WAC_PlayFile	plays a music file directly
WAC_Prev	plays the previous title of the playlist
WAC_SetPlaylistPosition	sets the position in the playlist
WAC_SetPosition	Sets the position marker in the media player with the ID __InstanceID__ to the specified __Seconds__. The title must be played at execution time already.
WAC_SetRepeat	switches the repetition of the playlist on/off
WAC_SetShuffle	switches the shuffle playback on/off
WAC_SetVolume	sets the volume
WAC_Stop	stops the playback

POP3

Functions of the POP3 module for retrieving e-mails

POP3_DeleteMail	deletes a mail with a specific e-mail(UID)
POP3_GetCachedMails	The command returns an array containing information about the cached e-mails of a POP3 instance with ID __InstanceID__. Should the instance not exist, an alert is generated.
POP3_GetMailEx	returns an array with information for a specific e-mail (UID)

POPUP MODULE

Stylistic module for display in Visualization

SMS

Functions for sending SMS via SMS module

SMS_Send	sends an SMS to a number

SMTP

Functions for the SMTP module for sending e-mails

SMTP_SendMail	sends an e-mail to the default address
SMTP_SendMailAttachment	sends an e-mail with attachment to the default address
SMTP_SendMailAttachmentEx	sends an email with an attachment to any address
SMTP_SendMailEx	sends an e-mail to any address
SMTP_SendMailMedia	sends an email with an attachment of a media object of type "image/sound" to the default address
SMTP_SendMailMediaEx	sends an e-mail with an attachment of a media object of the "image/sound" type to any address

SPOTIFY

Controls the playback of Spotify

SPO_MakeAPIRequest	Makes a request to the Spotify web API with the specified parameters
SPO_NextTrack	If a current playback exists, the next song will be played.
SPO_Pause	If a current playback exists, it will be paused.
SPO_Play	If a current, possibly paused, playback exists, it will be continued.
SPO_PlayURI	Plays the Spotify resource with the URI __URI__ on the currently selected device.
SPO_PreviousTrack	If a current playback exists, the previous song will be played.
SPO_ResetToken	Resets the OAuth-token and thus the link to the Spotify account.
SPO_SetRepeat	Sets the repeat to the value Repeat.
SPO_SetShuffle	If Shuffle is true, random playback is enabled, otherwise disabled.

FAULT MANAGER

Indicates when selected variables are active and can acknowledge them depending on the setting.

SYMCONREPORT

Creates a report as CSV or PDF depending on the modules

MailReport	Creates a CSV report of archive values and sends it by e-mail(SMTP)

MR_SendInfo	Sends an info mail about the last completed time interval
MR_SetActive	Enables or disables the module

PDFReport (Energy)	Creates a report of archive values of two variables of a time span as PDF

RAC_GenerateEnergyReport	Generates a PDF with the values specified in the module with the InstanceID

PDFReport (Multi)	Creates a report of archive values of several variables of a time span as PDF

RAC_GenerateReport	Generates a PDF with the values specified in the module with the InstanceID

PDFReport (Multi Energy)	
PDFReport (Single)	Creates a report of archive values of a variable with min/max/avg of a time span as PDF

RAC_GenerateReport	Generates a PDF with the values specified in the module with the InstanceID

PHONE ANNOUNCEMENT

The Telephone Announcement Module enables the convenient linking of a VoIP instance and a text-to-speech instance (AWS Polly) to call a phone number and output a text when the call is answered. In addition, the module can respond to DTMF tones and output other texts.

TA_StartCall	Starts a call based on the current setting.
TA_StartCallEx	Starts a call to a phone number and outputs the text

PHONE CHAIN

The 'Phone Chain' module allows to call a list of telephone numbers one after another.

TELEGRAMBOT

Allows to send messages and perform actions

TEXT TO SPEECH

Functions for controlling a TTS module

TTS_GenerateFile	generates a WAV file with the desired text
TTS_Speak	speaks any text on the selected sound card

TTSAWSPOLLY

Can generate sound files in various formats

TTSAWSPOLLY_GenerateData	Queries the text via AWS and returns the speech data in the return.
TTSAWSPOLLY_GenerateFile	Queries the text via AWS and returns the file name to the vocie data.

CONSUMPTION ALERT

Monitors a counter variable for unusual consumption.

VBA_CheckAlert	Checks whether a limit value is exceeded

WATER ALERT

Monitors a meter variable for unnatural water consumption (leak/break).

WAA_CheckAlert	Checks whether the limit values have been exceeded

WATCHDOG

Monitors variables and triggers an alarm if they have not been updated or changed for too long.

WD_GetAlertTargets	returns an array with the overdue variables
WD_SetActive	de-/activate the instance

Alerting

Require: Symcon >= 5.3
The module triggers an alarm when one of the sensor variables becomes active. In this case, target variables are set to the maximum value or On (True) in the event of an alarm. Once an alarm has been switched, it is not automatically deactivated; it must be reset manually.

FUNCTION SCOPE

Configuration of sensor and target variables via list selection, which trigger the alarm or are switched in the event of an alarm
Adjustable switch-on delay
Switching on/off via Visualization button or script function
Conversion function for old versions of the alarm module

SOFTWARE INSTALLATION

Install the 'Alerting' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Alerting' module can be found using the quick filter.
More information in the Instances documentation.

CONFIGURATION PAGE

NAME	DESCRIPTION
Button "Conversion"	(Only shown if the lists are empty and old links are present) If an old version of the module has been detected, the old links can be merged into the new lists by pressing a button. If this is successful, a message window appears.
Sensor variables	This list contains the variables that trigger an alarm when updated to an active value. Variables with the value true or a value unequal 0 are considered active. If the variable has a .reversed profile, the values false and 0 are considered active.
Target variables	This list contains the variables that are switched on alarm. These must contain a default action or action script.
Switch-on delay	If greater than 0, the alarm will only be active after the set time.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Active	Boolean	Disables/enables alerting. If the alarm is deactivated, any existing alarm will also be deactivated as well as all selected target variables.
Time to activation	Integer	Displays the remaining time during the activation process. The value contains the Unix timestamp of the upcoming activation and is displayed using the Duration presentation.
Alarm	Boolean	Deactivates/activates the alarm and all selected target variables.
Active Sensors	String	Lists all active sensors and is hidden if none are active.

VISUALIZATION

The Visualization can be used to disable/enable alarms.
It is also displayed whether there is an alarm or not. A list of all still active sensors is displayed. The alarm can also be de-/activated manually.

ARM_GetLastAlertID

Require: Symcon >= 5.3

```
int ARM_GetLastAlertID (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

ID of the variable that last triggered an alarm.

DESCRIPTION

Returns the ID of the variable that last triggered an alarm.

EXAMPLE

```
ARM_GetLastAlertID(12345); //Get variable ID
```

ARM_SetActive

Require: Symcon >= 5.3

```
bool ARM_SetActive (int $InstanceID, bool $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Value to which the module is to be switched

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

Switches the alarming module with the InstanceID **InstanceID** to the value **Value** (true = On; false = Off).

EXAMPLE

```
ARM_SetActive(12345, true); //Activate Alarmierungsmodul
```

ARM_SetAlert

Require: Symcon >= 5.3

```
bool ARM_SetAlert (int $InstanceID, bool $Value)
```

PARAMETERS

InstanceID	ID of the device to be switched
Value	Value to which the alarm is to be switched

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

Switches the alarm with the InstanceID **InstanceID** to the value **Value** (true = On; false = Off).

EXAMPLE

```
ARM_SetAlert(12345, false); //Deactivate alarm
```

Notification

Require: Symcon >= 5.1
This module enables multi-level notification, where the level increases after a defined time and is reset when acknowledged. Various actions can be performed when a new level is reached.

FUNCTION SCOPE

Starting a multi-stage notification chain with a variable
Actions can be set individually at each stage:
    Execute scripts
    Send push messages, e-mails or SMS
    Make a phone call with an announcement (if the "Phone Announcement" module is installed)
    Make an announcement over the loudspeaker (if the "Announcement" module is installed)
    Send message via Telegram (if "TelegramBot" module is installed)
    Increase to next level after certain time
Acknowledgement via attached script or push message ends notifications
Individual levels can be deactivated if required

SOFTWARE INSTALLATION

Via the Module Store install the 'Notification' module.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Notification' module can be found using the quick filter.
    More information about adding instances in the documentation of instances.
Select a variable at 'Trigger
    If this variable takes a value that is not neutral (false for boolean, 0 for integer and float, '' for string), the notification chain is started at level 1
Add any number of levels to the notification levels list and configure them
    Duration defines the time until the next level is activated
    Actions are executed when a level is reached
    Active defines whether the stage should be executed. Stages that are not active are skipped
    'Status' contains error messages if something is not correct when configuring the stage, otherwise "OK".

SET UP ACTIONS

Each action has parameters for the action type, a recipient object, a recipient address, a title, a message and a message variable. The "action" defines what should be called. (e.g. A script or an email sent).

Further information like "title" and "message" will be sent to the set "recipient object". The content of the message is defined by the text in the configuration field "Message". This is concatenated with the text in the "Message variable".
For generic messages, the content of the message variable is available as '{variable}'. Likewise, line breaks can be inserted by '\n'. The recipient address has different functionality depending on the action.

SCRIPT

The script selected as recipient object will be executed. During this call the following system variables can be used:

SYSTEM VARIABLE	DESCRIPTION
$_IPS['RECIPIENT']	The contents of the recipient address table field
$_IPS['TITLE']	The content of the table field Title
$_IPS['MESSAGE']	The content of the table field Message
$_IPS['MESSAGE_VARIABLE']	The ID of the message variable

PUSH

A push message is sent to all devices in the selected web front. This message links the acknowledgement script. So by tapping on the push message, the notification can be acknowledged. The recipient address has no effect with this action type. The message has a maximum length of 256 characters.

E-MAIL (SMTP)

An e-mail is sent via the selected SMTP instance. If a recipient address is specified, the email will be sent to that address. If no recipient address is specified, the e-mail will be sent to the specified recipient of the SMTP instance. If the 'Advanced Response' option is enabled, a block of links can be inserted with the keyword '{actions}', through which the available actions can be executed.

SMS

An SMS will be sent via the selected SMS instance to the phone number specified in the recipient address. If the 'Extended response' option is enabled, a link can be inserted with the '{actions}' keyword, through which the available actions can be performed. If a message is longer than 160 characters (limitation by SMS), it will be split up to 2 additional SMS (maximum 459 characters).

TELEPHONE ANNOUNCEMENT (ONLY AVAILABLE IF THE TELEPHONE ANNOUNCEMENT MODULE IS INSTALLED)

The phone number specified in the recipient's address will be called and the title and the message will be read out. If the 'Extended answer' option is enabled, the corresponding

action can be performed with the 0-9 keys. If desired, the '{actions}' keyword can be used to include the available actions in the message.

PASSAGE (ONLY AVAILABLE IF THE ANNOUNCEMENT MODULE IS INSTALLED)

The title and message are read out using the selected instance. The recipient address has no effect with this action type.

TELEGRAM (ONLY AVAILABLE IF THE TELEGRAM BOT MODULE IS INSTALLED)

The title and message are sent to the recipient using the selected instance. The recipient address can be either the name or the UserID of the Telegram recipient. If the recipient address is left empty, all recipients stored in the Telegram bot will be notified.

ADVANCEDREPLY

If extended reply is enabled, different actions can be defined in the corresponding list. To define what is additionally executed on an action, a triggering event can be created. The variable 'Response action' is selected as the triggering variable and 'On certain value' as the trigger. The desired action can now be selected as the value.

STATUSVARIABLES

STATUSVARIABLE	DESCRIPTION
Response action	Can be clicked via the Visualization or called by command to reset the notification
Notification active	Controls whether the notification module is active or not
Notification level	Contains the current notification level

VISUALIZATION

Via the Visualization "Reply action" can be executed to reset the notification.
The Visualization can be used to switch the notification on/off via "Notification active".
The current notification level is displayed.

BN_IncreaseLevel

Require: Symcon >= 5.1

```
bool BN_IncreaseLevel (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

If another level exists, the current level is increased. In this case, the actions of the new level are executed.

EXAMPLE

```
BN_IncreaseLevel(12345);
```

BN_Reset

Require: Symcon >= 5.1

```
bool BN_Reset (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Resets the notification level and disables the notification chain

EXAMPLE

```
BN_Reset(12345);
```

BN_SetNotifyLevel

Require: Symcon >= 5.1

```
bool BN_SetNotifyLevel (int $InstanceID, int $Level)
```

PARAMETERS

InstanceID	ID of the Instance
Level	Notification module level

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Switches the notification module to the specified level. The actions of the new level are executed directly. Actions of skipped stages are NOT executed.

EXAMPLE

```
BN_SetNotifyLevel(12345, 2);
```

Announcement

Require: Symcon >= 5.4
The 'Announcement' module offers the possibility to play back audio data generated by AWS Polly via Sonos or under Windows the Media Player.

FUNCTION SCOPE

Playback of audio data generated by AWS Polly via a Sonos player or the "Symcon" media player
Volume of the announcement is adjustable
Announcement on change of text variable. Alternatively via offered function

SOFTWARE INSTALLATION

Install the 'Announcement' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Announcement' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Text-to-Speeach instance	AWS Polly instance, which should be used to create the announcement. The output format must be mp3. Sonos can only be used with sampling rates 16000 Hz, 22050 Hz, 24000 Hz, 32000 Hz, 44100 Hz, 48000 Hz
Output Device	Output Instance Type
Symcon IP	The IP address where the Sonos player reaches IP Symcon.
Sonos/Media Player	Instance over which the announcement is played
Volume	Sonos: the volume change of the announcement (0 → no change , 50 → 50, +10 → louder by 10) Media Player: the volume of the announcement in percent (will not be reset)

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Text	String	Text which is used for the announcement. When the variable is updated, the content is played back.

DS_Play

Require: Symcon >= 5.4

```
bool DS_Play (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the Instance
Text	Text to be played

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Plays the text as an announcement.

EXAMPLE

```
DS_Play(12345, "This is a text");
```

Dynamic Mail

Require: Symcon >= 7.1
Sends an e-mail with a dynamic text.

RANGE OF FUNCTIONS

Sending an e-mail with a fixed/dynamic text
Dynamic by replacing placeholders with the variable values

SOFTWARE INSTALLATION

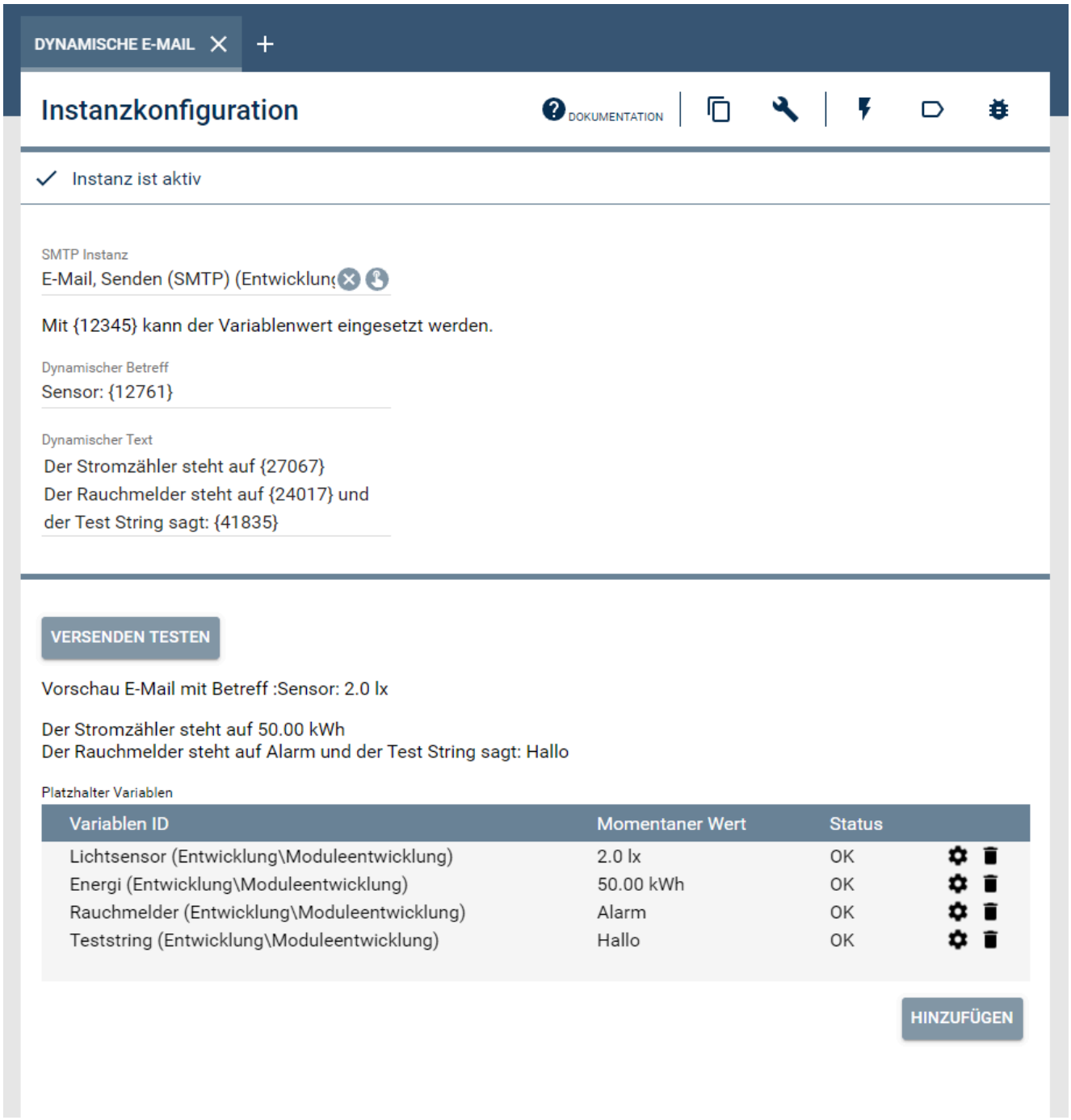
Install the 'Dynamic e-mail' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add instance', the 'Dynamic E-Mail' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation
Configuration page:

NAME	DESCRIPTION
SMTP instance	SMTP instance which is fully configured
Dynamic subject	Text input which can be set.
Dynamic text	Multi-line text input.

The subject and the text are given a dynamic by inserting variable values. These can be set within the text by the variable ID placed in curly brackets. If the ID is too short or not a variable, the brackets with the number remain.



Action area:

NAME	DESCRIPTION
Send Test	Sends an e-mail with the set subject and text
Preview e-mail with subject	A preview with set variable values
Placeholder variables	Table with the given placeholders

Table:
The placeholders found are listed in the table with their current value and status. If a placeholder is not available, the number entered is not an ID. If a placeholder is invalid, the ID found is not a variable.

STATUS VARIABLES AND PROFILES

No variables or profiles are created.

VISUALIZATION

The instance has no functionality in the visualization.

DM_SendMail

Require: Symcon >= 7.1

```
bool DM_SendMail (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

Use the subject and text to send an e-mail.

EXAMPLE

```
DM_SendMail(12345);
```

Done Notifier

Require: Symcon >= 4.2
The ready indicator signals whether a device is ready. For this purpose, the variable of the power consumption of the device is selected and a limit value is defined. As soon as this limit value is exceeded for the first time, the ready detector is started. If the power consumption falls below this limit value and does not exceed it again within an adjustable period of time, the status variable is set to "Ready".

FUNCTION SCOPE

Switching the entire module on/off
Selection of a source variable
Setting the limit value
Time span until ready message is set after limit value is undershot

SOFTWARE INSTALLATION

Under "Add Instance" is 'Done Notifier' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Source	Source variable which is used for comparison with the limit value.
Limit	Value below which a ready message is set.
Interval	Period until a ready message is set. Only becomes active after the value falls below the limit value.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Active	boolean	Turns the module on/off
Status	integer	Indicates the status (Off/Running/Complete)

PROFILE:

NAME	TYPE
FM.Status	Integer

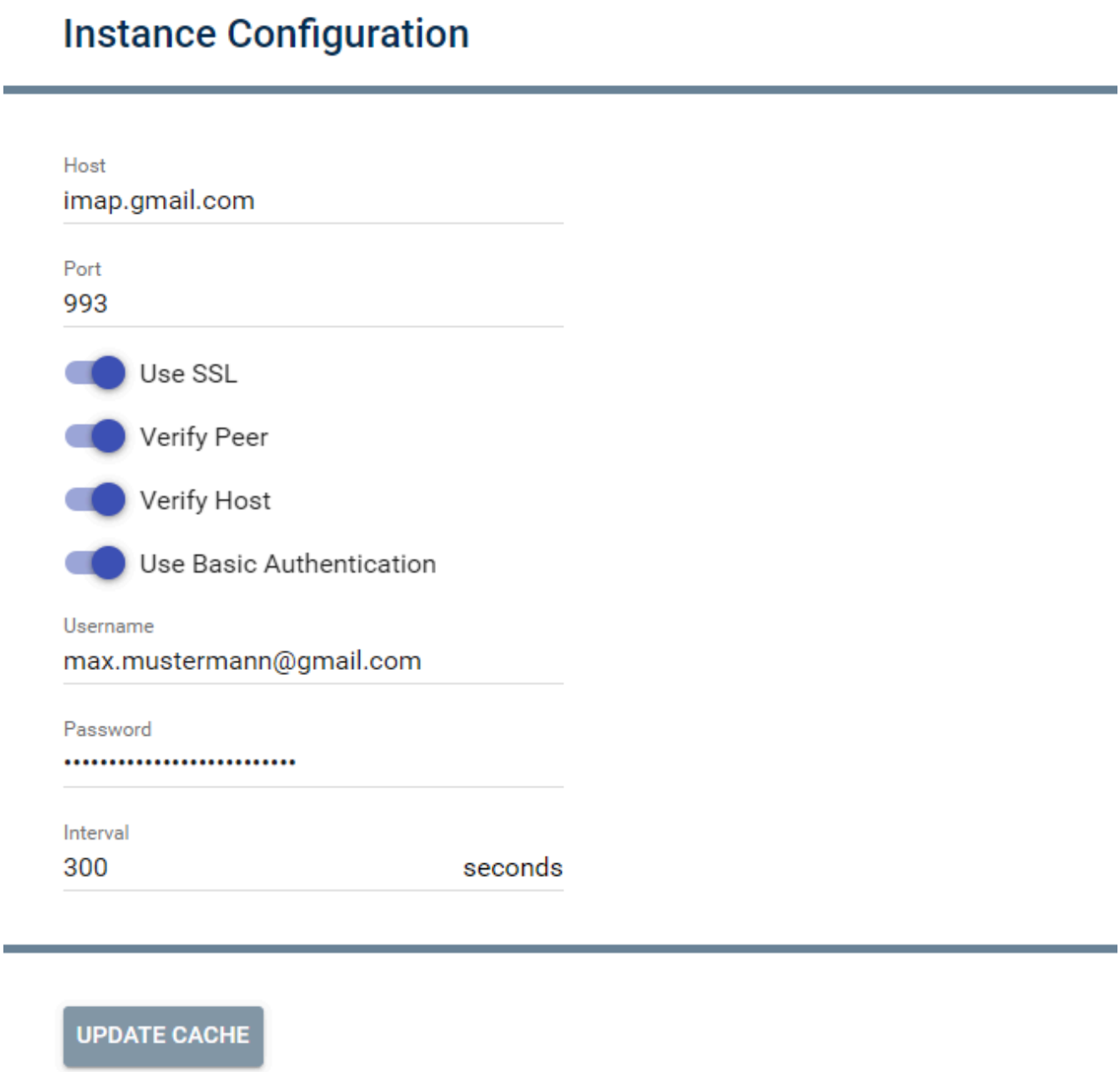
IMAP

Require: Symcon >= 2.2
The IMAP module can be configured using the configuration page within the Object Tree of the Management Console. The parameters for host and port can be taken from the manual of the mail provider. The username and password are the login data for the e-mail account.



Since October 2022 a connection to Microsoft 365 (outlook.office365.com) isn't supported. This is because of the only use of OAuth2 authentification supported by Microsoft.

To display the e-mails in the visualization, the interval must be set. It determines the cyclic request of the inbox. Only the subject line is requested. The complete e-mail will only be loaded when requested by the visualization.
On the variable "Last Message" can be determined if and when new mails arrive. If no e-mails are in the mailbox, the variable has the value 0. The "Unread Messages" variable returns the number of unread messages.



PROPERTY	DESCRIPTION
Host	Server name or IP of email provider
Port	Port, that is handled by the server
Use SSL	Defines if SSL should be used
Verify Peer	Verify that hosts SSL certificate is correct (Only visible if Use SSL is active)
Verify Host	Verify that host URL from the SSL certificate matches the property Host (Only visible if Use SSL is active)
Use Basic Authentication	Defines if a username password is required
Username	Username
Password	Password
Interval	Number of seconds in which the inbox is requested



An overview for host, port, and authentication for many providers can be found here: Table

IMAP_DeleteMail

Require: Symcon >= 5.0

```
bool IMAP_DeleteMail (int $InstanceID, string $UID)
```

PARAMETERS

InstanceID	ID of the IMAP-Instance
UID	UID of the email to be deleted

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes an email with the UID **UID** from the IMAP-Instance with the ID **InstanceID**. If the instance or an e-mail with the transferred UID does not exist, a warning is generated.

EXAMPLE

```
// Deletes the mail with the e-mail(UID) 1234
IMAP_DeleteMail (12345, "1234"));
```

IMAP_GetCachedMails

Require: Symcon >= 2.2

```
array IMAP_GetCachedMails (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the IMAP-Instance

RETURNS

If the command was executed successfully, it returns as its result an array of data to the cached emails, otherwise a Boolean with the value FALSE.

DESCRIPTION

The command returns an array containing information about the cached emails of an IMAP instance with ID InstanceID. Should the instance not exist, an alert is generated.

EXAMPLE

```
print_r(IMAP_GetCachedMails(12345));

/* returns e.g.:
Array
(
[0] => Array
(
[Date] => 1295756412
[Flags] => SEEN
[Recipient] => recipient@test.test
[SenderAddress] => sender@test.test
[SenderName] => Test Sender
[Subject] => 3 2 1 Test
[UID] => 1234
)
)
*/
```

IMAP_GetMailEx

Require: Symcon >= 2.2

```
array IMAP_GetMailEx (int $InstanceID, string $UID)
```

PARAMETERS

InstanceID	ID of teh IMAP-Instance
UID	UID of the email to be loaded

RETURNS

If the command was executed successfully, it returns as its result an array of data to the cached emails, otherwise a Boolean with the value FALSE.

DESCRIPTION

This command loads an email with the UID **UID** from the IMAP instance with ID **InstanceID** and returns an array containing the data of the e-mail. Should the instance or an e-mail with the given UID not exist, an alert is generated.

EXAMPLE

```
print_r(IMAP_GetMailEx(12345, "1234"));

/* returns e.g.:
Array
(
[ContentType] => text/plain
[Date] => 1295756412
[Flags] => SEEN
[Recipient] => ips@test.test
[SenderAddress] => sender@test.test
[SenderName] => Test Sender
[Subject] => 3 2 1 Test
[Text] => This is a test!
[UID] => 1234
)
*/
```

MediaPlayer

With the media player module, various sound files as Wave, Midi or MP3 can be played. The module allows the use of several sound cards. This allows speech outputs (Information, Warnings) or your favorite music to be played.

TIPS & TRICKS

You can create as many Media Player instances, and even configure them on the same sound card. This allows to continue to running the music in the background (quiet), while a second instance makes an announcement.

WAC_AddFile

```
bool WAC_AddFile (int $InstanceID, string $Filename)
```

PARAMETERS

InstanceID	ID of the controlled media player
Filename	Path to the file to be played

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Adds a file to the internal playlist of the Media Player with ID **InstanceID**.
It does not check whether the file already exists. The same file can appear several times in the playlist.
To start playing, you must call WAC_Play.

EXAMPLE

```
//Please note the slashes in the path.
//The exact meaning you can read in the PHP manual:
//http://de.php.net/manual/de/language.types.string.php
WAC_AddFile(12345, "D:/MP3s/favoritesong.mp3");
```

WAC_ClearPlaylist

```
bool WAC_ClearPlaylist (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the controlled media player

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Clears the internal playlist in the Media Player with ID **InstanceID**.

EXAMPLE

```
WAC_ClearPlaylist(12345);
```

WAC_GetPlaylistLength

```
int WAC_GetPlaylistLength (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the controlled media player

RETURNS

Number of tracks, which are registered in the playlist.

DESCRIPTION

Returns the length of the internal playlist of the Media Player with ID **InstanceID**.

EXAMPLE

```
$len = WAC_GetPlaylistLength(12345);
if($len > 0)
WAC_Play(12345);
```

WAC_GetPlaylistPosition

```
int WAC_GetPlaylistPosition (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

Value between 1 and WAC_GetPlayListLength, if the playlist should be empty, it returns -1.

DESCRIPTION

Returns the current position of the internal playlist of the Media Player with ID **InstanceID**.

EXAMPLE

```
$pos = WAC_GetPlaylistPosition(12345);
```

WAC_Next

```
bool WAC_Next (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the controlled media player

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Goes in the internal playlist by the media player with ID **InstanceID** forward a title. If shuffle is on, a song is selected randomly. If playback is started, after the retreat, the new current track will be played from the beginning.
If the playlist is empty, an error message is generated.
If the playlist arrived at the end, an error message will be generated or, if repeat is activated, restarts the playlist from the beginning.

EXAMPLE

```
WAC_Next(12345);
```

WAC_Pause

```
bool WAC_Pause (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the controlled media player

RETURNS

If the command was executed successfully, it returns the result **TRUE**, otherwise **FALSE**. It is also returned **TRUE** if no active playback is available.

DESCRIPTION

Pause the playback of the internal playlist in Media Player with ID **InstanceID**. By a renewed call of WAC_Pause or a call of WAC_Play the playback can be continued.

EXAMPLE

```
WAC_Pause(12345);
```

WAC_Play

```
bool WAC_Play (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the controlled media player

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Starts playing the playlist from the internal media player with ID **InstanceID**. If you have not added any files, you can do so via WAC_AddFile. A direct function to play a single file is WAC_PlayFile. Please also note the supported file formats that can be found on WAC_PlayFile.
Should be a playback has already been started, by calling this command starts playback from the beginning of the file.
If the playlist is empty, an error message is generated.

EXAMPLE

```
WAC_Play(12345); //Play
```

WAC_PlayFile

```
bool WAC_PlayFile (int $InstanceID, string $Filename)
```

PARAMETERS

InstanceID	ID of the controlled media player
Filename	Path to the file to play

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Plays a file with a **Filename** directly from the media player with the ID **InstanceID**. The command is only available for simplification. They can also be replaced by a sequence of WAC_ClearPlaylist, WAC_AddFile, WAC_Play.
Please refer to the supported file formats.
Currently supported formats: ***.wav,*** **.mp3, *.wma** (each without DRM)

EXAMPLE

```
//Please note the slashes in the path.
//The exact meaning, read the PHP manual:
http://de.php.net/manual/de/language.types.string.php
WAC_PlayFile(12345, "D:/MP3s/Favorite song.mp3");
```

WAC_Prev

```
bool WAC_Prev (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the controlled media player

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Goes in the internal playlist in Media Player with ID **InstanceID** back a title. If the playback is already started, the new current track will be played from the beginning after the retreat.
If the playlist is empty, an error message is generated.
If the playlist has reached at the beginning, the current track will be played from the beginning.

EXAMPLE

```
WAC_Prev(12345);
```

WAC_SetPlaylistPosition

```
bool WAC_SetPlaylistPosition (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the current position of the internal playlist in Media Player with ID **InstanceID** to the new **Position**. After running the WAC_Play must be execute when the playback should be started from the new position. Otherwise the next song will be played after the end of the current track.

EXAMPLE

```
WAC_SetPlaylistPosition(12345, 0); //An Anfang setzen
```

WAC_SetPosition

```
bool WAC_SetPosition (int $InstanceID, int $Seconds)
```

PARAMETERS

InstanceID	ID of the controlled media player
Seconds	Label in seconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the position marker in the media player with the ID **InstanceID** to the specified **Seconds**. The title must be played at execution time already.

EXAMPLE

```
WAC_PlayFile(12345, "my.mp3"); //Play title
WAC_SetPosition(12345, 60); //jump on the 1 minute mark in the song
```

WAC_SetRepeat

```
bool WAC_SetRepeat (int $InstanceID, bool $Repeat)
```

PARAMETERS

InstanceID	ID of the controlled media player
Repeat	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Enables / Disables the repetition of the playlist by the media player with the ID InstanceID. The command takes effect immediately, but will be noticeable until the next track in the current playlist. The current playback is not affected.

EXAMPLE

```
WAC_SetRepeat(12345, true); //Activate playlist repeat
```

WAC_SetShuffle

```
bool WAC_SetShuffle (int $InstanceID, bool $Shuffle)
```

PARAMETERS

InstanceID	ID of the controlled media player
Shuffle	TRUE for On, FALSE for Off

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Enables / disables the shuffle by the media player with the ID InstanceID.
The command takes effect immediately, but will be noticeable until the next track in the current playlist. The current playback is not affected.

EXAMPLE

```
WAC_SetShuffle(12345, true); //Zufallswiedergabe einschalten
```

WAC_SetVolume

```
bool WAC_SetVolume (int $InstanceID, int $Volume)
```

PARAMETERS

InstanceID	ID of the controlled media player
Volume	0-100 (%)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Fades the volume from the media player with the ID **InstanceID** within 500ms to the desired volume.

EXAMPLE

```
WAC_SetVolume(12345, 85);
```

WAC_Stop

```
bool WAC_Stop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the controlled media player

RETURNS

If the command was executed successfully, it returns the result **TRUE**, otherwise **FALSE**. It is also returned **TRUE** if no active playback is available.

DESCRIPTION

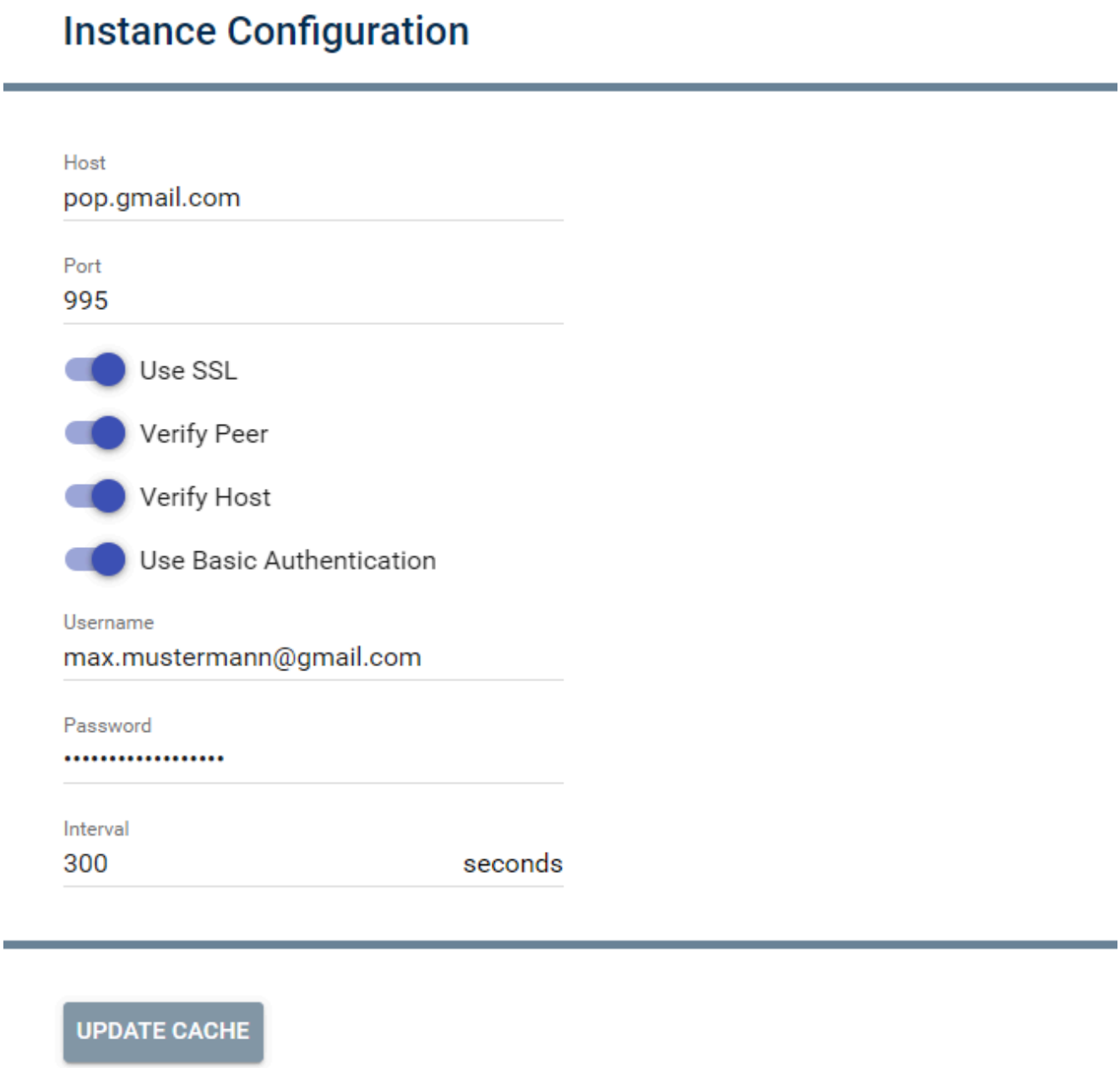
Stops the playback of the internal playlist in Media Player with ID **InstanceID**.

EXAMPLE

```
WAC_Stop(12345);
```

POP3

Require: Symcon >= 2.2
The POP3 module can be configured using the configuration page within the Object Tree of the Management Console. The parameters for host and port can be taken from the manual of the mail provider. The username and password are the login data for the e-mail account. To display the e-mails in the visualization, the interval must be set. It determines the cyclic request of the inbox. Only the subject line is requested. The complete e-mail will only be loaded when requested by the visualization.
On the variable "Last Message" can be determined if and when new mails arrive. If no e-mails are in the mailbox, the variable has the value 0. Unlike IMAP, the display of the number of unread variables is technically not possible for POP3.



PROPERTY	DESCRIPTION
Host	Server name or IP of email provider
Port	Port, that is handled by the server
Use SSL	Defines if SSL should be used
Verify Peer	Verify that hosts SSL certificate is correct (Only visible if Use SSL is active)
Verify Host	Verify that host URL from the SSL certificate matches the property Host (Only visible if Use SSL is active)
Use Basic Authentication	Defines if a username password is required
Username	Username
Password	Password
Interval	Number of seconds in which the inbox is requested



An overview for host, port, and authentication for many providers can be found here: Table

POP3_DeleteMail

Require: Symcon >= 5.0

```
bool POP3_DeleteMail (int $InstanceID, string $UID)
```

PARAMETERS

InstanceID	ID of the POP3-Instance
UID	UID of the email to be deleted

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function deletes an email with UID **UID** from the POP3-Instance with ID **InstanceID**. If the instance or an e-mail with the transferred UID does not exist, a warning is generated.

EXAMPLE

```
// Deletes the mail with the e-mail(UID) 1234
POP3_DeleteMail(12345, "1234");
```

POP3_GetCachedMails

Require: Symcon >= 2.2

```
array POP3_GetCachedMails (int $InstanceID)
```

PARAMETERS

InstanceID	ID of POP3-Instance

RETURNS

If the command was executed successfully, it returns as result an array of data of the cached emails, otherwise a Boolean with the value **FALSE**.

DESCRIPTION

The command returns an array containing information about the cached e-mails of a POP3 instance with ID **InstanceID**. Should the instance not exist, an alert is generated.

EXAMPLE

```
print_r(POP3_GetCachedMails(12345));

/* returns e.g.:
Array
(
[0] => Array
(
[Date] => 1295756412
[Flags] =>
[Recipient] => recipient@test.test
[SenderAddress] => sender@test.test
[SenderName] => Test Sender
[Subject] => 3 2 1 Test
[UID] => 1234
)
)
*/
```

POP3_GetMailEx

Require: Symcon >= 2.2

```
array POP3_GetMailEx (int $InstanceID, string $UID)
```

PARAMETERS

InstanceID	ID of POP3-Instance
UID	UID of the email to be loaded

RETURNS

If the command was executed successfully, it returns the result in an array with data from the e-mail, otherwise a Boolean with FALSE.

DESCRIPTION

This command loads an email with the UID **UID** from POP3 instance with ID **InstanceID** and returns an array containing the data of the e-mail. Should the instance or an e-mail with the given UID not exist, an alert is generated.

EXAMPLE

```
print_r(POP3_GetMailEx(12345, "1234"));

/* returns e.g.:
Array
(
[ContentType] => text/plain
[Date] => 1295756412
[Flags] =>
[Recipient] => ips@test.test
[SenderAddress] => sender@test.test
[SenderName] => Test Sender
[Subject] => 3 2 1 Test
[Text] => This is a test!
[UID] => 1234
)
*/
```

Popup Module

The Popup Module is an instance that does not contain its own function or status variables. The module can be used as a stylistic means to display objects within a popup.

APPLICATION

Several objects belong to a subject group, which should only be viewed quickly when required. In order to display the objects in the popup, they must be copied into the popup instance themselves or via a link.

VISUALIZATION

How the module is displayed can be seen here:
Popup display



In the mobile apps, the popup instance has the same representation as a Dummy Module, as these do not allow nesting.

SMS

Require: Symcon >= 4.1
SMS is an IP-Symcon module which enables sending SMS via Clickatell. A connection to IP-Symcon is established via the REST API.



Since November 2016, Clickatell has only offered the REST API, which IP-Symcon has integrated with the SMS (Clickatell) module.

INSTALLATION (REST API)

In order to be able to use the SMS module, an account at clickatell.com is required.
The following step-by-step instructions explain the registration process:

The following URL opens the login page for registering with the SMS gateway:
clickatell.com/sign-up
After confirming the verification email, one must log in at portal.clickatell.com . "Platform" must be selected.
Telephone numbers must be verified and an "integration" set up via the dashboard on the portal.clickatell.com page.
An integration configures the type of communication.
Credit is managed through portal.clickatell.com .

In order for IP-Symcon to be able to communicate with the gateway, the API key for the integration in IP-Symcon must also be entered.

The following URL must be opened:
portal.clickatell.com
The "**API key**" option must be copied.
The APIKey received must be specified in IP-Symcon on the configuration page of the SMS module.



The sender number must be verified in the Clickatell account before this SMS can be sent.

INSTALLATION (DEPRECATED, HTTPS API)

This API is no longer supported by Clickatell as of November 2016.
For this reason, the module has been revised.

INTEGRATION IN IP-SYMCON

The configuration consists exclusively of the API key that was set up in the account.

So far there is no way to query the credit with the new REST API.
SMS_RequestBalance is still functional only for the old API.

COUNTRY CODE AREA CODE NUMBER

No leading zeros need to be specified.
Example: +49171123456789
There is also the option of integrating an automatic country code on the Settings page of the Clickatell homepage.
"Convert mobile numbers into international format"
Thus, the +49 for the country code is omitted.

SMS_Send

```
bool SMS_Send (int $InstanceID, string $Number, string $Text)
```

PARAMETERS

InstanceID	ID of the device to be switched
Number	Mobile number of the recipient
Text	Text of the SMS (max. 160 characters)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends from the instance with ID **InstanceID** a SMS with the text to the number.



It is used the sender number that was specified in the properties page.

EXAMPLE

```
SMTP_SendMailEx(12345, "me@example.com", "Alarm!", "The heating system
has broken.!");
```

SMTP

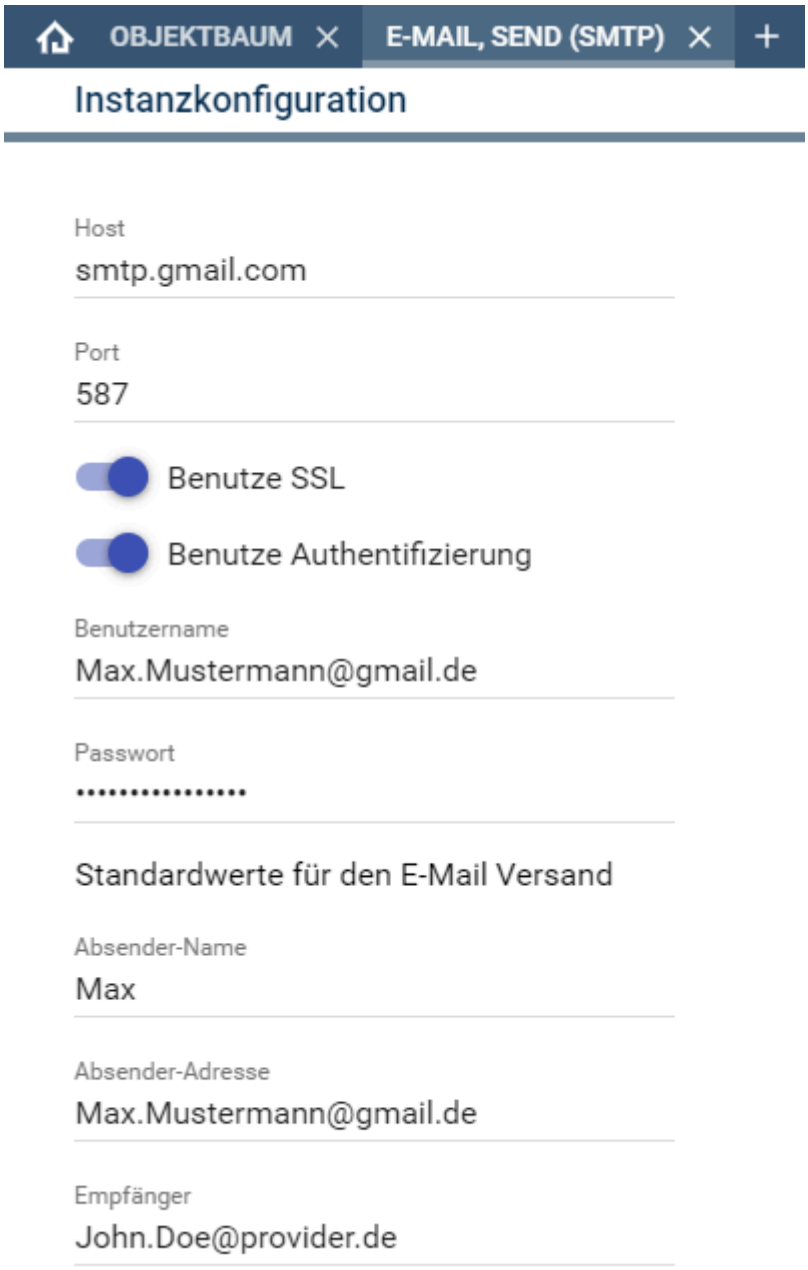
Require: Symcon >= 2.2
SMTP is a module of IP-Symcon that can send e-mails. The module is set up as an instance in IP-Symcon. It can be found under Manufacturer "None" and "Send Email (SMTP)". The individual properties can be found in the table.



In order to send a message to one or more e-mail addresses, the SMTP_SendMailEx function must be used.



The emails cannot be specially formatted. /n /r are functional but any formatting such as bold, italics etc will not work. A PHPMailer, for example, must be used for this instead.



PROPERTY	MEANING
Host	Server name or IP to the mail provider
Port	Port on which the server is working
SSL	Indicates whether SSL should be used
Authentication	Indicates whether a username/password is required
Username	Username (details are available from the mail provider)
Password	Password
Sender-Name	The name visible to the recipient
Sender-Address	The address visible to the recipient. Depending on the provider, the sender address must match ones own address.
Recipient	Only required for the SMTP_SendMail function, which automatically sends to the recipient of this field.

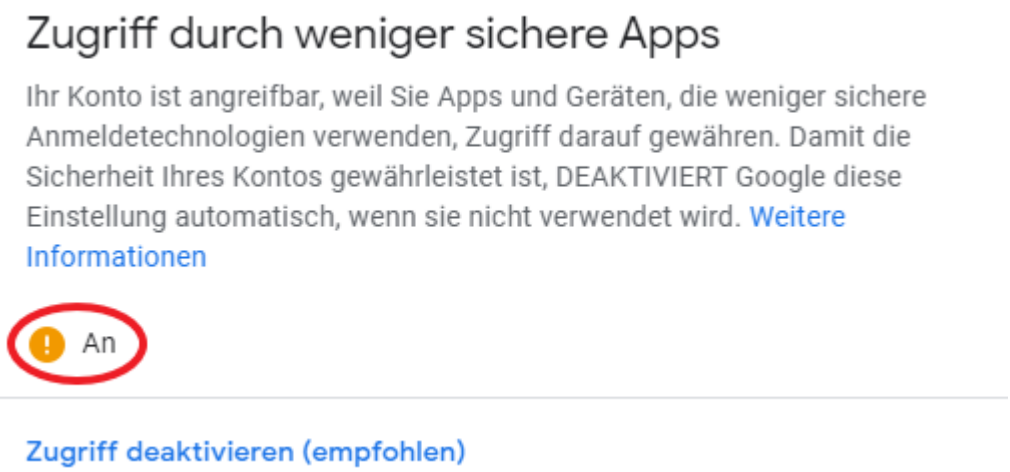
An overview of host, port and authentication for many providers can be viewed here: table

EXAMPLE WITH GMAIL

Gmail does not allow direct access with the username and password without further settings. With these settings, it depends on whether two-factor authentication is activated.

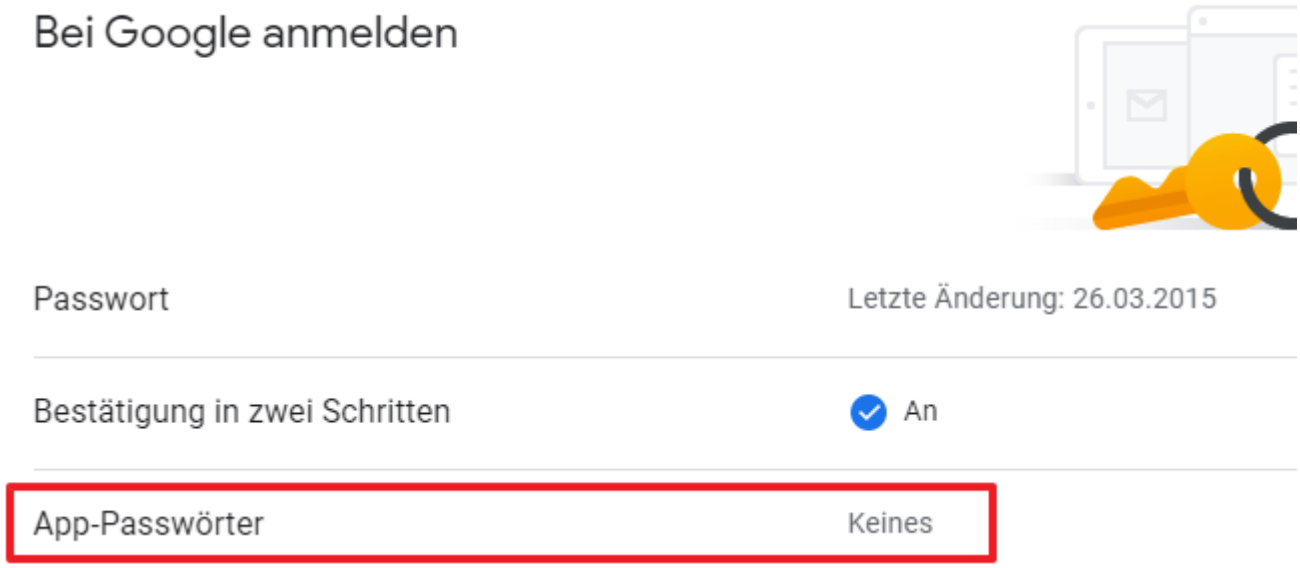
WITHOUT TWO-FACTOR AUTHENTICATION

For use without two-factor authentication, "Access through less secure apps" must be activated in the administration of the Google account under "Security". A direct link would be myaccount.google.com/lesssecureapps .
When access is enabled, it looks like this. Username and password can now be used to log in.

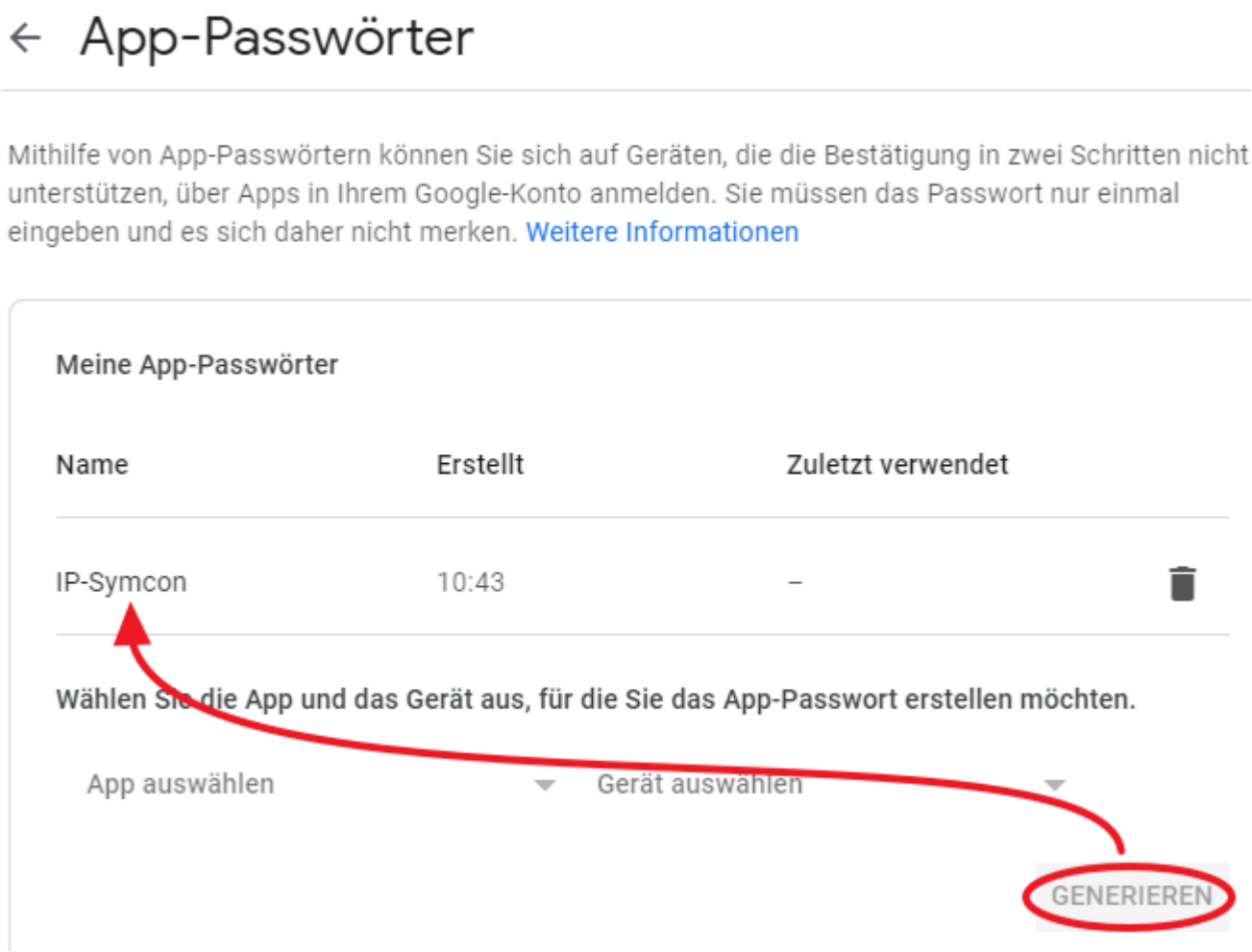


WITH TWO-FACTOR AUTHENTICATION

When using two-factor authentication, an app password must be assigned in the administration of the Google account under "Security" in the "Sign in to Google" section. Click on the line "App passwords".

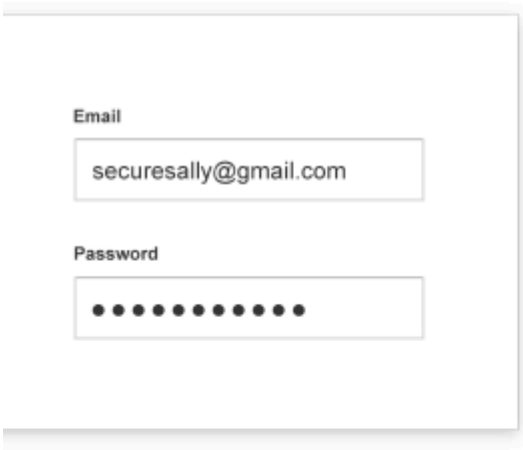


A new password with an individual name can be generated in the overview.



The generated password must then be entered in IP-Symcon.

Generiertes App-Passwort



Ihr App-Passwort für Ihr Gerät

eymx reme ojye pauj

Verwendung

Öffnen Sie in der App oder auf dem Gerät, die bzw. das Sie einrichten möchten, die Einstellungen Ihres Google-Kontos. Ersetzen Sie Ihr Passwort durch das oben angezeigte 16-stellige Passwort.
Wie mit Ihrem normalen Passwort haben Sie auch mit diesem App-Passwort vollen Zugriff auf Ihr Google-Konto. Sie müssen es sich jedoch nicht merken und sollten es weder aufschreiben noch an Dritte weitergeben.

FERTIG

SMTP_SendMail

Require: Symcon >= 2.2

```
bool SMTP_SendMail (int $InstanceID, string $Subject, string $Body)
```

PARAMETERS

InstanceID	ID of the device to be switched
Subject	Subject of mail
Body	Content of the email. Sent as HTML (since 7.0) if <html> and </html> is found, otherwise as text

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends from the instance with ID **InstanceID** an e-mail with the subject and text.



It is used the sender that was specified in the properties page.

EXAMPLE

```
SMTP_SendMail(12345, "Alarm!", "The heating system has broken.!");
```

SMTP_SendMailAttachment

Require: Symcon >= 2.2

```
bool SMTP_SendMailAttachment (int $InstanceID, string $Subject, string
$Body, string $FilePath)
```

PARAMETERS

InstanceID	ID of the SMTP instance
Subject	Email subject
Body	Content of the email. Sent as HTML (since 7.0) if <html> and </html> is found, otherwise as text
FilePath	Path to the file

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends an email with the subject, attachment and text via the instance with the ID **InstanceID**.



The recipient specified in the properties page will be used. If different recipients are to be addressed, the SMTP_SendMailAttachmentEx function should be used.

EXAMPLE

```
SMTP_SendMailAttachment(12345, "Alarm!", "The heater is down!",
"media/webcam.jpg");
```

SMTP_SendMailAttachmentEx

Require: Symcon >= 2.2

```
bool SMTP_SendMailAttachmentEx (int $InstanceID, string $Recipient,
string $Subject, string $Body, string $FilePath)
```

PARAMETERS

InstanceID	ID of the SMTP instance
Recipient	Recipient's e-mail address
Subject	Subject of the email
Body	Content of the email. Sent as HTML (since 7.0) if <html> and </html> is found, otherwise as text
FilePath	Path to the file

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends an email with the subject, attachment and text to recipients via the instance with the ID **InstanceID**. In contrast to the SMTP_SendMail function, the recipient can be specified explicitly in the function call.

EXAMPLE

```
SMTP_SendMailAttachmentEx(12345, "me@example.com", "Alarm!", "The
heater is down!", "media/webcam.jpg");
```

SMTP_SendMailEx

Require: Symcon >= 2.2

```
bool SMTP_SendMailEx (int $InstanceID, string $Receiver, string
$Subject, string $Body)
```

PARAMETERS

InstanceID	ID of the device to be switched
Receiver	E-mail Address of the receiver
Subject	Subject of mail
Body	Content of the email. Sent as HTML (since 7.0) if <html> and </html> is found, otherwise as text

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends from the instance with ID **InstanceID** an e-mail with the subject and text to receiver

EXAMPLE

```
SMTP_SendMailEx(12345, "me@example.com", "Alarm!", "The heating system
has broken.!");
```

SMTP_SendMailMedia

Require: Symcon >= 4.0

```
bool SMTP_SendMailMedia (int $InstanceID, string $Subject, string
$Body, int $MediaID)
```

PARAMETERS

InstanceID	ID of the SMTP instance
Subject	Subject of the email
Body	Content of the email. Sent as HTML (since 7.0) if <html> and </html> is found, otherwise as text
MediaID	Media object ID (image/sound and document only)

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sends an email with the subject, media object and text via the instance with the ID InstanceID.



The recipient specified in the properties page will be used. If different recipients are to be addressed, the SMTP_SendMailMediaEx function is to be used.

EXAMPLE

```
SMTP_SendMailMedia(12345, "Alarm!", "Someone rang the doorbell, but
you're not here!", 23456);
```

SMTP_SendMailMediaEx

Require: Symcon >= 4.0

```
bool SMTP_SendMailMediaEx (int $InstanceID, string $Recipient, string
$Subject, string $Body, int $MediaID)
```

PARAMETERS

InstanceID	ID of the SMTP instance
Recipient	Recipient's e-mail address
Subject	Subject of the email
Body	Content of the email. Sent as HTML (since 7.0) if <html> and </html> is found, otherwise as text
MediaID	Media object ID (image/sound and document only)

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sends an email with the subject, media object and text to recipients via the instance with the ID InstanceID. In contrast to the SMTP_SendMailMedia function, the recipient can be specified explicitly in the function call.

EXAMPLE

```
SMTP_SendMailMediaEx(12345, "me@example.com", "Alarm!", "Someone rang
the doorbell, but you're not here!", 23456);
```

Spotify

Require: Symcon >= 5.3
The Spotify module allows linking to a Spotify Premium account. Playback can be controlled via the account. Thus, it is possible to start a playback or to change or stop the current playback. Options like repeat or random playback can also be set. Favorites are stored within the module, which can then be conveniently retrieved from the visualization.

RANGE OF FUNCTIONS

Linking to a Spotify Premium account
Play or pause playback
Switch to next or previous song
Select or change device for playback
Save and recall favorites
Enable repeat or shuffle playback

REQUIREMENTS

Spotify Premium account

SOFTWARE INSTALLATION

Via the Module Store, install the 'Spotify' module.

SETTING UP INSTANCES IN IP-SYMCON

Under 'Add Instance', the 'Spotify' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

Initially, the module needs to be linked to a Spotify Premium account by clicking on the 'Register' button. The click opens a login dialog from Spotify. After entering the user data, the link must be confirmed. After that, all functions of the module are available.

ADVANCED SETTINGS

NAME	DESCRIPTION
Refresh interval	In this interval the values of the status variables are matched with the current playback of Spotify. This includes the device, play or pause, shuffle and repeat

NAME	DESCRIPTION
Cover: Maximum width	Maximum width of the displayed cover
Cover: Maximum height	Maximum height of the displayed cover

SEARCH

NAME	DESCRIPTION
Search	This field is used to enter a search term for a search
Search by Albums	Specifies whether the next search should include albums
Search for Artists	Specifies whether the next search should include artists
Search by Playlists	Specifies whether the next search should include playlists
Search for Songs	Specifies whether the next search should include songs
Start Search	Starts a search
Search results	This list contains the results of the current search - By activating the field "Favorite" a result can be added to the favorites

The list "Your Playlists" contains the playlists of the linked Spotify account. These can be added to the favorites via the "Favorite" field. The "Favorites" list contains the set favorites. Favorites can be removed by clicking on the trash can icon.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Current Cover	HTMLBox	The cover of the currently played album is displayed here, if available
Current Song	string	The name of the currently played song is displayed here
Current Artist	string	The artist of the currently played song is displayed here
Current Album	string	The album of the currently played song is displayed here

NAME	TYPE	DESCRIPTION
Action	integer	Here you can start or pause a playback. It is also possible to switch to the previous or next song
Device	integer	With this variable a playback device can be selected. If a playback is currently active, it will be switched to the new device
Favorite	integer	This variable represents the favorites configured in the module. By selecting a favorite, it will be played on the currently selected device
Repeat	integer	This variable can be used to activate the repeat function - "Off": No repeat, "Context": The current context, i.e. album, playlist, ... is repeated, "Song": The current song is repeated
Random Play	boolean	Enables or disables random play

PROFILE:

NAME	TYPE
Spotify.Favorites.<InstanceID>	Integer
Spotify.Devices	Integer
Spotify.Repeat	Integer

VISUALIZATION

Via the Visualization or in the mobile apps status variables are displayed and can be switched.

SPO_MakeAPIRequest

Require: Symcon >= 5.3

```
string SPO_MakeAPIRequest (int $InstanceID, string $Method, string
$Url, string $Body)
```

PARAMETERS

InstanceID	ID of the Instance
Method	Type of HTML request
Url	Path, which follows `https://api.spotify.com/v1
Body	Message to be sent to the recipient

RETURNS

The function returns the API return as a string or false on error.

DESCRIPTION

Makes a request to the Spotify web API with the specified parameters. Only the endpoint specific path needs to be specified, the general part, i.e. https://api.spotify.com/v1 is added automatically

EXAMPLE

```
SPO_MakeAPIRequest(12345, 'GET', '/me', '');
```

SPO_NextTrack

Require: Symcon >= 5.3

```
bool SPO_NextTrack (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

If a current playback exists, the next song will be played.

EXAMPLE

```
SPO_NextTrack(12345);
```

SPO_Pause

Require: Symcon >= 5.3

```
bool SPO_Pause (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

If a current playback exists, it will be paused.

EXAMPLE

```
SPO_Pause(12345);
```

SPO_Play

Require: Symcon >= 5.3

```
bool SPO_Play (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

If a current, possibly paused, playback exists, it will be continued. Otherwise, the currently selected favorite is played on the currently selected device.

EXAMPLE

```
SPO_Play(12345);
```

SPO_PlayURI

Require: Symcon >= 5.3

```
bool SPO_PlayURI (int $InstanceID, string $URI)
```

PARAMETERS

InstanceID	ID of the Instance
URI	URI of the Spotify resource

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Plays the Spotify resource with the URI URI on the currently selected device.

EXAMPLE

```
SPO_PlayURI(12345, 'spotify:artist:1Lw1vZvhgNZk7hVSvdY4OA');
```

SPO_PreviousTrack

Require: Symcon >= 5.3

```
bool SPO_PreviousTrack (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

If a current playback exists, the previous song will be played.

EXAMPLE

```
SPO_PreviousTrack(12345);
```

SPO_ResetToken

Require: Symcon >= 5.3

```
bool SPO_ResetToken (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Resets the OAuth-token and thus the link to the Spotify account.

EXAMPLE

```
SPO_ResetToken(12345);
```

SPO_SetRepeat

Require: Symcon >= 5.3

```
bool SPO_SetRepeat (int $InstanceID, string $Repeat)
```

PARAMETERS

InstanceID	ID of the Instance

Repeat

WERT	BEDEUTUNG
0	Wiederholung deaktiviert
1	Wiederholung des Kontext
2	Wiederholung des Songs

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

Sets the repeat to the value **Repeat**.

EXAMPLE

```
SPO_SetRepeat(12345, 2);
```

SPO_SetShuffle

Require: Symcon >= 5.3

```
bool SPO_SetShuffle (int $InstanceID, bool $Shuffle)
```

PARAMETERS

InstanceID	ID of the Instance
Shuffle	De-/Aktiviert die zufällige Wiedergabe

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

If **Shuffle** is true, random playback is enabled, otherwise disabled.

EXAMPLE

```
SPO_SetShuffle(12345, true);
```

Fault Manager

Require: Symcon >= 6.0
The module is used to display active variables in the Visualization and, depending on the setting, to hide them after they have been entered or deactivated. If 'Auto hide' but not 'Confirmation' has been activated, a link will be visible as soon as the monitored variable is active. This link disappears as soon as the monitored variable is deactivated. If 'Confirmation' and 'Auto Hide' have been activated, a variable will be created as soon as the monitored variable is active. This disappears as soon as the value of the created variable has been changed to 1 or 'In process' and the monitored variable is inactive. If 'Confirmation' but not 'Auto Hide' has been activated, a variable will be created as soon as the monitored variable is active. This disappears as soon as the value of the created variable has been changed to over 2 or 'All right'.

FUNCTION SCOPE

Display of active variables
Possibility to acknowledge them depending on the selection

SOFTWARE INSTALLATION

Install the 'Fault Manager' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Fault Manager' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Variable	Variable to be monitored.
Confirmation	Checkbox to confirm or not the message.
Hide automatically	Checkbox, so that the message is automatically hidden again when the state is deactivated.
Message handling	Selection how to handle the variables when they are active.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Link: Variable Name	Link	Link which is visible or invisible. Will be created if 'Message disappears automatically' is selected.
Variable Names - Status	Integer	Variable which is created when 'Acknowledge message' or 'Acknowledge message, disappears when error is cleared' is selected.

PROFILE:

NAME	DESCRIPTION
STA.Confirm	Profile for created variables, with 3 levels
STA.ConfirmHide	Profile for created variables, with 2 levels

VISUALIZATION

Via the Visualization the created variables are displayed and can be changed if they need to be confirmed.

SymconReport

Creates a report as CSV or PDF depending on the modules

MailReport		Creates a CSV report of archive values and sends it by e-mail(SMTP)
	MR_SendInfo	Sends an info mail about the last completed time interval
	MR_SetActive	Enables or disables the module
PDFReport (Energy)		Creates a report of archive values of two variables of a time span as PDF
	RAC_GenerateEnergyReport	Generates a PDF with the values specified in the module with the InstanceID
PDFReport (Multi)		Creates a report of archive values of several variables of a time span as PDF
	RAC_GenerateReport	Generates a PDF with the values specified in the module with the InstanceID
PDFReport (Multi Energy)		
PDFReport (Single)		Creates a report of archive values of a variable with min/max/avg of a time span as PDF
	RAC_GenerateReport	Generates a PDF with the values specified in the module with the InstanceID

MailReport

Require: Symcon >= 5.5
The module regularly sends the logged aggregated data of a variable via e-mail.

FUNCTION SCOPE

Send e-mails with aggregated data of a variable
E-mails are sent after each completion of the selected time interval (Daily, Weekly or Monthly)
E-mails contain aggregated data, which would also be displayed in a corresponding graph (Hourly for daily time interval, Daily for weekly or monthly time interval)
The email contains a CSV file, which is formatted analogously to the CSV export of graphs.

PREREQUISITES

Set up SMTP instance for sending e-mails

SOFTWARE INSTALLATION

The module can be found via the Module Store under the category "Information"=>"Preparation" or directly by searching for "Report Module". By pressing the button "Install" the module is made available to IP-Symcon.

SETTING UP THE INSTANCES IN IP-SYMCON

Via "Add Instance" the 'Report(Mail)' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Email SMTP	Selection of the SMTP instance through which the emails will be sent
Aggregated Variable	Selection of the variable whose aggregated data should be sent
Decimal separator	Selection of decimal separator comma or point.
Interval of messages	Select the time interval in which the messages should be sent
Send data	Click to send an info mail about the last completed time interval

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLEN

NAME	TYPE	DESCRIPTION
Mail Report active	Variable	Activates or deactivates the module

VISUALIZATION

The Visualization can be used to activate or deactivate the module.

MR_SendInfo

Require: Symcon >= 5.5

```
bool MR_SendInfo (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Sends an info mail about the last completed time interval

EXAMPLE

```
MR_SendInfo(12345);
```

MR_SetActive

Require: Symcon >= 5.5

```
bool MR_SetActive (int $InstanceID, bool $Active)
```

PARAMETERS

InstanceID	ID of the Instance
Active	De-/Activate the module

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Enables or disables the module depending on Active

EXAMPLE

```
// Activates the module MR_SetActive(12345, true);
```

PDFReport (Energy)

Require: Symcon >= 5.5
This module provides the function of displaying the archive values of two variables as graphs, and providing selected information as PDF for the past month.

SCOPE OF FUNCTIONS

Allows to create and download created PDF.
Setting via instance configuration
Logo selection for header is possible

SOFTWARE INSTALLATION

Via the Module Store the module can be found under the category "Information"=>"Preparation" or directly via the search for "Report Module". By pressing the button "Install" the module is made available to IP-Symcon.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Report (PDF, Energy)' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

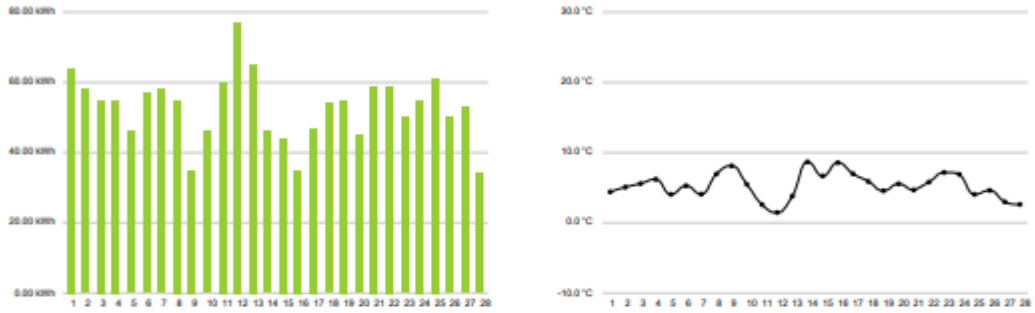
NAME	DESCRIPTION
Logo	(optional) Selectable graphic as PNG
Consumption type	Free text which is below the date
Decimal separator	Selection of decimal separator comma or point.
Consumption counter	Variable logged as counter
Temperature	(optional) Logged variable
prediction	(optional) Variable which can predict consumption based on behavior, for example from the module consumption behavior
CO2 type	(optional) CO2 equivalent for consumption type

EXAMPLE DOCUMENT

Jaetzel

FEBRUAR 2022
IHR GASVERBRAUCH

VERBRAUCH



Im Februar haben Sie 1478.00 kWh verbraucht.

VERHALTEN

Februar

Die Durchschnittstemperatur betrug: 4.6 °C
Erwarteter Verbrauch aufgrund Ihres Verhaltens: 1537.60 kWh
Tatsächlicher Verbrauch: 1478.00 kWh

Sie haben in diesen Zeitraum Ihren Verbrauch um 3.88% gesenkt.

CO2 Emmisionen

353.242 kg

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

The PDF is generated as a "Report (PDF)".

NAME	TYPE	DESCRIPTION
Report (PDF)	Media	Generated PDF which can be downloaded from Visualization or sent by e-mail.

VISUALIZATION

The generated PDF can be downloaded via the Visualization.

RAC_GenerateEnergyReport

Require: Symcon >= 5.5

```
bool RAC_GenerateEnergyReport (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

Generates a PDF with the values specified in the module with the InstanceID **InstanceID**. The PDF is then available as a media file.
If a file already exists, it will be updated.

EXAMPLE

```
RAC_GenerateEnergyReport(12345);
```

PDFReport (Multi)

Require: Symcon >= 5.5
This module provides the function to summarize archive values as a report in a PDF.

FUNCTION SCOPE

Allows to create and download created PDF.
Setting via instance configuration
Logo selection for header is possible
Adjustable number of records, aggregation strength, min and max values of selected variables

SOFTWARE INSTALLATION

The module can be found via the Module Store under the category "Information"=>"Preparation" or directly by searching for "Report Module". By pressing the button "Install" the module is made available to IP-Symcon.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Report (PDF, Multi)' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Logo	Selectable graphic as PNG
Company	Company name
Title	PDF Title
Footer	Footer line
Data Source	Multiple variables from which the records per column are created
Decimal separator	Selection of decimal separator comma or point.
Aggregation	Defines aggregation level of the listed records (hour - year)
Number	Number of records listed
Skip record	If this option is enabled, the most recent incomplete record will be discarded

EXAMPLE DOCUMENT



Symcon GmbH
29.10.2021

Titel

Datum	Variable 1	Variable 2	Variable 3
29.10.2021 23:26	100 %	100 %	100 %

Fußzeile

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

The PDF is generated as "Report (PDF)".

NAME	TYPE	DESCRIPTION
Report (PDF)	Media	Generated PDF which can be downloaded via Visualization or sent by e-mail.

VISUALIZATION

The generated PDF can be downloaded via the Visualization.

RAC_GenerateReport

Require: Symcon >= 5.5

```
bool RAC_GenerateReport (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Generates a PDF with the values specified in the module with the InstanceID **InstanceID**. The PDF is then available as a media file.
If a file already exists, it will be updated.

EXAMPLE

```
RAC_GenerateReport(12345);
```

PDFReport (Multi Energy)

This module offers the function of displaying the total consumption of the selected variables as well as the individual consumption in the time period.

SCOPE OF FUNCTIONS

Enables the creation and download of created PDFs.
Setting via instance configuration

SOFTWARE INSTALLATION

The module can be found via the Module Store under the category "Information"=>"Preparation" or directly by searching for "Report module". The "Install" button makes the module available to IP-Symcon.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add instance", the 'Report (PDF, Energy)' module can be found using the quick filter.
Further information on adding instances can be found in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Aggregation Level	Selection of the level of detail of the data used
Decimal Separator	Selection of how the decimal separator should be displayed
Energy Counters	List of energy counters to be used

EXAMPLE DOCUMENT

VERBRAUCH

Von 01.03.2024 bis 23.10.2024

Gesamtverbrauch: 250407.7 kwh

Zähler	Verbrauch	Prozentsatz
Stromzähler	63132.9 kWh	25.21 %
Stromzähler (Haus)	187274.8 kWh	74.79 %

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

The PDF is generated as a "Report (PDF)".

NAME	TYPE	DESCRIPTION
Start	Integer	Specifies the start date from which the data should be used
End	Integer	Specifies the end date until when the data should be used
Generate	Script	Executable to update the PDF
Report (PDF)	Media	Created PDF, which can be downloaded via the visualization or sent by e-mail.

VISUALIZATION

The start and end date can be entered via the visualization and the PDF can be generated and downloaded.

PDFReport (Single)

Require: Symcon >= 5.5
This module provides the function to summarize archive values as a report in a PDF. A single target variable is displayed with its min/max/avg values.

SCOPE OF FUNCTIONS

Allows to create and download created PDF.
Setting via instance configuration
Logo selection for header is possible
Adjustable number of records, aggregation strength, min and max values of selected variables

SOFTWARE INSTALLATION

The module can be found via the Module Store under the category "Information"=>"Preparation" or directly by searching for "Report Module". By pressing the button "Install" the module is made available to IP-Symcon.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Report (PDF, Single)' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Logo	Selectable graphic as PNG
Company	Company name
Title	PDF Title
Footer	Footer line
Data Source	Variable from which the records are created
Aggregation	Defines aggregation level of the listed records (hour - year)
Decimal separator	Selection of decimal separator comma or point.
Number	Number of records listed
Skip record	If this option is enabled, the most recent incomplete record will be discarded
Tolerance (Min)	Accepted minimum value of aggregated records. Values outside the tolerance will be ignored.
Tolerance (Max)	Accepted maximum value of the aggregated records. Values outside the tolerance will be ignored.



Firma
16.08.2019

Titel

Datum	Min	Max	Mittel	Toleranz

Fußzeile

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

The PDF is generated as "Report (PDF)".

NAME	TYPE	DESCRIPTION
Report (PDF)	Media	Generated PDF which can be downloaded via Visualization or sent by e-mail.

VISUALIZATION

The generated PDF can be downloaded via the Visualization.

RAC_GenerateReport

Require: Symcon >= 5.5

```
bool RAC_GenerateReport (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Generates a PDF with the values specified in the module with the InstanceID **InstanceID**. The PDF is then available as a media file.
If a file already exists, it will be updated.

EXAMPLE

```
RAC_GenerateReport(12345);
```

Phone Announcement

Require: Symcon >= 5.5
The Telephone Announcement Module allows the convenient linking of a VoIP instance and a text-to-speech instance (Polly) to call a phone number and output a text when the call is answered. In addition, the module can respond to DTMF tones and output other texts.

SCOPE OF FUNCTIONS

Calling a phone number and outputting a text
Evaluating DTMF tones
Output of further texts on an existing connection

PREREQUISITES

Installed VoIP instance
Installed text-to-speech instance (AWS Polly)

SOFTWARE INSTALLATION

Using the Module Store, install the 'Phone Announcement module.

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'Phone Announcement' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
VoIP instance	The VoIP instance that manages the calls
Text-to-Speech instance (Polly)	The text-to-speech instance that converts the texts into audio data
Duration until disconnected	The maximum time to wait for a call to be answered

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Phone Number	String	The phone number that will be called when "Start Call" is pressed
Text	String	The text that will be output initially when "Start call" is pressed. If this variable is switched while a connection is established, the new text will be output directly
DTMF Tone	String	If a DTMF tone is used during the call, it will be stored in this variable
Start Call	Script	Starts a call with the currently set phone number and the current text

TA_StartCall

Require: Symcon >= 5.5

```
void TA_StartCall (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

The function does not return any value.

DESCRIPTION

Starts a call based on the current setting.

EXAMPLE

```
TA_StartCall(12345);
```

TA_StartCallEx

Require: Symcon >= 5.5

```
void TA_StartCallEx (int $InstanzID, string $Telefonnummer, string
$Text)
```

PARAMETERS

InstanzID	ID of the Instance
Telefonnummer	Phone number to call
Text	Text to be output

RETURNS

The function does not return any value.

DESCRIPTION

Starts a call to the phone number **PhoneNumber** and outputs the text **text**.

EXAMPLE

```
TA_StartCallEx(12345, "+4945130500511", "I love IP-Symcon!");
```

Phone Chain

Require: Symcon >= 5.5
The 'Telefonkette' module allows to call a list of telephone numbers one after another.

FUNCTION SCOPE

A list of phone numbers can be called in descending order
If accepted, a TTS file can be played
Further calls are terminated when a called party confirms by a DTMF character
DTMF character freely selectable
Duration until a call is terminated freely selectable
Boolean variable as trigger

REQUIREMENTS

Custom VoIP processing script for audio announcement or a TTSAWSPOLLY instance established

SOFTWARE INSTALLATION

Via the Module Store install the 'Phone Chain' module

SETTING UP THE INSTANCES IN IP-SYMCON

Under 'Add Instance', the 'Phone Chain' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Trigger	Boolean variable which is used as trigger
VoIP instance	VoIP instance to be used for calls
TTS instance	TTSAWSPOLLY Instance to be used for the announcement
TTS Type	**Static**, static text for an announcement or dynamic announcement via a string variable
Phone numbers can be switched in the visualization	Option to switch the phone numbers on and off in the visualization
Phone Numbers	List and sequence of numbers with description which should be called
Number of simultaneous calls	Number of calls that will be made in parallel

NAME	DESCRIPTION
Call duration	Time in seconds to wait for call to be answered
DTMF confirmation key	key with which a call is confirmed

Note on the 'Phone numbers switchable in visualization' option: This ensures that the phone numbers in the list are created as variables. The telephone numbers are only called if the corresponding variable has the value 'True' and is activated in the visualization.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUS VARIABLES

NAME	TYPE	DESCRIPTION
Call confirmed by	String	Contains the number that confirmed the call
Status	Integer	Shows at which point the chain is currently
Reset status	Script	Resets the status

TelegramBot

Require: Symcon >= 6.0
Allows sending messages to specified persons and performing actions within IP-Symcon.

FUNCTION SCOPE

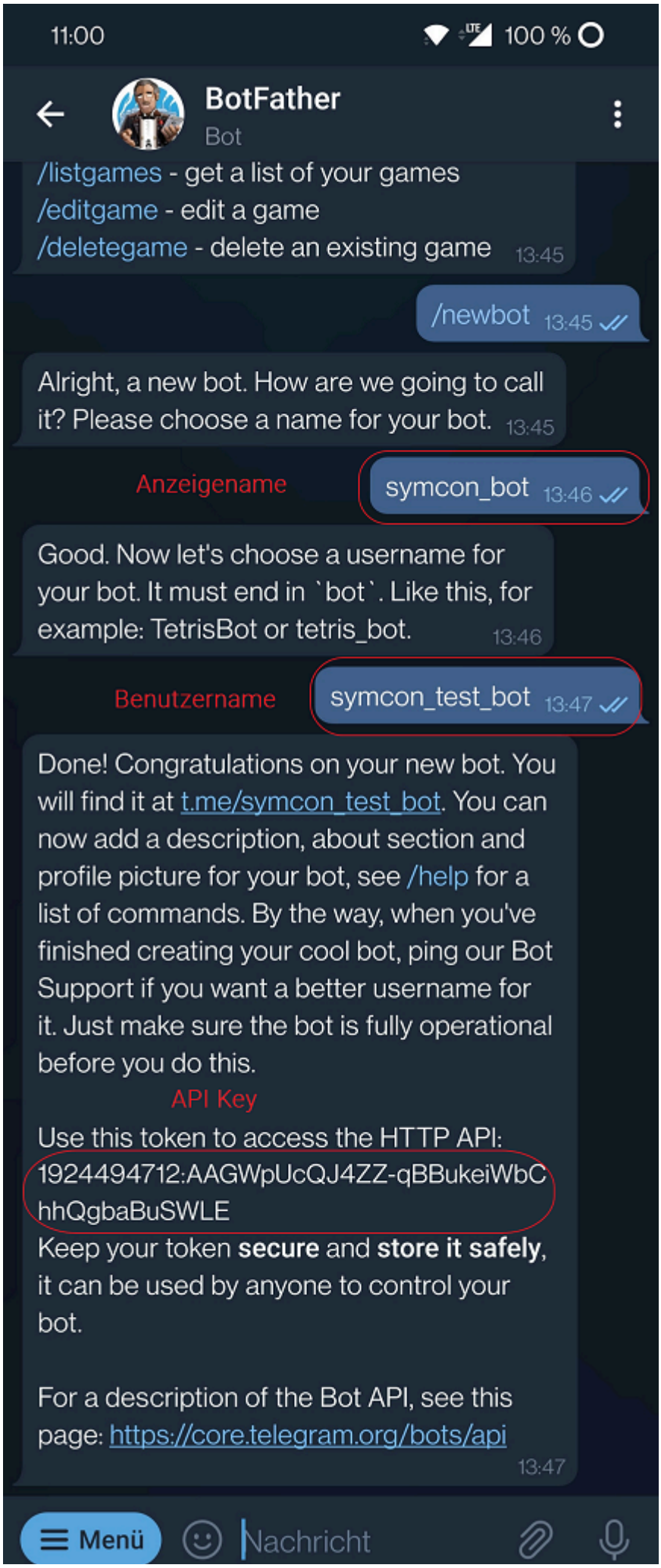
Send messages (to all or individual persons)
Send images (from media) (to all or individual people)
Respond to chat commands with actions

REQUIREMENTS

A Telegram Bot

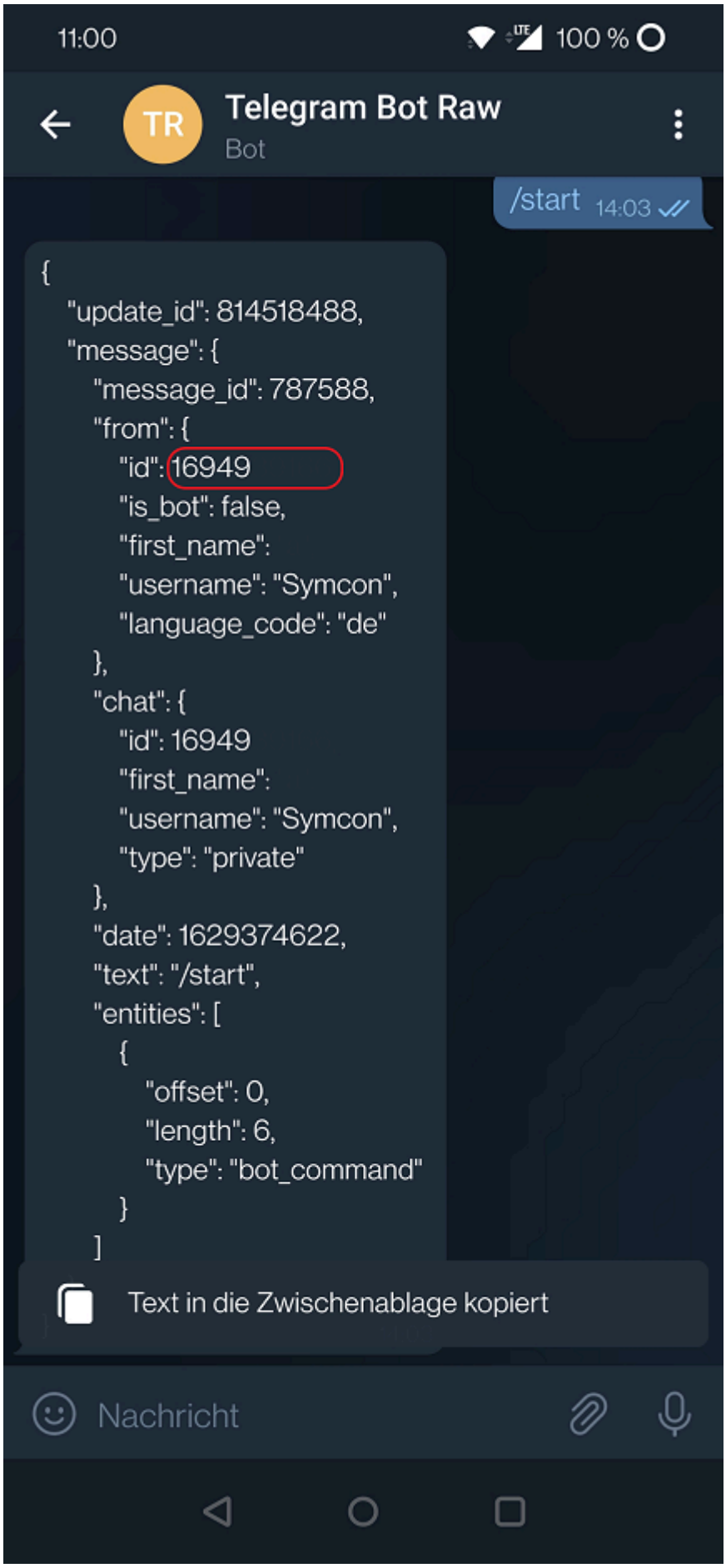
CREATE TELEGRAM BOT

In the Telegram client, search for BotFather
Use */newbot* to create a new bot
Under the username the bot can be found and should be populated with for example random numbers


11:00
100 %
BotFather
Bot
/listgames - get a list of your games
/editgame - edit a game
/deletegame - delete an existing game 13:45
/newbot 13:45
Alright, a new bot. How are we going to call it? Please choose a name for your bot. 13:45
Anzeigename
symcon_bot 13:46
Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 13:46
Benutzername
symcon_test_bot 13:47
Done! Congratulations on your new bot. You will find it at t.me/symcon_test_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.
API Key
Use this token to access the HTTP API:
1924494712:AAGWpUcQJ4ZZ-qBBukeiWbChhQgbaBuSWLE
Keep your token secure and store it safely, it can be used by anyone to control your bot.
For a description of the Bot API, see this page: https://core.telegram.org/bots/api
13:47
Menü
Nachricht



In the Telegram client search for RawDataBot
When the bot is contacted, it returns some data, from which you can get your own userID
This process must be repeated for each user that should be reached by the bot



Write the bot with */start* to start it

SOFTWARE INSTALLATION

Install the 'TelegramBot' module via the Module Store.

INSTANCE SETUP IN IP-SYMCON

Under 'Add Instance', the 'TelegramBot' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATIONPAGE:

NAME	DESCRIPTION
Bot API Key	The API key of the created bot
Bot Username	The username of the created bot
Users	A list of users that can be contacted by the bot
Actions	A list of commands and their actions

ACTIONS

SEND-IMAGE

This action can be used to send an image to the selected recipient

PARAMETER:

Image: the media object to be sent
Recipient: A specific person or all persons defined in the user list

SENDMESSAGE

With this action a message can be sent to the selected recipient

PARAMETER:

Text: The content of the message
Recipient: A specific person or all persons defined in the user list
More information about using actions

Text to Speech

The TTS module can convert text to speech. The system-internal speech software or a post-installed one is used for this.



The TTS module can currently only be used in the Windows installation. Alternatively, the "Text to Speech (AWS Polly)" module from the Module Store can be used. This works on all operating systems.

SETUP IN IP-SYMCON

The speech engine and the audio output device can be set in the configuration tab.
In order to make a voice output, the PHP function TTS_Speak or TTS_GenerateFile must be called from a script.



The TTS_Speak function is deprecated and no longer supported on Windows Vista or higher. TTS_GenerateFile works in all versions.

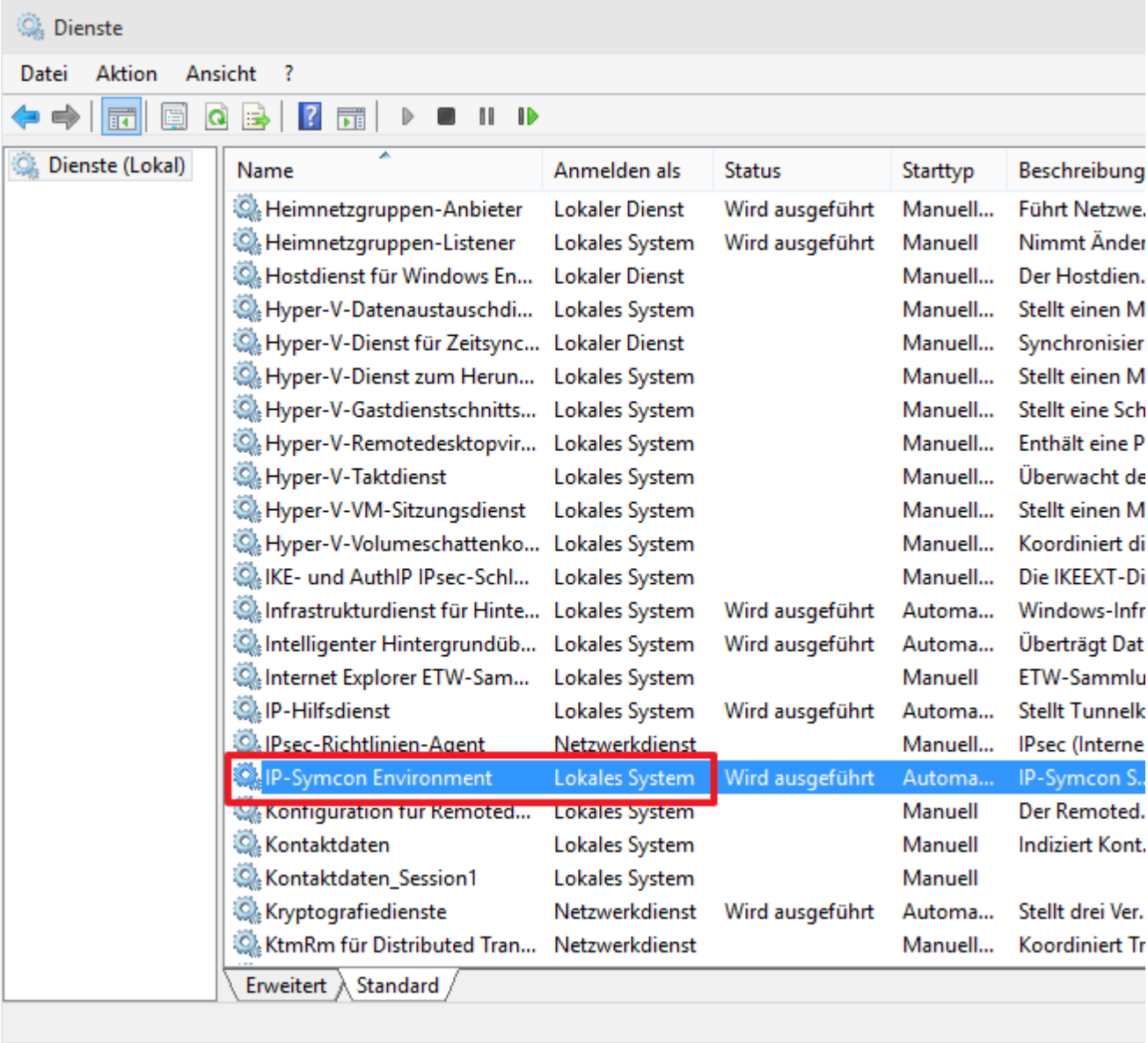
When using TTS_GenerateFile, the use of MediaPlayer is necessary.
An example code can be found under TTS_GenerateFile.

INSTALLATION

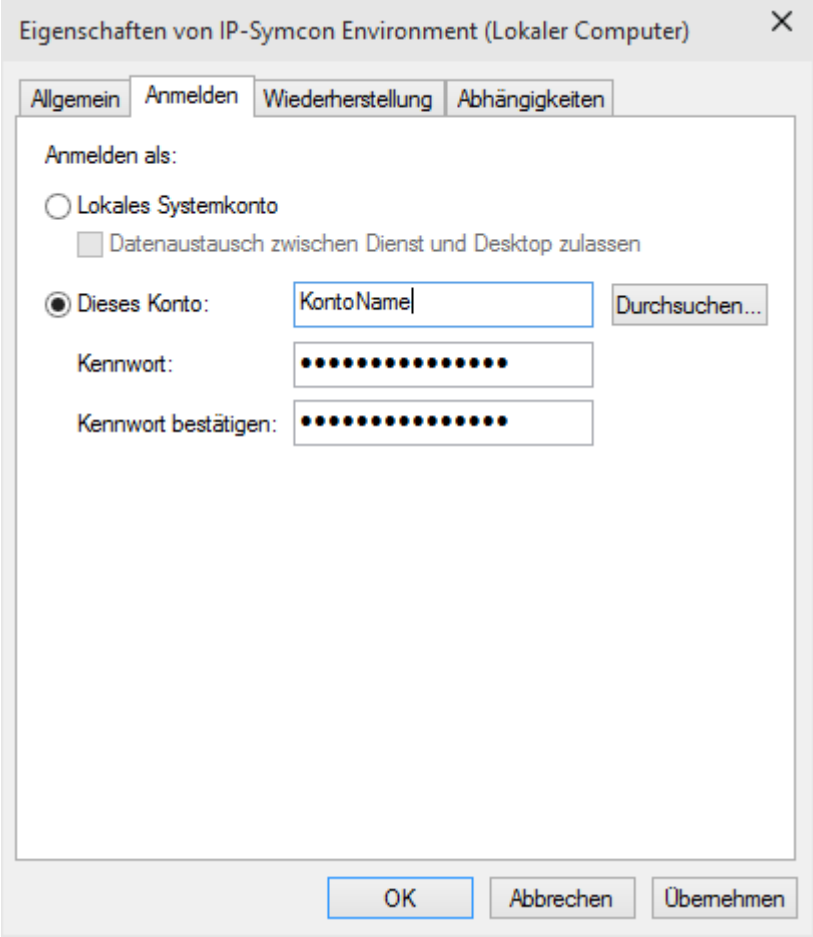
In addition to possible standard voices, other German and international versions can be found here .

EXCEPTION WINDOWS VISTA/7

In order to be able to use the speech output under Windows Vista/7, the IP-Symcon service must be run as a local user. It should be noted that this means that the IPS_ExecuteEx function is no longer available.
Call up the service management via Start > Run
%SystemRoot%\system32\services.msc /s



Double click on the entry IP-Symcon Environment. The following dialog should open and the password for the current user must be entered in the "Login" tab.



After confirming with Apply/OK, the IP-Symcon service must be restarted. The easiest way to stop and start this is via the tray.

TTS_GenerateFile

```
bool TTS_GenerateFile (int $InstanceID, string $Text, string $Filename,
int $Format)
```

PARAMETERS

InstanceID	ID of the device to be switched
Text	Text to be said
Filename	Filename/-path in the system

Format

FORMAT	VALUE
Default	0
8kHz, 8Bit, Mono	4
8kHz, 8Bit, Stereo	5
8kHz, 16Bit, Mono	6
8kHz, 16Bit, Stereo	7
11kHz, 8Bit, Mono	8
11kHz, 8Bit, Stereo	9
11kHz, 16Bit, Mono	10
11kHz, 16Bit, Stereo	11
12kHz, 8Bit, Mono	12
12kHz, 8Bit, Stereo	13
12kHz, 16Bit, Mono	14
12kHz, 16Bit, Stereo	15
16kHz, 8Bit, Mono	16
16kHz, 8Bit, Stereo	17
16kHz, 16Bit, Mono	18
16kHz, 16Bit, Stereo	19
22kHz, 8Bit, Mono	20
22kHz, 8Bit, Stereo	21
22kHz, 16Bit, Mono	22
22kHz, 16Bit, Stereo	23
24kHz, 8Bit, Mono	24
24kHz, 8Bit, Stereo	25
24kHz, 16Bit, Mono	26
24kHz, 16Bit, Stereo	27
32kHz, 8Bit, Mono	28
32kHz, 8Bit, Stereo	29
32kHz, 8Bit, Stereo	30
32kHz, 16Bit, Stereo	31
44kHz, 8Bit, Mono	32
44kHz, 8Bit, Stereo	33
44kHz, 16Bit, Mono	34
44kHz, 16Bit, Stereo	35
48kHz, 8Bit, Mono	36
48kHz, 8Bit, Stereo	37
48kHz, 16Bit, Mono	38
48kHz, 16Bit, Stereo	39

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Generated on the instance with ID **InstanceID** a sound file in WAV format in the file **Filename**. The format parameter determines the quality of the wav file.
Warning: % was not found in [...] on line …
If the error message appears, it indicates an access problem. Likely the file to be written is locked by another process.
Furthermore, it is not possible to write on **NetworkPath**.

EXAMPLE

```
TTS_GenerateFile(44007, "Hello World", "C:/HelloWorld.WAV", 19);
```

TTS_Speak

```
bool TTS_Speak (int $InstanceID, string $Text, bool $Waiting)
```

PARAMETERS

InstanceID	ID of the device to be switched
Text	Text to be said
Waiting	**TRUE** for On, **FALSE** for Off

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Talks about the instance with ID **InstanceID** in the **Text** specified text. The **Waiting** parameter specifies whether to wait until the speech is over.
This function does not work under Windows **Vista/2003** or newer. This is because services such as IP-Symcon, for security reasons, can not access interactive components in the system. The Text To Speech output is such an interactive component. The problem manifests itself with the error message: **OLE error 80045063**
extract MSDN : SPERR_NOT_ACTIVE_SESSION.
“Neither audio output nor input is supported for non-active console sessions.”
As *Alternative* should the function TTS_GenerateFile in connection with the media player be used.

EXAMPLE

```
TTS_Speak(12345, "Hallo Welt!", true); //Wait until it was spoken to
the end
```

TTSAWSPolly

Require: Symcon >= 5.1
The Text to Speech (AWS Polly) module is used to generate sound data/files, which can be used e.g. for audio notifications or VoIP announcements.

FUNCTION SCOPE

Create sound data that can be stored in a media file for output via audio notifications, for example
Create sound files that can be used for an output in the VoIP module

REQUIREMENTS

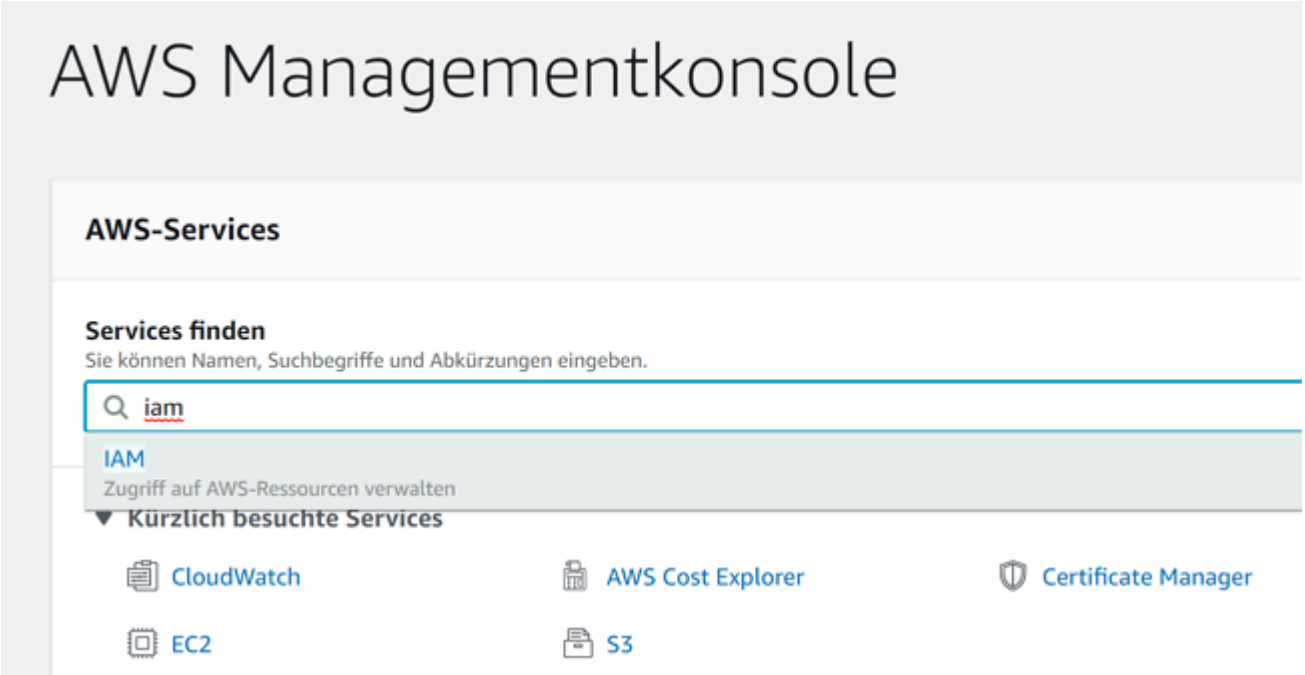
Account with Amazon Web Services
User with appropriate Access Key/Secret Key and access rights to Polly (e.g. AmazonPollyFullAccess)

SOFTWARE INSTALLATION

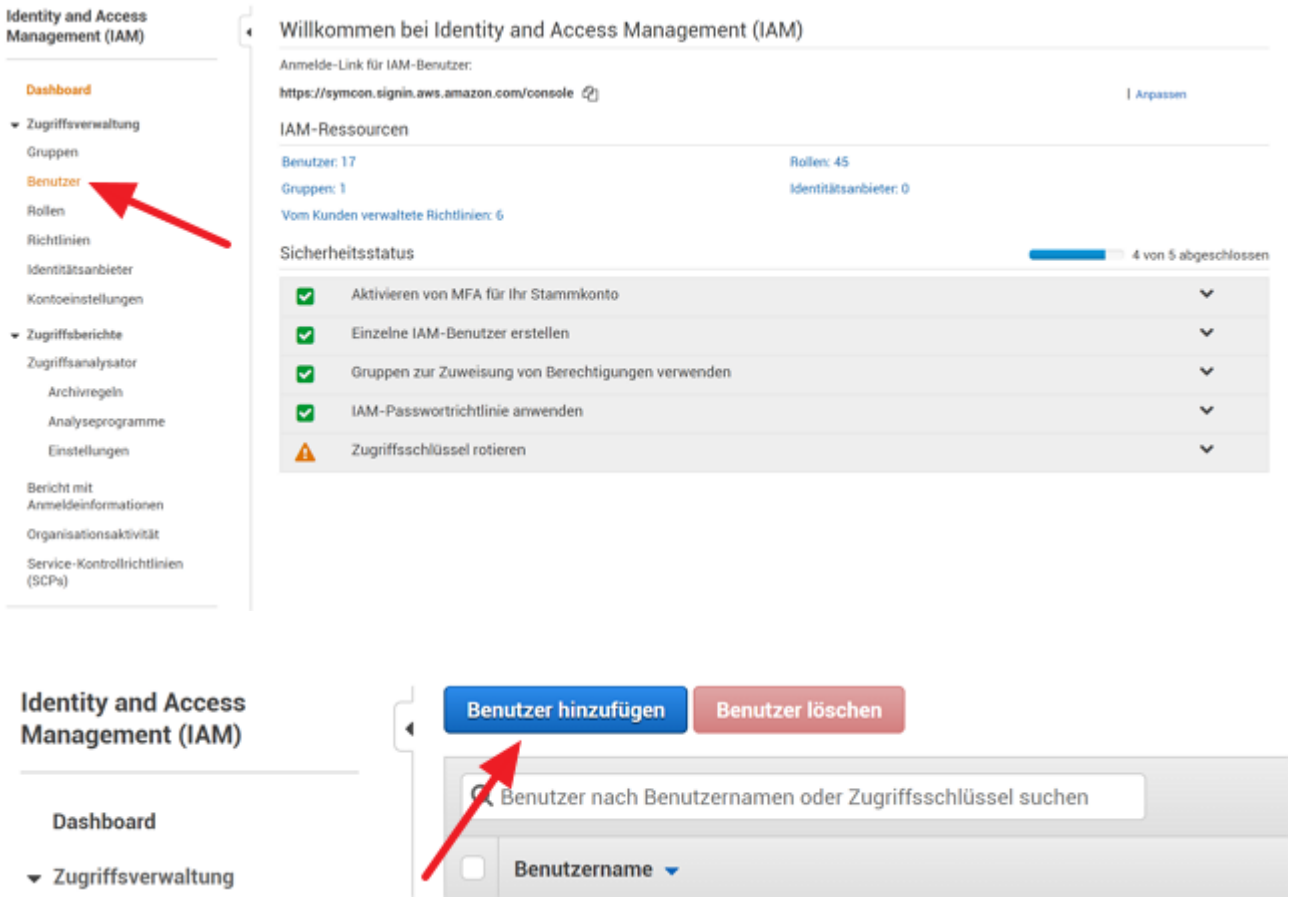
The "Text to Speech (AWS Polly)" module can be installed directly from the Module Store.

USER SETUP IN AWS IAM

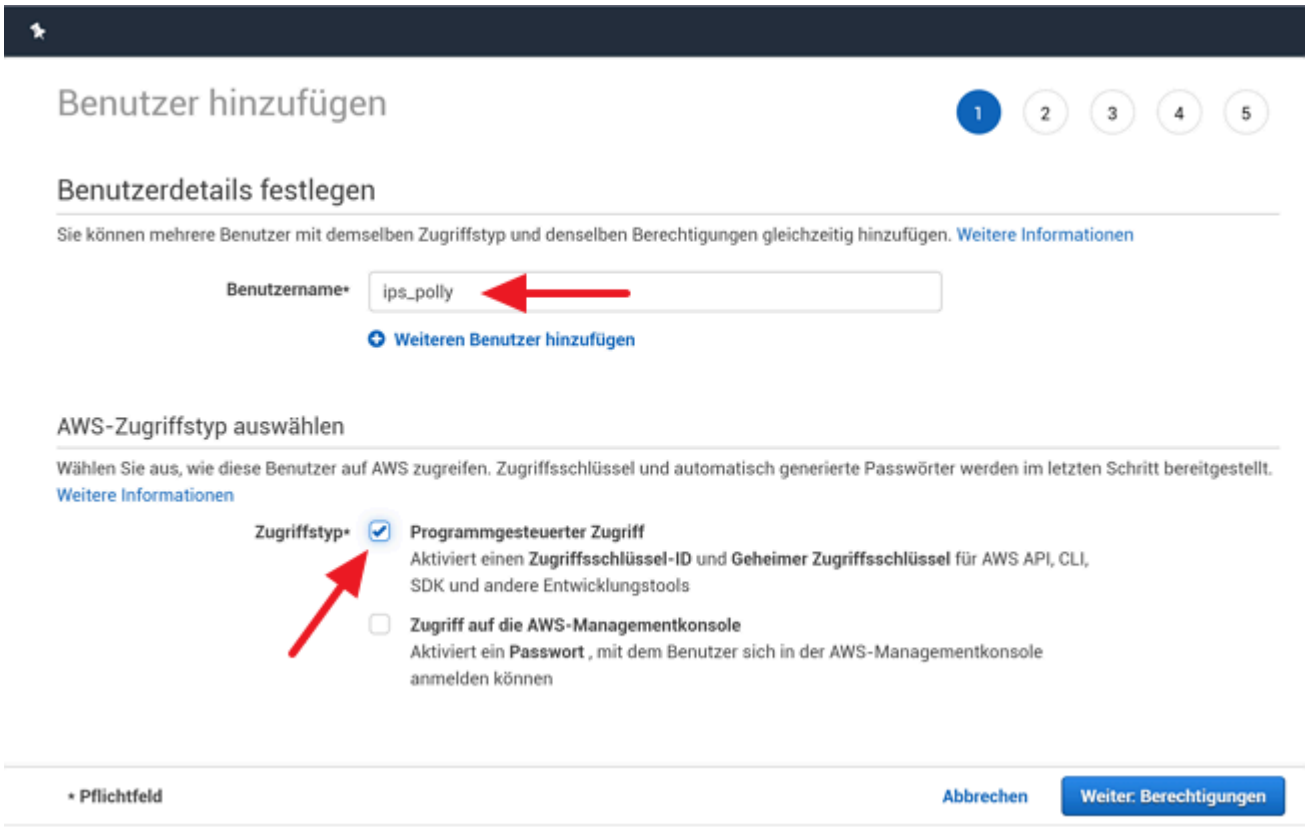
On the Amazon AWS home page, you need to search for the IAM in AWS Services.



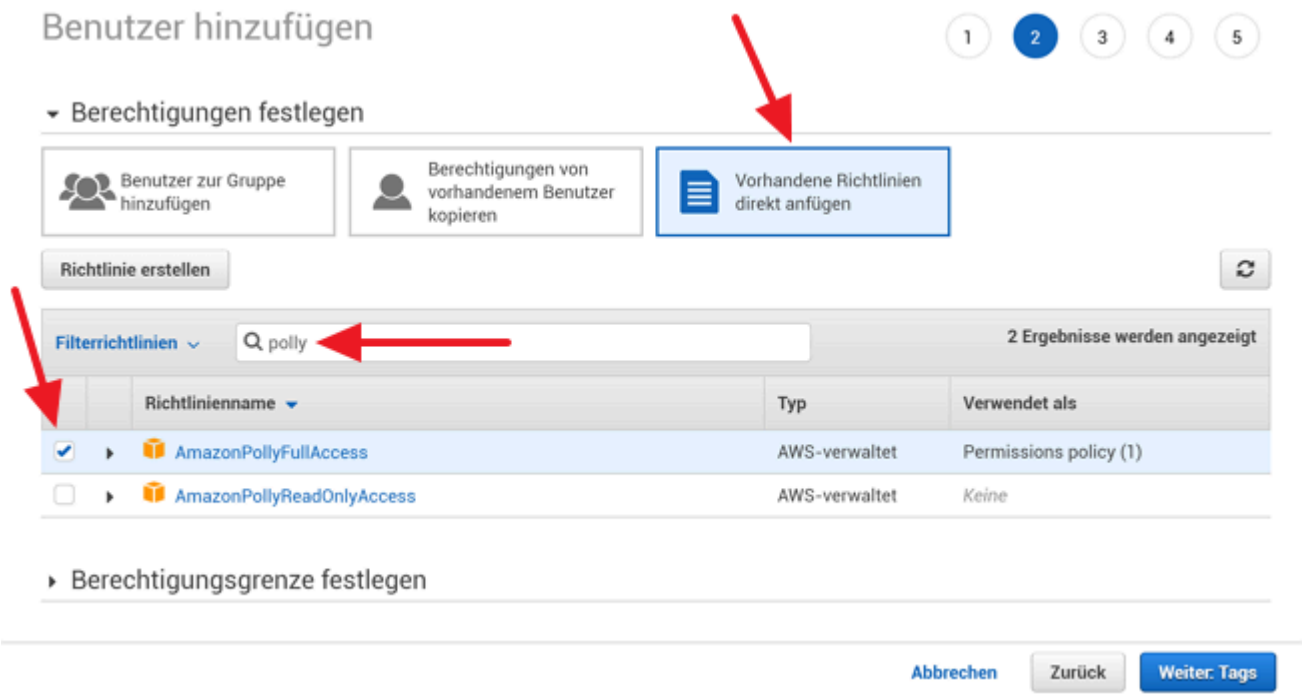
Within the IAM a new user has to be added



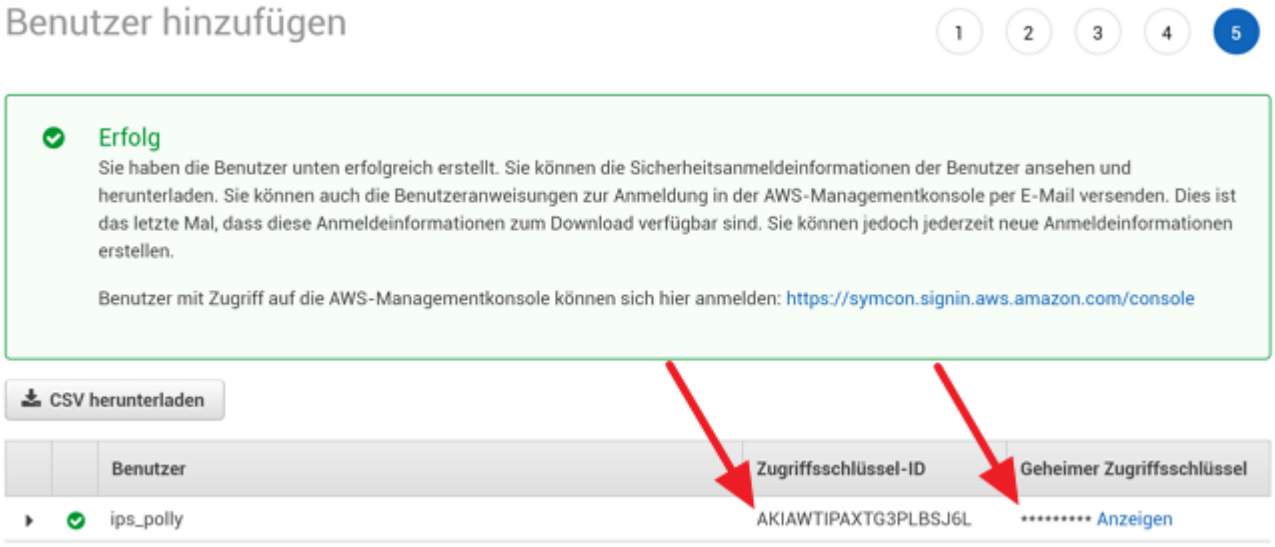
Afterwards, a speaking username should be assigned and "Program controlled access" should be activated.



As a policy "AmazonPollyFullAccess" must be activated.



Skip the next dialog with "Next: Check".
Create the user by clicking on "Create user".
After creating the user, it will be assigned an access key ID and a secret access key.



SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance", the 'Text to Speech (AWS Polly)' module can be found using the quick filter.

For more information on adding instances, see the Instances documentation

CONFIGURATIONPAGE:

NAME	DESCRIPTION
Access Key	Access Key from AWS user who has access to Polly

NAME	DESCRIPTION
Secret Key	Secret Key of the AWS user who has access to Polly
Region	Region in which Polly should be used
Language	Language in which the output should take place
Output Format	Format (MP3/WAV/OGG) of the output
Sample Rate	Sample Rate (Standard, 8000 Hz, 16000 Hz, 22050 Hz) of the output
Text type	Type of text. For SSML the special SSML tags can be used

After entering the Access Key/Secret Key the configuration must be saved to load the available languages.

TTSAWSPOLLY_GenerateData

Require: Symcon >= 5.1

```
string TTSAWSPOLLY_GenerateData (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the Instance
Text	Voice data

RETURNS

Base64 encoded voice data

DESCRIPTION

Queries the text via AWS and returns the voice data base64 encoded in the return.

EXAMPLE

```
echo TTSAWSPOLLY_GenerateData(12345, "This is a test");
```

TTSAWSPOLLY_GenerateFile

Require: Symcon >= 5.1

```
string TTSAWSPOLLY_GenerateFile (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the Instance
Text	Voice data

RETURNS

Voice data filename

DESCRIPTION

Queries the text via AWS and returns the file name to the vocie data.

EXAMPLE

```
echo TTSAWSPOLLY_GenerateFile(12345, "this is a test");
```

Consumption Alert

Require: Symcon >= 6.0
The module is used to detect unusual consumption. It reacts to a counter variable and switches an alarm under certain conditions. There are two state variables.
A high consumption state, which switches when an adjustable limit value is exceeded. A low consumption state, which ticks up in 7 steps if a set limit value (e.g. dripping faucet) is exceeded over a longer period of time. The interval for both controls can be set via the configuration. An alarm, which switches when the high consumption state is switched to alarm, or the low consumption alarm exceeds a set value.

FUNCTION RANGE

Selection of the counter variable
Large/small consumption timer adjustable in minutes
Large/small consumption limit value adjustable
7 steps display for small consumption
Alarm display for large consumption

SOFTWARE INSTALLATION

Install the 'VerbrauchsAlarm' module via the Module Store.

SET UP INSTANCES IN IP-SYMCON

Under 'Add Instance' the 'VerbrauchsAlarm' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Counter variable	Variable that represents the counter value.
Alarm trigger	Default: 6 Low consumption level at which the alarm is triggered.
Low Consumption Interval	Default: 1min Time interval to check if the consumption is too high.
Low consumption limit	Default: 0 Limit value at which the state is changed. If this is not set, it will not be checked.
High consumption interval	Default: 5min Time interval in which it is checked if the consumption is too high.
Limit value for large consumption	Default: 0 Limit value at which the state is changed. If this is not set, it is not checked.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES:

NAME	TYPE	DESCRIPTION
Alarm notification	Boolean	Alarm when large consumption switched or small consumption exceeds value.
Low Consumption State	Integer	7 level indicator for the state of the alarm level.
High consumption state	Boolean	Alarm if the flow is too high.

PROFILE:

NAME	DESCRIPTION
VBA.LeakLevel	Profile for small consumption - 7 alarm levels with different symbols and color indicators
VBA.ThresholdValue	Profile for small/large consumption threshold value

Breakdown VBA.LeakLevel

LEVEL	VALUE
No activity	0
Everything okay	1
Normal activity	2
Increased activity	3
Abnormal Activity	4
Pre-Alarm	5
Alarm	6

VISUALIZATION

The limit values can be set via the Visualization.
It is additionally displayed whether an alarm is present or not.

VBA_CheckAlert

Require: Symcon >= 6.0

```
void VBA_CheckAlert (int $InstanceID, string $BorderValue, string
$OldValue)
```

PARAMETERS

InstanceID	ID of the Instance
BorderValue	Border Value
OldValue	Old Value

RETURNS

The function does not return any value.

DESCRIPTION

Checks within the consumer alarm with the InstanceID **InstanceID** whether a limit value was exceeded and sets the alarm variables.

EXAMPLE

```
VBA_CheckAlert(12345, "SmallUserThreashold", "SmallUserBuffer");
```

Water Alert

Require: Symcon >= 4.2
The module is used to detect unnaturally high water consumption. It reacts to a counter variable and switches an alarm under certain conditions. There are two alarm variables. A pipe burst alarm, which switches when large quantities flow at once. A leakage alarm, which ticks up in 7 steps when a small amount flows over a longer period of time (e.g. dripping faucet). The interval for both controls can be set via the configuration.

FUNCTION RANGE

Selection of water meter variable
Pipe burst timer adjustable in minutes
Leakage timer adjustable in minutes
7 steps display for leakage
Alarm display for pipe breakage

SOFTWARE INSTALLATION

Install the 'Water Alert' module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Water Alert' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Counter variable	Variable that represents the counter value.
Leakage interval	Default: 1min Time interval in which it is checked whether too much water has flowed through.
Pipe burst interval	Default: 15min Time interval in which it is checked whether too much water has flowed through.

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

STATUSVARIABLES

NAME	TYPE	DESCRIPTION
Leakage Limit	Float	Difference value, which is checked in the set "Leakage Interval" and if exceeded, the "Leakage" is increased by 1. If the difference is below or equal to the limit, the alarm level will be reset.
Leakage	Integer	7 step display for the status of the alarm level.
Pipe breakage limit	Float	Limit value, which is checked and may be the trigger for an alarm.
Pipe break	Boolean	Alarm if the flow is too high.

PROFILE:

NAME	DESCRIPTION
WAA.LeakLevel	Profile for leakage - 7 alarm levels with different symbols and color indicators
WAA.ThresholdValue	Profile for leakage/pipe breakage threshold value

VISUALIZATION

The threshold values can be set via the Visualization.
It is additionally displayed whether an alarm is present or not.

WAA_CheckAlert

Require: Symcon >= 4.2

```
bool WAA_CheckAlert (int $InstanzID, string $BorderValue, string
$OldValue)
```

PARAMETERS

InstanzID	ID of the Instance
BorderValue	Border Value
OldValue	Old Value

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Checks within the WaterAlarm with the InstanceID InstanceID whether a limit value was exceeded and sets the alarm variables

EXAMPLE

```
WAA_CheckAlert(12345, "LeakThreashold", "LeakBuffer");
```

Watchdog

Require: Symcon >= 5.1
Checks whether variables defined in a list are overdue. If variables are overdue, an alarm is set and a list of them is displayed in the Visualization.

SCOPE

Monitoring of listed variables.
Set whether the variables are to be checked for changes or updates.
Set how long the listed variables may be overdue.
Switching on/off via Visualization button or script function.
Display when the listed variables were last checked.
Display of the original path or an individual name.

SOFTWARE INSTALLATION

Install the Watchdog module via the Module Store.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Watchdog' module can be found using the quick filter.
More information about adding instances in the Instances documentation.
All variables to be switched must be added to the "Variables" list in the instance configuration.

CONFIGURATIONPAGE:

NAME	DESCRIPTION
Variables	Variables to watch are added to this list.
Time	Duration of inactivity until the listed variables trigger the alarm.
Unit	Unit of time.

If variables have different allowed inactivity time, it is recommended to use multiple instances of the watchdog module.

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

NAME	TYPE	DESCRIPTION
Active alarms	String	Contains the table for the display in the Visualization.

NAME	TYPE	DESCRIPTION
Alarm	Boolean	The variable indicates whether an alarm is present. True = Alarm; False = OK;
Last Check	Integer	UnixTimestamp which indicates the time when the last check was done.
Watchdog active	Boolean	Indicates whether the watchdog is activated or not. True = Enabled; False = Disabled;

VISUALIZATION

The watchdog can be enabled/disabled via the Visualization.
Additionally the information is displayed at which time the watchdog was last checked.

WD_GetAlertTargets

Require: Symcon >= 5.1

```
array WD_GetAlertTargets (int $InstanzID)
```

PARAMETERS

InstanzID	ID of the Instanz

RETURNS

Array of active alarms of the watchdog instance with InstanceID **InstanceID**.

DESCRIPTION

The function returns an array with the active alarms of the watchdog instance with InstanceID **InstanceID**.
The function returns an array with overdue objects. It contains the optional name, VariableID and the last time (UnixTimestamp) of the update or change.

EXAMPLE

```
WD_GetAlertTargets(12345);
```

WD_SetActive

Require: Symcon >= 5.1

```
bool WD_SetActive (int $InstanzID, bool $SetActive)
```

PARAMETERS

InstanzID	ID of the Instanz
SetActive	Enables/disables the watchdog

RETURNS

If the command could be executed successfully, it returns **TRUE** as result, otherwise **FALSE**.

DESCRIPTION

SetActive activates (true) or deactivates (false) the watchdog with the InstanzID **InstanzID**.

EXAMPLE

```
WD_SetActive(12345, true);
```

Core Instances

Archive Control	Functions for managing the data logging
AC_AddLoggedValues	adds further data records to a logged variable
AC_ChangeVariableID	migrates the data of a variable into a variable that has not yet been logged
AC_DeleteVariableData	deletes all data records of a logged variable in a certain period of time
AC_GetAggregatedValues	gets aggregated data from the archive
AC_GetAggregationType	indicates the aggregation type of a variable
AC_GetAggregationVariables	returns an array of all logged variables
AC_GetCompaction	returns the compaction entries of a variable
AC_GetCounterIgnoreZeros	gets the status whether zeros and negative values are ignored
AC_GetGraphStatus	asks whether a variable is being visualized
AC_GetLoggedValues	gets raw data from the Archive Control
AC_GetLoggingStatus	asks whether a variable is logged
AC_ReAggregateVariable	starts the re-aggregation of a variable
AC_SetAggregationType	sets the aggregation type of a variable
AC_SetCompaction	Configures the compression of the variable
AC_SetCounterIgnoreZeros	sets the status whether zeros and negative values are ignored
AC_SetGraphStatus	sets the property for the visualization of a variable
AC_SetLoggingStatus	sets the property for logging a variable
Data Format	
Permission Control	User management and synchronization with LDAP server
Calendar Control	Calendar module
Connect Control	Connection module used to enable a secure connection from the outside
Cutter	
Cutter_ClearBuffer	clears the buffer of the cutter instance
DNS-SD Control	manages established Bonjour services
ZC_QueryService	requests information from a single service
ZC_QueryServiceType	requests information from services of a certain type
ZC_QueryServiceTypes	Queries all the services that have been set up

Event Control	Module to start scripts on set events
Location Control	calculates twilight times based on latitude and longitude
Module Control	manages external and added PHP modules
Notification Control	a module for sending push notifications
Presence Control	Functions for detecting presence within an area

PC_Enter	sets presence to true
PC_Leave	sets presence to false

RegisterVariable	RegisterVariables serve as an interface for forwarding and processing data

RegVar_GetBuffer	reads the internal buffer
RegVar_SendEvent	sends an event to the parent HID instance
RegVar_SendPacket	sends a data packet to the specified IP address
RegVar_SendText	sends data to the parent instance
RegVar_SetBuffer	sets the internal buffer

Skin Control	manages external and uploaded skins
SSDP Control	enables the search for UPnP devices

YC_SearchDevices	sends a request to the local network

System Information	Functions for querying system information and hardware

Sys_GetBattery	displays information about the battery
Sys_GetCPUInfo	shows the CPU usage
Sys_GetHardDiskInfo	shows the hard disk space/usage
Sys_GetMemoryInfo	shows the memory usage
Sys_GetNetworkInfo	shows information about the network adapters
Sys_GetProcessInfo	lists all processes
Sys_GetSpooler	displays information about the printer queues
Sys_GetURLContent	reads the content from an URL as String
Sys_GetURLContentEx	Reads the contents of the __URL__ and returns it as a return value. Here, some __Parameter__ for timeouts, authentication and proxy are made. -
Sys_Ping	sends a ping to a network device

Tailscale VPN	Use of Tailscale VPN on the SymBox
TextParser	provides text snippets from websites/files
Translation Control	
Util Control	serves as an interface with extended functions between the IP-Symcon service and the management console

UC_FindReferences	searches all places where an object is used
UC_GetEventStatistics	returns the number of updates of events
UC_GetIconList	returns the names of all available icons
UC_GetKernelStatistics	returns key figures on the message and script queue of the kernel
UC_GetLastLogMessages	returns the last log messages of a message type
UC_GetLogMessageStatistics	returns the number of log messages per message type
UC_GetMessageSenderIDList	returns the number of messages per sender
UC_GetMessageSenderIDSizeList	returns the amount of message data per sender
UC_GetMessageTypeList	returns the number of messages per message type
UC_GetScriptSenderList	returns the number of script executions per trigger
UC_ResetLastLogMessages	clears the stored last log messages
UC_ResetLogMessageStatistics	resets the number of log messages per message type

WebHook Control	Module for calling scripts via browser

WC_PushMessage	sends a WebSocket message to all connected clients
WC_PushMessageEx	sends a WebSocket message to a specific client

WebServer	Module which makes the Visualization accessible via port

Archive Control

The Archive Control module takes care of storing the variable values in a database.

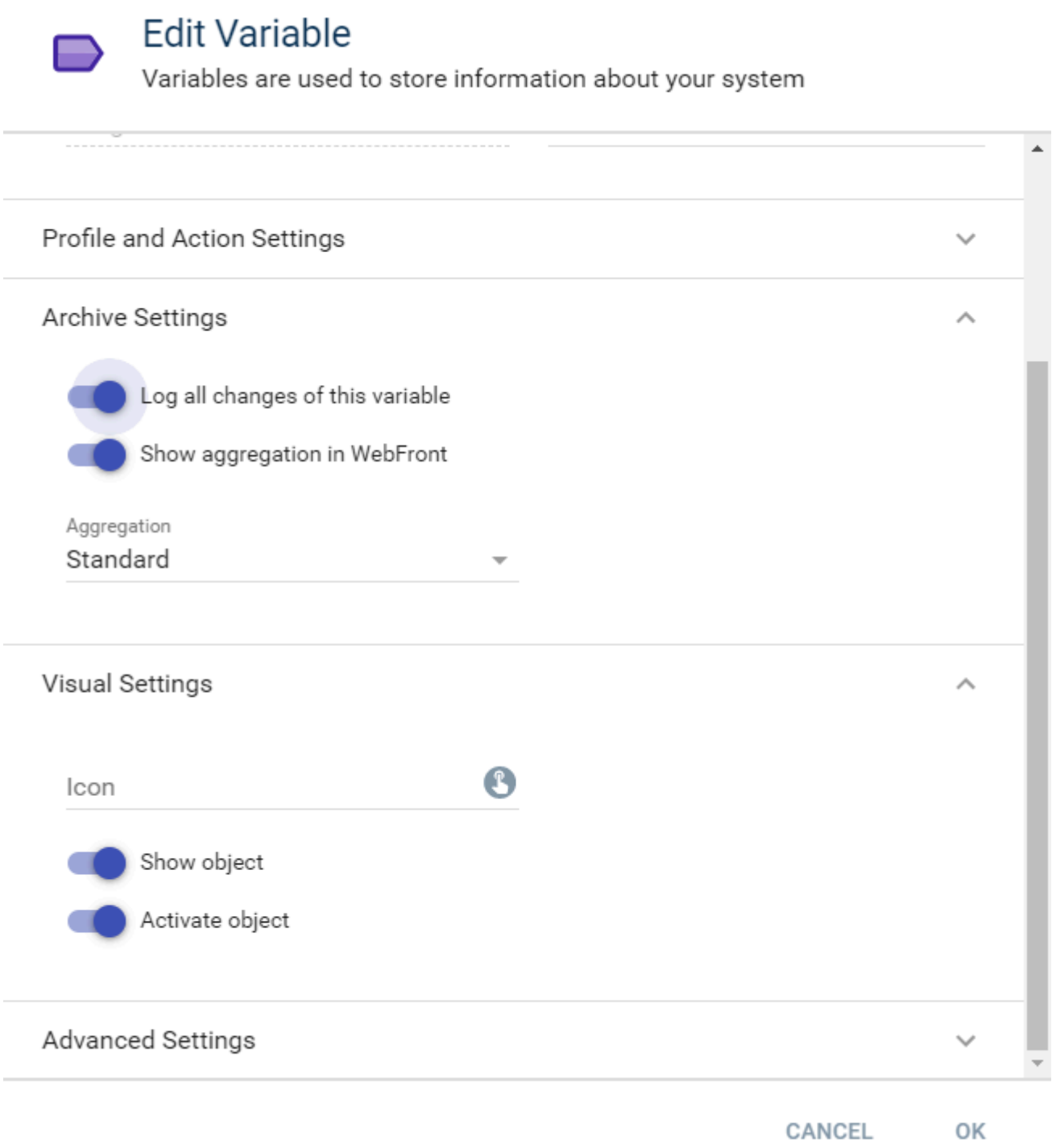
STORAGETYPE

With version 4.0 was the SQLite database deleted for an easier usability, using CSV files.

IPS VERSION	STORAGETYPE
to Version 3.4	SQLite database
from Version 4.0	CSV files

ACTIVATE LOGGING

To log a variable in the database, it must be selected for it. You will need to edit the variable and select the checkbox "Log all changes of this variable". If no button for showing the graph is wanted, deactivate the option "Show aggregation in Visualization".



AGGREGATION TYPE	DESCRIPTION
Standard	For all aggregationlevels (hour, day, week, month, year) will min / max / average calculated.
Counter	For all aggregationlevels (hour, day, week, month, yea) will the delta (difference) of the values as min / max / sum calculated.

If the Counter aggregation is used, the first value logged in the archive is used as reference value and the sum of positive delta starts with the following value. That means a variable with the logged values 500, 505, 520, and 521 is displayed with the values 0, 5, 15, 1.

ARCHIVE SETTINGS	DESCRIPTION
Log all changes of this varaible	Activates the logging of the variable.
Show aggregation in Visualization	Deactivate/Hide the button for showing the visualization of the logged variable. The logging ist still active for the varabelDas Aufzeichnen der Datenpunkte der Variable ist immernoch aktiv.

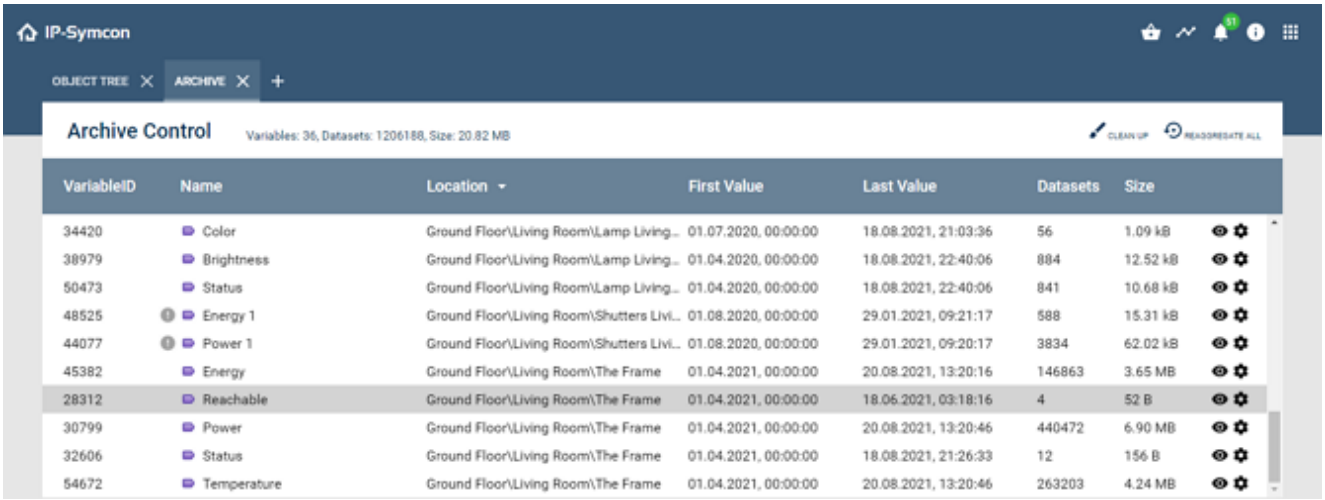
If the aggregationtype is changed, the aggregation of this variable starts automatically. This could take some time.

As soon as logging is selected, all data is saved in the database. The data is saved two times. All raw data is logged in the database and these datasets are summarized as individual intervals (days, weeks, months, years) for faster generation of graphs. The graphs can, if activated for the variable, be accessed via Visualization.

ARCHIVE HANDLER

The Archive Handler is a function to manage all logged variables. All values can be viewed and edited.

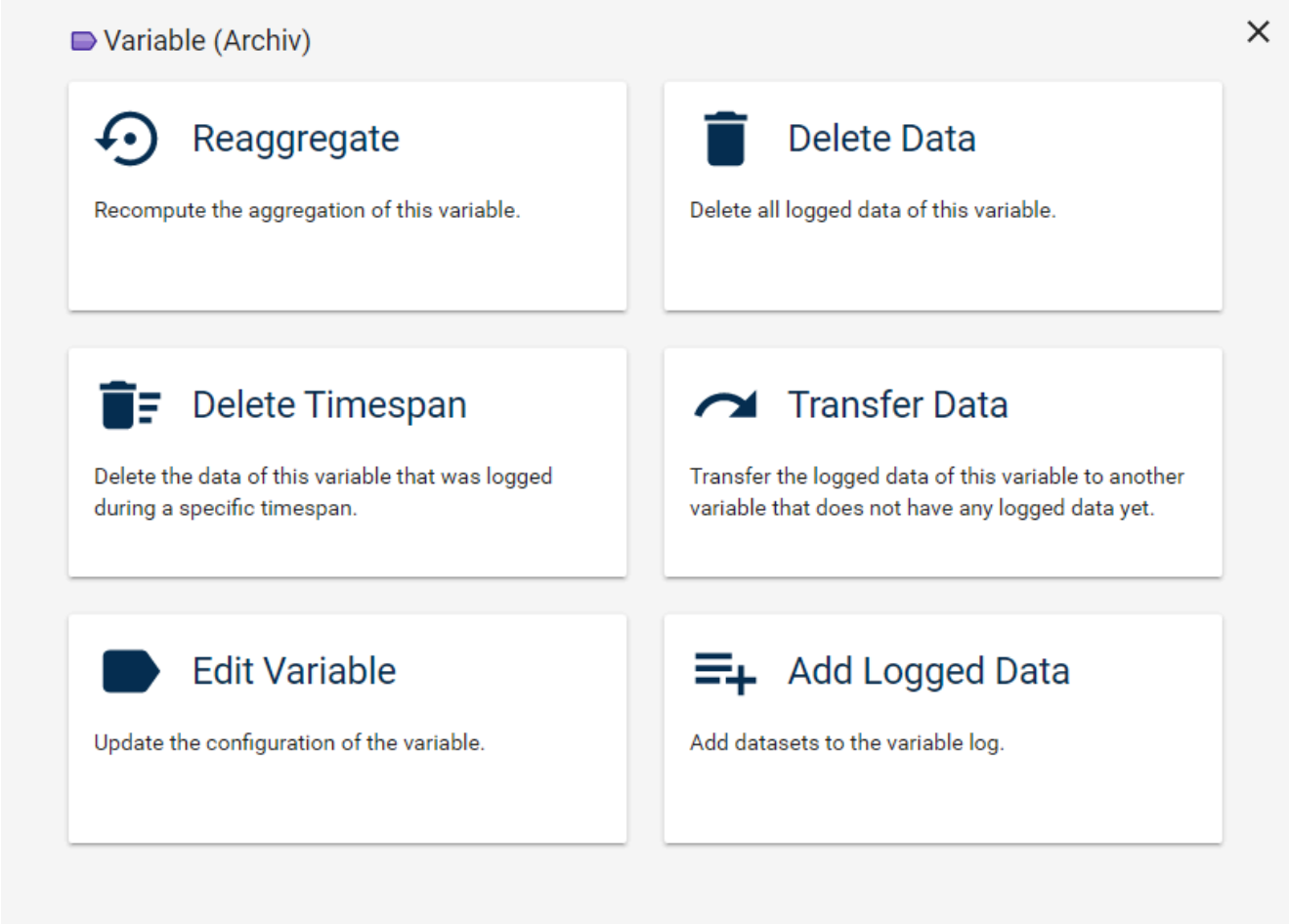




The header provide following functions:

FUNCTION	DESCRIPTION
Clean up	Removes all archivedata from nonexistent variables. Data of deactivated variables won't get deleted.
Reaggregate all	Aggregates all logged variables again. Depending on the scope, this may take a very long time. Regular status and finishing messages are available in the tab "Message Log".

Clicking the eye icon opens the view with saved raw data of the selected variable and allows the deletion of individual datasets. **After deletion, it is required to reaggregate the variable data.** This function is not available for variables that do not exist any more. A click on the cog wheel opens a dialog with extended functions for the selected variable



FUNCTION	DESCRIPTION
Reaggregate	Aggregates the selected variable again. Depending on the scope, this could take very long. Regular status and finishing messages are available in the tab "Message Log".
Delete	Deletes all the raw data from the archive.
Delete timespan	A time interval can be selected here. All datasets within that interval are deleted.

FUNCTION	DESCRIPTION
Transfer data	The raw data of a variable is transferred to another variable. This can be useful if a VariableID does not exist any more or a sensor is replaced. A variable must be selected which is not logged yet and whose variable type fits the data. The data of the old variable will be automatically deleted in this process.
Edit variable	Opens the dialog to edit the Variable
Add logged data	Additional datasets can be added here. Datasets can either be entered manually or from files. For manual adding, another dialog opens, in which a list of times and values can be entered, that are added to the variable with a click on "Add Values". If datasets from files should be added, accordingly prepared csv files are required. In an additional step the values can be checked manually and finally be added to the variable.

POSSIBLE DATASTRUCTURES FOR "ADD LOGGED DATA"

In .csv files for adding logged data, every row must note a point in time and a data value. Two basic notations are possible:

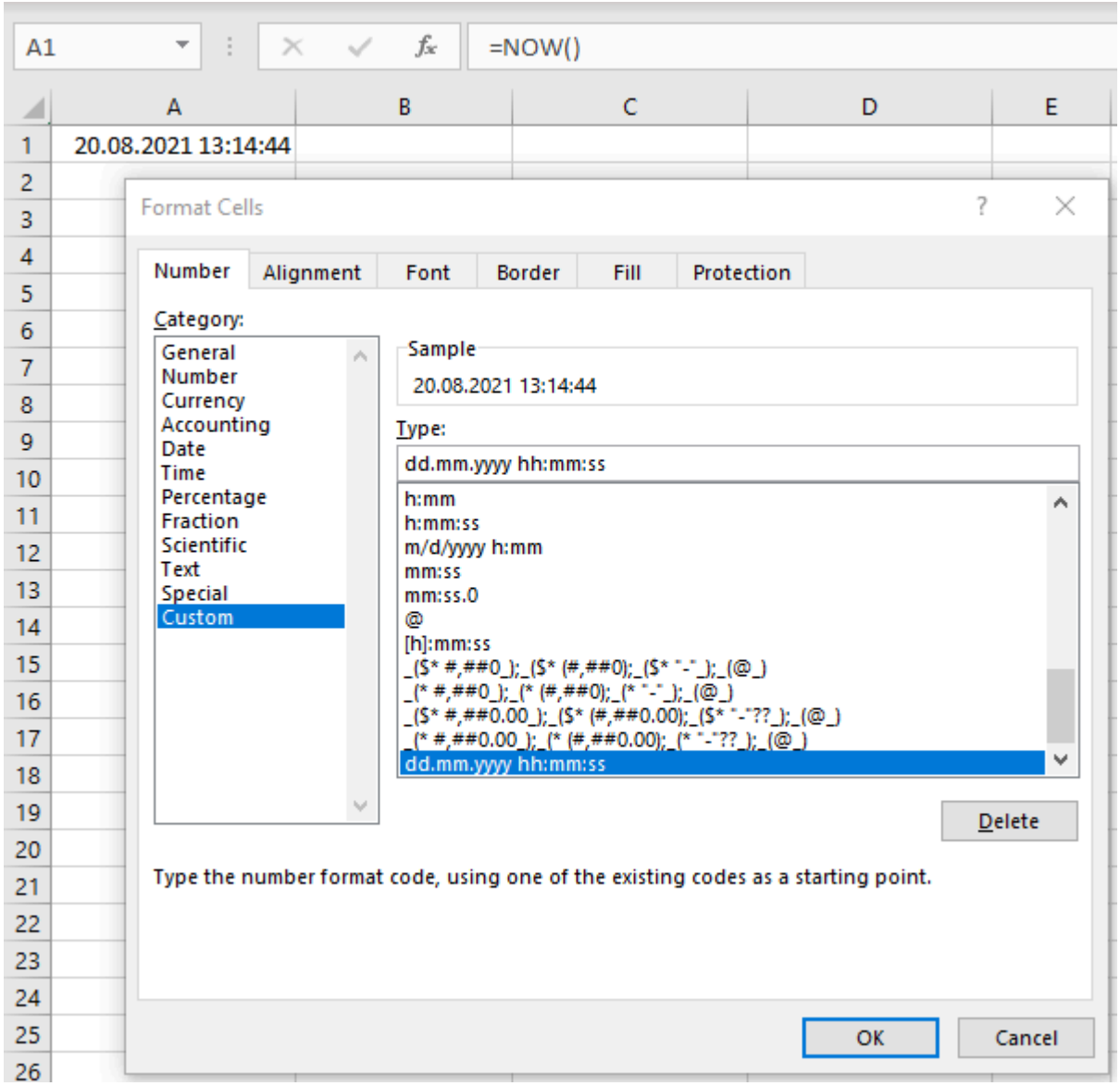
Comma as seperator and dot as decimal seperator (Date, 1.5)
Semicolon as seperator and comma as decimal seperator (Date; 1,5)

Several formats for the date are supported:

Unix Timestamp, e.g., 1522527010
German format, e.g., 31.03.2018 22:10:10
American format, e.g., 03/31/2018 10:10:10 PM
RFC2822, e.g., Sat, 31 Mar 2018 22:10:10 +0000
ISO8601, e.g., 2018-03-31 22:10:10

ADD FILES FROM EXCEL FILE

It is possible to prepare the .csv files for adding with Microsoft Excel. Here, the points in time are entered into the first column and the values into the second. Verify that the point of time is formatted in one of the supported formats. For example, the German format can be used with the user defined number format DD.MM.YYYY hh:mm:ss.



Finally, the list can be saved as .csv file via "Save as" and the type "CSV (Seperator delimited)" and finally confirming with "Save". The generated .csv file kann be added in IP-Symcon via "Add Data" of a Variable.

AC_AddLoggedValues

Require: Symcon >= 5.1

```
bool AC_AddLoggedValues (int $InstanceID, int $VariableID, array
$Datasets)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to which the datasets are to be added
Datasets	An array with datasets, where each set is an array with the following **key => value** pairs:

PARAMETERS	TYPE	DESCRIPTION
TimeStamp	Integer	Time of the new data record as a Unix Timestamp
Value	by variable	The value of the new data record - the type of the value must match the type of the variable

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function adds the data records **Datasets** to the logged raw data of the variable with the ID **VariableID**.



After this function, the aggregated data of the variable is invalid and must be re-aggregated.

EXAMPLE

```
// Add three additional data records to an Integer variable
AC_AddLoggedValues(12345, 34567, [
[
```

```
'TimeStamp' => 1128255120,
'Value' => 30
],
[
'TimeStamp' => 1128257851,
'Value' => 50
],
[
'TimeStamp' => 1128278514,
'Value' => 130
]
]);
```

AC_ChangeVariableID

Require: Symcon >= 3.0

```
bool AC_ChangeVariableID (int $InstanceID, int $OldVariableID, int
$NewVariableID)
```

PARAMETERS

InstanceID	ID for the archive
OldVariableID	ID of the variable to be migrated
NewVariableID	ID of the new variable

RETURNS

True if successful, otherwise **False**

DESCRIPTION

This function migrates the data of the variable with the ID **OldVariableID** into the variable with the ID **NewVariableID**.



The variable with the ID **NewVariableID** must not be logged. This also means that no old data records may be available through logging.

EXAMPLE

```
// The data of the 34567 variable are migrated to the 56789 variable
AC_ChangeVariableID(12345, 34567, 56789);
```

AC_DeleteVariableData

Require: Symcon >= 3.0

```
int AC_DeleteVariableData (int $InstanceID, int $VariableID, int
$StartTime, int $EndTime)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable in which the data records are to be deleted
StartTime	Date/time as Unix Timestamp (0 = from the beginning)
EndTime	Date/time as Unix Timestamp (0 = until now)

RETURNS

Returns the number of deleted records

DESCRIPTION

This function deletes all logged data records of the variable with the ID **VariableID** in the period from **start time** up to and including **end time**. If the start time and the end time = 0, the logging of the variable is also deactivated.



If data records are to be deleted at a single point in time, the end time must be set equal to the start time



It should be noted that the Unix Timestamp is based on UTC+0. Many European countries have summer time (CEST/UTC+2) and winter time (CET/UTC+1).



After execution, the aggregation of the variable is invalid and must first be re-aggregated. AC_ReaggregateVariable can be used for this.



If the parameters start and end time = 0, the data records of the variable are completely deleted.

EXAMPLE

```
// Deletes all data of the variable "TestVariable" in the period
07.10.2015 16:00 UTC+2(CEST) to 07.10.2015 17:00 UTC+2(CEST)
AC_DeleteVariableData(12345 /*[Archive]*/, 45678 /*[Variable]*/,
1444226400, 1444237200);

// Deletes all data of the variable "TestVariable" in the period
07.10.2015 16:00 UTC+2(CEST) until now
AC_DeleteVariableData(12345 /*[Archive]*/, 45678 /*[Variable]*/,
1444226400, 0);

// Delete completely and remove from the Archive Control
// Deletes all data of the variable and deactivates the variable in the
Archive Control
AC_DeleteVariableData(12345 /*[Archive]*/, 45678 /*[TestVariable]*/, 0,
0);
```

AC_GetAggregatedValues

Require: Symcon >= 3.0

```
array AC_GetAggregatedValues (int $InstanceID, int $VariableID, int $AggregationLevel, int $StartTime, int $EndTime, int $Limit)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried
AggregationLevel	

AGGREGATION LEVEL	DESCRIPTION
0	Hourly Aggregation (00:00 - 59:59)
1	Daily Aggregation (00:00:00 - 23:59:59)
2	Weekly Aggregation (Monday 00:00:00 - Sunday 23:59:59)
3	Monthly Aggregation (First day of month 00:00:00 - Last day of month 23:59:59)
4	Annual Aggregation (01.01. 00:00:00 - 31.12. 23:59:59)
5	5-Minute Aggregation (Calculated from raw data)
6	1-Minute Aggregation (Calculated from raw data)
8	15-Minute Aggregation (Calculated from raw data) (from version 8.1)

StartTime	Date/time as Unix Timestamp (0 = from the beginning)
EndTime	Date/time as Unix Timestamp (0 = until now)
Limit	Maximum number of datasets. (0 = no limit, 10000 is the hard limit, which always applies)

RETURNS

An array with the following **key => value** pairs.

The output starts with the newest dataset and then, in descending order, with the older data sets.

Meaning of the fields for the aggregation type **Standard**

INDEX	TYPE	DESCRIPTION
Avg	variant	Average value within this aggregation period
Duration	integer	Duration of the aggregation period in seconds
Max	variant	Largest value within this aggregation period
MaxTime	variant	Date/time of **Max** as Unix Timestamp
Min	variant	Smallest value within this aggregation period
MinTime	variant	Date/time of **min** as Unix Timestamp
TimeStamp	integer	Date/time of the start of the aggregation period as a Unix Timestamp

Meaning of the fields for the aggregation type **Counter**

INDEX	TYPE	DESCRIPTION
Avg	variant	Sum of the positive delta within this aggregation period
Duration	integer	Duration of the aggregation period in seconds
Max	variant	Largest positive delta within this aggregation period
MaxTime	variant	Date/time of **Max** as Unix Timestamp
Min	variant	Smallest positive delta within this aggregation period
MinTime	variant	Date/time of **min** as Unix Timestamp
TimeStamp	integer	Date/time of the start of the aggregation period as a Unix Timestamp

DESCRIPTION

This command returns the aggregated data that was logged and calculated via the archive for a variable with the **VariableID**. The **Start-** and **EndTime** can be specified as parameters. The parameter **Limit** limits the maximum number of data records returned, whereby the limit of 10000 data records per query can never be exceeded.

This function guarantees that all data records between the StartTime and the EndTime are also available and are put out. This means that there are no time gaps between the data records. Data records are put out up to the current point in time.

Whole aggregated time periods are always queried. For this, the StartTime (depending on the **aggregation level**) of the hour/day/week/month/year in the period of **StartTime** and **EndTime** must be included.

The aggregation levels 15-minute/5-minute/1-minute aggregation are calculated from the raw data, so that if there is a lot of raw data, they may require more system performance than the other aggregation levels

EXAMPLE

```
// Query all data records from 01/01/2013 to 12/31/2013 (daily
aggregation level)
// e.g. to determine the consumption on the respective day or the
average temperature on the respective day
$values = AC_GetAggregatedValues(12345, 55554, 1 /* daily */, mktime(0,
0, 0, 1, 1, 2013), mktime(23, 59, 59, 12, 31, 2013), 0); //55554 is the
variable ID, 12345 from the archive

// Query all current data records (daily aggregation level)
// e.g. to determine today's consumption or today's average temperature
$values = AC_GetAggregatedValues(12345, 55554, 1 /* daily */,
strtotime("today 00:00"), time(), 0); //55554 is the ID of the
variable, 12345 from the archive

// Query all yesterday's records (hourly aggregation level)
// For example, to check yesterday's consumption or the average wind
speed every hour
$values = AC_GetAggregatedValues(12345, 55554, 0 /* hourly */,
strtotime("yesterday 00:00"), strtotime("today 00:00")-1, 0); //55554
is the ID of the variable, 12345 from the archive

// This part creates an output in the script window with the queried
values
foreach($values as $value) {
echo date("d.m.Y H:i:s", $value['TimeStamp']) . " -> " . $value['Avg']
. PHP_EOL;
}
```

AC_GetAggregationType

Require: Symcon >= 3.0

```
int AC_GetAggregationType (int $InstanceID, int $VariableID)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried

RETURNS

An integer that represents the type of aggregation.

RETURN VALUE	DESCRIPTION
0	Standard
1	Counter

DESCRIPTION

This function queries the aggregation type of the variable with the ID **VariableID** and returns it as an integer.

EXAMPLE

```
// The variable is aggregated as a counter
echo AC_GetAggregationType(39147 /*[Archive]*/, 53716 /*
[TestVariableCounter]*/);

// Output: 1
```

AC_GetAggregationVariables

Require: Symcon >= 3.0

```
array AC_GetAggregationVariables (int $InstanceID, bool $DatabaseQuery)
```

PARAMETERS

InstanceID	ID for the archive
DatabaseQuery	True, if additional information is to be displayed, otherwise False (with version 4.0 and above unimportant, but required)

RETURNS

An array with the following **key => value** pairs.

INDEX	TYPE	DESCRIPTION
FirstTime	integer	Date/time from the beginning of the aggregation period as a Unix Timestamp
LastTime	integer	Date/time of the last entry of the aggregation period as a Unix Timestamp
RecordCount	integer	Number of records
RecordSize	integer	Size of all data records in bytes
VariableID	integer	ID of the variable
AggregationType	integer	Aggregation type as an Integer. See also AC_GetAggregationType
AggregationVisible	boolean	Indicates whether the variable is displayed in the visualization. See also AC_GetGraphStatus
AggregationActive	boolean	Indicates whether logging is active for this variable. See also AC_GetLoggingStatus
Compaction	array	Array of compaction entries. Each entry contains MonthOffset and CompactionType. See also AC_SetCompaction

DESCRIPTION

With version 4.0 and above the specified parameter **DatabaseQuery** is no longer used. However, this must be specified for downward compatibility.

This function returns an array of all logged tags of the archive handler with the ID **InstanceID**. Additional information about each variable can be activated with **DatabaseQuery**. (Functional up to version 3.4)

EXAMPLE

```
// Output with additional information
var_dump(AC_GetAggregationVariables(39147 /*[Archive]*/, true));

// Sample output:
array(1) {
[0]=>
array(9) {
["FirstTime"]=>
int(1444221643)
["LastTime"]=>
int(1444221705)
["RecordCount"]=>
int(6)
["RecordSize"]=>
int(78)
["VariableID"]=>
int(53716)
["AggregationType"]=>
int(0)
["AggregationVisible"]=>
bool(true)
["AggregationActive"]=>
bool(true)
["Compaction"]=>
array(1) {
[0]=>
array(2) {
["MonthOffset"]=>
int(-1)
["CompactionType"]=>
int(1)
}
}
}
}
*/
```

AC_GetCompaction

Require: Symcon >= 6.3

```
array AC_GetCompaction (int $InstanceID, int $VariableID)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried

RETURNS

An array containing the compaction entries of the variable.
Each entry contains the following fields:

FIELD	TYPE	DESCRIPTION
MonthOffset	int	Month offset for the compaction
CompactionType	int	Type of compaction (see below)

Compaction Types

COMPACTIONTYPE	DESCRIPTION
0	Compact to one value per minute
1	Compact to one value per 5 minutes
2	Compact to one value per hour
3	Compact to one value per day
4	Compact to one value per week
5	Compact to one value per month
6	Compact to one value per year
7	Delete values

DESCRIPTION

This function queries the compaction entries of the variable with the ID **VariableID** and returns them as an array. Each entry in the array represents a configured compaction with the corresponding month offset and compaction type.

EXAMPLE

```
// Query the compaction entries for "TestVariable"
$compaction = AC_GetCompaction(39147 /*[Archive]*/, 53716 /*
[TestVariable]*/);
print_r($compaction);

/* Sample output:
Array
(
[0] => Array
(
[MonthOffset] => -1
[CompactionType] => 1
)
[1] => Array
(
[MonthOffset] => 3
[CompactionType] => 2
)
)
*/
```

AC_GetCounterIgnoreZeros

Require: Symcon >= 5.5

```
bool AC_GetCounterIgnoreZeros (int $InstanceID, int $VariableID)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried

RETURNS

True if ignoring zeros and negative values is active for the variable, otherwise False

DESCRIPTION

This function queries whether the variable with the ID **VariableID** in the Archive Control with the ID **InstanceID** ignores zeros and negative values.



The function only works with variables that are logged as counters. If the ignoring of zeros and negative values is active, they will still be logged, but will not be included in the aggregation.

EXAMPLE

```
// Query the variable "54321" from the ArchivControl "12345"
AC_GetCounterIgnoreZeros(12345, 54321);
```

AC_GetGraphStatus

Require: Symcon >= 3.0

```
bool AC_GetGraphStatus (int $InstanceID, int $VariableID)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried

RETURNS

True if the visualization of the variable is active, otherwise **False**

DESCRIPTION

This function queries whether the variable with the ID **VariableID** is displayed in the visualization.

EXAMPLE

```
// Query the variable "TestVariable"
var_dump(AC_GetGraphStatus(39147 /*[Archive]*/, 53716 /*
[TestVariable]*/));
echo AC_GetGraphStatus(39147 /*[Archive]*/, 53716 /*[TestVariable]*/);

/* Sample output
var_dump:
bool(true)
echo:
1
*/
```

AC_GetLoggedValues

Require: Symcon >= 3.0

```
array AC_GetLoggedValues (int $InstanceID, int $VariableID, int
$StartTime, int $End time, int $Limit)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried
StartTime	Date/time as Unix Timestamp. (0 = from the beginning)
End time	Date/time as Unix Timestamp. (0 = until now)
Limit	Maximum number of records. (0 = no limit, 10000 is the hard limit, which always applies)

RETURNS

An array with the following **key => value** pairs.



The output starts with the newest data set and then, in descending order, with the older dataset / datasets.

INDEX	TYPE	DESCRIPTION
Duration	integer	Duration in seconds that the data record was set
TimeStamp	integer	Date/time when the dataset / datasets was created as a Unix Timestamp
Value	variant	Value

DESCRIPTION

This command returns the raw data that were also logged via the Archive Control for a variable with the **VariableID**. The **Start** - and **End time** can be specified as parameters. The parameter **Limit** limits the maximum number of data records returned, whereby the limit of 10000 data records per query can never be exceeded.

The Archive Control only creates new data records for changed values! Updates are not taken into account.

This function places a considerable load on the system, since in most cases a large number of data records have to be processed. In any case, please use the AC_GetAggregatedValues function to switch to pre-aggregated values. This function gives you direct access to mean values or sums for different periods of time such as years, months, weeks and days.

This function was bugged in IP-Symcon 2.x. One more value, which was before the start time, was always put out . This was useful insofar as the previous value was known for, for example, the generation of graphs, and a graph could be created for the entire period of time. Otherwise a gap would be visible at the beginning of the graph. However, this functionality is not correct for a function with the parameter start time.
If one requires exactly this special functionality for their scripts, one can use the auxiliary function, which queries the previous data record which is before the start time and adds it to the array.

EXAMPLE

```
//Get the last value that was saved in the database
$last_value = AC_GetLoggedValues(12345, 55554, 0, 0, 1)[0]['Value'];

// Query all data records from 01/01/2013 to 01/07/2013
$values = AC_GetLoggedValues(12345, 55554, mktime(0, 0, 0, 1, 1, 2013),
mktime(23, 59, 59, 1, 7, 2013), 0); //55554 is the ID of the variable,
12345 from the Archive Control

//Query all of today's records
$values = AC_GetLoggedValues(12345, 55554, strtotime("today 00:00"),
time(), 0); //55554 is the ID of the variable, 12345 from the Archive
Control

//Query all of yesterday's records
$values = AC_GetLoggedValues(12345, 55554, strtotime("yesterday
00:00"), strtotime("today 00:00")- 1, 0); //55554 is the ID of the
variable, 12345 from the Archive Control

//This part creates an output in the script window with the queried
values
foreach($values as $value) {
echo date("d.m.Y H:i:s", $value['TimeStamp']) . " -> " .
$value['Value'] . PHP_EOL;
}
```

```
//Auxiliary function that simulates the functionality of IP-Symcon 2.x.
function AC_GetLoggedValuesCompatibility($instanceID, $variableID,
$startTime, $endTime, $limit) {
$values = AC_GetLoggedValues($instanceID, $variableID, $startTime,
$endTime, $limit );
if((sizeof($values) == 0) || (end($values)['TimeStamp'] > $startTime))
{
$previousRow = AC_GetLoggedValues($instanceID, $variableID, 0,
$startTime - 1, 1 );
$values = array_merge($values, $previousRow);
}
return $values;
}
```

AC_GetLoggingStatus

Require: Symcon >= 3.0

```
bool AC_GetLoggingStatus (int $InstanceID, int $VariableID)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried

RETURNS

True if logging is active for the variable, otherwise **False**

DESCRIPTION

This function queries whether the variable with the ID **VariableID** is logged.

EXAMPLE

```
// Query the variable "TestVariable"
var_dump(AC_GetLoggingStatus(39147 /*[Archive]*/, 53716 /*
[TestVariable]*/));
echo AC_GetLoggingStatus(39147 /*[Archive]*/, 53716 /*
[TestVariable]*/);

/* Sample output
var_dump:
bool(true)
echo:
1
*/
```

AC_ReAggregateVariable

Require: Symcon >= 3.0

```
bool AC_ReAggregateVariable (int $InstanceID, int $VariableID)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be re-aggregated

RETURNS

True if successful, otherwise **False**

DESCRIPTION

This function starts the re-aggregation of the variable with the ID **VariableID**.



Re-aggregation can take a long time. The current progress of the re-aggregation can be checked via regular entries in the message window.

EXAMPLE

```
// This starts the re-aggregation of the variable "test variable"
AC_ReAggregateVariable(39147 /*[Archive]*/, 53716 /*[TestVariable]*/);

/* Sample output in the message window
07.10.2015 15:50:05 | Archive Control | Reaggregation for VariableID
#53716 is in progress... 10/2015! (6 rows)
07.10.2015 15:50:05 | Archive Control | Reaggregation for VariableID
#53716 is complete! (6 rows)
*/
```

AC_SetAggregationType

Require: Symcon >= 3.0

```
bool AC_SetAggregationType (int $InstanceID, int $VariableID, int
$Aggregation_Type)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable for which the aggregation type is to be set
Aggregation_Type	

AGGREGATIONSTYP	DESCRIPTION
0	Standard
1	Counter

RETURNS

True if successful, otherwise **False**

DESCRIPTION

This function sets the aggregation type **AggregationType** for the variable with the ID **VariableID**



Afterwards a reaggregation of the variable is necessary.

EXAMPLE

```
// Sets the aggregation type of the "TestVariable" variable to 1
(counter)
AC_SetAggregationType(39147 /*[Archive]*/, 53716 /*[TestVariable]*/,
1);
```

AC_SetCompaction

Require: Symcon >= 6.3

```
bool AC_SetCompaction (int $InstanceID, int $VariableID, int
$MonthOffset, int $CompactionType)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable for which the compaction is to be set
MonthOffset	The number of months after which the compaction is applied; 0 = After the current month is ended, -1 = Compact values directly
CompactionType	

COMPACTIONTYPE	DESCRIPTION
-1	Deactivate compaction
0	Compact to one value per minute
1	Compact to one value per 5 minutes
2	Compact to one value per hour
3	Compact to one value per day
4	Compact to one value per week
5	Compact to one value per month
6	Compact to one value per year
7	Delete values

RETURNS

True if successful, otherwise **False**

DESCRIPTION

This function updates the compaction entry of the variable with the ID **VariableID** and the month offset **MonthOffset** and sets it to the compaction type **CompactionType**. Other compaction entries with different month offsets remain unaffected.



Afterwards a reaggregation of the variable is necessary.

EXAMPLE

```
// Set the direct compaction of the variable "TestVariable" to 1 (One
value per 5 minutes)
AC_SetCompaction(39147 /*[Archive]*/, 53716 /*[TestVariable]*/, -1, 1);
```

AC_SetCounterIgnoreZeros

Require: Symcon >= 5.5

```
bool AC_SetCounterIgnoreZeros (int $InstanceID, int $VariableID, bool $IgnoreZeros)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried
IgnoreZeros	True if zeros and negative values are to be ignored, otherwise False.

RETURNS

DESCRIPTION

This function changes the property for ignoring zeros and negative values of the variable with the ID **VariableID** in the Archive Control with the ID **InstanceID** to active.



The function only works with variables that are logged as counters. Zeros and negative values are still logged, but are not included in the aggregation.

// Sets the property of the "54321" tag in the Archive Control "12345" to ignore zeros and negative values to true
AC_SetCounterIgnoreZeros(12345, 54321, true);

EXAMPLE

AC_SetGraphStatus

Require: Symcon >= 3.0

```
bool AC_SetGraphStatus (int $InstanceID, int $VariableID, bool $Active)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried
Active	True if the display of the variable is to be activated in the visualization, otherwise False.

RETURNS

True if successful, otherwise **False**

DESCRIPTION

This function changes the property for visualizing the variable with the ID **VariableID** to **Active**.

EXAMPLE

```
// Sets the property of the "test variable" for visualization to true
AC_SetGraphStatus(39147 /*[Archive]*/, 53716 /*[TestVariable]*/, true);
```

AC_SetLoggingStatus

Require: Symcon >= 3.0

```
bool AC_SetLoggingStatus (int $InstanceID, int $VariableID, bool
$Active)
```

PARAMETERS

InstanceID	ID for the archive
VariableID	ID of the variable to be queried
Active	True if the logging of the variable is to be activated, otherwise False.

RETURNS

True if successful, otherwise **False**

DESCRIPTION

This function changes the property for logging the variable with the ID **VariableID** to **Active**.



The logging of the variables is activated immediately, but since the current value and changes to the variables are logged asynchronously, it is possible that these are not directly available when a script is executed. A possible workaround for this is to use the sleep command

EXAMPLE

```
// Sets the property of the "test variable" for logging to true
AC_SetLoggingStatus(39147 /*[Archive]*/, 53716 /*[TestVariable]*/,
true);
```

Data Format



Since version 4.0 of IP-Symcon CSV files are used for logging.
Before, an SQL database was used.

The archive saves two different types of values.
Primaryly, all raw data is stored.
Secondaryly, the aggregated data is calculated and stored for different timespans (year, month, week, day, hour)

CSV STRUCTURE

all files include the VariableID in their name and end with .csv
Raw data is written into individual folders per year and month
Aggregation data is stored in the db mainfolder
Files are only created when data is available
The decimal seperator is always a dot (.)

CONTENT OF RAW DATA

One set of data per row.
Values are seperated by comma (,).
Unix Timestamp
Raw Data



The logged values are always stored in UTC, independent of the configured server time.



String values are stored Base64 encoded.

FOLDER STRUCTURE OF RAW DATA

db\2010\01\12345.csv
db\2010\01\23456.csv
db\2010\02\12345.csv
db\2010\02\23456.csv

CONTENT OF AGGREGATION DATA

One set of data per row.
Values are seperated by comma (,).
  Unix Timestamp for Avg value
  Avg value
  Delta to the Unix Timestamp for Min value
  Min value
  Delta to the Unix Timestamp for Max value
  Max value

FOLDER STRUCTURE OF AGGREGATION DATA

  db\12345.year.csv
  db\12345.month.csv
  db\12345.week.csv
  db\12345.day.csv
  db\12345.hour.csv

Permission Control

Require: Symcon >= 8.0

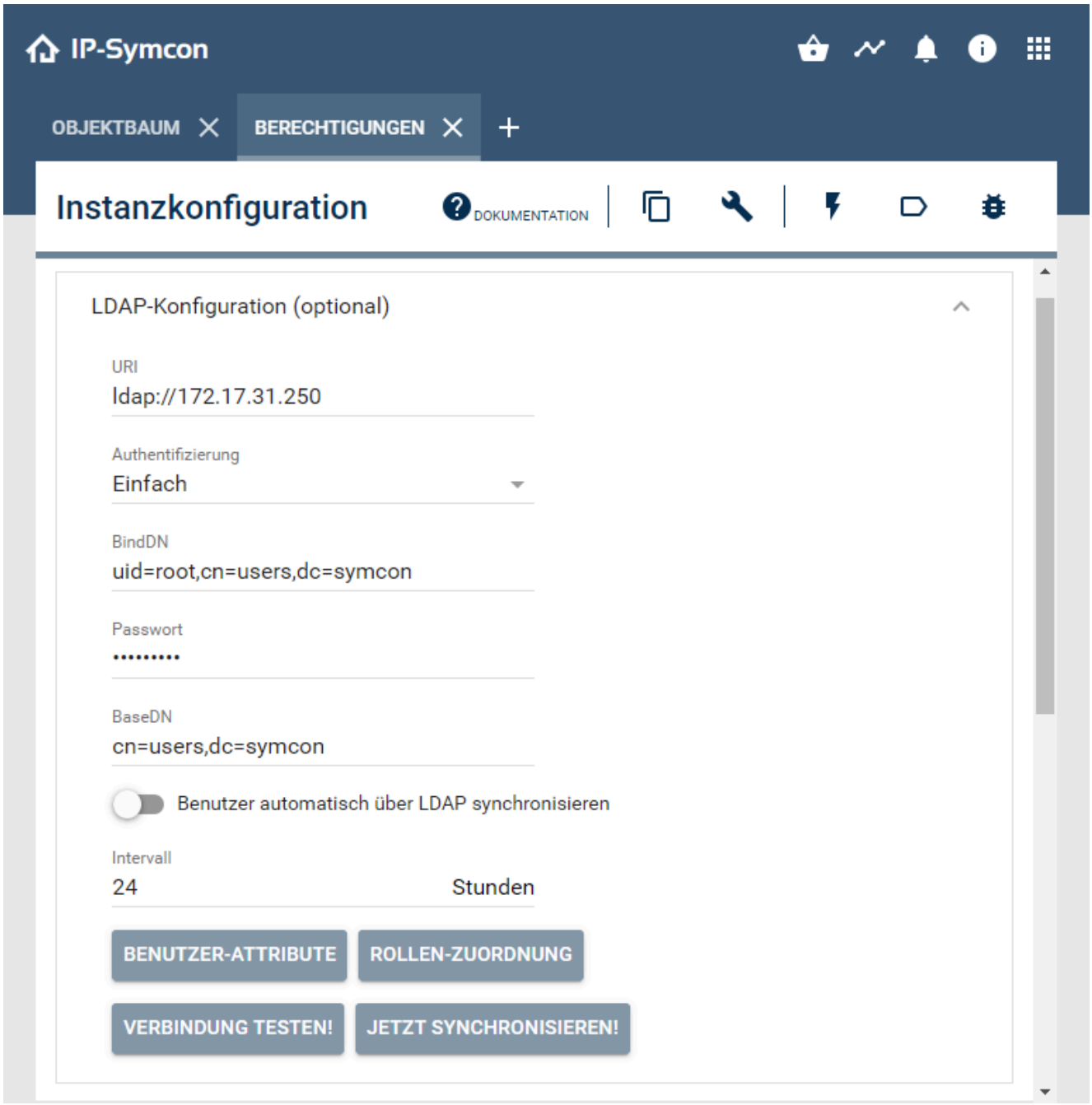


The authorization control is a paid extension, which can be purchased for every existing Symcon license. For a suitable demo version, please contact our Support. The extension can be purchased directly in the Shop.

The authorization control makes it possible to create users and roles and assign certain authorizations to them. Users can also be synchronized with an existing LDAP authentication server.

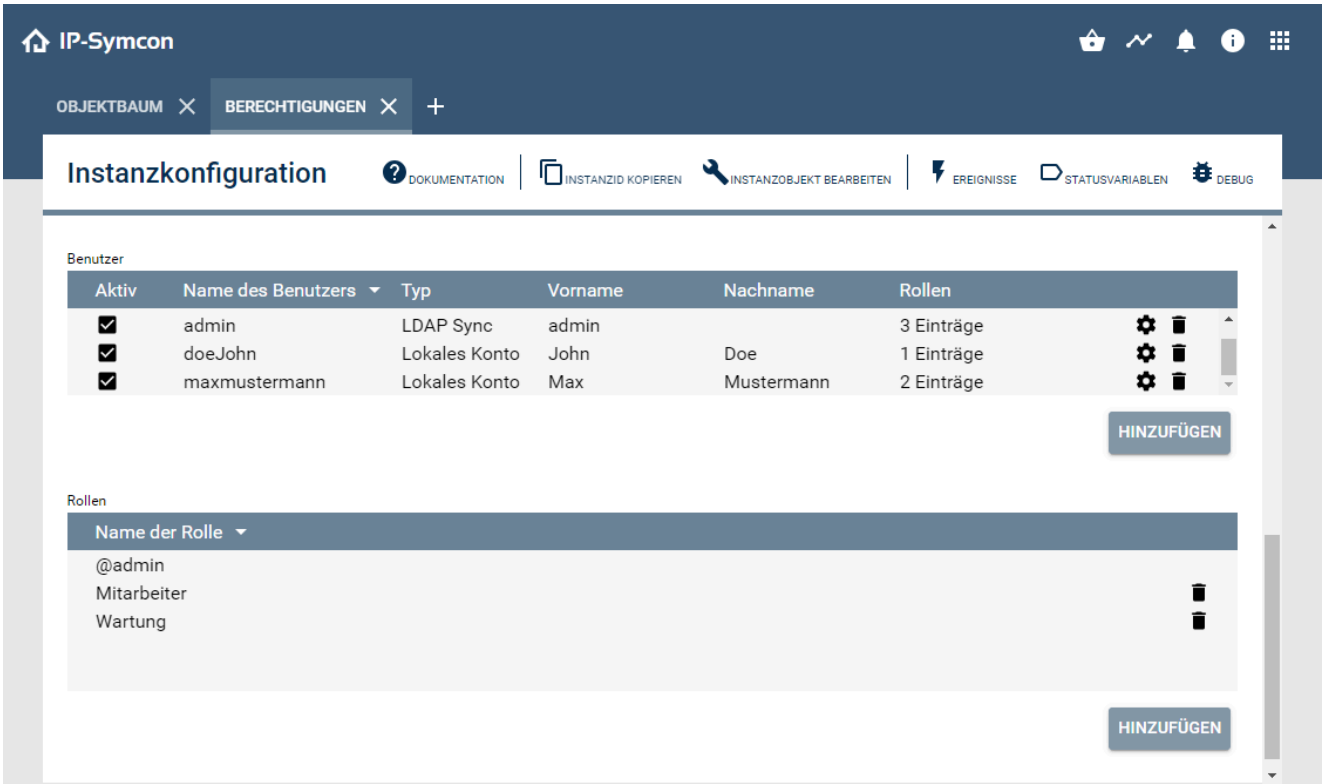
LDAP CONFIGURATION (OPTIONAL)

Once all the data for the connection has been entered, the user and role lists can be filled with the equivalents set up with LDAP by clicking on "Synchronize now". If automatic synchronization is activated, users and roles are synchronized at the specified interval.



USERS

By default there is the @admin user. This user cannot be edited and has all authorizations. The password for remote access is used to log in with this user. The permissions for accessing a visualization, for example, can be defined in the corresponding instance. If necessary, a user can be deactivated without deleting them directly.

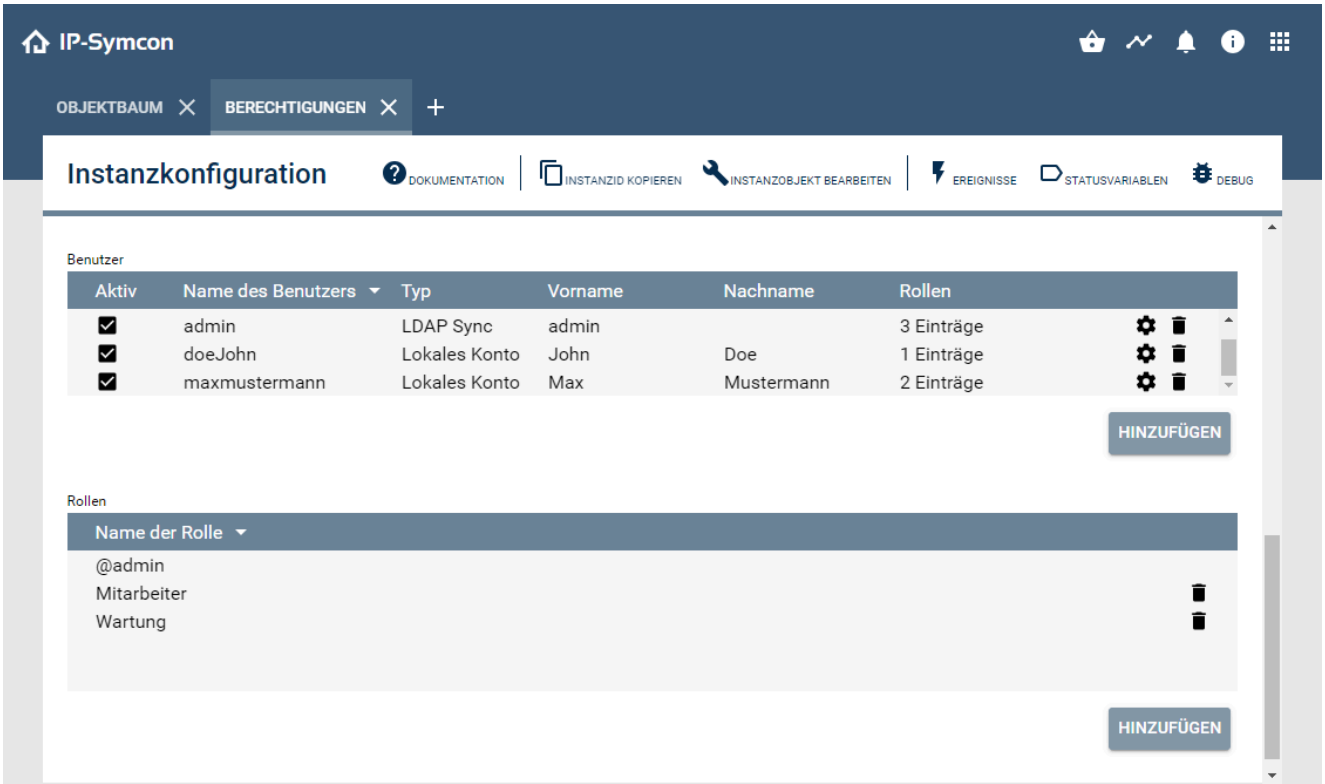


ADD USER

When adding a user, a first name and surname can be entered in addition to the password. These are displayed in the visualization, among other things. When creating, you can choose between the account types "Local account" and "LDAP Sync". If "LDAP Sync" is selected, the UserDN must be entered instead of a password. All desired roles can be selected in the list.

ROLES

By default there is the @admin role. This cannot be edited and has all authorizations. Created roles can be added to the created users. The permissions for accessing a visualization, for example, can be defined in the corresponding instance.



ADD ROLES

Each role can be given a name when it is created.

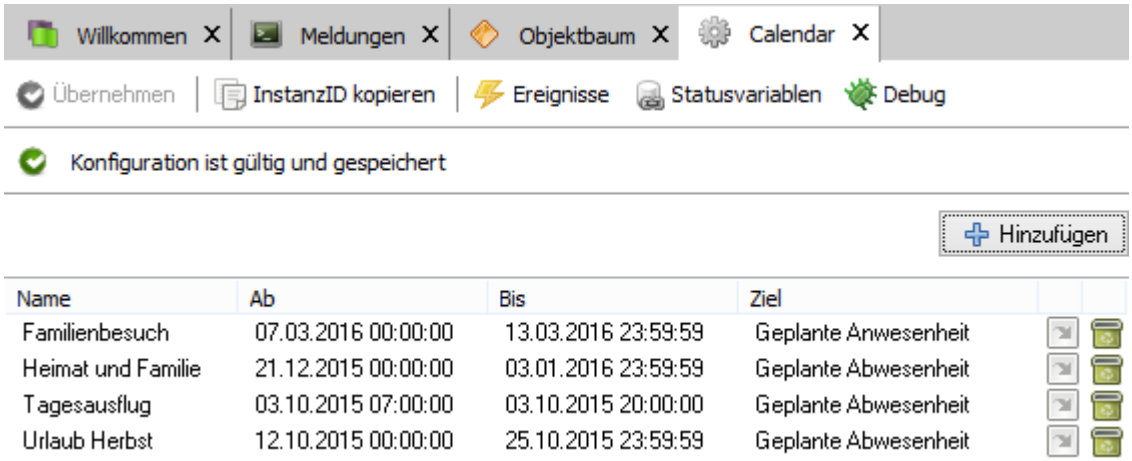
Calendar Control

Require: Symcon >= 4.0

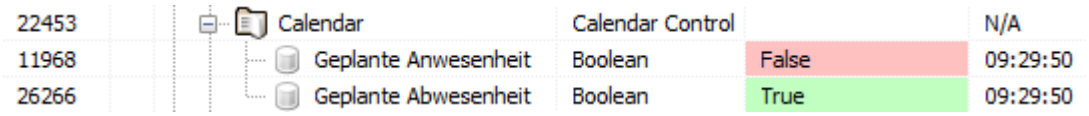
DESCRIPTION

The “Calendar Control” module offers the possibility to define periods in which one is expecting to be present or absent.

INTEGRATION IN IP-SYMCON



In the “Calendar” instance (“Logical tree view” -> “Core instances”), new time periods can be added via the configuration page.
During these periods, the appropriate variable is switched to “True”. If both variables are set to false, no time period is defined for the current time.



EXAMPLES

Planned presence

A family visit is recorded as a planned presence. At times when normally no one is in the home, the heating is automatically reduced to save energy. By taking into account the variable “planned presence”, this can be prevented without changing or manually switching off established scripts or weekly schedules.

Planned absence

The family vacation to the Baltic Sea is recorded as a planned absence. Due to the planned absence, an energy-saving automation system can be permanently activated for the period of time that would normally only be active at certain times in everyday life.

Connect Control

Require: Symcon >= 4.0

DESCRIPTION

The Connect Control module offers the possibility to securely access the system from the outside. This connection can be used to manage the system from anywhere via the management console or to access it via the visualizations and mobile apps.
The measures mentioned under Security must always be taken in order to prevent unwanted access from the outside after activating the Connect module.
The module works completely without port sharing or setting up a DynDNS. A URL is automatically provided by our server. Thanks to secure tunneling, no special firewall settings are required and access within a company network is also possible.

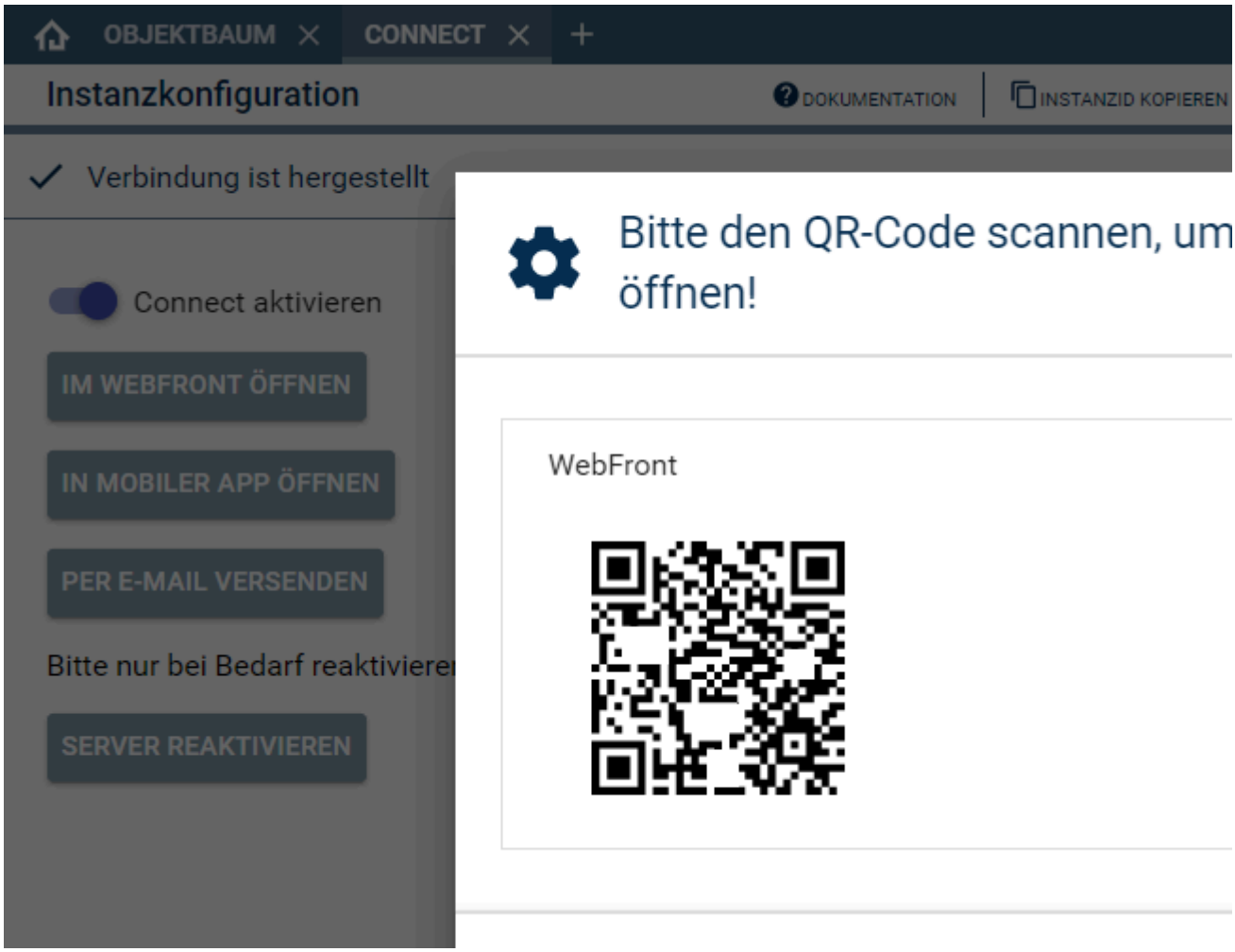


In the event of an error, further information can be viewed in the Notification Window. Further information can be found under Solutions in Case of an Error



INTEGRATION IN IP-SYMCON

The prerequisite for using the Symcon Connect service is an active subscription. A maximum of one active server connection is possible per license



In the instance "Connect" ("Logical tree view" -> "Core instances") the module can be activated via "Activate Symcon Connect".
The following 3 access methods are available:

"Open WebFront"

This opens the personal URL in the web browser, which has the following format (http://32-figure-lettersequence.ipmagic.de). The WebFront can now be accessed from anywhere using the address shown in the address bar. This can be entered in any browser and on any smartphone. An encrypted connection is then established.



It is recommended to secure the WebFront with a password.

"Open on smartphone"

The address can be read in on mobile devices via QR code. (Any QR code reading software is required for this. As of iOS11, scanning can be carried out directly via the integrated camera app.) In the QR code, the link is formatted in the form of a WebFront Share Link and thus automatically opens the required IP-Symcon app, including the selected WebFront.



Some QR code reading software may have issues with the created links. In this case, try using another QR code software.

"Send via email"
The default e-mail program will open automatically and it is only required to enter the recipient's address.
This way, the link and the associated IP-Symcon can be made available to third parties.



In order to be able to access the Management Console via the Connect module, Remote Access must be activated and a password must be set. An encrypted connection may now be established and used via an installed Management Console and URL input.

LIMITATIONS

DATA VOLUME

The total amount of data Connect Control can send each day is 1GB.

DAILY RECONNECTS

By default, 25 reconnects to the Connect service are possible per day. These can be increased to 100 using the Special switches (ConnectLimit).

SOLUTIONS IN CASE OF AN ERROR

In connection with the Connect Control, a few settings must be minded as they could lead to errors. In the event of an error, instructions are given in the Notification Window as to where an error has occurred.

STATUS

The current server status of Connect Control can be seen on this Status Page .

RESTART THE SYSTEM

If an unforeseeable connection error has occurred, restarting the service via the tray icon or reactivating the server may help. Information on this may be found under Reactivate Server .

ADJUSTMENTS IN THE FIREWALL

In order to grant IP-Symcon access to certain remote functions (e.g. Connect Control) it may be necessary to set them up in the firewall. Further information can be found under Firewall.

REACTIVATE THE SERVER

For additional security, the sending server authenticates itself using a token so that it can be clearly identified as the sender. If the server is changed, replaced, or reinstalled, reactivation may be required. This reactivation can be done every 24 hours and started using the "Reactivate server" button. You will be notified about this change by email. This is done in order to detect misuse of a license at an early stage.

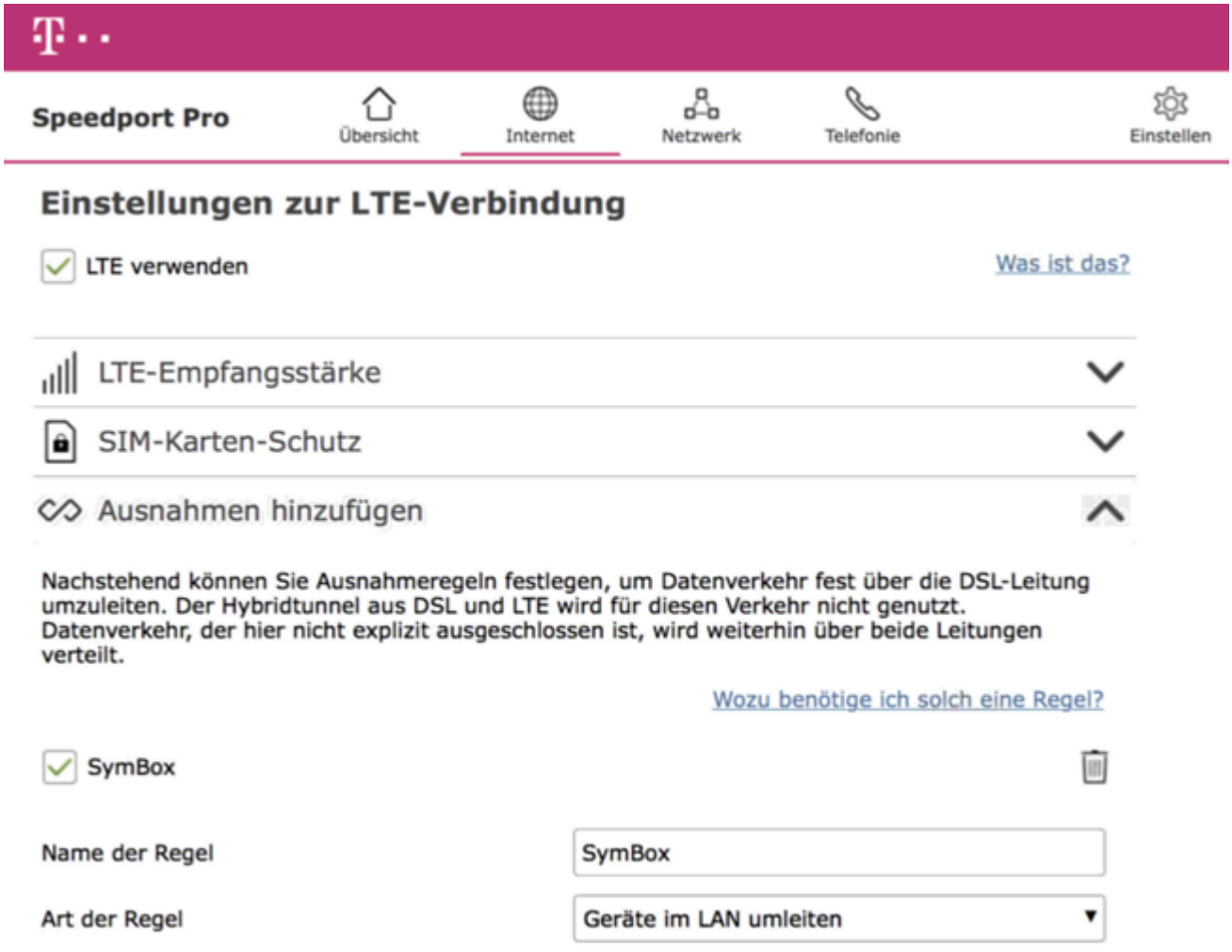
A reactivation is required if the following error message appears: Your server authentication token has changed. Please reactivate your server.

FIREWALL SETUP

Access to live.symcon.de (443/TCP) and ipmagic.de (50000/TCP, 60000/TCP) must be enabled. The TTL (Time To Live) for DNS requests must be equal to or less than 60 seconds, as AWS (Amazon Web Services) changes these every minute.

TELEKOM WITH DSL AND LTE

For Telekom customers using a hybrid DSL+LTE solution, an exception should be created for the SymBox or the IP-Symcon service so that it only communicates via DSL.



Cutter

The cutter is used to extract sections from an incoming data stream using separators or a fixed number of steps and forwards them.

INTEGRATION IN IP-SYMCON

The cutter acts as a splitter instance and is therefore often between the I/O and device instance.
Thus the cutter takes apart the incoming data stream and passes it on.

USE CHARACTERS FOR CUTS

This setting sets separators from left (start of data stream) to right (end of data stream). Everything that is between the separators is passed on. Optionally, hex characters can be switched on and thus the separators can be entered in hexadecimal. If only the left or right separator is entered, the entire data stream is forwarded before (only set on the right) or after (only set on the left).
In the drop-down menu there are standard control commands, which are often used and forward the 1:1.



The separators themselves are not passed on. These are also cut out so that only the pure user data is forwarded.

USE FIXED CUTS

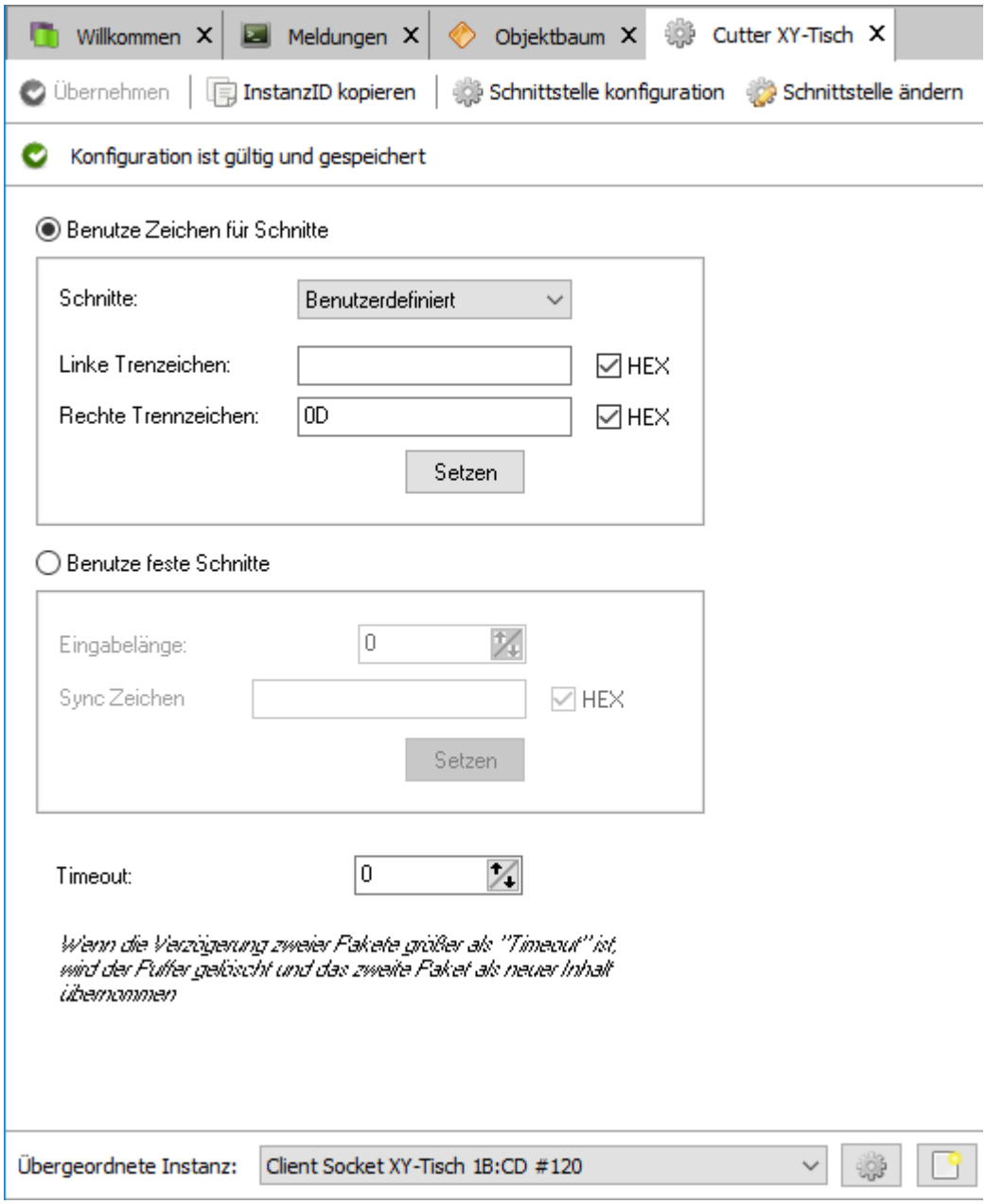
A search is done for the sync character in the data stream. The subsequent data stream (including sync characters) is cut off according to the set input length and forwarded. If the input length = 0, the entire data stream including sync characters is forwarded.

TRANSFER TO VARIABLE

The data stream can be transferred to a variable via a RegisterVariable (using a script).

SPECIAL OPTION TIMEOUT

If the delay between two data stream packets is greater than the set "timeout" time, the two packets are not treated as a coherent data stream, but rather as two separate data packets.


Willkommen
Meldungen
Objektbaum
Cutter XY-Tisch
Übernehmen
InstanzID kopieren
Schnittstelle konfiguration
Schnittstelle ändern
Konfiguration ist gültig und gespeichert
Benutze Zeichen für Schnitte
Schnitte:
Benutzerdefiniert
Linke Trenzeichen:
HEX
Rechte Trennzeichen:
0D
HEX
Setzen
Benutze feste Schnitte
Eingabelänge:
0
Sync Zeichen
HEX
Setzen
Timeout:
0
Wenn die Verzögerung zweier Pakete größer als "Timeout" ist, wird der Puffer gelöscht und das zweite Paket als neuer Inhalt übernommen
Übergeordnete Instanz:
Client Socket XY-Tisch 1B:CD #120

Cutter_ClearBuffer

```
bool Cutter_ClearBuffer (int $InstanceID)
```

PARAMETERS

InstanceID	ID to the cutter

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function clears the buffer of the cutter instance with the ID InstanceID.

EXAMPLE

```
// Cleans the buffer of the clutter instance with the ID 12345
Cutter_ClearBuffer(12345);
```

DNS-SD Control

Require: Symcon >= 5.0
The DNS-SD control module manages established Bonjour services.

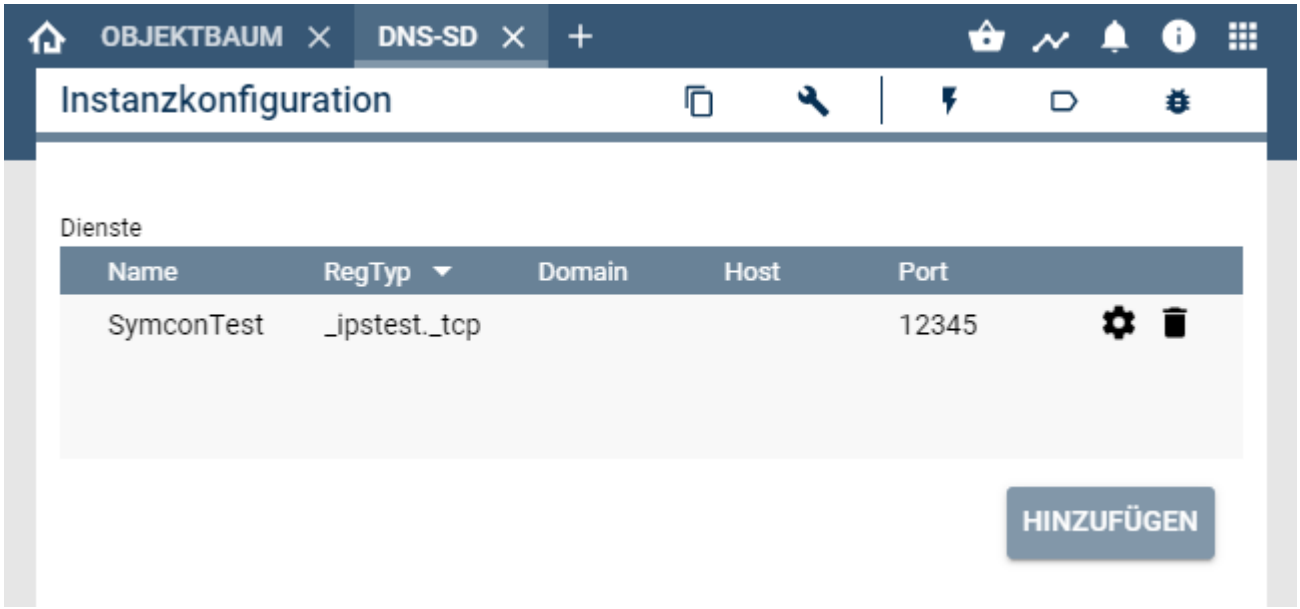
For further information one can search for Zeroconf or mDNS.

INSTALLATION

The DNS-SD Control is automatically installed as a core instance. This can be found in the object tree under core instances.

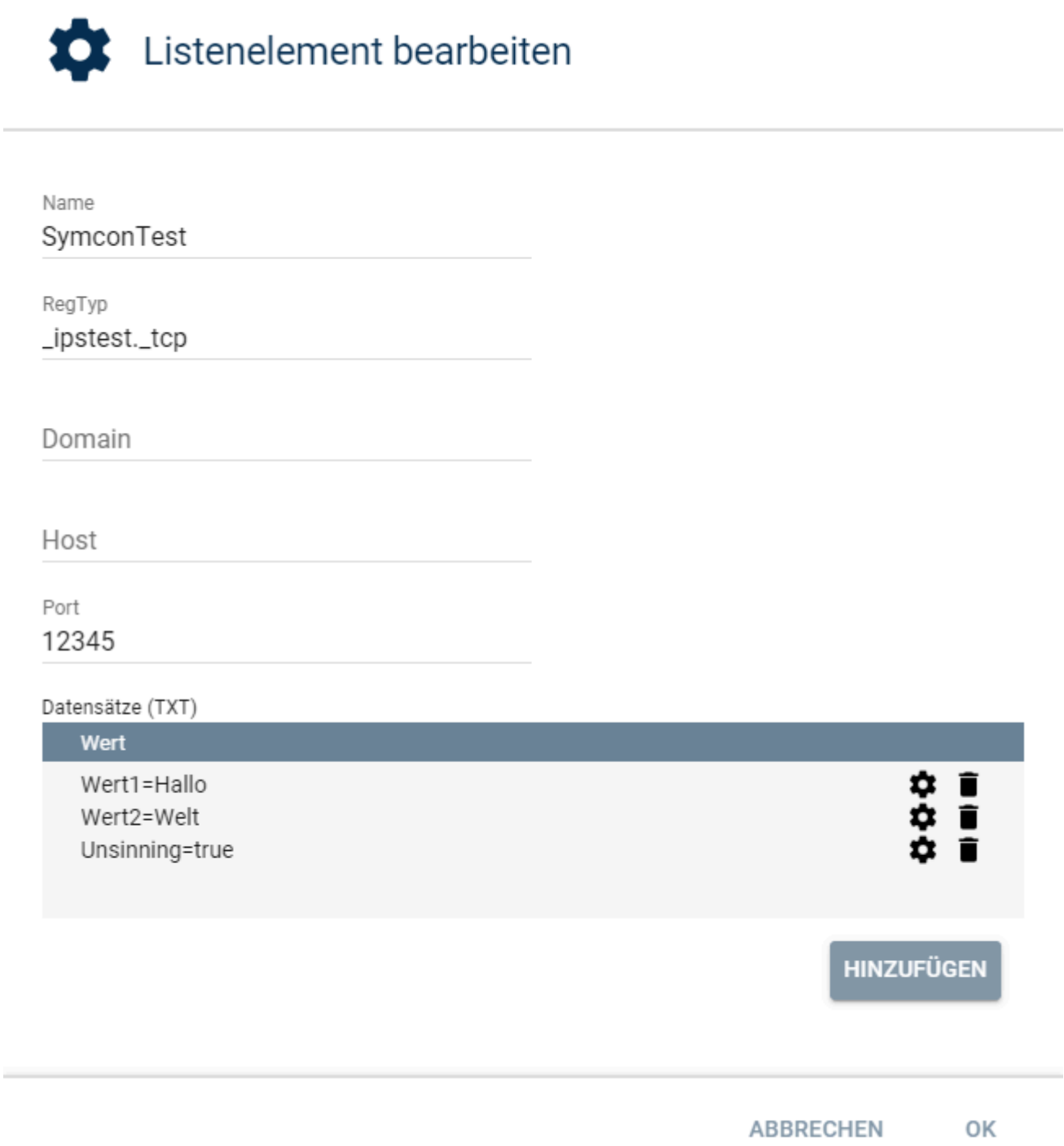
CONFIGURATION

New services (list element) can be added via the configuration page of the DNS-SD Control.

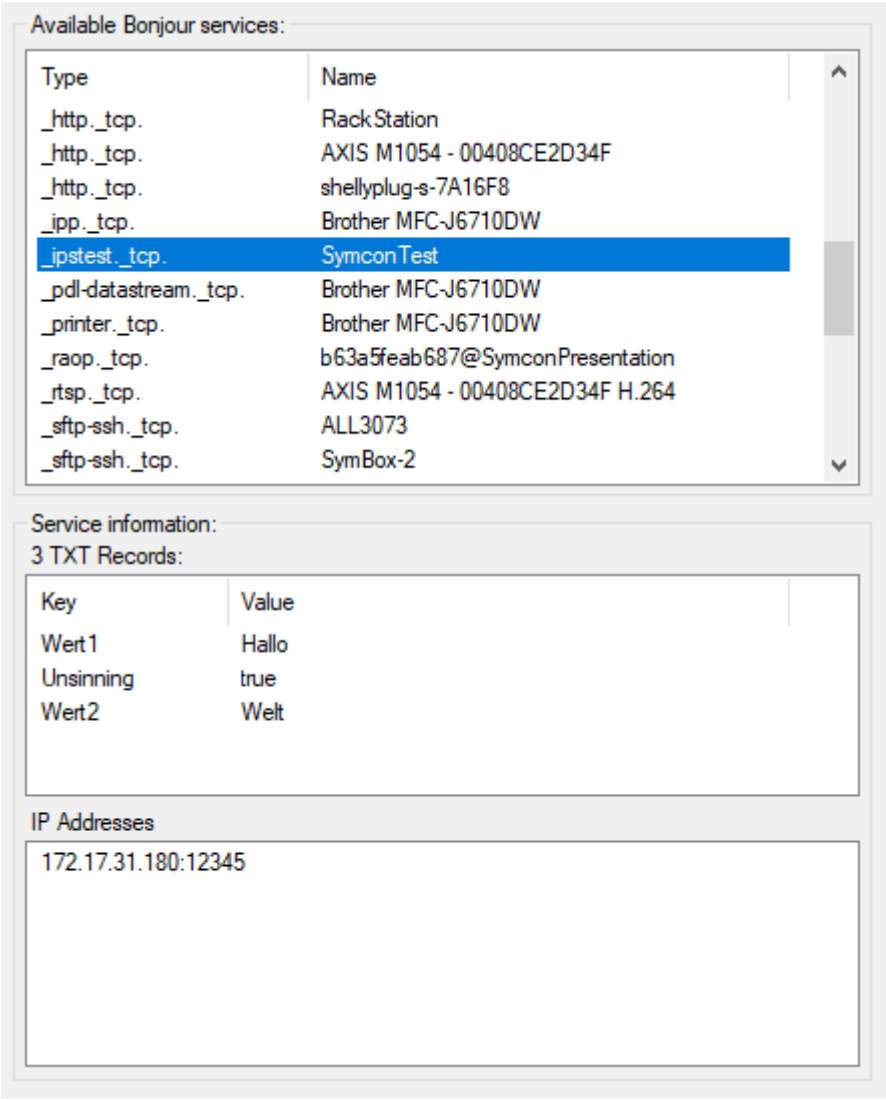


These need a Name, RegType and Port. Domain and Host are optional, as are values. These are saved as key -> value pairs.

The same Name and RegType may not be reused for different list elements.



After successfully adding, the new RegType and Name can be found and used directly via a Bonjour browser.


Available Bonjour services:
Type
Name
_http._tcp.
RackStation
_http._tcp.
AXIS M1054 - 00408CE2D34F
_http._tcp.
shellyplug-s-7A16F8
_ipp._tcp.
Brother MFC-J6710DW
_ipstest._tcp.
SymconTest
_pdl-datastream._tcp.
Brother MFC-J6710DW
_printer._tcp.
Brother MFC-J6710DW
_raop._tcp.
b63a5feab687@SymconPresentation
_rtsp._tcp.
AXIS M1054 - 00408CE2D34F H.264
_sftp-ssh._tcp.
ALL3073
_sftp-ssh._tcp.
SymBox-2
Service information:
3 TXT Records:
Key
Value
Wert1
Hallo
Unsinning
true
Wert2
Welt
IP Addresses
172.17.31.180:12345

ZC_QueryService

Require: Symcon >= 5.3

```
array ZC_QueryService (int $InstanceID, string $Name, string $RegType,
string $Domain)
```

PARAMETERS

InstanceID	ID of the DNS-SD control to be requested
Name	Name of the service
RegType	The RegType of the service
Domain	Domain of the service

RETURNS

Returns all information about the service found as an array.

DESCRIPTION

Queries the service of the DNS-SD control with the ID **InstanceID** and the appropriate name **Name**, RegType **RegType** and the domain **Domain**.



The domain must have a valid value and must not be empty

EXAMPLE

```
ZC_QueryService(12345, "NAS", "_http._tcp", "local.");

// Sample output:
/*
Array
(
[0] => Array
(
[Name] => NAS._http._tcp.local.
[Host] => NAS.local.
[Port] => 5000
[TXTRecords] => Array
```

```
(
[0] => vendor=Synology
[1] => model=RS818+
[2] => serial=xxxxx
[3] => version_major=6
[4] => version_minor=2
[5] => version_build=24922
[6] => admin_port=5000
[7] => secure_admin_port=5001
[8] => mac_address=xxxxxx
)

[IPv4] => Array
(
[0] => 192.168.1.105
)

[IPv6] => Array
(
[0] => fe60::cc4b:5ccf:fac7:5d82
)
)
)
*/
```

ZC_QueryServiceType

Require: Symcon >= 5.3

```
array ZC_QueryServiceType (int $InstanceID, string $RegType, string
$Domain (optional))
```

PARAMETERS

InstanceID	ID of the DNS-SD control to be requested
RegType	The RegType of the service
Domain (optional)	Domain of the service

RETURNS

Returns all found services as an array.

DESCRIPTION

Queries the services of the DNS-SD control with the ID **InstanceID** and the appropriate RegType **RegType** and the domain **Domain**.



The Domain is optional and can contain an empty string

EXAMPLE

```
ZC_QueryServiceType(12345, "_http._tcp", "");

// Sample output
/*
Array
(
[0] => Array
(
[Name] => NAS
[Type] => _http._tcp.
[Domain] => local.
)
```

```
)
*/
```

ZC_QueryServiceTypes

Require: Symcon >= 5.3

```
array ZC_QueryServiceTypes (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the DNS-SD control to be requested

RETURNS

Returns all services and information as an array.

DESCRIPTION

Queries the services of the DNS-SD control with the ID **InstanceID** and returns this as an array.

EXAMPLE

```
// Request the registered services of the DNS-SD Control with the ID
12345
ZC_QueryServiceTypes(12345);

// Sample output
/*
Array
(
[0] => Array
(
[Name] => _afpovertcp
[Type] => _tcp.local.
[Domain] => .
)


[1] => Array
(
[Name] => _airplay
[Type] => _tcp.local.
[Domain] => .
)
```

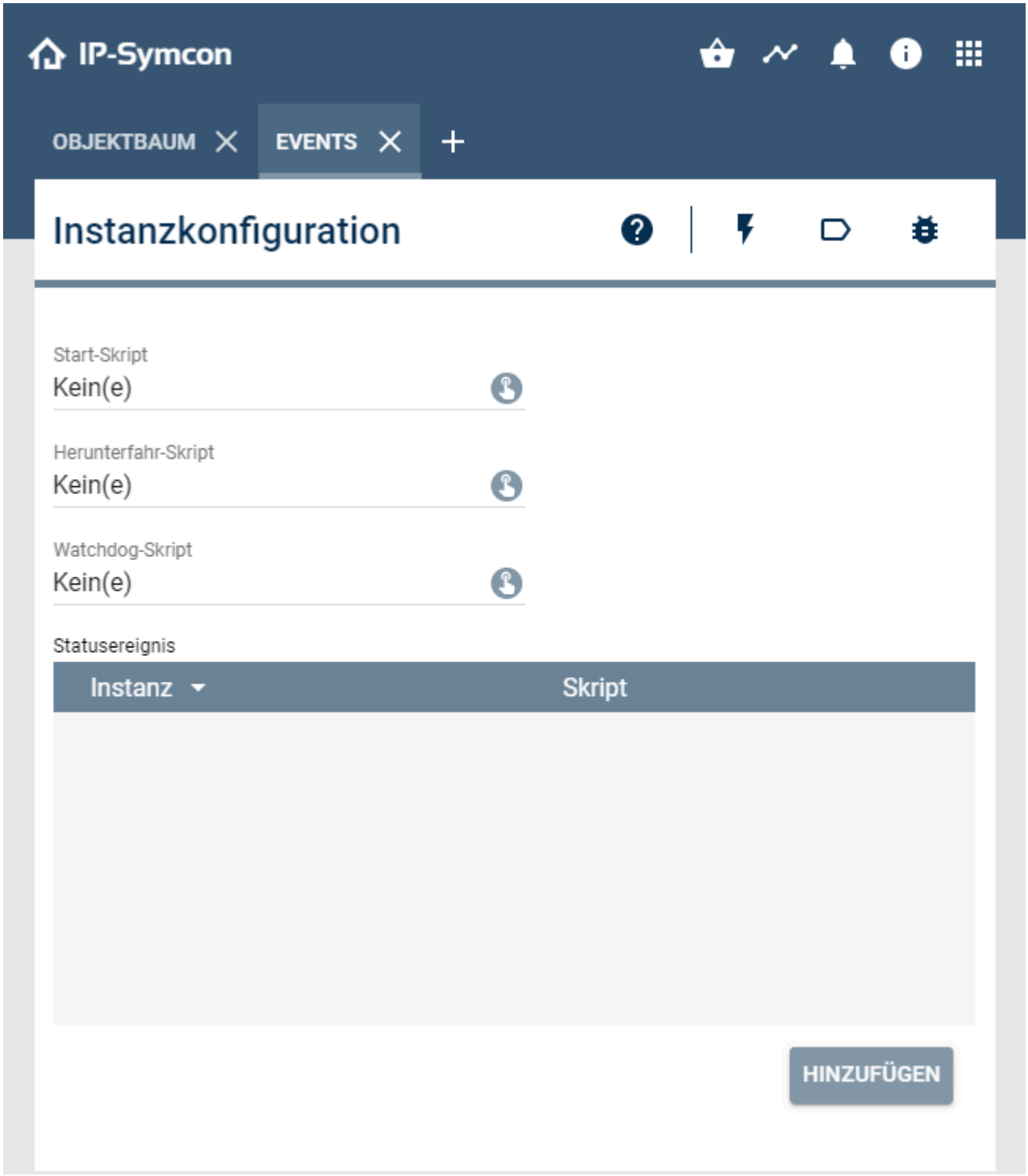
```
[2] => Array
(
[Name] => _airtame
[Type] => _tcp.local.
[Domain] => .
)


[3] => Array
(
[Name] => _apple-mobdev2
[Type] => _tcp.local.
[Domain] => .
)


[4] => Array
(
[Name] => _axis-video
[Type] => _tcp.local.
[Domain] => .
)
)
*/
```

Event Control

This module allows the user to react to certain events in IP-Symcon and to start a script. It is created automatically and is available to the user at any time. It can be found in the object tree at the following location:
Core Instances -> Events



The following events are available:

EVENT	DESCRIPTION
At startup (start-script)	After IP-Symcon has started, this event is called. At this point in time, all interfaces should already be available and all devices should be able to be switched. The following system variables are available within the start-script.
When shutting down (shutdown-script)	This event is called just before IP-Symcon shuts down. At this point, all devices should still be switchable. The following system variables are available within the shutdown script.
Profile border crossing (watchdog-script)	This event is called as soon as a variable (integer/float only) comes outside the limit range of the associated variable profile. A variable must have a profile to be monitored. The following system variables are available within the watchdog script.
On status update (status event)	When the status of an instance updates, a specified script can act on that update. The following system variables are available within the status event script.



This module should not be deleted.
If it is deleted, it will be recreated on the next start.

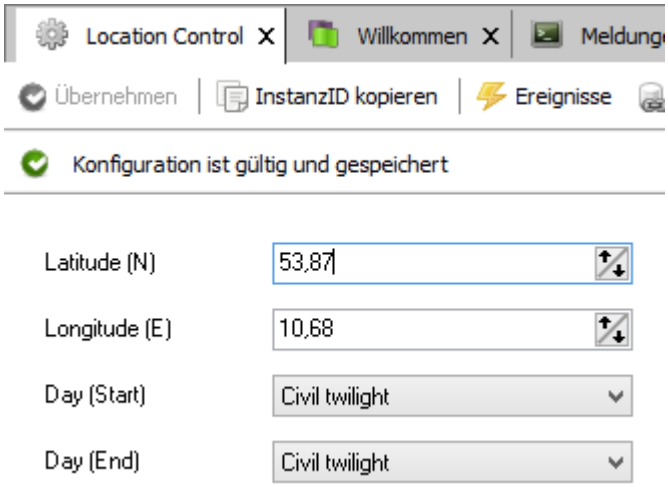
Location Control

Require: Symcon >= 4.0

DESCRIPTION

The location control module offers the possibility to calculate twilight times. This simply requires the longitude and latitude of the location to be calculated.

INTEGRATION IN IP-SYMCON



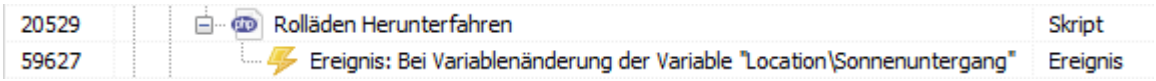
In the “Location” instance (“Logical Tree View” -> “Core Instances”), the longitude N and latitude E can be entered to exactly 2 decimal places. A method for determining latitude and longitude can be found under Tips & Tricks .
Thereby civil, nautical and astronomical twilight values for sunrise and sunset are calculated. The two selection menus start of day and end of day form the boundary between which the boolean variable “Is it day” is set to true.

53993	Location Control	Location Control		N/A
18456	Ist es Tag	Boolean	True	15:15:42
30060	Sonnenaufgang	Integer	28.08.2015 06:18:15	15:15:42
58785	Sonnenuntergang	Integer	27.08.2015 20:18:58	15:15:42
45869	Ziviler Dämmerungsbeginn	Integer	28.08.2015 05:38:39	15:15:42
52072	Ziviles Dämmerungsende	Integer	27.08.2015 20:59:09	15:15:42
10583	Nautischer Dämmerungsbeginn	Integer	28.08.2015 04:50:56	15:15:42
50634	Nautisches Dämmerungsende	Integer	27.08.2015 21:46:51	15:15:42
37357	Astronomischer Dämmerungsstart	Integer	28.08.2015 03:55:22	15:15:42
14038	Astronomisches Dämmerungsende	Integer	27.08.2015 22:42:25	15:15:42

The twilight values also determine the time of recalculation. This entails that when the time for sunrise is reached, the new value for the next sunrise is calculated.

EXAMPLE

An example would be a script to move down the shutters. This is started automatically when it is executed by an event that reacts to the variable change of “Sunset”.



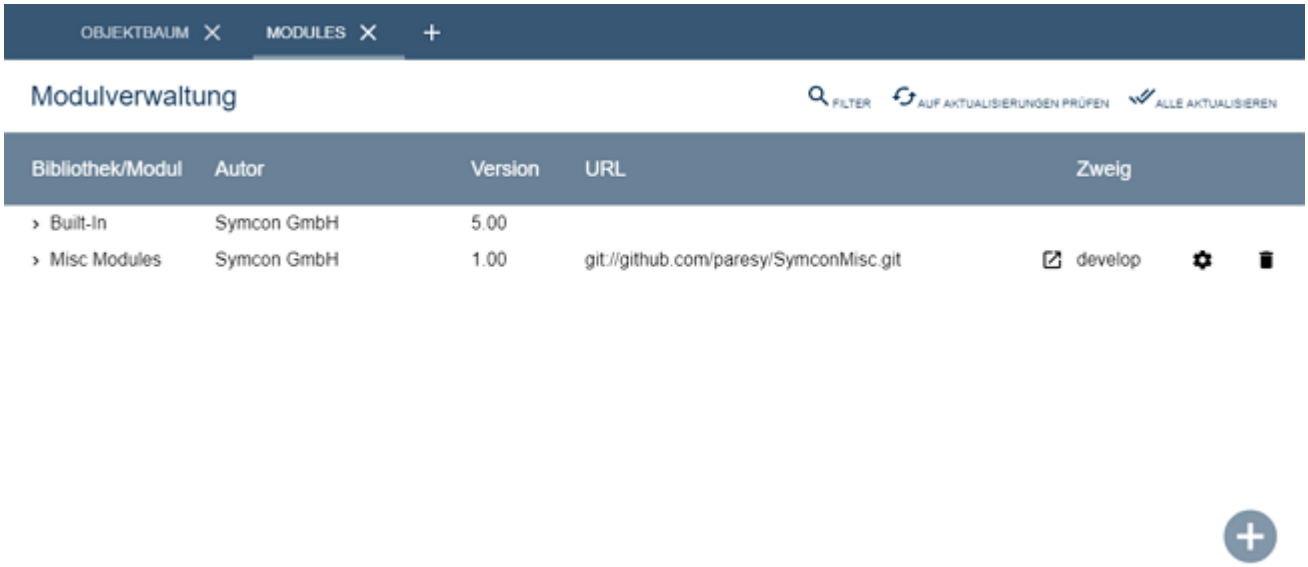
Module Control

Require: Symcon >= 4.0

DESCRIPTION

The "Module Control" module offers the possibility to integrate own PHP modules or ones provided by other users.

INTEGRATION IN IP-SYMCON



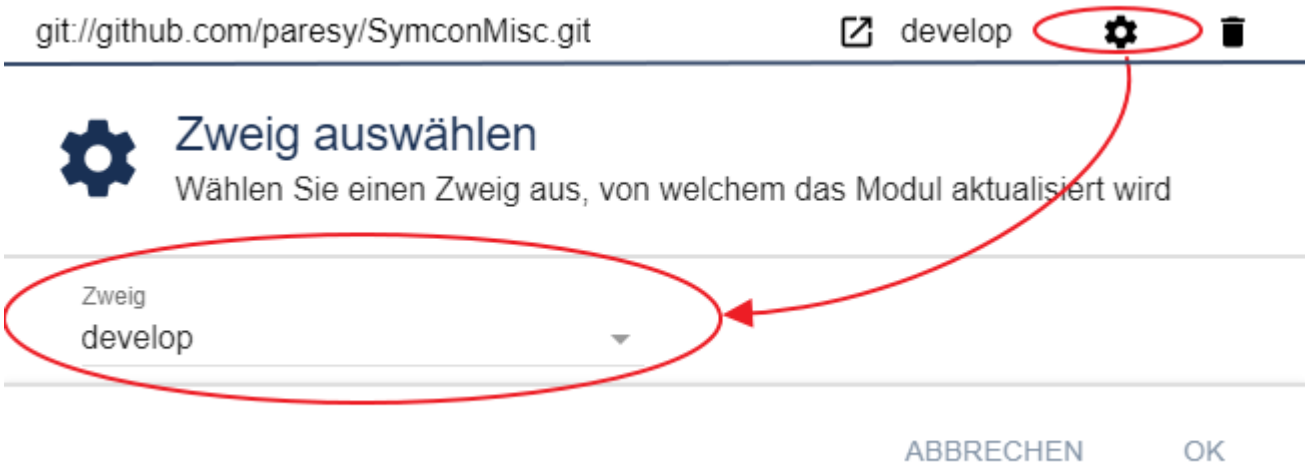
In the "Modules" instance ("Logical tree view"->"Core instances"), a Git-Repository link/URL can be entered via "+". The module is then installed via this and is immediately available in the "Add instance" dialog. Which link is needed for which module can usually be seen in the documentation or forum entry of the module. If the data is incomplete, the respective module developer can be contacted via the forum using a private message.

UPDATE

The "Check for updates" button checks whether there is an update for modules that have already been added. If an update is possible, the respective module can be updated via the appearing update button. If a module is up to date, a tick is displayed instead.

CHANGE BRANCH

The cogwheel next to the respective module allows to change the branch of the repository. What these are called depends on the naming by the repository creator.



DEVELOP INDIVIDUAL MODULES

In the developer area all the information necessary to develop individual modules is available.
Further information can be found directly in the SDK for PHP .

EXAMPLES

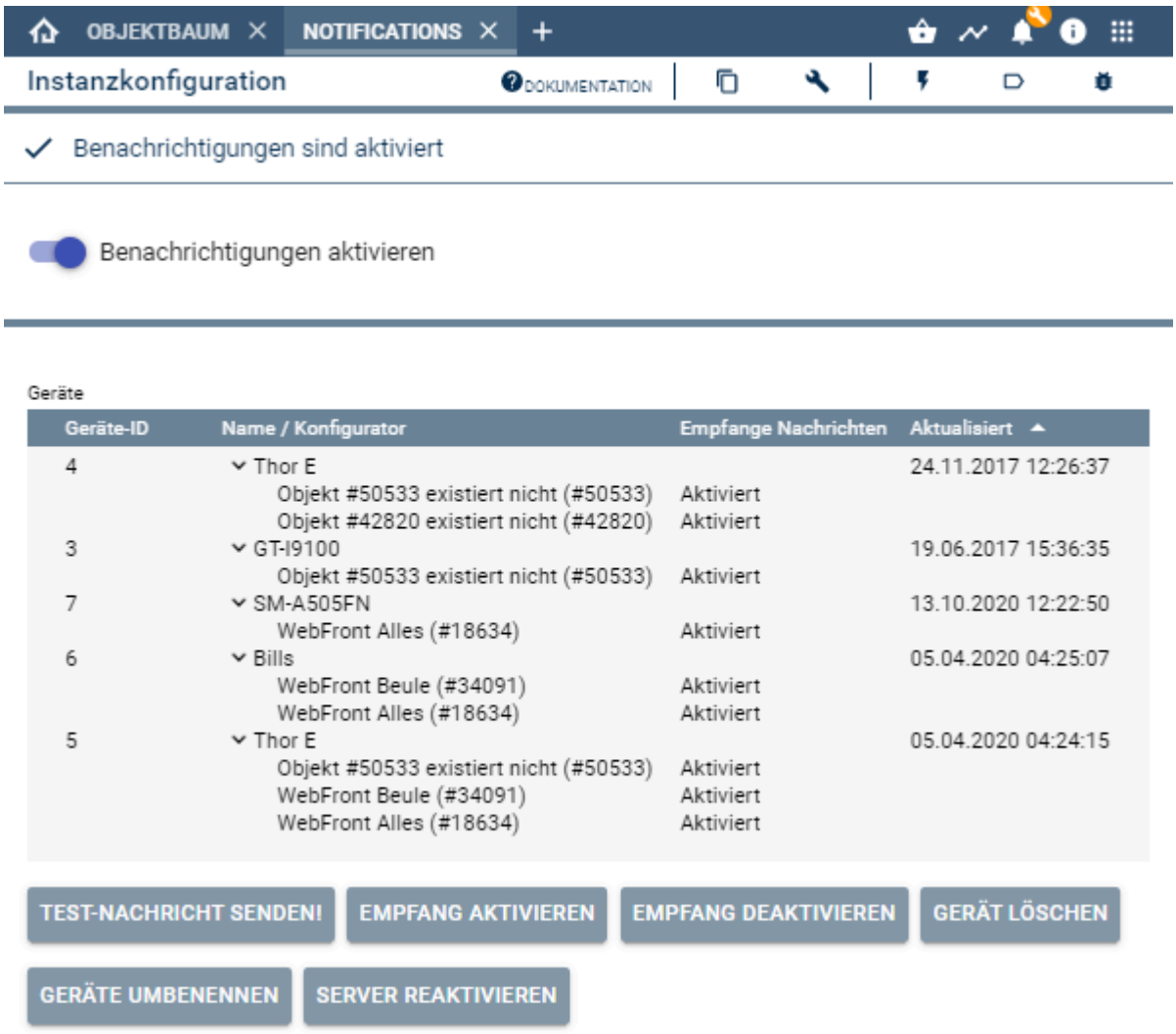
There is a link collection in the forum, which lists PHP modules developed by users.

Notification Control

The Notification Control module provides a quick overview of the devices currently registered for push notifications. In the list all devices are displayed with a unique ID, the changeable name and the assigned configurators. By selecting a device and clicking the "Send test message" button, a test message can be sent to the selected device.
Push notifications can be sent via WFC_PushNotification.

Per license only one IP-Symcon server can send push notifications. Further information can be found in the Rights of Use section.

In order to grant IP-Symcon access to certain functions (e.g. push notifications), it may be necessary to set them up in the firewall. Further information can be found under Firewall.



ADD DEVICES

New devices can only be added via the mobile apps. To do this, the configuration of the server must be carried out again within the respective app. In the last step, when selecting the configurator, the option of activating/deactivating the notifications for each configurator prevails. As soon as the device has been registered on the server, it appears in the registered configurator and in the notification control overview.

RENAME DEVICES

The name of a device can be changed at any time. To do this, the small arrow icon in the list must be clicked.

REMOVE DEVICES

In the Notification Control, devices can be removed from the notification completely. If devices are to be removed from an individual configurator, this can be set in the respective configurator under the "Notifications" tab.

REACTIVATE SERVER

The push notifications are sent via the IP-Symcon. For additional security, the sending server authenticates itself so that it can be clearly identified as the sender. If the server is changed, replaced, or reinstalled, reactivation may be required. This reactivation can be done every 24 hours and started using the "Reactivate server" button. You will be notified about this change by email. This is done in order to detect misuse of a license at an early stage.

DAILY PUSH NOTIFICATIONS

As a standard, 250 push messages from Notification Control are possible per day. These can be increased to 1000 using the special switch (NotificationLimit).



A reactivation is required if the following error message appears: “Your server authentication token has changed. Please reactivate your server.”

Presence Control

Require: Symcon >= 5.1
The "Presence Control" offers the possibility to evaluate presence within an area. For this evaluation, sensors are diveded into two groups. Impulse variables trigger a signal on activation which confirms presence for a specific duration. The other group are state variables that evaluate presence by their current state.
The status variable "Presence" displays the current presence. It is set if at least one variable confirms presence within the area. Alternatively, the presence can be set manually. In this case, the set presence is held for a lock time. During this time, updates of impulse and state variables are not evaluated. The status variable "Detection" informs whether the "Presence" variable is currently locked by the lock time after a manual setting or is active, i.e., is evaluated by impulse and state variables.

IMPULSE VARIABLES

OPTION	DESCRIPTION
Variable	Boolean variable that contains the impulse
Validity	Validity in seconds - An impulse on the variable confirms presence for this duration
Invert?	By default, the module reacts to variable updates to true. If "Invert?" is set, the module reacts to false instead

If an impulse variable is updated to true (or false, if "Invert?" is set), the presence is confirmed for the validity duration.



Changes to the validity or removing an impulse variable does not affect current pulses. Current impulses use the settings from the time the impulse was triggered.

STATE VARIABLES

OPTION	DESCRIPTION
Variable	Boolean variable that contains the state
Invert?	By default, presence is confirmed while the variable is true. If "Invert?" is set, the module reacts to false instead

If a state variable is true (or false, if "Invert?" is set), the presence is confirmed.

LOCK TIME

The lock time in seconds after a manual adjustment to presence can be defined here. During the lock time Presence stays at its set value. Impulses are ignored during the lock time. After the lock time, Presence is evaluated as usual.

A change to the lock time is only applied at the next manual adjustment. Current lock times use the settings from the time of manual adjustment.

PC_Enter

Require: Symcon >= 5.1

```
bool PC_Enter (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Presence-Control-Instance

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function sets the presence variable of the instance with ID InstanceID to true.
In addition, the presence is fixed for the dead-time set in the module and is not changed by status or pulse variables. During the dead-time, pulses on pulse-variables and changes in state-variables are completely ignored. After the dead-time, the presence is initially set based on the state variable.

EXAMPLE

```
// Sets Presence-Control-Instance 12345 to "true"
PC_Enter(12345);
```

PC_Leave

Require: Symcon >= 5.1

```
bool PC_Leave (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Presence-Control-Instance

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function sets the presence variable of the instance with ID InstanceID to false.
In addition, the presence is fixed for the dead-time set in the module and is not changed by status or pulse variables. During the dead-time, pulses on pulse-variables and changes in state-variables are completely ignored. After the dead-time, the presence is initially set based on the state variable.

EXAMPLE

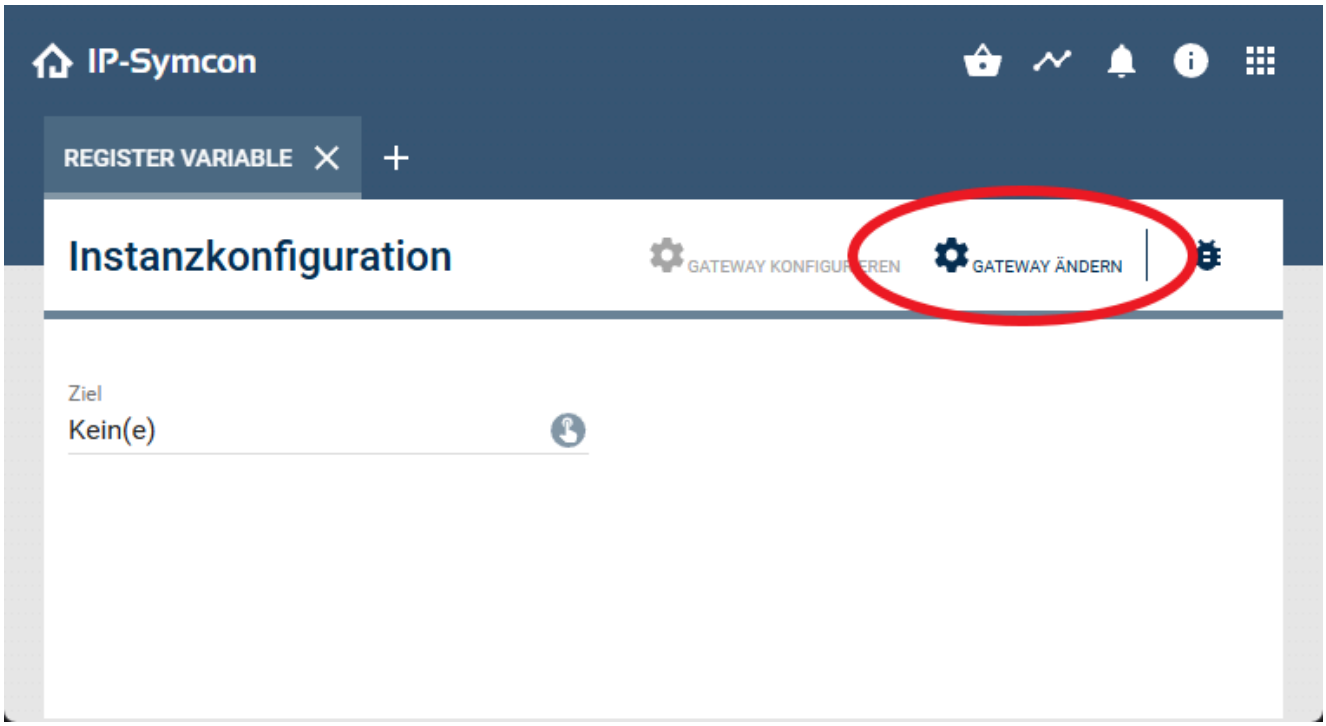
```
// Sets the presence of Presence-Control-Instance 12345 to "false"
PC_Leave(12345);
```

RegisterVariable

The RegisterVariable provides a data forwarding interface and data processing interface.

CONFIGURATION

If you have not set up your device, please follow the steps on this page: [embed devices] [2]
The RegisterVariable module connects automatically with a parent instance. You need to select parent instance or create a new one. You can also add another splitter (e.g. cutter) to synchronize the incoming data directly before evaluation.
The parent instance can be adjusted easily by using the function "Change Gateway" at the top of the configuration.



The execution of scripts is **serial**. It means that you do not have to protect your scripts with semaphores.

DATE PROCESSING

If the the parent instance (communication interface) receives new data, the target script of the RegisterVariable instance will be run. In this the received data over $ _IPS ['VALUE'] is available.
There are data sources that provide a meaningful evaluation after receiving several data transmissions. Therefore, the received data is temporarily stored. Problem can occur, if you store binary data in a normal string variable in IP-Symcon (defective IP-Symcon configuration). There is the function RegVar_SetBuffer (integer $InstanzID, string $buffer) that stores the data in the buffer associated to the RegisterVariable instance. The data stored in the buffer can be read with the function RegVar_GetBuffer (integer $InstanzID).
The function RegVar_SendText (integer $InstanzID, string $ text) can send data strings over the communication interface. Internally, the appropriate transmission function (e.g. COMPort_SendText) is executed.

The buffer is cleared after restarting IP-Symcon

Please DO NOT use VARIABLE as a buffer. Otherwise, it may cause instability within IP-Symcon and at worst cause a completely destroyed configuration. Since this problem can also affect the incremental backup folder, you may not even have a backup configuration. Take advantage of the internal buffer of the instance, to cache data.
RegVar_SetBuffer / RegVar_GetBuffer

EXAMPLE

The following example chaines received data and gives out by ; separated records at completion:

```
// wenn das Skript von einer RegisterVariable-Instanz aus aufgerufen
worden ist
if ($_IPS['SENDER'] == "RegisterVariable")
{
// bereits im Puffer der Instanz vorhandene Daten in $data kopieren
$data = RegVar_GetBuffer($_IPS['INSTANCE']);
// neu empfangene Daten an $data anhängen
$data .= $_IPS['VALUE'];

// wenn das Trennzeichen ; in $data gefunden worden ist
if (strpos($data, ';'))
{
// $data in durch ; separierte Datensätze zerlegen
$datasets = explode(';', $data);

// alle nicht durch ; terminierten Datensätze ausgeben
for ($i = 0; $i < count($datasets) - 1; $i++)
{
echo "empfangener Datensatz: ".$datasets[$i]."\n";
}

// $data auf den Inhalt des letzten (unvollständigen) Datensatzes
setzen
$data = $datasets[count($datasets) - 1];
}

// Inhalt von $data im Puffer der RegisterVariable-Instanz speichern
RegVar_SetBuffer($_IPS['INSTANCE'], $data);
}
```

Following example chaines received data and outputs blocks from a length of exactly 16 characters:

```
// wenn das Skript von einer RegisterVariable-Instanz aus aufgerufen
worden ist
if ($_IPS['SENDER'] == "RegisterVariable")
{
// bereits im Puffer der Instanz vorhandene Daten in $data kopieren
$data = RegVar_GetBuffer($_IPS['INSTANCE']);
// neu empfangene Daten an $data anhängen
$data .= $_IPS['VALUE'];

// wenn $data mindestens 16 Zeichen lang ist
if (strlen($data) >= 16)
{
// $data in Blöcke von bis zu 16 Zeichen zerlegen
$datasets = str_split($data, 16);
// $data leeren
$data = "";

// alle Datensätze durcharbeiten
for ($i = 0; $i < count($datasets); $i++)
{
// vollständige Datensätze (genau 16 Zeichen lang) ausgeben
if (strlen($datasets[$i]) == 16)
{
echo "empfangener Datensatz: ".$datasets[$i]."\n";
}
else
{
// Unvollständige Datensätze in $data schreiben
$data = $datasets[$i];
}
}
}

// Inhalt von $data im Puffer der RegisterVariable-Instanz speichern
RegVar_SetBuffer($_IPS['INSTANCE'], $data);
}
```

RegVar_GetBuffer

```
bool RegVar_GetBuffer (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the device to be switched

RETURNS

String of the current buffer

DESCRIPTION

Reads the internal buffer of the instance with ID **InstanceID**.

EXAMPLE

```
$buf = RegVar_GetBuffer(12345);
$buf .= $_IPS['VALUE']; //Concatenating

//verarbeiten ...
RegVar_SetBuffer(12345, $buf); //Write back remaining buffer
```

RegVar_SendEvent

Require: Symcon >= 4.1

```
bool RegVar_SendEvent (int $InstanceID, int $ReportID, string $Text)
```

PARAMETERS

InstanceID	ID of the RegisterVariable-Instance
ReportID	The record type as ReportID
Text	Buffer to send

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a string **Text** through the instance with ID **InstanceID** as the record type **ReportID**.



RegVar_SendEvent can only be used with a HID instance

EXAMPLE

```
//Send an event “Hello World” to the parent instance with the ReportID
0
RegVar_SendEvent(12345, 0, "Hello World");
```

RegVar_SendPacket

Require: Symcon >= 4.1

```
bool RegVar_SendPacket (int $InstanceID, string $Text, string
$ClientIP, string $ClientPort)
```

PARAMETERS

InstanceID	ID of the RegisterVariable-Instance
Text	Buffer to send
ClientIP	IP address to send to
ClientPort	Port to send to

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet **Text** via the instance with the ID **InstanceID**. In addition, the IP address **ClientIP** and the port **ClientPort** are specified.



RegVar_SendPacket can only be used in connection with a ServerSocket instance

EXAMPLE

```
// Sends the packet "Hello World" via ServerSocket to IP:Port
192.168.1.2:1234
RegVar_SendPacket(12345, "Hello World", "192.168.1.2", 1234);
```

RegVar_SendText

```
bool RegVar_SendText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the device to be switched
Text	Buffer to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a string **Text** from the instance with ID **InstanceID**

EXAMPLE

```
RegVar_SendText(12345, "Hallo World");
```

RegVar_SetBuffer

```
bool RegVar_SetBuffer (int $InstanceID, string $Buffer)
```

PARAMETERS

InstanceID	ID of the device to be switched
Buffer	Data to be written into the buffer

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Set the internal buffer of the instance with ID **InstanceID** to value **Buffer**

EXAMPLE

```
$buf = RegVar_GetBuffer(12345);
$buf .= $_IPS['VALUE']; //Concatenating

//verarbeiten ...
RegVar_SetBuffer(12345, $buf); //Write back remaining buffer
```

Skin Control

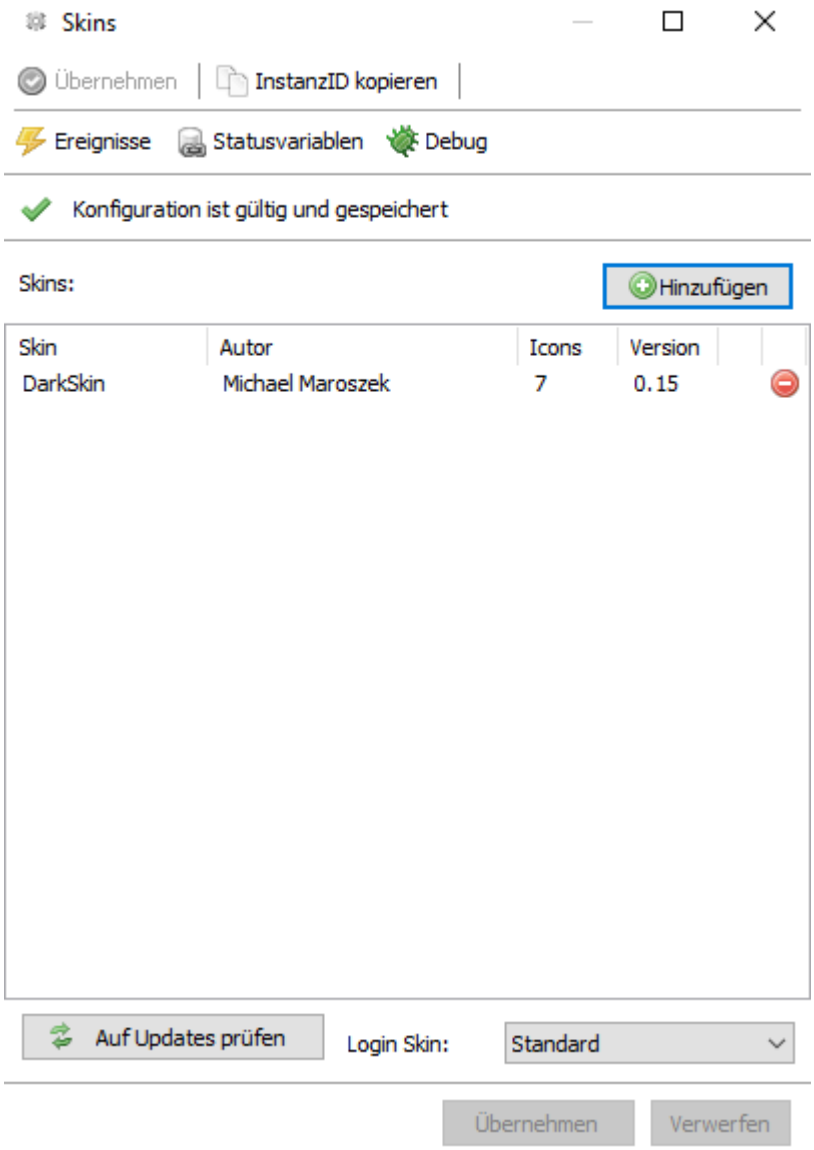
Require: Symcon >= 4.0

DESCRIPTION

The "Skin Control" module offers the possibility to integrate individual skins or those provided by other users.

INTEGRATION IN IP-SYMCON

In the "Modules" instance ("Logical tree view"->"Core instances"), a Git repository link can be entered via "Add". Here, the skin is then installed and is immediately available.



UPDATE

The "Check for updates" button checks whether an update for already added skins is available. If an update is possible, the respective skin can be updated via the displayed update button. If a skin is up to date, a green tick is displayed instead.

CHOOSE LOGIN SKIN

Separate to the respective WebFront skins, which are selected in the respective WebFront configurator, the skin of the login area of the WebFront can also be changed. This can be set via selection in the skin control.

DEVELOP INDIVIDUAL SKINS

Information on developing individual skins can be found here:
SDK for Skins

EXAMPLES

The following skins are maintained and provided by us:
DarkSkin: github.com/symcon/SkinDark
LightSkin: github.com/symcon/SkinLight

SSDP Control

The SSDP (Simple Service Discovery Protocol) control module allows searching for UPnP devices in the network.
SSDP is a UDP-based protocol that allows UPnP devices to identify themselves in the network and communicate only the most important information. These include, for example, device name, device type, or a URL describing the device.



Further information can be found for example on Wikipedia

INSTALLATION

The SSDP Control is automatically installed as a core instance. This can be found in the object tree under core instances.

CONFIGURATION

The module does not require any further configuration. By default it uses port 1900 to search for devices.

YC_SearchDevices

Require: Symcon >= 5.3

```
array YC_SearchDevices (int $InstanceID, string $SearchTarget)
```

PARAMETERS

InstanceID	ID of the SSDP control to be requested
SearchTarget	.

RETURNS

Returns all information about found devices as an array.

DESCRIPTION

Sends a request via the SSDP Control with the ID **InstanceID** and the matching search target **SearchTarget**.
---param2---

SEARCHTARGET	DESCRIPTION
"ssdp:all"	All devices responding to the SSDP
"upnp:rootdevice"	All devices with the subtype "rootdevice"
"uuid:<UID>"	All devices with unique ID **UID**
"urn:<URN>"	All devices with the unique URN **URN**

EXAMPLE

```
YC_SearchDevices(27253, "ssdp:all");

// Sample output
var_dump(YC_SearchDevices(27253, "ssdp:all"));
/*
array(97) {
[...]
[11]=>
array(9) {
["CacheControl"]=>
string(14) "max-age = 1800"
["Date"]=>
```

```
string(0) ""
["Ext"]=>
string(0) ""
["Location"]=>
string(52) "http://172.17.31.142:1400/xml/device_description.xml"
["Server"]=>
string(38) "Linux UPnP/1.0 Sonos/53.2-70210 (ZPS5)"
["ST"]=>
string(47) "urn:schemas-upnp-org:service:DeviceProperties:1"
["USN"]=>
string(78) "uuid:RINCON_000E58507DEA01400::urn:schemas-upnp-
org:service:DeviceProperties:1"
["Fields"]=>
array(5) {
[0]=>
string(52) "X-RINCON-HOUSEHOLD: HHID_tZkieK2lkPYES3tycishsbwfXIz"
[1]=>
string(21) "X-RINCON-BOOTSEQ: 636"
[2]=>
string(20) "X-RINCON-WIFIMODE: 0"
[3]=>
string(19) "X-RINCON-VARIANT: 0"
[4]=>
string(83) "HOUSEHOLD.SMARTSPEAKER.AUDIO:
HHID_tZkieK2lkPYES3tycishsbwfXIz.hPqvaa7W35eMY2P3XidJ"
}
["IPv4"]=>
string(13) "172.17.31.142"
)
[...]
}
*/
```

System Information

With the system features, you can read out information over your system and the existing hardware to visualize it or otherwise use it in scripts.

Sys_GetBattery

```
array Sys_GetBattery ()
```

RETURNS

Array with information about the battery

DESCRIPTION

Gives information about the presence of a battery, whether it is in use and what percentage of the battery charge is left. The remaining time and maximum battery life are indicated in seconds.

EXAMPLE

```
print_r(Sys_GetBattery());
/*
// Sample output on a desktop PC without battery
// HasBattery is available with IP-Symcon 7.0+
Array
(
[HasBattery] =>
[OnBattery] =>
[IsCharging] =>
[BatteryLevel] => -1
[BatteryRemainingTime] => -1
[BatteryMaxTime] => -1
)
*/
```

Sys_GetCPUInfo

```
array Sys_GetCPUInfo ()
```

RETURNS

Array with information about CPU utilization

DESCRIPTION

Provides for each CPU the current load (CPU_*) and the average load of the last 60 seconds for all CPU cores (CPU_AVG). The average load is calculated by taking a measurement of each second of the value of the current load and is stored afterwards.The average load is calculated by adding the last 60 measured values and then divided by 60.



The average load is not the average load of the current load, but the average load of the last 60 seconds.

EXAMPLE

```
print_r(Sys_GetCPUInfo());

/*
Array
(
[CPU_0] => 3
[CPU_AVG] => 3
)
*/
```

Sys_GetHardDiskInfo

```
array Sys_GetHardDiskInfo ()
```

RETURNS

Array of information about all the hard drives

DESCRIPTION

Provides information on the existing disks in the system, including the size and the already used memory.

EXAMPLE

```
print_r(Sys_GetHarddiskInfo());

/*
Array
(
[HDD0] => Array
(
[LETTER] => c:\
[LABEL] =>
[TOTAL] => 53684989952
[FREE] => 23275171840
)

[NUMDRIVES] => 1
)
*/
```

Sys_GetMemoryInfo

```
array Sys_GetMemoryInfo ()
```

RETURNS

Array of information on the overall memory usage and virtual memory usage

DESCRIPTION

Provides information about the memory usage of the operating system.

EXAMPLE

```
print_r(Sys_GetMemoryInfo());

/*
Array
(
[TOTALPHYSICAL] => 1072467968
[AVAILPHYSICAL] => 526647296
[TOTALPAGEFILE] => 2420019200
[AVAILPAGEFILE] => 1386422272
[TOTALVIRTUAL] => 2147352576
[AVAILVIRTUAL] => 1906978816
)
*/
```

Sys_GetNetworkInfo

```
array Sys_GetNetworkInfo ()
```

RETURNS

Array with information about all network adapters

DESCRIPTION

Provides information about all network adapters.

EXAMPLE

```
print_r(Sys_GetNetworkInfo());
/*
// Example output

Array
(
[0] => Array
(
[InterfaceIndex] => 10
[IP] => 192.168.1.2
[MAC] => 00:A0:03:AD:14:BD
[Description] => Siemens I BT USB Remote NDIS Network Device
[Speed] => 9728000
[InTotal] => 40236
[OutTotal] => 247248
)

[1] => Array
(
[InterfaceIndex] => 13
[IP] => 172.12.1.200
[MAC] => 00:A0:03:AD:14:BD
[Description] => Qualcomm Atheros AR8151 PCI-E Gigabit Ethernet
Controller (NDIS 6.30)
[Speed] => 1000000000
[InTotal] => 169987950
[OutTotal] => 86029648
)
)
*/
```

Sys_GetProcessInfo

```
array Sys_GetProcessInfo ()
```

RETURNS

Array of information, such as e.g. Memory usage and number of threads

DESCRIPTION

Provides information about the process of IP Symcon.

EXAMPLE

```
print_r(Sys_GetProcessInfo());
/*
Array
(
[IPS_HANDLECOUNT] => 635
[IPS_NUMTHREADS] => 53
[IPS_VIRTUALSIZE] => 240373760
[IPS_WORKINGSETSIZE] => 32706560
[IPS_PAGEFILE] => 52719616
[PROCESSCOUNT] => 53
)
*/
```

Sys_GetSpooler

```
array Sys_GetSpooler ()
```

RETURNS

Array of information about all the printing queues and being printed documents

DESCRIPTION

Provides information about all print queues



This function only works in Windows

EXAMPLE

```
print_r(Sys_GetSpooler());
```

Sys_GetURLContent

```
string Sys_GetURLContent (string $URL)
```

PARAMETERS

URL	Complete URL

RETURNS

If the command was executed successfully, it returns the result of the website contents (including binary code), otherwise **FALSE**.

DESCRIPTION

Reads the contents of the **URL** and returns it as a return value.



The function has a default timeout of 10 seconds. If it should be changed, the function Sys_GetURLContentEx can be used

EXAMPLE

```
echo Sys_GetURLContent("http://www.google.de");
```

Sys_GetURLContentEx

```
string Sys_GetURLContentEx (string $URL, array $Parameter)
```

PARAMETERS

URL — Complete URL

Parameter — Array with index => value pairs

INDEX	TYPE	DESCRIPTION
Timeout	Integer	**Default = 10000**. Timeout in milliseconds
AuthUser	String	**Default = ""**. Username for basic authentication
AuthPass	String	**Default = ""**. Password for basic authentication
VerifyPeer	Boolean	**Default = true**. Verify that hosts SSL certificate is correct
VerifyHost	Boolean	**Default = true**. Verify that host URL from the SSL certificate matches the property Host

RETURNS

If the command was executed successfully, it returns the result of the website contents (including binary code), otherwise **FALSE**.

DESCRIPTION

Reads the contents of the **URL** and returns it as a return value. Here, some **Parameter** for timeouts, authentication and proxy are made.

EXAMPLE

```
echo Sys_GetURLContentEx("http://www.google.de", Array("AuthUser"=>
"test", "AuthPass"=> "test"));
```

Sys_Ping

```
bool Sys_Ping (string $Host, int $Timeout)
```

PARAMETERS

Host	Hostname or IP address
Timeout	Time in milliseconds

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a ping to **Host** and waits **Timeout** milliseconds.

EXAMPLE

```
Sys_Ping("meinrechner", 1000); //Wait max. 1 second
```

Tailscale VPN

Require: Symcon >= 7.1
This module offers the possibility to use Tailscale VPN on the SymBox.



This module is only available for the SymBox!

RANGE OF FUNCTIONS

Downloading and updating Tailscale VPN
Authentication via Auth Keys
Access via VPN to the SymBox (incl. SSH)
Access to devices behind the SymBox (subnets)
Activate/deactivate VPN via variable

SOFTWARE INSTALLATION

Install the 'Tailscale VPN' module via the Module Store.

SET UP THE INSTANCES IN IP-SYMCON

Under "Add instance" the 'Tailscale VPN' module can be found using the quick filter.
Further information on adding instances in the Documentation of the instances

SET UP TAILSCALE

Create Auth Key: https://login.tailscale.com/admin/settings/keys
Open Tailscale instance
Press the buttons to download, install and start Tailscale.
The authentication can then be carried out using the Auth Key created at the beginning. The Auth Key is no longer required afterwards and does not need to be saved anywhere.

SHARE NETWORK

Switch the VPN in IP-Symcon off and on again.
To access devices behind the SymBox, the "Advertise Routes" function must be set up
To do this, enter the subnet e.g. 192.168.178.0/24 in the CIDR notation in the list and accept the configuration.
The routes must still be activated in the Tailscale Dashboard. To do this, open the following page: https://login.tailscale.com/admin/machines
Select the correct machine there, click Review and confirm the activation of the subnet.

CONFIGURATION PAGE

NAME	DESCRIPTION
Advertise Routes	Subnet to be shared in CIDR notation e.g. 192.168.178.0/24

TextParser

The TextParser is a module that provide text snippets from websites or files.

INTEGRATION INTO IP-SYMCON

The TextParser can be searched and added as instance via the "+" in the Object Tree. Within the configuration of the TextParser, a gateway that provides the raw text can be selected via "Change Gateway". For example, an HTTP Client can be selected.

DESCRIPTION

The TextParser in IP-Symcon allows to cut off certain parts e.g. of a web page or a file by executing several rules consecutively. The following rules are available:

FUNCTION	DESCRIPTION
Cut Before	Cuts off all text before Tag1
Cut After	Cuts off all text after Tag1
Get Text	Brings all text until Tag1 in the variable
Get Text Between	Cuts the text between Tag1 and Tag2 and writes it in the variable

Multiple rules can be applied consecutively and even additional "Get Text" - "Get Text Between" operations can be used.

EXAMPLE

A small example will show how to read the first headline of heise.de to show it e.g. in the Visualization.

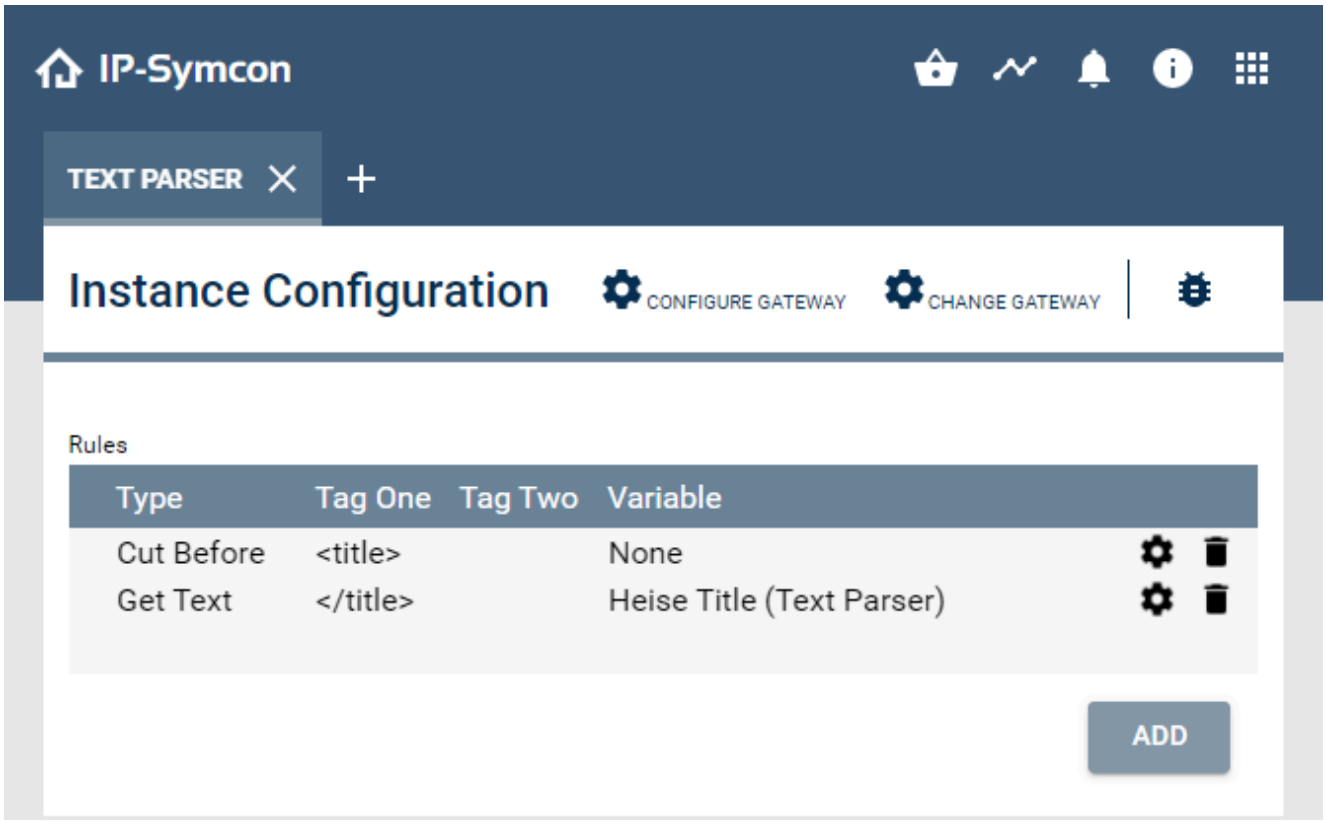
Create a Text Parser.
Next, create a parent instance. In this example, a HTTP Client is required.
In the HTTP Client, the URL heise.de must be entered and eventually the timer activatec, which requests the page cyclically.
Back in the Text Parser, the HTTP Client needs to be configured as Gateway. It can be selected via "Change Gateway".
In the Text Parser, the following rules need to be defined to read the title.
These rules can be deducted by looking at the HTML Code that is read by the HTTP Client. In the code, it should be searched for significant parts, that should be cut. This is repeated until the desired text remains. Those rules are the entered into IP-Symcon.
With "Get Text" a variable is defined where the completely cur text is saved.

RULES

DESCRIPTION	RULE
Cut Text Before	<title>

DESCRIPTION	RULE
Get Text	</title>

SCREENSHOT



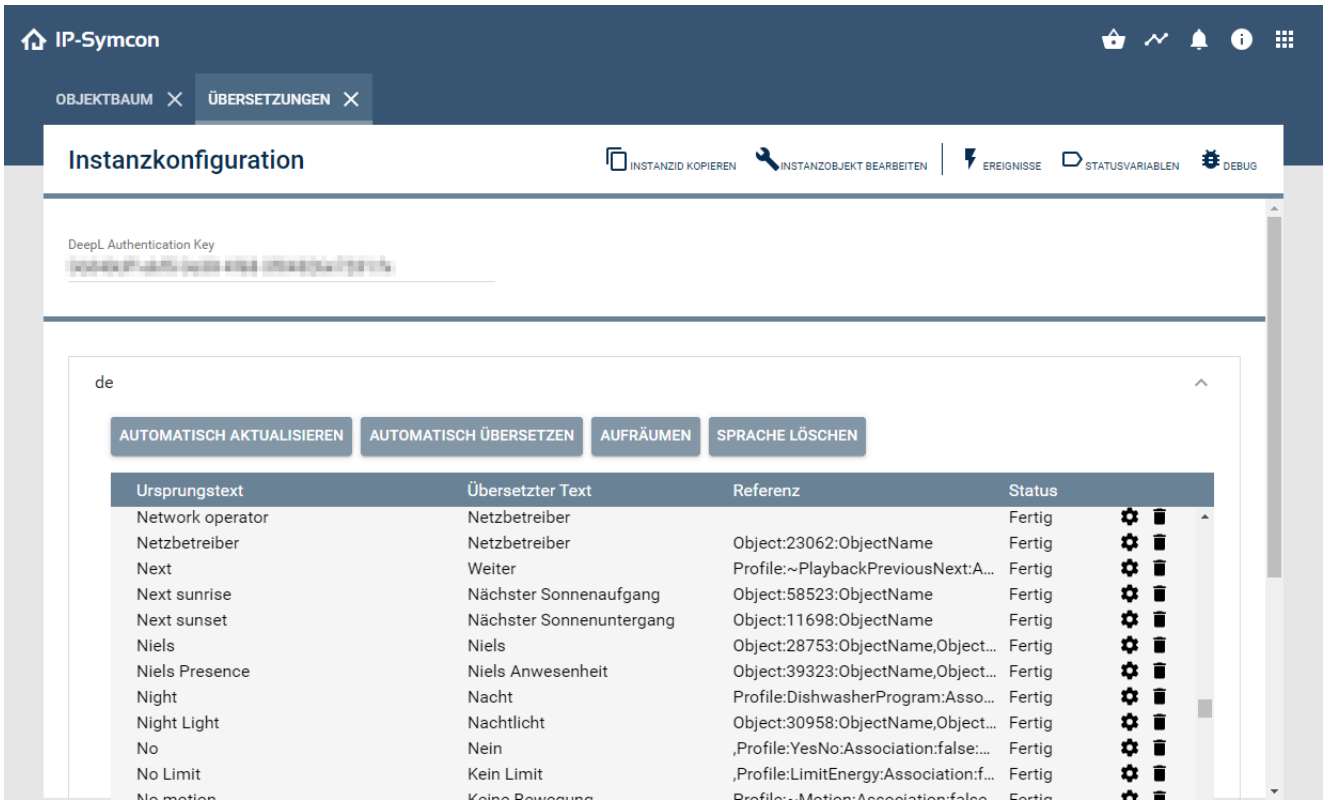
Translation Control

Require: Symcon >= 7.1

The translation module is a paid extension that can be purchased for any existing Symcon license. With each Symcon license, up to 10 strings per language can be translated free of charge. The extension can be purchased directly in the Shop.

It is recommended to translate from English into different languages. Therefore, the Symcon system should be run in English. The language of the operating system can be changed for this purpose. If the device on which Symcon is installed is not to be influenced, the special switch Locale can be set to en

The translation control allows any text in Symcon to be translated into different languages. The various translations can be adapted here at any time. The entire system can be automatically translated using the service DeepL. An API key from DeepL is required for this. The translations apply to both the console and the visualization.



ADD LANGUAGE

A new language can be added with the corresponding ISO language code.

ACTIONS

ACTION	DESCRIPTION
Auto update	Adds all texts from the system to the translation table. These can be translated manually or automatically.
Auto translate	All values of the 'Original text' column that do not have the status 'Unused' are translated using DeepL and transferred to the 'Translated text' column.
Cleanup	All entries that are no longer in the system are removed
Delete language	Deletes the translation table

TRANSLATION TABLE

COLUMN	DESCRIPTION
Source text	The text found in the system and used for the translation with DeepL
Translated text	Initially empty. Can be filled manually or by the automatic translation. If the field is empty, the original text is displayed.
Reference	Shows whether the text originates from a profile or object and whether it is, for example, an object name or the suffix of a profile trade. This context can be used to improve the translation.
Status (New, Unused, Done)	The current status of the individual translations. New is set when the translation is initially recognized. If the translation has been successfully completed, the status is set to Finished. Entries with the status Unused are not sent for translation.

Util Control

This module is an interface between the service and the IP-Symcon console. It allows advanced features such as finding and replacing in files.



This module should not be deleted.
If it is cleared, the next time it will be re-created.

UC_FindReferences

```
array UC_FindReferences (int $InstanceID, int $ID)
```

PARAMETERS

InstanceID	ID for the Util Control
ID	ID of the object whose usages are searched

RETURNS

An array with one entry per location. Each entry contains the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
LineContent	string	For scripts the found line without leading and trailing spaces, otherwise the kind of location (see table)
LineNumber	integer	For scripts the line number (starting at 1), otherwise 0
LinePosition	integer	For scripts the position of the first match in the line in bytes (starting at 0), otherwise 0
ObjectID	integer	ID of the object that uses **ID**

DESCRIPTION

The function searches the entire system for places where the object **ID** is used, e.g. before a variable is deleted or replaced. The management console uses it to show the usages of an object. The following locations are recognized, in this order:

LINECONTENT	OBJECTID	LOCATION
(line of the script)	Script	The script content contains **ID** as text. One entry per line, **LineNumber** and **LinePosition** are set
Variable	Variable	**ID** is the custom action (action script) of the variable
Event (Action)	Event	A parameter of the action of the event or of an action of the weekly schedule contains **ID** as text
Event (Condition)	Event	A condition of the event checks the variable **ID** or compares with it
Event	Event	**ID** is the triggering variable of the triggered event
Link	Link	**ID** is the target of the link
Instance	Instance	The instance has **ID** in its reference list (see IPS_GetReferenceList)

LINECONTENT	OBJECTID	LOCATION
Instance (Connected)	Instance	The instance is connected to the instance **ID** as its parent
IPSView	Media object (IPSView)	The IPSView file uses **ID**
Chart	Media object (Chart)	A dataset of the chart uses the variable **ID**

The name in **LineContent** may be translated into the language of the system. An object can appear several times in the result, e.g. a script with several matches or an event with several matching conditions. If nothing is found, the result is an empty array. Whether **ID** exists is not checked. Searching for the root (ID 0) is not possible and fails with "Search ID cannot be 0".



Scripts and action parameters are searched as text. Searching for 12345 therefore also finds 123456 or a number 12345 that is not an object ID at all. Conversely, IDs that a script only determines at runtime, e.g. with IPS_GetObjectIDByIdent, are not found.



Since all script files are read, the search can take some time in large systems.

The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

// Output all usages of the variable 12345
foreach (UC_FindReferences($id, 12345) as $reference) {
if ($reference['LineNumber'] > 0) {
echo IPS_GetName($reference['ObjectID']) . ', line ' .
$reference['LineNumber'] . ': ' . $reference['LineContent'] . PHP_EOL;
} else {
echo IPS_GetName($reference['ObjectID']) . ' (' .
$reference['LineContent'] . ')' . PHP_EOL;
}
}
```

UC_GetEventStatistics

```
array UC_GetEventStatistics (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
CyclicUpdateCount	integer	Number of updates of cyclic events and weekly schedules
TriggerUpdateCount	integer	Number of executions of triggered events

DESCRIPTION

The function evaluates the message EM_UPDATE, which an event sends after each execution and on each recalculation of its next run time, and assigns it by the type of the event.
Triggered events only send the message when they are executed, so **TriggerUpdateCount** counts their executions. Cyclic events and weekly schedules count towards **CyclicUpdateCount**, also when their next run time is recalculated without an execution, e.g. when they are activated, deactivated or their schedule is changed.
If one of the values grows strongly between two queries, a very large number of events is executed, e.g. triggered events on a variable that is updated very frequently. The values are summed up in memory since the start of IP-Symcon and cannot be reset (only a restart starts again at 0).
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

print_r(UC_GetEventStatistics($id));
```

```
/* e.g. returns:
Array
(
[CyclicUpdateCount] => 0
[TriggerUpdateCount] => 0
)
*/
```

UC_GetIconList

```
array UC_GetIconList (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with the names of the icons (without file extension).

DESCRIPTION

The function returns the names of all icons that come with IP-Symcon. For this purpose, the SVG files in the directory "webfront/img/icons" of the installation are read. The names can be used e.g. with IPS_SetIcon. The sub-icons of adaptive icons (name with "-", e.g. "Battery-50") are not included. The order of the entries is not defined. If the directory does not exist, the result is an empty array.



These icons apply to the classic WebFront. In the tile visualization every Font Awesome icon can be used (light style and brand icons, see Icons), e.g. "lightbulb" or "temperature-half". The classic names listed here are shown there as a similar Font Awesome icon.

The icons including their content and the icons of the skins are returned by IPS_GetIcons. The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

print_r(UC_GetIconList($id));

/* e.g. returns:
```

```
Array
(
[0] => Aircraft
[1] => Alert
[2] => ArrowRight
[3] => Backspace
[4] => Basement
...
)
*/
```

UC_GetKernelStatistics

```
array UC_GetKernelStatistics (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
MessageCounter	integer	Number of messages put into the message queue of the kernel since the start of IP-Symcon
MessageQueueDelay	integer	Waiting time of the oldest message not yet distributed in milliseconds (0 if the queue is empty)
MessageQueueSize	integer	Number of messages currently waiting in the message queue of the kernel to be distributed
MessageSlowCounter	integer	Number of messages whose processing in a kernel component took longer than "MessageQueueWatchLimit". Only with "MessageQueueWatch" enabled, otherwise always 0
RequestQueueSize	integer	Number of script executions currently waiting for a free PHP thread

DESCRIPTION

The function returns key figures that allow to assess the load of the kernel. All events in the system (variable updates, status changes, log messages, ...) are put into a queue as messages and distributed from there to all receivers. If **MessageQueueSize** and **MessageQueueDelay** keep rising, more messages are created than can be processed. The cause can be narrowed down with UC_GetMessageTypeList (which messages) and UC_GetMessageSenderIDList (which objects). A permanently high value of **RequestQueueSize** means that more scripts are started than PHP threads are available.
MessageCounter counts since the start of IP-Symcon, the rate results from the difference of two queries. **MessageSlowCounter** is only increased if the special switch "MessageQueueWatch" is enabled (e.g. with IPS_SetOption); the limit is defined by "MessageQueueWatchLimit" (default 50 milliseconds). The processing by the individual instances is recorded by IPS_GetInstanceMessageStatistics, their queues are returned by IPS_GetInstanceMessageQueueSize.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-

24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

print_r(UC_GetKernelStatistics($id));

/* e.g. returns:
Array
(
[MessageCounter] => 184
[MessageQueueDelay] => 0
[MessageQueueSize] => 0
[MessageSlowCounter] => 0
[RequestQueueSize] => 0
)
*/

// Determine messages per second
$before = UC_GetKernelStatistics($id)['MessageCounter'];
sleep(10);
$after = UC_GetKernelStatistics($id)['MessageCounter'];
echo 'Messages per second: ' . (($after - $before) / 10);
```

UC_GetLastLogMessages

```
array UC_GetLastLogMessages (int $InstanceID, int $Type)
```

PARAMETERS

InstanceID	ID for the Util Control
Type	Message type from 0 to 6 (see table in the description)

RETURNS

An array with one entry per message, starting with the oldest one. Each entry contains the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
Message	string	Text of the message
Sender	string	Name of the sender, e.g. "Kernel" or the module name. Shorter names are padded with spaces to 20 characters
SenderID	integer	ID of the sender, e.g. the InstanceID for messages of instances. 0 for messages of the kernel and from IPS_LogMessage
TimeStamp	integer	Unix timestamp of the message

DESCRIPTION

The function returns the messages of the message type **Type** that were written to the log most recently. For this purpose, the Util Control keeps a separate buffer in memory for each message type, which contains the last 25 messages. If another message is added, the oldest one is discarded. The number can be changed with the special switch "LogMessageCount"; the value is read when the instance is created, i.e. when IP-Symcon starts. After a restart the buffers are empty, UC_ResetLastLogMessages clears them as well.

TYPE	CONSTANT	NAME IN THE LOG	NOTE
0	KL_MESSAGE	MESSAGE	
1	KL_SUCCESS	SUCCESS	
2	KL_NOTIFY	NOTIFY	
3	KL_WARNING	WARNING	
4	KL_ERROR	ERROR	
5	KL_DEBUG	DEBUG	Always empty, debug messages are not forwarded

TYPE	CONSTANT	NAME IN THE LOG	NOTE
6	KL_CUSTOM	CUSTOM	e.g. messages from IPS_LogMessage



Type is not the message ID of the constant but its offset to KL_MESSAGE, e.g. **KL_ERROR - KL_MESSAGE** = 4. Values outside of 0 to 6, e.g. KL_ERROR directly, result in the error "Type must be between 0 and 6".

Only messages that occur while IP-Symcon is ready are recorded (not during startup and shutdown). Debug messages (KL_DEBUG) are not forwarded and are therefore never recorded, neither are messages that are filtered out by the special switch "LogfileFilter". The number of messages per type is returned by UC_GetLogMessageStatistics.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

// Output all stored error messages
foreach (UC_GetLastLogMessages($id, KL_ERROR - KL_MESSAGE) as $message)
{
echo date('d.m.Y H:i:s', $message['TimeStamp']) . ' | ' .
trim($message['Sender']) . ' | ' . $message['Message'] . PHP_EOL;
}

print_r(UC_GetLastLogMessages($id, KL_SUCCESS - KL_MESSAGE));

/* e.g. returns:
Array
(
[0] => Array
(
[Message] => *** IPS READY
[Sender] => Kernel
[SenderID] => 0
[TimeStamp] => 1700000000
)

)
*/
```

UC_GetLogMessageStatistics

```
array UC_GetLogMessageStatistics (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
MessageCustomCount	integer	Number of messages of type KL_CUSTOM (e.g. from IPS_LogMessage)
MessageDefaultCount	integer	Number of messages of type KL_MESSAGE
MessageErrorCount	integer	Number of messages of type KL_ERROR
MessageNotifyCount	integer	Number of messages of type KL_NOTIFY
MessageSuccessCount	integer	Number of messages of type KL_SUCCESS
MessageTimeStamp	integer	Sequence number of the last message (of any type) processed by the Util Control. Not a timestamp!
MessageWarningCount	integer	Number of messages of type KL_WARNING
ResetTimeStamp	integer	Unix timestamp from which counting starts (start of IP-Symcon or last call of UC_ResetLogMessageStatistics)

DESCRIPTION

The function returns how many messages per message type have been written to the log since **ResetTimeStamp**. For this purpose, the Util Control receives all messages of the kernel and counts the log messages. The counters can be queried regularly, e.g. to detect whether new errors or warnings have occurred. The last messages themselves are returned by UC_GetLastLogMessages, the counters are reset with UC_ResetLogMessageStatistics.
Only messages that occur while IP-Symcon is ready are recorded (not during startup and shutdown). Debug messages (KL_DEBUG) are not forwarded and are therefore never recorded, neither are messages that are filtered out by the special switch "LogfileFilter".

Despite its name, **MessageTimeStamp** is not a timestamp. The kernel numbers all messages it distributes consecutively since the start (1, 2, 3, ...). **MessageTimeStamp** is the number of the last message processed by the Util Control and changes with every message, not only with log messages. If no messages are waiting to be processed, the value roughly equals **MessageCounter** of UC_GetKernelStatistics. It is not reset by UC_ResetLogMessageStatistics.

The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

print_r(UC_GetLogMessageStatistics($id));

/* e.g. returns:
Array
(
[MessageCustomCount] => 0
[MessageDefaultCount] => 0
[MessageErrorCount] => 2
[MessageNotifyCount] => 0
[MessageSuccessCount] => 1
[MessageTimeStamp] => 184
[MessageWarningCount] => 1
[ResetTimeStamp] => 1700000000
)
*/
```

UC_GetMessageSenderIDList

```
array UC_GetMessageSenderIDList (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with one entry per sender that has sent a message since the start. Each entry contains the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of messages of this sender
SenderID	integer	ID of the object that caused the message (0 = kernel or no object)

DESCRIPTION

The function counts all messages distributed by the kernel, separated by the sender. The sender is the object the message refers to, e.g. the variable for a variable update or the instance for a status change. This allows to find objects that create an unusually large number of messages, e.g. a variable that is updated several times per second. The amount of data per sender is returned by UC_GetMessageSenderIDSizeList, the distribution among the message types by UC_GetMessageTypeList.
The values are summed up in memory since the start of IP-Symcon and cannot be reset (only a restart starts again at 0). The order of the entries is not defined.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

// Output the three most active senders
$list = UC_GetMessageSenderIDList($id);
usort($list, function ($a, $b) {
```

```
return $b['Count'] <=> $a['Count'];
});
foreach (array_slice($list, 0, 3) as $entry) {
echo $entry['SenderID'] . ': ' . $entry['Count'] . PHP_EOL;
}

/* e.g. returns:
0: 26
10008: 25
10012: 11
*/
```

UC_GetMessageSenderIDSizeList

```
array UC_GetMessageSenderIDSizeList (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with one entry per sender that has sent a message since the start. Each entry contains the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
SenderID	integer	ID of the object that caused the message (0 = kernel or no object)
Size	integer	Summed up size of the message data of this sender in bytes (estimate)

DESCRIPTION

The function sums up per sender the size of the data transported in its messages. While UC_GetMessageSenderIDList only counts the number of messages, this list shows which objects create particularly large amounts of data, e.g. a string variable that is frequently updated with long texts.
The size is an estimate: strings count with their length in bytes, integers and floats with 8 bytes and booleans with 1 byte. Management data of the message itself (sender, message ID) is not counted.
The values are summed up in memory since the start of IP-Symcon and cannot be reset (only a restart starts again at 0). The order of the entries is not defined.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

// Output the three senders with the largest amount of data
```

```
$list = UC_GetMessageSenderIDSizeList($id);
usort($list, function ($a, $b) {
return $b['Size'] <=> $a['Size'];
});
foreach (array_slice($list, 0, 3) as $entry) {
echo $entry['SenderID'] . ': ' . $entry['Size'] . ' bytes' . PHP_EOL;
}

/* e.g. returns:
10015: 1648 bytes
10007: 729 bytes
10008: 439 bytes
*/
```

UC_GetMessageTypeList

```
array UC_GetMessageTypeList (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with one entry per message type that has occurred since the start. Each entry contains the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of messages of this type
Message	integer	Message ID, e.g. 10603 for VM_UPDATE

DESCRIPTION

The function counts all messages distributed by the kernel, separated by their message ID. The meaning of the IDs is listed under Messages. If the message load increases (see UC_GetKernelStatistics), the list shows which kind of messages is responsible, e.g. a flood of variable updates (VM_UPDATE). Which objects cause the messages is returned by UC_GetMessageSenderIDList.
The values are summed up in memory since the start of IP-Symcon and cannot be reset (only a restart starts again at 0). The order of the entries is not defined.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

// Output the three most frequent message types
$list = UC_GetMessageTypeList($id);
usort($list, function ($a, $b) {
return $b['Count'] <=> $a['Count'];
```

```
});
foreach (array_slice($list, 0, 3) as $entry) {
echo $entry['Message'] . ': ' . $entry['Count'] . PHP_EOL;
}

/* e.g. returns:
11403: 27
10401: 20
10412: 20
*/
```

UC_GetScriptSenderList

```
array UC_GetScriptSenderList (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

An array with one entry per trigger that has executed a script since the start. Each entry contains the following **key => value** pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of script executions of this trigger
Sender	string	Trigger of the script execution as in $_IPS['SENDER'], e.g. "TimerEvent", "Variable" or "RunScript"

DESCRIPTION

The function counts every started script execution (message SE_RUNNING) and assigns it to the trigger. The trigger corresponds to the value of the system variable $_IPS['SENDER'] in the script, e.g. "TimerEvent" for cyclic events, "Variable" for triggered events, "WebFront" for operations in the visualization or "RunScript" for IPS_RunScript. This allows to identify where a high script load comes from, e.g. if UC_GetKernelStatistics shows a permanently filled **RequestQueueSize**. The currently running scripts are returned by IPS_GetScriptThreadList. The values are summed up in memory since the start of IP-Symcon and cannot be reset (only a restart starts again at 0). The order of the entries is not defined.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

// Output script executions per trigger
foreach (UC_GetScriptSenderList($id) as $entry) {
```

```
echo $entry['Sender'] . ': ' . $entry['Count'] . PHP_EOL;
}
```

UC_ResetLastLogMessages

```
bool UC_ResetLastLogMessages (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function clears the buffers of all message types whose content is returned by UC_GetLastLogMessages. Afterwards UC_GetLastLogMessages only returns messages that were written to the log after the call. The log or the log files themselves are not changed. The counters of UC_GetLogMessageStatistics are kept, they are reset with UC_ResetLogMessageStatistics.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

UC_ResetLastLogMessages($id);

// Now returns an empty array until another error message occurs
print_r(UC_GetLastLogMessages($id, KL_ERROR - KL_MESSAGE));
```

UC_ResetLogMessageStatistics

```
bool UC_ResetLogMessageStatistics (int $InstanceID)
```

PARAMETERS

InstanceID	ID for the Util Control

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets all counters of UC_GetLogMessageStatistics to 0 and **ResetTimeStamp** to the current time. Afterwards only what is written to the log from this point on is counted. **MessageTimeStamp** remains unchanged. The stored messages of UC_GetLastLogMessages are kept as well, they are cleared with UC_ResetLastLogMessages.
The ID of the Util Control instance can be determined e.g. with IPS_GetInstanceListByModuleID and the module GUID {B69010EA-96D5-46DF-B885-24821B8C8DBD}. If **InstanceID** does not exist, the function fails with "Instance #... does not exist", if the instance belongs to another module, with "Instance does not implement this function".

EXAMPLE

```
// Determine the ID of the Util Control instance
$id = IPS_GetInstanceListByModuleID('{B69010EA-96D5-46DF-B885-
24821B8C8DBD}')[0];

UC_ResetLogMessageStatistics($id);

// Count anew from now on
echo 'Counting since ' . date('d.m.Y H:i:s',
UC_GetLogMessageStatistics($id)['ResetTimeStamp']);
```

WebHook Control

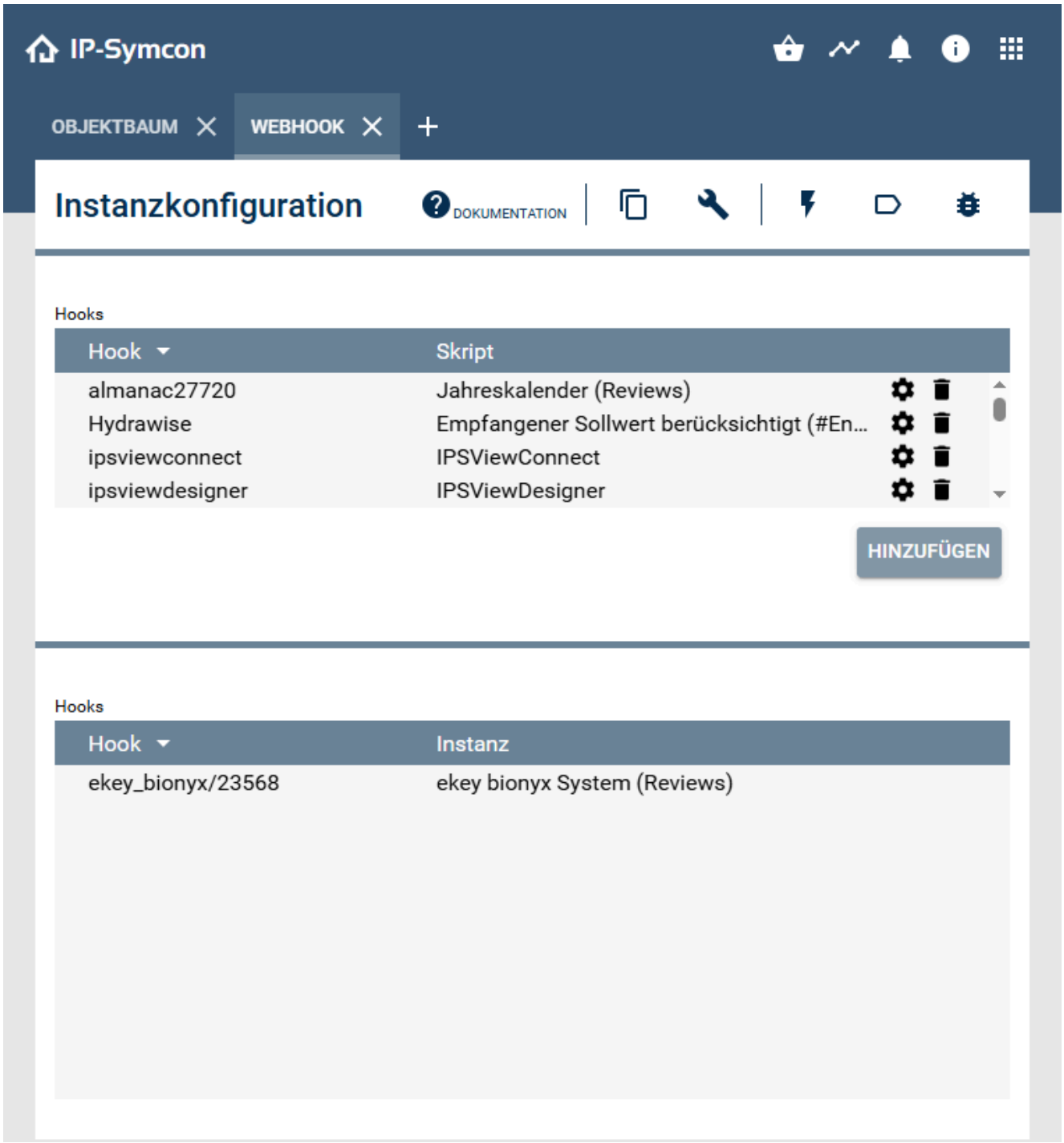
Require: Symcon >= 4.0

DESCRIPTION

The WebHook Control module offers the possibility to call up scripts via a browser. Calls outside the local network are also possible via the Symcon Connect service.
It also offers the option of supplying data to the script.



Scripts executed by the WebHook include these Systemvariables



DETERMINE THE BEST WEBHOOK

The called hook must not match exactly. The **Best Match**, which is the **longest match**, will always be used. This means that the hook path can be shorter than the hook called, for example, in the browser.
Example Configuration:
  Hook: /hook/ocpp, Target: 12345
  Hook: /hook/ocpp/extra, Target: 55555
Example Calls:
  Call: /hook/ocpp, Best Match: 12345
  Call: /hook/ocpppui, Best Match: 12345
  Call: /hook/ocpp/wallbox1, Best Match: 12345
  Call: /hook/ocpp/extra, Best Match: 55555
  Call: /hook/tester, No Match

WEBSOCKET SUPPORT

Since Symcon Version 5.2, the WebHook Control uses WebSockets and thus enables real-time transmission.
Received data can be recorded and output as follows:

```
// When receiving, the data end up in the php://input stream.
IPS_LogMessage("WebSocket", file_get_contents("php://input"));
```

INTEGRATION IN SYMCON

The WebHook Control is one of the core instances and can therefore be configured immediately.
Within a browser, the respectively linked script can be called with "IP:PORT/hook/HOOKNAME".
To set up a hook, "Add" of the first list must be clicked on in the "WebHook Control" instance. The call name of the hook must first be entered and then the script to be executed by the hook must be selected.



Calls via Connect Control are also possible.

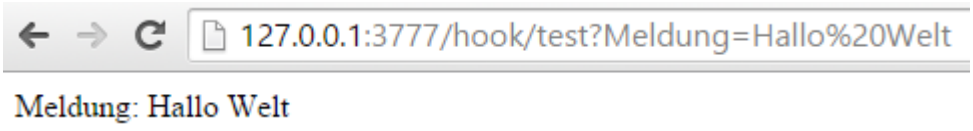
EXAMPLE

SCRIPT CALL

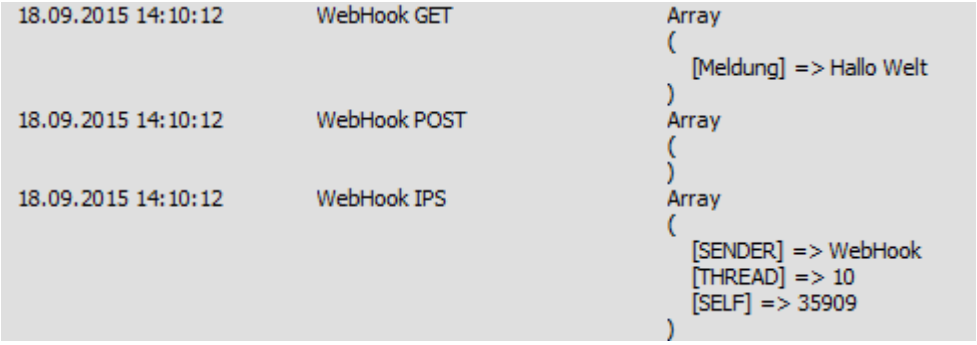
The sample script to be called is "WebHook TestScript" with ObjectID "35909".

```
IPS_LogMessage("WebHook GET", print_r($_GET, true));
IPS_LogMessage("WebHook POST", print_r($_POST, true));
IPS_LogMessage("WebHook IPS", print_r($_IPS, true));
IPS_LogMessage("WebHook RAW", file_get_contents("php://input"));
echo "Message: " . $_GET['Message'];
```

The call and the result would be as follows:



The data is passed to the script in arrays at Symcon.



AUTHENTICATION

An example of how automatic authentication was implemented via the Control module. "Symcon" was selected as the user name and "password" was selected as the password.



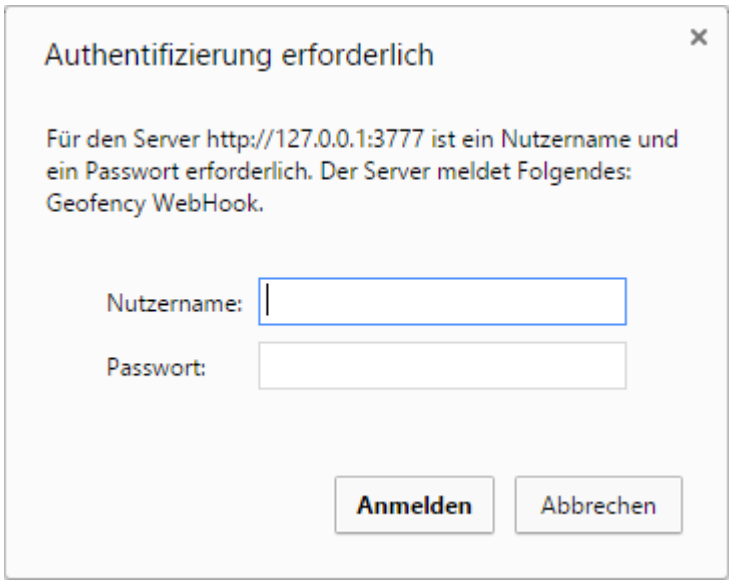
Authentication is always recommended to prevent unauthorized access to Symcon from outside.

Source code

```
if(!isset($_SERVER['PHP_AUTH_USER']))
$_SERVER['PHP_AUTH_USER'] = "";
if(!isset($_SERVER['PHP_AUTH_PW']))
$_SERVER['PHP_AUTH_PW'] = "";

if(($_SERVER['PHP_AUTH_USER'] != "Symcon") || ($_SERVER['PHP_AUTH_PW']
!= "passwort")) {
header('WWW-Authenticate: Basic Realm="Geofency WebHook"');
header('HTTP/1.0 401 Unauthorized');
echo "Authorization required";
return;
}
echo "Welcome to the protected area";
```

If the hook is called, the query for username and password appears. And the data is sent to the script via $post array.



INTEGRATION VIA PHP-MODULE

In addition to the call via script, it is also possible to react to hook calls within a PHP module using the ProcessHookData function.
To do this, a hook is registered per RegisterHook on the own InstanceID. Subsequently, the "ProcessHookData()" function is automatically called by the Hook Control.

EXAMPLE

The example HookServe contains the source code for the complete handling of a WebHook via PHP module with the modern IPSModuleStrict. Usage with the old IPSModule is listed here: HookServe (Legacy)

WC_PushMessage

Require: Symcon >= 5.2

```
bool WC_PushMessage (int $InstanceID, string $HookPath, string
$Message)
```

PARAMETERS

InstanceID	ID of the WebHook Control
HookPath	Path of the hook
Message	Message to send

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends message **Message** to all connected clients on hook **HookPath** of WebHook control with ID **InstanceID**.

EXAMPLE

```
// Sends "Hello World" to the "test" hook of the WebHook Control with
ID 12345
WC_PushMessage(12345, "/hook/test", "Hello World");
```

WC_PushMessageEx

Require: Symcon >= 5.3

```
bool WC_PushMessageEx (int $InstanceID, string $HookPath, string $Text,
string $DestinationAddress, int $DestinationPort)
```

PARAMETERS

InstanceID	ID des WebHook Control
HookPath	Path of the hook
Text	Message to send
DestinationAddress	IP address to send to
DestinationPort	Port to send to

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the message **Message** to a specific client on the hook **HookPath** of the WebHook Control with ID **InstanceID**. The client can be determined via the parameters of the IP address **Target address** and the target port **Port**.

EXAMPLE

```
// Sends "Hello World back" to the "test" hook of the WebHook Control
with ID 12345. In addition, the client with the address
$_SERVER["REMOTE_ADDR"] and port $_SERVER["REMOTE_PORT"] is determined
as the recipient.
WC_PushMessageEx(12345, "/hook/test", "Hello world back",
$_SERVER["REMOTE_ADDR"], $_SERVER["REMOTE_PORT"]);
```

WebServer

The WebServer module in IP-Symcon is used to make the Visualization available via another port. By default, the Visualization is available via port 3777. In particular, another WebServer is advantageous if SSL encryption is to be used.
Optionally, it is possible to enable basic authentication, which provides additional protection for the "user" folder.

Scripts executed by the WebServer include these system variables .

The JSON-RPC API (/api/) is available on any WebServer and is secured by remote access. Since IP-Symcon 4.0, the WebHooks (/hook/) are also available, each of which is secured by the authentication of the respective WebHook.

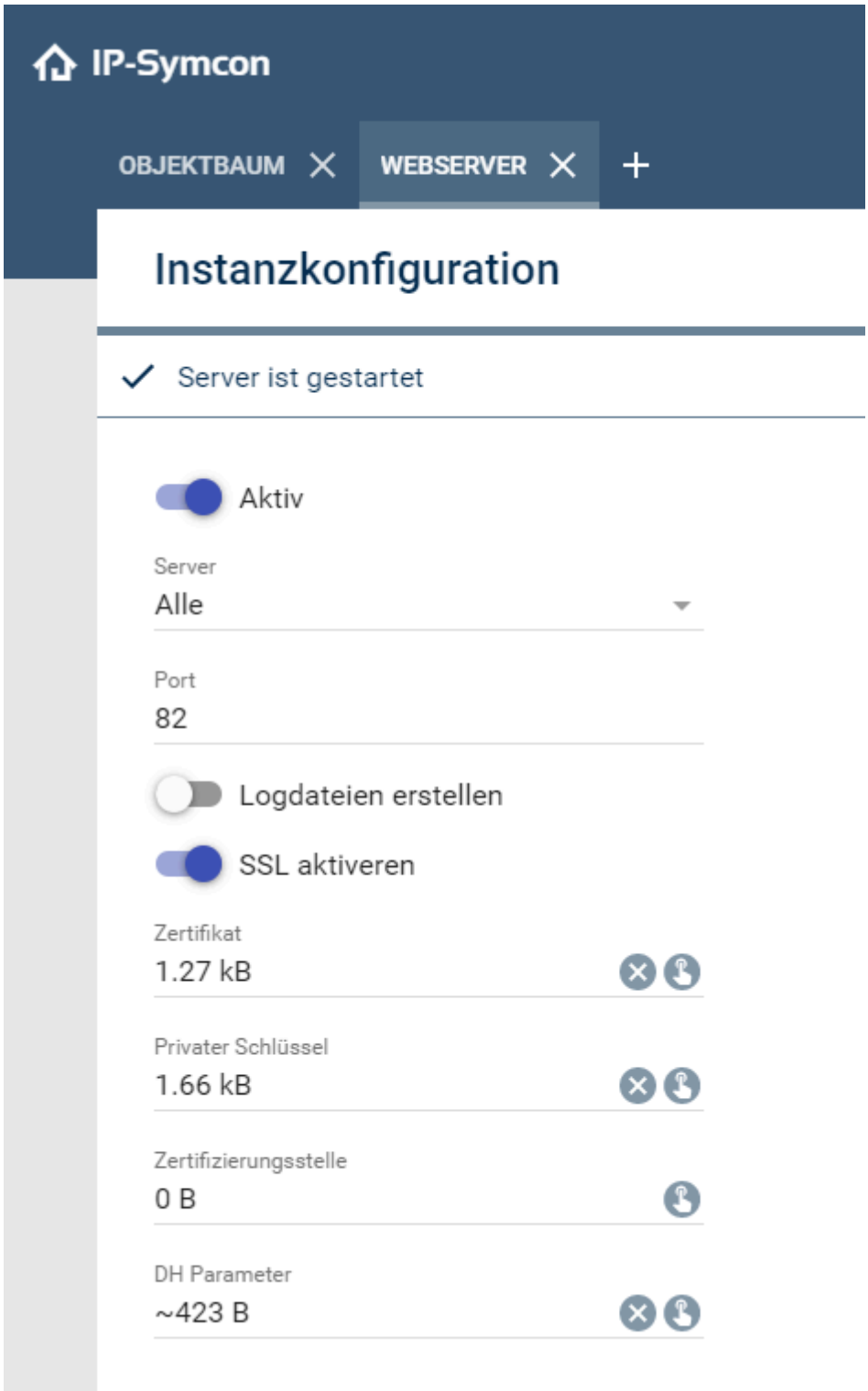
Since version 4.0, the WebServer is no longer added automatically. To be able to use it, it must be added via "Add object -> Add instance -> WebServer"

Basic authentication applies to the "user" folder only. The start page with the list of available configurators is always accessible without a password. If a Visualization password is to be assigned for each Visualization configurator, this can be set up in the configuration of the respective Visualization configurator

SSL ENCRYPTION

To use SSL, at least the certificate and the private key are required.

OPTION	DESCRIPTION
Certificate	Selection of the *.pem file containing the certificate
Private key	Selection of the *.pem file containing the private key
Certification body (optional)	Selection of the *.pem file which contains the certification authority for a trusted access
DH parameters (optional)	Select the dhparam.pem file for the Diffie-Hellman parameter which prevents a logjam. This brings increased safety



CREATE CERTIFICATE

Using OpenSSL, certificates and private keys can be created.

```
// Generate a private key
openssl genrsa -out pk.pem 1024

// Create a certificate signing request
openssl req -new -key pk.pem -out req.csr
```

```
// Self-sign the csr
openssl x509 -req -days 3650 -in req.csr -signkey pk.pem -out cert.pem
```

For a Browser to accept a certificate, IP-Symcon must be accessible via a public domain and a certificate issued by a recognized certificate authority must be present.

"USER" FOLDER

The "user" folder contains user-defined contents of IP-Symcon (e.g. additional display scripts).
The path to this folder depends on the operating system used.

SymBox: /var/lib/symcon/webfront/user/
Windows: C:\ProgramData\Symcon\webfront\user\
MacOS: /Library/Application Support/Symcon/webfront/user/
Linux: /var/lib/symcon/webfront/user/
Raspberry Pi: /var/lib/symcon/webfront/user/

If files are stored under another path, they will be deleted automatically during the next update of IP-Symcon. This is the only way to ensure that the installation of IP-Symcon always remains free of file version conflicts.

Up to and including IP-Symcon 6.4 the paths were the following:

SymBox: /var/lib/symcon/user/
Windows: C:\ProgramData\Symcon\user\
MacOS: /Library/Application Support/Symcon/user/
Linux: /var/lib/symcon/user/
Raspberry Pi: /var/lib/symcon/user/

PROTECT "USER" FOLDER

If the content has been stored in the special "user" folder, it is advisable to activate basic-authentication. In this way, these contents are additionally protected and require the respective username/password combination within the Visualization or the mobile Apps.
The authentication setup is located at the Special Switches.

If the authentication base has been activated in a previous IP-Symcon version (up to and including 5.0) in a web server, this is valid until it is deactivated.

LOG FILES

The option "Create log files" creates a file named access_12345.log in the "logs" directory of IP-Symcon for each WebServer, where the number 12345 stands for the InstanceID of the WebServer.
The created log files are Webalizer compatible and can thus be evaluated graphically.
The path to this folder depends on the operating system used.

SymBox: /var/log/symcon/
Windows: C:\ProgramData\Symcon\logs\
MacOS: /Library/Logs/Symcon/
Linux: /var/log/symcon/
Raspberry Pi: /var/log/symcon/

TIPS AND TRICKS

Forum: "Order SSL certificate and set it up in IP-Symcon"

I/O Instances

I/O Instances deliver data from devices or sources outside of IP Symcon and make it available. These often form the first interface to the physical device.

INTEGRATION IN IP-SYMCON

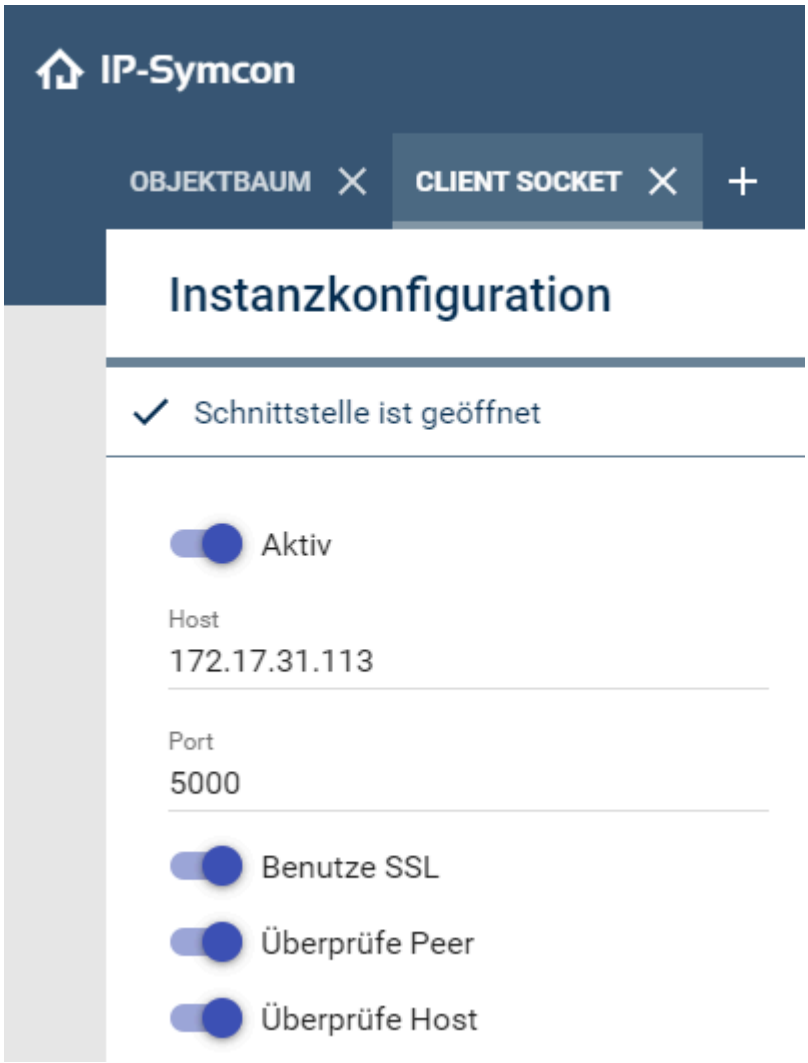
When creating a device instance, which requires an I/O instance as a parent instance, the appropriate instance is often created and set up automatically.
Only the settings need to be checked.
An I/O instance can also be inserted if required. This is often used in combination with a RegisterVariable, which has additional values available via System Variables.
I/O instances are also used in PHP Module Development. Similar to native modules, I/O instances are inserted there as an interface.

Client Socket

Require: Symcon >= 2.0
The client socket opens an interface, which is often used for communication between the gateway/ splitter instance and the connected device. It uses the TCP protocol.

INTEGRATION IN IP-SYMCON

On the configuration page, the IP/ host name must be specified under "Host" and the port of the device to be controlled under "Port".
If SSL is activated, it can also be selected whether the host or peer should be checked.
Only a click on "Apply" saves the settings and then an attempt is made to establish the connection.



CSCK_SendText

Require: Symcon >= 2.0

```
bool CSCK_SendText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the client socket to be updated
Text	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the string **Text** on the client socket with the ID **InstanceID**.

EXAMPLE

```
CSCK_SendText(12345, "Any data record"); // Sends the text "Any data
record" on the client socket with the ID 12345
```

HID

Require: Symcon >= 2.0
The HID (Human Interface Device) opens a USB interface, which is often used for communication between the gateway/ splitter instance and the connected device.

INTEGRATION IN IP-SYMCON

The device to be controlled must be specified on the configuration page under "Device".
Checking "Open HID device" sets the connection to active after "Apply".
All settings/changes are only saved after "Apply" is clicked.



HID_SendEvent

Require: Symcon >= 2.0

```
bool HID_SendEvent (int $InstanceID, int $ReportID, string $Text)
```

PARAMETERS

InstanceID	ID of the HID to be updated
ReportID	ID of the ReportID to be sent
Text	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the string **Text** and the data record type **ReportID** on the HID with the ID **InstanceID**.

EXAMPLE

```
HID_SendEvent(12345, 0, "Any data record"); //Sends the text "Any data
record" to the HID with the ID 12345
```

HTTP Client

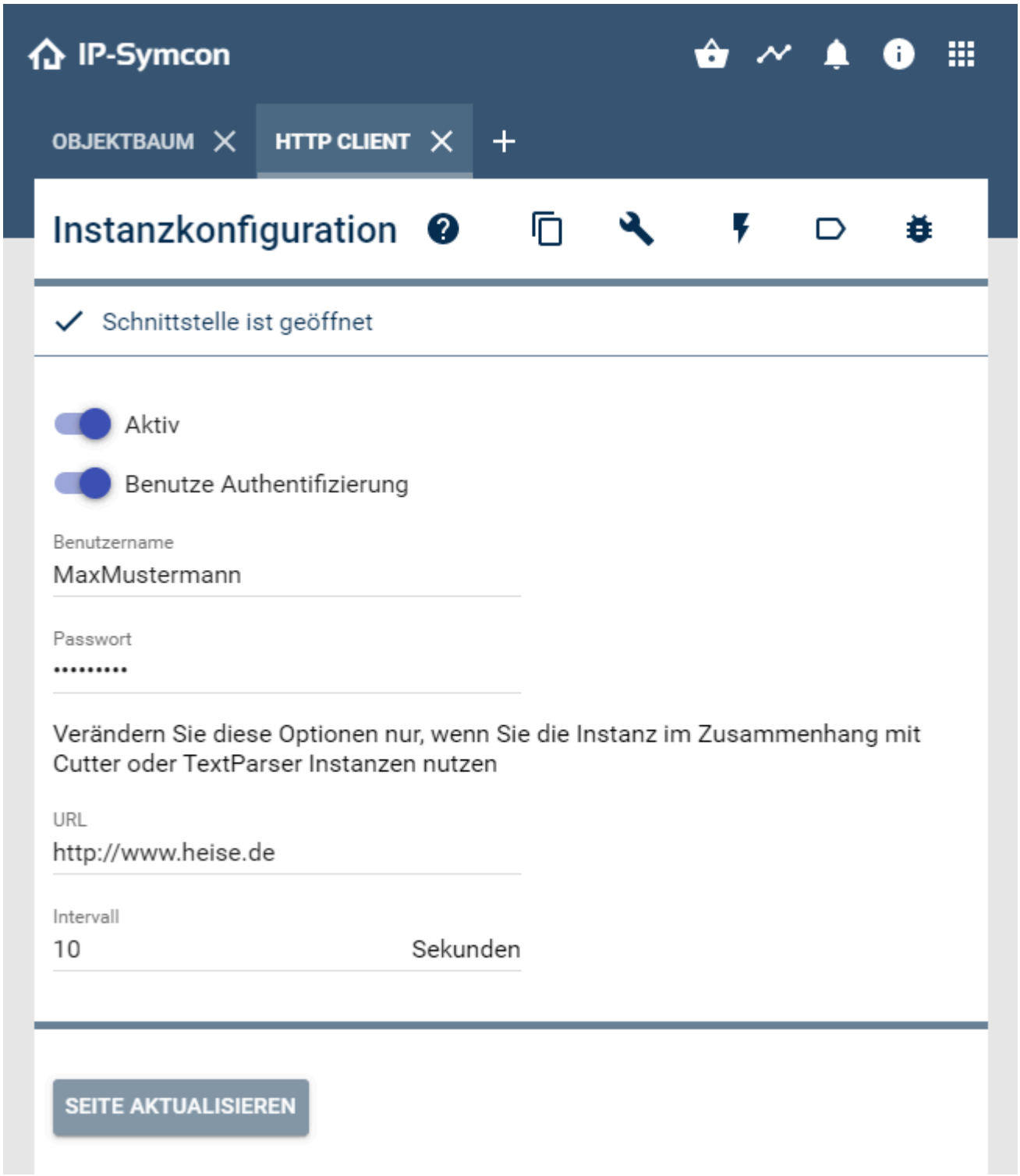
Require: Symcon >= 2.3
The HTTP Client reads data from a URL and makes it available.

INTEGRATION IN IP-SYMCON

Authentication via username and password can optionally be set up on the configuration page.

READ WEBSITE

If a website is to be read in, the URL must be entered.
The interval determines how often this query is carried out.


IP-Symcon
OBJEKTBAUM
HTTP CLIENT
Instanzkonfiguration
Schnittstelle ist geöffnet
Aktiv
Benutze Authentifizierung
Benutzername
MaxMustermann
Passwort
Verändern Sie diese Optionen nur, wenn Sie die Instanz im Zusammenhang mit Cutter oder TextParser Instanzen nutzen
URL
http://www.heise.de
Intervall
10
Sekunden
SEITE AKTUALISIEREN

WWW_UpdatePage

Require: Symcon >= 2.3

```
bool WWW_UpdatePage (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the HTTP Client to be updated

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Queries the URL of the HTTP Client with the ID **InstanceID**.



This function only has a function if the URL-mask is used on the configuration page.

EXAMPLE

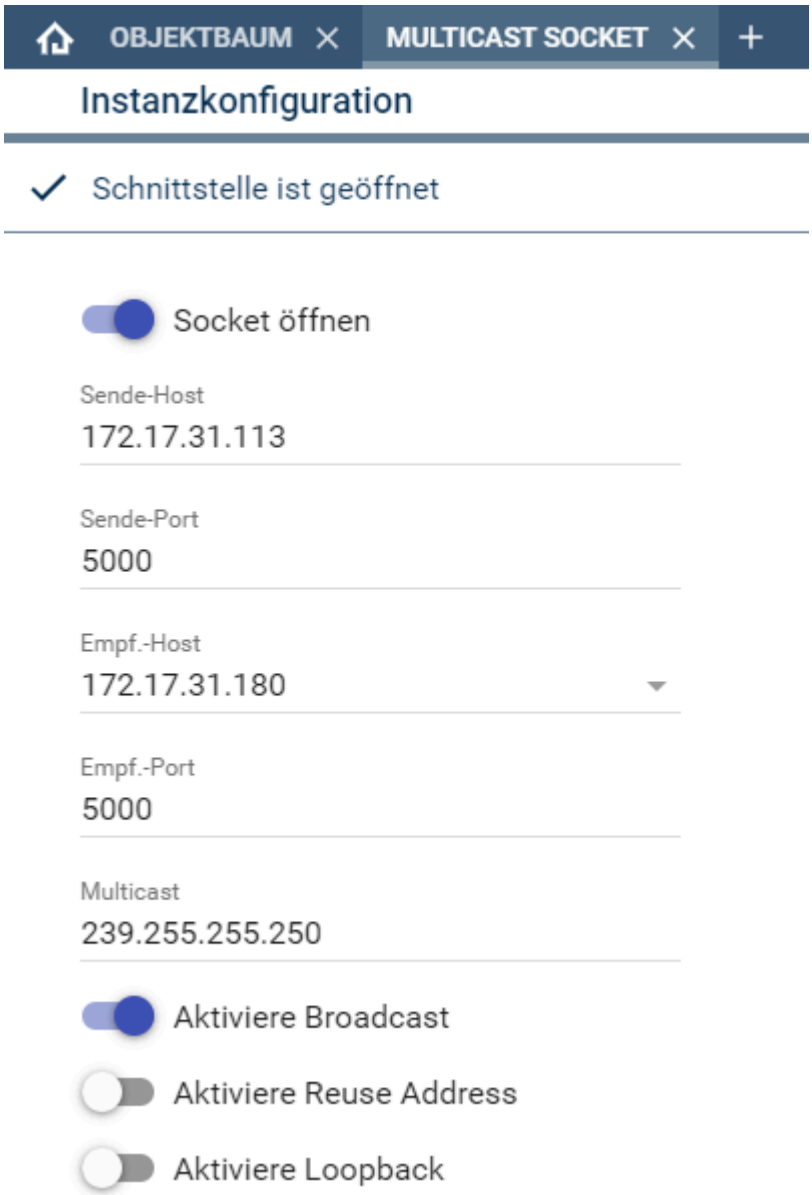
```
//Manual query of the configured URL of the HTTP Client with the
InstanceID 12345
WWW_UpdatePage(12345);
```

Multicast Socket

Require: Symcon >= 4.1
The Multicast Socket opens an interface which is often used for communication between the gateway/splitter instance and the connected device. It uses the UDP protocol.

INTEGRATION IN IP-SYMCON

The multicast socket can be set up and activated on the configuration page.
Only a click on "Apply" saves the settings and then an attempt is made to establish the connection with activated "Open socket".



SEND-HOST/PORT

IP and port of the device to be controlled.

RECEIVE-HOST/PORT

Determines on which network and via which IP and port the data is received.

MULTICAST

Multicast group address that is logged on and obeyed.

ACTIVATABLE OPTIONS

OPTION	DESCRIPTION
Broadcast	Allows broadcast messages to be sent
Reuse Address	Allows the socket to be connected to an address that is already in use.
Loopback	Activates the reception of Multicast-Packets on the send socket, provided this is a member of the Multicast Group.

MSCK_SendText

Require: Symcon >= 4.1

```
bool MSCK_SendText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the Multicast Socket to be updated
Text	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the string **Text** on the Multicast Socket with the ID **InstanceID**.

EXAMPLE

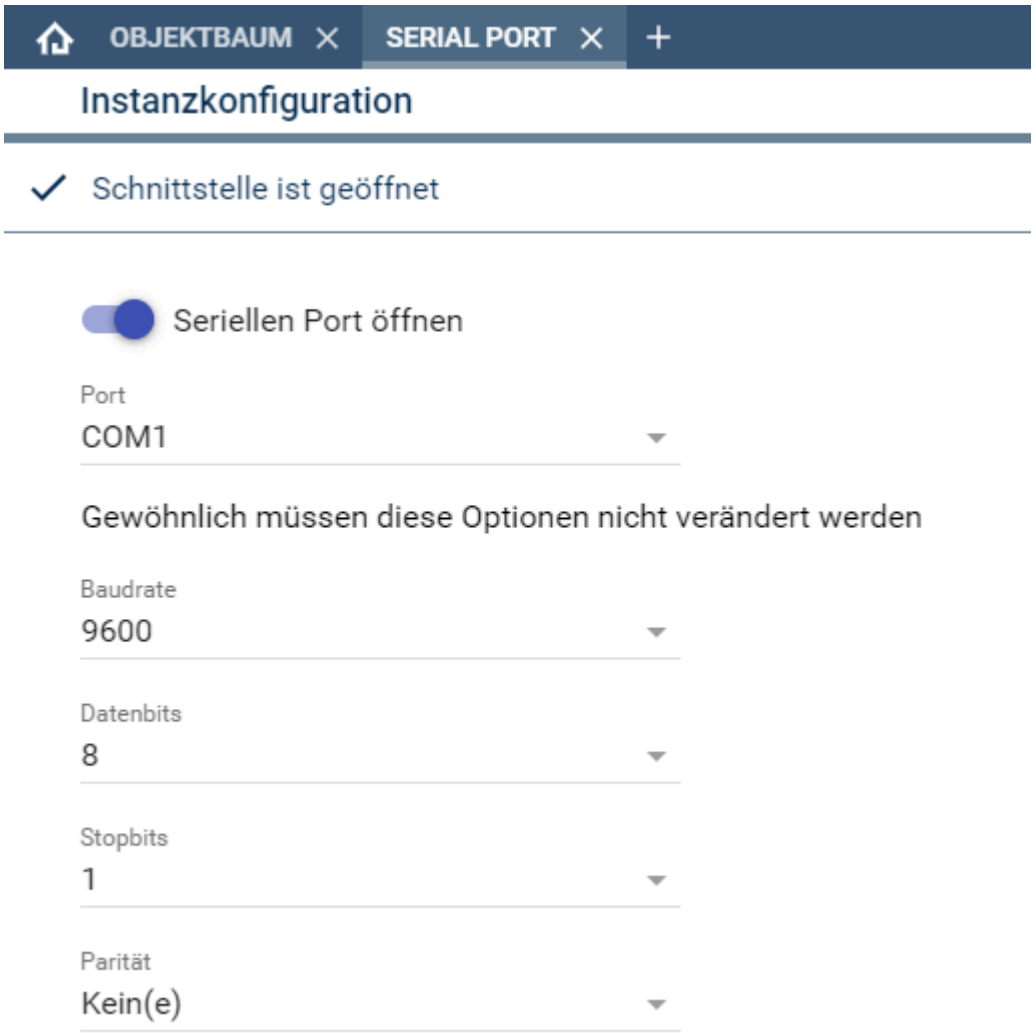
```
//Sends the string "Any data record" on the Multicast Socket with the
ID 12345
MSCK_SendText(12345, "Any data record");
```

Serial Port

Require: Symcon >= 2.0
The serial port opens an interface which is often used for communication between the gateway/splitter instance and the connected device.

INTEGRATION IN IP-SYMCON

The serial port, to which the device to be controlled is connected, must be specified on the configuration page.
If necessary, the baud rate, data bits, stop bits and parity can be set.
Only a click on "Apply" saves the settings and then an attempt is made to establish the connection.



SPRT_SendText

Require: Symcon >= 2.0

```
bool SPRT_SendText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the serial port to be updated
Text	The string to be sent

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sends the string Text on the serial port with the ID InstanceID.

EXAMPLE

```
//Sends the text "Any data record" to the serial port with the ID 12345
SPRT_SendText(12345, "Any data record");
```

SPRT_SetBreak

Require: Symcon >= 2.6

```
bool SPRT_SetBreak (int $InstanceID, bool $OnOff)
```

PARAMETERS

InstanceID	ID of the serial port to be updated
OnOff	Switches the break On **True** or Off **False**

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the break to **OnOff** (True = On; False = Off) on the serial port with the ID **InstanceID**.

EXAMPLE

```
// switches the break of the serial port instance with the ID 12345 to
"On"
SPRT_SetBreak(12345, true);
```

SPRT_SetDTR

Require: Symcon >= 2.6

```
bool SPRT_SetDTR (int $InstanceID, bool $OnOff)
```

PARAMETERS

InstanceID	ID of the serial port to be updated
OnOff	Switches the DTR On **True** or Off **False**

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the DTR to **OnOff** (True = On; False = Off) on the serial port with the ID **InstanceID**.

EXAMPLE

```
// switches the DTR of the serial port instance with the ID 12345 to
"On"
SPRT_SetDTR(12345, true);
```

SPRT_SetRTS

Require: Symcon >= 2.6

```
bool SPRT_SetRTS (int $InstanceID, bool $OnOff)
```

PARAMETERS

InstanceID	ID of the serial port to be updated
OnOff	Switches the RTS On **True** or Off **False**

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Switches the RTS to **OnOff** (True = On; False = Off) on the serial port with the ID **InstanceID**.

EXAMPLE

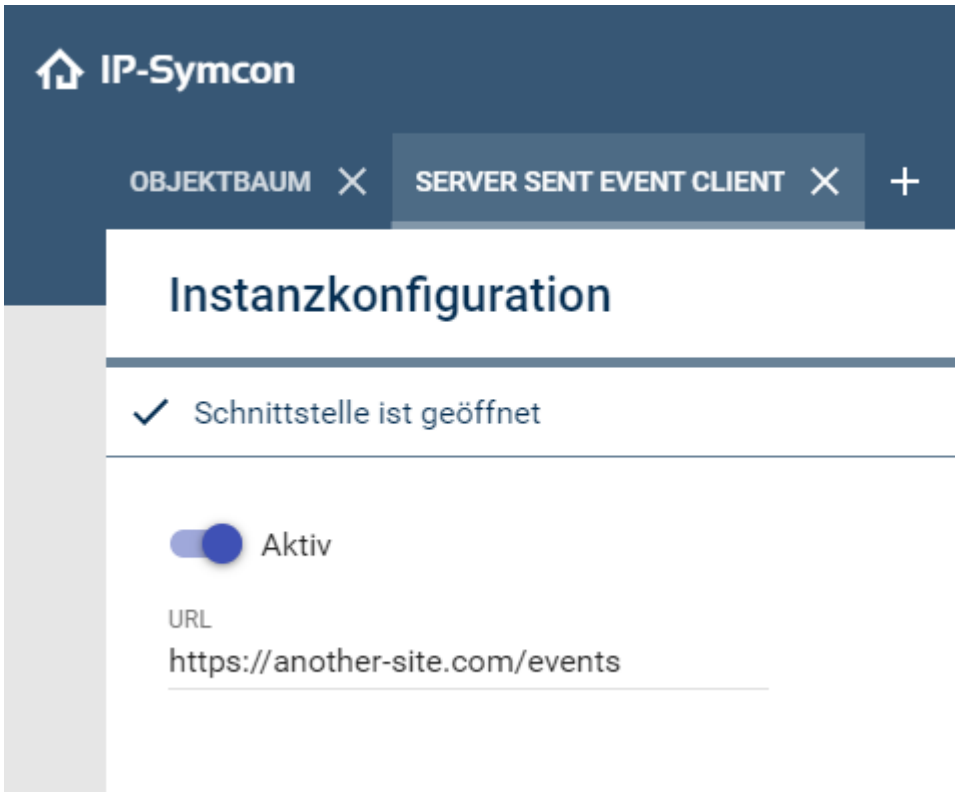
```
// switches the RTS of the serial port instance with the ID 12345 to
"On"
SPRT_SetRTS(12345, true);
```

Server Sent Event Client

The Server Sent Event Client opens an interface which automatically reacts to the Server Event. It uses a HTTP connection.

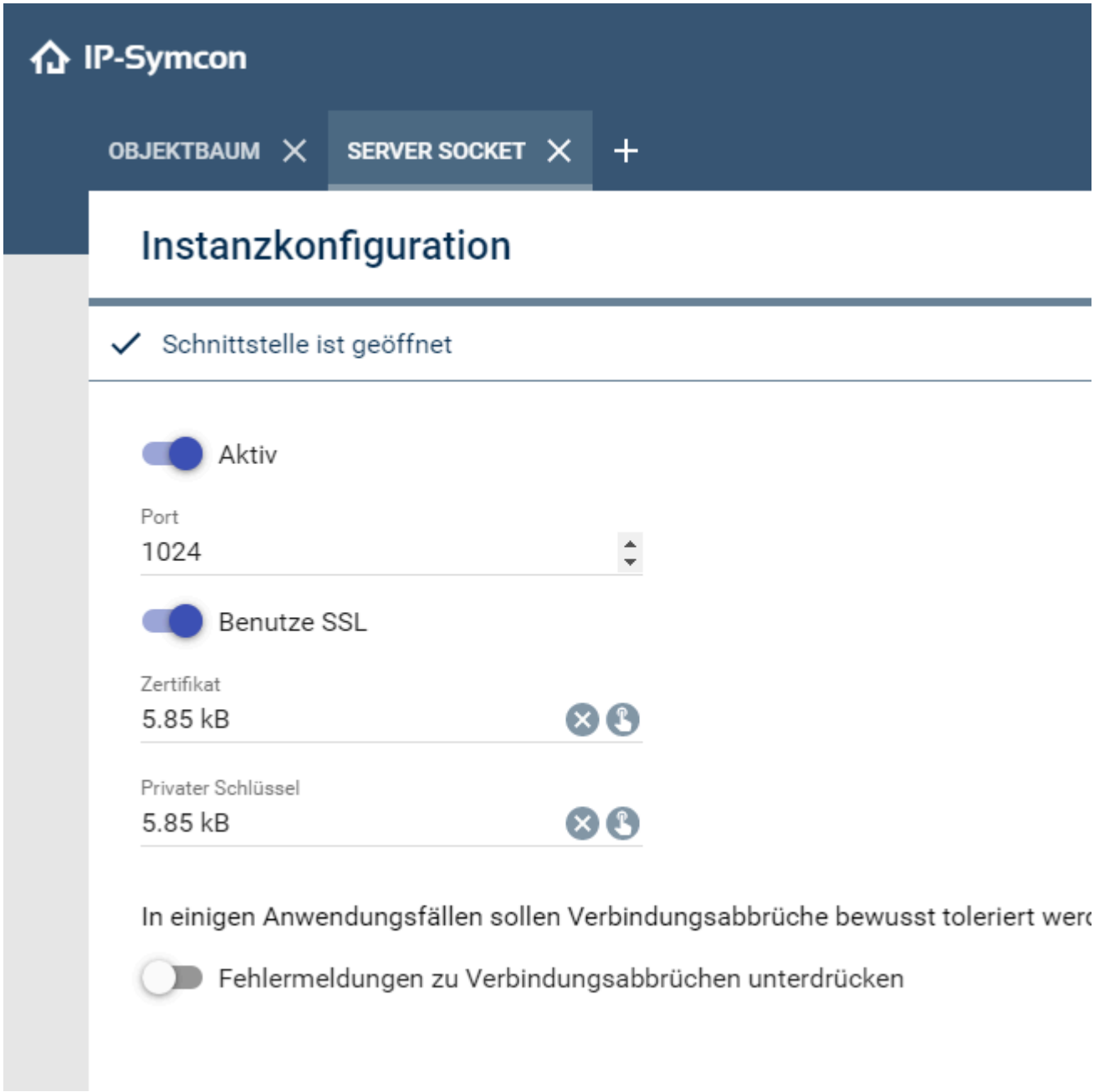
INTEGRATION IN IP-SYMCON

On the configuration page, the address of the server to which one is responding must be specified under "URL".
Only a click on "Apply" saves the settings and then an attempt is made to establish the connection.



Server Socket

Require: Symcon >= 4.1
The server socket opens a port for all networks on the server. The port can then be listened on and data can be received on it.
Only a click on "Apply" saves the settings and then the port is opened.
If SSL is activated, the certificate and the private key must be specified.



CREATE CERTIFICATE

Using OpenSSL, certificates and private keys can be created.

```
// Generate a private key
openssl genrsa -out pk.pem 1024

// Create a certificate signing request
openssl req -new -key pk.pem -out req.csr

// Self-sign the csr
openssl x509 -req -days 3650 -in req.csr -signkey pk.pem -out cert.pem
```



The certificate for the certification authority is optional.

SSCK_SendPacket

Require: Symcon >= 4.1

```
bool SSCK_SendPacket (int $InstanceID, string $Text, string $ClientIP,
int $ClientPort)
```

PARAMETERS

InstanceID	ID of the server socket to be updated
Text	The string to be sent
ClientIP	Recipient's IP
ClientPort	Recipient's port

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the string **Text** on the server socket with the ID **InstanceID**. The IP **ClientIP** and port **ClientPort** are used as recipient addressing.



If there is no connection from the IP address on the corresponding port, the string to be sent is discarded

EXAMPLE

```
//Sends the packet with the data record "Any data record" on the server
socket with ID 12345 to the IP:Port 192.168.1.123:1234
SSCK_SendPacket(12345, "Any data record", "192.168.1.123", 1234);
```

SSCK_SendText

Require: Symcon >= 4.1

```
bool SSCK_SendText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the server socket to be updated
Text	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the string **Text** on the server socket with the ID **InstanceID**.

EXAMPLE

```
SSCK_SendText(12345, "Any data record"); //Sends the text "Any data
record" to the server socket with the ID 12345
```

UDP Socket

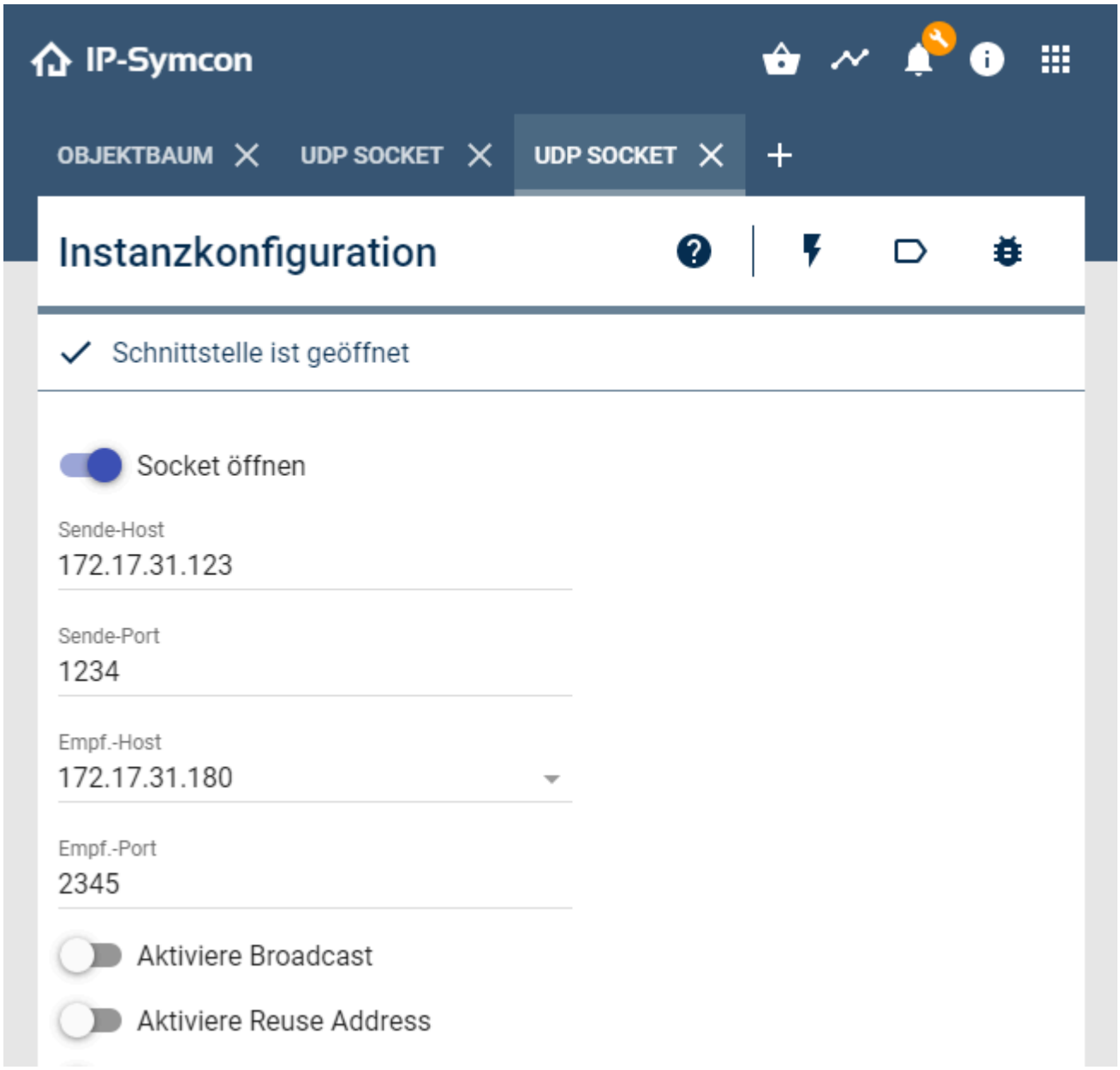
Require: Symcon >= 3.3
The UDP socket opens an interface which is often used for communication between the gateway/splitter instance and the connected device. It uses the UDP protocol.

INTEGRATION IN IP-SYMCON

A new UDP socket instance can be created via "+" in the object tree.
The following settings can be made on the configuration page.

OPTION	DESCRIPTION
Open Socket	Whether the socket should be open or closed.
Send-Host	Which IP address the data should be sent to.
Send-Port	On which port the data should be sent.
Receive-Host	On which TCP interface the data should be listened to.
Receive-Port	On which port the data should be listened to.
Activate Broadcast	When activated, the UDP socket sends to the broadcast address of the selected Receive-Host interface. The entered Send-Host address is irrelevant.
Activate Reuse Address	Allows other programs to listen to the Receive-Port when the "Reuse address" is also activated.

Only a click on "Apply" saves the settings and then an attempt is made to establish the connection.


IP-Symcon
OBJEKTBAUM
UDP SOCKET
UDP SOCKET
Instanzkonfiguration
Schnittstelle ist geöffnet
Socket öffnen
Sende-Host
172.17.31.123
Sende-Port
1234
Empf.-Host
172.17.31.180
Empf.-Port
2345
Aktiviere Broadcast
Aktiviere Reuse Address

USCK_SendPacket

Require: Symcon >= 5.0

```
bool USCK_SendPacket (int $InstanceID, string $Text, string $ClientIP,
int $ClientPort)
```

PARAMETERS

InstanceID	ID of the UDP socket to be updated
Text	The string to be sent
ClientIP	Recipient's IP
ClientPort	Recipient's port

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the string **Text** on the UDP socket with the ID **InstanceID**. The IP **ClientIP** and port **ClientPort** are used as recipient addressing.



If ClientIP is left blank, content is sent to the host set in the UDP socket. If broadcast is active, the ClientIP parameter is ignored and the content of $Text is sent to the broadcast address. If ClientPort = 0, then the port set in the UDP socket is used.

EXAMPLE

```
//Sends the packet with the data record "Any data record" on the server
socket with ID 12345 to the IP:Port 192.168.1.123:1234
USCK_SendPacket(12345, "Any data record", "192.168.1.123", 1234);
```

USCK_SendText

Require: Symcon >= 3.3

```
bool USCK_SendText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the UDP socket to be updated
Text	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the string **Text** on the UDP socket with the ID **InstanceID**.

EXAMPLE

```
USCK_SendText(12345, "Any data record"); // Sends the text "Any data
record" to the UDP socket with the ID 12345
```

Virtual I/O

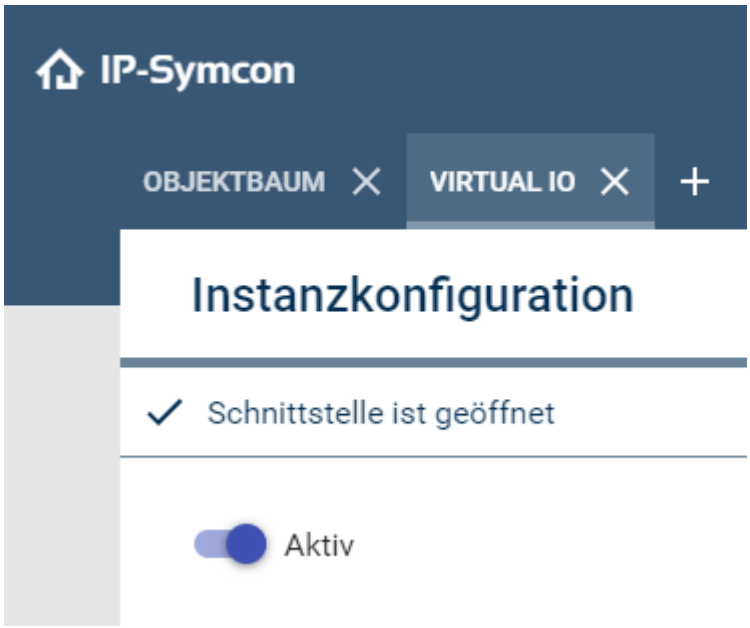
Require: Symcon >= 3.3
The virtual I/O opens an interface that is used for communication between the gateway/splitter instance and the connected device. It does not use a specific protocol, but is used for test purposes to ensure correct behavior of the connected entities.
The virtual I/O is able to send data packets of the type Simple and Extended (Socket). The rule is that functions with "text" use such simple data packets. Connect, Disconnect and Functions with "Packet" use the extended data packet (Socket).

INTEGRATION IN IP-SYMCON

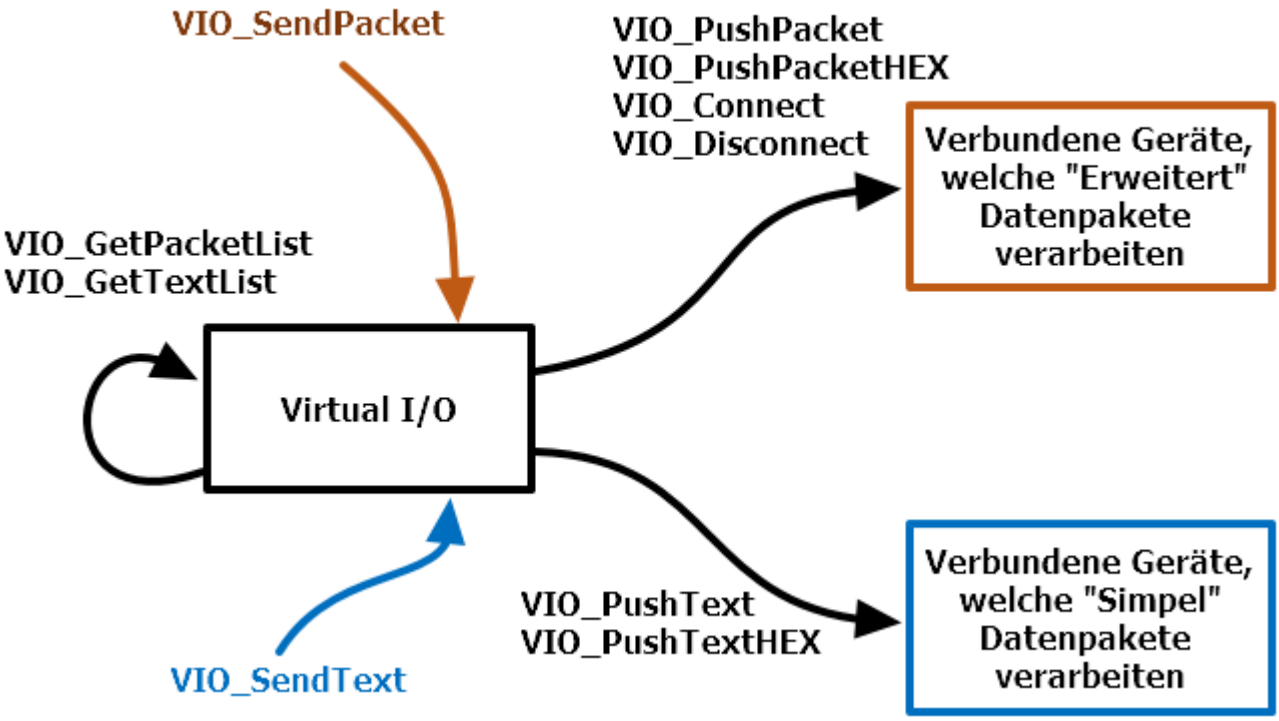
A new virtual I/O instance can be created via "+" in the object tree.
The following settings can be made on the configuration page.

OPTION	DESCRIPTION
Active	Whether the interface should be open or closed.

Only a click on "Apply" saves the settings and then an attempt is made to establish the connection.



For a better understanding, the graphic shows which direction and data packet types are used by the individual commands.


VIO_SendPacket
VIO_PushPacket
VIO_PushPacketHEX
VIO_Connect
VIO_Disconnect
Verbundene Geräte, welche "Erweitert" Datenpakete verarbeiten
VIO_GetPacketList
VIO_GetTextList
Virtual I/O
VIO_PushText
VIO_PushTextHEX
Verbundene Geräte, welche "Simpel" Datenpakete verarbeiten
VIO_SendText

VIO_Connect

Require: Symcon >= 5.1

```
bool VIO_Connect (int $InstanceID, string $ClientIP, int $ClientPort)
```

PARAMETERS

InstanceID	ID of the Virtual I/O to be updated
ClientIP	The ClientIP to be connected
ClientPort	The port to be connected

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Extended (Socket) to all instances connected to the Virtual I/O with the ID **InstanceID**. This includes: Type = 1 (Connect), Buffer = "" (Buffer empty), ClientIP = **ClientIP** and ClientPort = **ClientPort**.

EXAMPLE

```
// Sends a data packet to the virtual I/O instance with the ID 12345
with the IP-address 192.168.0.8 and port 502
VIO_Connect(12345, "192.168.0.8", 502);
```

VIO_Disconnect

Require: Symcon >= 5.1

```
bool VIO_Disconnect (int $InstanceID, string $ClientIP, int
$ClientPort)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated
ClientIP	The ClientIP to be connected
ClientPort	The port to be connected

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Extended (Socket) to all instances connected to the Virtual I/O with the ID **InstanceID**. This includes: Type = 2 (Disconnect), Buffer = "" (Buffer empty), ClientIP = **ClientIP** and ClientPort = **ClientPort**.

EXAMPLE

```
// Sends a data packet to the virtual I/O instance with the ID 12345
with the IP-address 192.168.0.8 and port 502
VIO_Disconnect(12345, "192.168.0.8", 502);
```

VIO_GetPacketList

Require: Symcon >= 5.1

```
array VIO_GetPacketList (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated

RETURNS

Supplies an array of extended Data-Packets

DESCRIPTION

Returns an array of all received data packets of the virtual I/O instance with the ID **InstanceID** of the type Extended (Socket).

EXAMPLE

```
// Returns an array of all extended data packets of the virtual I/O
instance with the ID 12345
$allPackets = VIO_GetPacketList(12345);
print_r($allPackages);

/* returns, for example:
Array
(
[0] => Array
(
[Type] => 0
[Buffer] => Hello world
[ClientIP] => 192.168.0.8
[ClientPort] => 502
)

[1] => Array
(
[Type] => 0
[Buffer] => IP-Symcon
[ClientIP] => 192.168.0.4
[ClientPort] => 3777
)
...
```

```
)
*/
```

VIO_GetTextList

Require: Symcon >= 5.1

```
array VIO_GetTextList (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated

RETURNS

Returns an array of simple Data-Packets

DESCRIPTION

Returns an array of all received data packets of the virtual I/O instance with the ID **InstanceID** of the type Simple.

EXAMPLE

```
// Returns an array of all simple data packages of the virtual I/O
instance with the ID 12345
$allTexts = VIO_GetTextList(12345);
print_r($allTexts);

/* returns, for example:
Array
(
[0] => Hello world
[1] => IP-Symcon
...
)
*/
```

VIO_PushPacket

Require: Symcon >= 5.1

```
bool VIO_PushPacket (int $InstanceID, string $Text, string $ClientIP,
int $ClientPort)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated
Text	The string to be sent
ClientIP	On the ClientIP to be sent
ClientPort	On the port to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Extended (Socket) to all instances connected to the Virtual I/O with the ID **InstanceID**. This includes: Type = 0 (Data), Buffer = **Text**, ClientIP = **ClientIP** and ClientPort = **ClientPort**.

EXAMPLE

```
// Sends the text "Hello World" to the virtual I/O instance with the ID
12345, and with the IP address 192.168.0.8 and port 502
VIO_PushPacket(12345, "Hello World", "192.168.0.8", 502);
```

VIO_PushPacketHEX

Require: Symcon >= 5.1

```
bool VIO_PushPacketHEX (int $InstanceID, string $Text, string
$ClientIP, int $ClientPort)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated
Text	The string to be sent in HEX
ClientIP	On the ClientIP to be sent
ClientPort	On the port to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Extended(Socket) to all instances connected to the Virtual I/O with the ID **InstanceID**. This includes: Type = 0 (Data), Buffer = **Text**, ClientIP = **ClientIP** and ClientPort = **ClientPort**.



The **Text** parameter is case-insensitive and spaces are ignored. Furthermore, only the characters 0..9, Aa..Ff are allowed.

EXAMPLE

```
// Sends the hex string "1a2fee" with the IP address 192.168.0.8 and
port 502 to the virtual I/O instance with the ID 12345
VIO_PushPacketHEX(12345, "1A 2F EE", "192.168.0.8", 502);
```

VIO_PushText

Require: Symcon >= 3.3

```
bool VIO_PushText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated
Text	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Simple to all instances connected to the virtual I/O with the ID **InstanceID**. This includes: Buffer = **Text**.

EXAMPLE

```
// Sends the text "Hello world" to the virtual I/O instance with the ID
12345
VIO_PushText(12345, "Hello World");
```

VIO_PushTextHEX

Require: Symcon >= 3.3

```
bool VIO_PushTextHEX (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated
Text	The string to be sent in HEX

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Simple to all instances connected to the virtual I/O with the ID **InstanceID**. This includes: Buffer = **Text**.



The **Text** parameter is case-insensitive and spaces are ignored. Furthermore, only the characters 0..9, Aa..Ff are allowed.

EXAMPLE

```
// Sends the hex string "1a2fee" to the virtual I/O instance with ID
12345
VIO_PushTextHEX(12345, "1A 2F EE");
```

VIO_SendPacket

Require: Symcon >= 5.1

```
bool VIO_SendPacket (int $InstanceID, string $Text, string $ClientIP,
int $ClientPort)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated
Text	The string to be sent
ClientIP	On the ClientIP to be sent
ClientPort	On the port to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Extended(Socket) to the Virtual I/O with the ID **InstanceID** itself. This includes: Type = 0 (Data), Buffer = **Text**, ClientIP = **ClientIP** and ClientPort = **ClientPort**.

EXAMPLE

```
// Sends the text "Hello world" with the IP address 192.168.0.8 and the
port 502 to the Virtual I/O instance with the ID 12345
VIO_PushPacket(12345, "Hello World", "192.168.0.8", 502);
```

VIO_SendText

Require: Symcon >= 3.3

```
bool VIO_SendText (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the virtual I/O to be updated
Text	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends a data packet of the type Simple to the virtual I/O with the ID **InstanceID** itself. This includes: Buffer = **Text**.

EXAMPLE

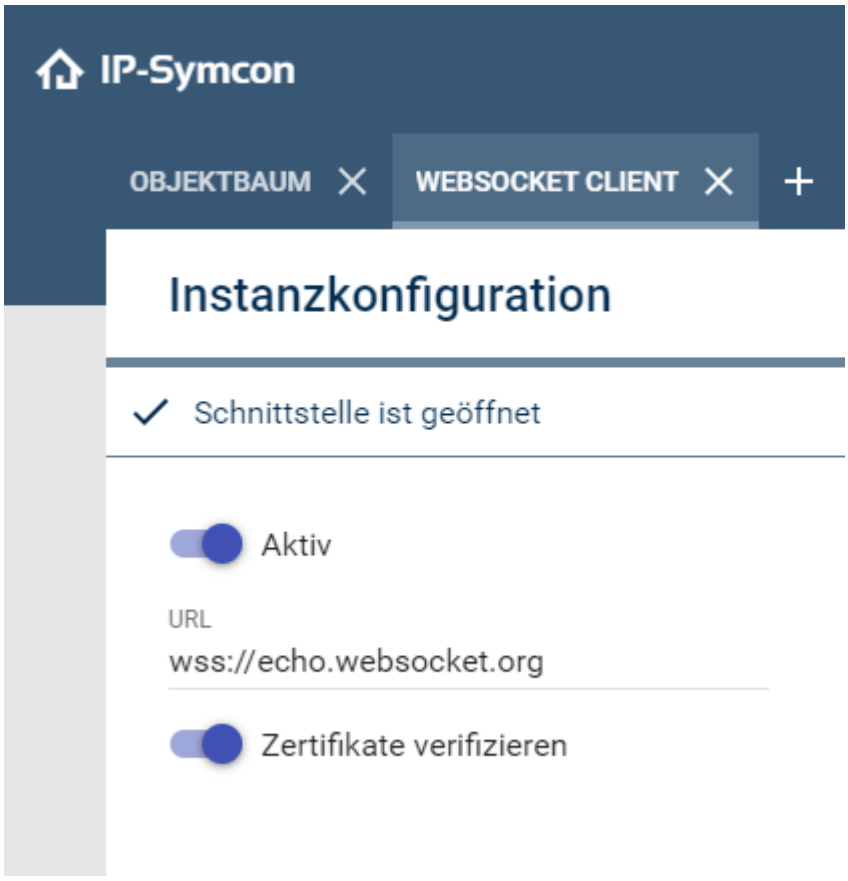
```
// Sends the text "Hello world" to the virtual I/O instance with ID
12345
VIO_SendText(12345, "Hello World");
```

WebSocket Client

Require: Symcon >= 5.5
The WebSocket Client opens an interface which is often used for bidirectional communication with web applications/web servers. It uses the TCP protocol.

INTEGRATION IN IP-SYMCON

The address of the web application/ web server must be specified under "URL" on the configuration page. If "Verify Certificates" is activated, the SSL certificates are checked. Only a click on "Apply" saves the settings and then an attempt is made to establish the connection.



WSC_SendMessage

Require: Symcon >= 5.5

```
bool WSC_SendMessage (int $InstanceID, string $Message)
```

PARAMETERS

InstanceID	ID of the WebSocket client to be updated
Message	The string to be sent

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sends the message **Message** to the WebSocket Client with the ID **InstanceID**.

EXAMPLE

```
// Sends the message "Hello World" to the WebSocket Client instance
with ID 12345
WSC_SendMessage(12345, "Hello World");
```

Backups

Backup	Creates a backup via SFTP, FTP or FTPS
SB_CreateBackup	Creates or updates a backup

Backup

Require: Symcon >= 6.3
Creates a backup via SFTP, FTP or FTPS.

RANGE OF FUNCTIONS

Creates a backup via SFTP, FTP or FTPS on a server
Updates a backup via SFTP, FTP or FTPS on a server

SOFTWARE INSTALLATION

Install the 'Backup (SFTP/FTP/FTPS)' module via the Module Store.

SET UP THE INSTANCES IN IP-SYMCON

The 'Backup (SFTP/FTP/FTPS)' module can be found under 'Add instance' using the quick filter.
Further information on adding instances can be found in the Instances documentation
Configuration page:

NAME	DESCRIPTION
Connection type	Selection of the connection type
Host	IP address of the server
Port	Port on which the client connects to the server. For FTPS and FTP this is normally 21, for SFTP this is port 22
User name	User name for the SFTP connection
Password	Password for the SFTP connection
Mode	Mode for which a backup is to be created
Change folder to	A new folder is created at the beginning of the period
Destination folder	Folder on the server where the backup is to be created
Search target folder	Browse through the folders and a valid path can be created
Activate automatic backups	Activates a daily update
Daily at	Time when the update starts daily
Expert options	----------
Size limit	If a file is larger than this limit, it is ignored
Filtered folders	Filter to not transfer certain folders
Create backup	Button which starts an update immediately.
Test connection	Tests whether a connection can be established

Mode:
Full backup: Creates a complete copy in a separate folder according to the pattern: symcon-backup-{year}-{month}-{day}-{hour}-{minute}-{second}
Incremental backup: Updates an existing backup.
If the 'Change by folder' option is not set to 'Never', the folder is changed after the set time.
The folders that are created during the incremental backup follow the following pattern:

Never: symcon-backup
Week: symcon-backup-{year}-0{calendar-week}
Month: symcon-backup-{year}-{month}
Year: symcon-backup-{year}

STATUS VARIABLES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

NAME	TYPE	DESCRIPTION
Last completed backup	Integer	Time at which the last backup was completed
Megabytes transferred	Float	Megabytes transferred during the last update

SB_CreateBackup

```
bool SB_CreateBackup (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the command could be executed successfully, it returns TRUE as the result, otherwise FALSE

DESCRIPTION

Creates or updates a backup according to the settings of the instance.

EXAMPLE

```
SB_CreateBackup(12345);
```

Legacy

RRDTool	Direct integration of RRDTool
RRD_Exexute	Directs a command to the RRD Tool Library and executes it. Once the command has been executed, the execution will be proceed.
Shutter Control (legacy)	Functions for controlling shutter components (legacy)
SC_Move	moves the shutter to a specific position
SC_MoveDown	moves the shutter down to the end position
SC_MoveUp	moves the shutter up to the end position
SC_Stop	stops a motion
USBMapper	Automatically sets the correct USB port to the entered serial ports.
USBM_FixPorts	Checks whether the USB ports are still configured correctly
Web Graph	Any graphs can be displayed independently via the WebHook
Wunderground Weather	Gets weather data (current/hourly/12-hourly/weather warning) via API from www.wunderground.com
WGW_UpdateStormWarningData	Updates the severe weather warnings of the Weatherground module
WGW_UpdateWeatherData	Updates the weather data (current/hourly/daily) of the Weatherground module

RRDTool

RRDtool is the OpenSource industry standard, high performance data logging and graphing system for time series data. Use it to write your custom monitoring shell scripts or create whole applications using its Perl, Python, Ruby, TCL or PHP bindings.
The RRDTool module is a direct integration of RRDTool as PHP function in IP-Symcon.
No other functions are offered.
The RRDTool manual is the first point of contact to find information about using this module.
The current version number you can find out as follows:

```
echo RRD_Execute("version -");
```

RRD_Exexute

```
bool RRD_Execute (int $InstanceID, string $Command)
```

PARAMETERS

InstanceID	ID of the device to be switched
Command	RRD command. See

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION



The feature was removed with IP Symcon 2.4.

Directs a command to the RRD Tool Library and executes it. Once the command has been executed, the execution will be proceed.

EXAMPLE

```
//No example available
```

Shutter Control (legacy)

This is a deprecated version. From version 5.0 onwards it is recommended to use the new Shutter Control version.

To make the IP-Symcon shutter module hardware-independent, the module needs a script that takes care of passing the functions to the hardware.
Anyone with the following hardware can easily copy and use the script. Other hardware is easy to add. (More on that below the script).

Scripts executed by the shutter control include these System variables

Eaton Xcomfort Shutter
1-Wire Shutter
Homematic (Thanks to hengesb)
LCN (Thanks to philipp)
FS20MS
This script normally only needs to exist once in IP-Symcon and can be accessed by any number of instances. Nothing needs to be edited.

CALIBRATE SHUTTERS

In the second step, the times it takes for the shutter to move to the individual positions are measured. This calibration must be carried out twice. For up and down. To start, the corresponding button is pressed, whereupon the shutter should start moving. The button must be pressed again when the position labelled on it is reached.

After each teach-in, the values must be transferred with the “Set” button.

TIPS & TRICKS

When moving to 0% and 100%, five seconds are added to the driving time to reach a defined position.

If the runtime of the shutter is above the PHP script limit (180sec), it must be increased in php.ini. (see PHP)

INFORMATION FOR LCN USERS

With the script, the shutter module can be used universally to control the shutter/blind via the outputs as well as via the relays.
If the shutter module is to control a shutter/blind that is connected to the two outputs of a UPP/SH/HU, then the instance of output 1 of the module must be selected under “Instance1” and the instance of output 2 of the module for “instance2”. This configuration then corresponds to the standard direction, as it is activated from the LCN-Pro by moving up or down. If the port is reversed, it is only necessary to change the assignment of output 1 and output 2 to the settings “Instance1” and “Instance2”.
When operating with relays, the first relay of the two must always be specified for “Instance1". The first relay is the one that switches the power ON/OFF. “Instance2” is always the relay that controls the direction.
If the direction of travel is to be rotated, this must be changed manually in the script. In the “case block for LCN” of the shutter script, there is a corresponding comment in both places.
If differently wired shutters/blinds on the relays are used, two shutter scripts must therefore be used for the respective configurations.

SHUTTER SCRIPT

```
//Variables provided by ShutterControl Module
//IPS_LogMessage("InstanceID", $_IPS['INSTANCE']); /* InstanceID */
//IPS_LogMessage("Direction", $_IPS['DIRECTION']); /* {0..2} Stop, Up,
Down */
//IPS_LogMessage("Duration", $_IPS['DURATION']); /* ms */

if($_IPS['SENDER'] != "ShutterControl") {
die("This script can only be started by the ShutterControl Module");
}

define("SC_DIRECTION_STOP", 0);
define("SC_DIRECTION_UP", 1);
define("SC_DIRECTION_DOWN", 2);

$instance = IPS_GetInstance($_IPS['INSTANCE']);
switch($instance['ModuleInfo']['ModuleID']) {

//Siemens Device S7/LOGO
case "{932076B1-B18E-4AB6-AB6D-275ED30B62DB}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
S7_WriteBit($_IPS['INSTANCE'], false);
S7_WriteBit($_IPS['INSTANCE2'], false);
break;

case SC_DIRECTION_UP:
S7_WriteBit($_IPS['INSTANCE'], true);
```

```
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
S7_WriteBit($_IPS['INSTANCE'], false);
}
break;

case SC_DIRECTION_DOWN:
S7_WriteBit($_IPS['INSTANCE2'], true);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
S7_WriteBit($_IPS['INSTANCE2'], false);
}
break;
}
break;

//FS20
case "{48FCFDC1-11A5-4309-BB0B-A0DB8042A969}":
$running = CreateVariableByName($_IPS['INSTANCE'], "Moving",
VARIABLETYPE_BOOLEAN);
$value = GetValue(IPS_GetObjectIDByIdent("StatusVariable",
$_IPS['INSTANCE']));
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
if(GetValue($running)) {
FS20_SwitchMode($_IPS['INSTANCE'], $value);
SetValue($running, false);
}
break;

case SC_DIRECTION_UP:
if(!GetValue($running)) {
FS20_SwitchMode($_IPS['INSTANCE'], true);
SetValue($running, true);
}
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
FS20_SwitchMode($_IPS['INSTANCE'], true);
SetValue($running, false);
}
break;

case SC_DIRECTION_DOWN:
if(!GetValue($running)) {
FS20_SwitchMode($_IPS['INSTANCE'], false);
SetValue($running, true);
}
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
FS20_SwitchMode($_IPS['INSTANCE'], false);
SetValue($running, false);
}
break;
}
break;

//EnOcean
case "{1463CAE7-C7D5-4623-8539-DD7ADA6E92A9}":
switch($_IPS['DIRECTION']) {
```

```
case SC_DIRECTION_STOP:
ENO_ShutterStop($_IPS['INSTANCE']);
break;

case SC_DIRECTION_UP:
ENO_ShutterMoveUp($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
ENO_ShutterStop($_IPS['INSTANCE']);
}
break;

case SC_DIRECTION_DOWN:
ENO_ShutterMoveDown($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
ENO_ShutterStop($_IPS['INSTANCE']);
}
break;
}
break;

//digitalStrom
case "{3DDA1E2B-B807-4680-AB6D-E7E8FBD6093A}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
DS_ShutterStop($_IPS['INSTANCE']);
break;

case SC_DIRECTION_UP:
DS_ShutterMoveUp($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
DS_ShutterStop($_IPS['INSTANCE']);
}
break;

case SC_DIRECTION_DOWN:
DS_ShutterMoveDown($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
DS_ShutterStop($_IPS['INSTANCE']);
}
break;
}
break;

//xComfort
case "{1B7B5B7D-CAA9-4AB5-B9D8-EC805EC955AD}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
MXC_ShutterStop($_IPS['INSTANCE']);
break;

case SC_DIRECTION_UP:
MXC_ShutterMoveUp($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
MXC_ShutterStop($_IPS['INSTANCE']);
```

```
}
break;

case SC_DIRECTION_DOWN:
MXC_ShutterMoveDown($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
MXC_ShutterStop($_IPS['INSTANCE']);
}
break;
}
break;

//LCN Unit
case "{2D871359-14D8-493F-9B01-26432E3A710F}":
$type=IPS_GetProperty($_IPS['INSTANCE'],'Unit');
switch($type) {
//Outputs
case 0:
switch($_IPS['DIRECTION'])
{
case SC_DIRECTION_STOP:
LCN_SetIntensity($_IPS['INSTANCE'],0,0);
LCN_SetIntensity($_IPS['INSTANCE2'],0,0);
break;

case SC_DIRECTION_UP:
LCN_SetIntensity($_IPS['INSTANCE'],100,4);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
LCN_SetIntensity($_IPS['INSTANCE'],0,0);
}
break;

case SC_DIRECTION_DOWN:
LCN_SetIntensity($_IPS['INSTANCE2'],100,4);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
LCN_SetIntensity($_IPS['INSTANCE2'],0,0);
}
break;
}
break;

//Relay
case 2:
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
LCN_SwitchRelay($_IPS['INSTANCE'],false);
break;

case SC_DIRECTION_UP:
LCN_SwitchRelay($_IPS['INSTANCE2'],false); //To change relay direction,
please set it to true
LCN_SwitchRelay($_IPS['INSTANCE'],true);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
LCN_SwitchRelay($_IPS['INSTANCE'],false);
}
```

```
break;

case SC_DIRECTION_DOWN:
LCN_SwitchRelay($_IPS['INSTANCE2'],true);//To change relay direction,
please set false
LCN_SwitchRelay($_IPS['INSTANCE'],true);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
LCN_SwitchRelay($_IPS['INSTANCE'],false);
}
break;
}
break;
}
break;

//LCN Shutter
case "{C81E019F-6341-4748-8644-1C29D99B813E}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
LCN_ShutterStop($_IPS['INSTANCE']);
break;

case SC_DIRECTION_UP:
LCN_ShutterMoveUp($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
LCN_ShutterStop($_IPS['INSTANCE']);
}
break;

case SC_DIRECTION_DOWN:
LCN_ShutterMoveDown($_IPS['INSTANCE']);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
LCN_ShutterStop($_IPS['INSTANCE']);
}
break;
}
break;

//1-Wire Shutter Modul (e-service Online)
case "{BD0F2622-F67C-4248-9A04-316DF13914C3}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
OW_SetPin($_IPS['INSTANCE'], 0, true);
OW_SetPin($_IPS['INSTANCE'], 1, true);
IPS_Sleep(100);
OW_SetPin($_IPS['INSTANCE'], 0, false);
OW_SetPin($_IPS['INSTANCE'], 1, false);
break;

case SC_DIRECTION_UP:
OW_SetPin($_IPS['INSTANCE'], 0, true);
IPS_Sleep(100);
OW_SetPin($_IPS['INSTANCE'], 0, false);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
OW_SetPin($_IPS['INSTANCE'], 0, true);
```

```
OW_SetPin($_IPS['INSTANCE'], 1, true);
IPS_Sleep(100);
OW_SetPin($_IPS['INSTANCE'], 0, false);
OW_SetPin($_IPS['INSTANCE'], 1, false);
}
break;

case SC_DIRECTION_DOWN:
OW_SetPin($_IPS['INSTANCE'], 1, true);
IPS_Sleep(100);
OW_SetPin($_IPS['INSTANCE'], 1, false);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
OW_SetPin($_IPS['INSTANCE'], 0, true);
OW_SetPin($_IPS['INSTANCE'], 1, true);
IPS_Sleep(100);
OW_SetPin($_IPS['INSTANCE'], 0, false);
OW_SetPin($_IPS['INSTANCE'], 1, false);
}
break;
}
break;

//1-Wire Shutter (1-wire.de)
case "{6A75828A-25CD-4CF3-83EA-DAAB914030A7}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
OneWireShutter($_IPS['INSTANCE'], 0, $_IPS['DURATION']);
break;

case SC_DIRECTION_UP:
if($_IPS['DURATION'] == 0) {
$_IPS['DURATION'] = 120000;
}
OneWireShutter($_IPS['INSTANCE'], 0, $_IPS['DURATION']);
break;

case SC_DIRECTION_DOWN:
if($_IPS['DURATION'] == 0) {
$_IPS['DURATION'] = 120000;
}
OneWireShutter($_IPS['INSTANCE'], 1, $_IPS['DURATION']);
break;
}
break;

//Homematic Shutter
case "{EE4A81C6-5C90-4DB7-AD2F-F6BBD521412E}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
HM_WriteValueBoolean($_IPS['INSTANCE'], "STOP", true);
break;

case SC_DIRECTION_UP:
HM_WriteValueFloat($_IPS['INSTANCE'], "LEVEL", 1.0);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
HM_WriteValueBoolean($_IPS['INSTANCE'], "STOP", true);
}
```

```
break;

case SC_DIRECTION_DOWN:
HM_WriteValueFloat($_IPS['INSTANCE'], "LEVEL", 0.0);
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
HM_WriteValueBoolean($_IPS['INSTANCE'], "STOP", true);
}
break;
}
break;

//DMXOUT
case "{E19C2E41-7347-4A3B-B7D9-A9A88E0D133E}":
switch($_IPS['DIRECTION']) {
case SC_DIRECTION_STOP:
DMX_SetValue($_IPS['INSTANCE'],1,0); //Relay Up
DMX_SetValue($_IPS['INSTANCE2'],1,0); //Relay Down
break;

case SC_DIRECTION_UP:
DMX_SetValue($_IPS['INSTANCE2'],1,0); //Relay Down
DMX_SetValue($_IPS['INSTANCE'],1,255); //Relay Up
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
DMX_SetValue($_IPS['INSTANCE'],1,0); //Relay Up
}
break;

case SC_DIRECTION_DOWN:
DMX_SetValue($_IPS['INSTANCE'],1,0); //Relay Up
DMX_SetValue($_IPS['INSTANCE2'],1,255); //Relay Down
if($_IPS['DURATION'] > 0) {
IPS_Sleep($_IPS['DURATION']);
DMX_SetValue($_IPS['INSTANCE2'],1,0); //Relay Down
}
break;
}
break;

default:
die("No Handler for Module ".$instance['ModuleInfo']['ModuleName']."
found");
}

function OneWireShutter($ins, $dir, $sec) {
@OW_SetStrobe($ins, True);
$res = ($dir * 128) + ($sec / 1000);
@OW_SetPort((integer)$ins, (integer)$res);
}

function CreateVariableByName($id, $name, $type) {
$vid = @IPS_GetVariableIDByName($name, $id);
if($vid===false) {
$vid = IPS_CreateVariable($type);
IPS_SetParent($vid, $id);
IPS_SetName($vid, $name);
IPS_SetInfo($vid, "This Variable was created by Script");
}
```

```
return $vid;
}
```

INFORMATION TO EXTEND THE SCRIPT

Three parameters are always transferred when a call is made.
The InstanceID of the “Transmit Device” set in the shutter control instance
The direction in which to move.
The time for which should be moved. (This value is 0 for STOP and for the TEST buttons in the properties page)
The GUIDs for the respective module must be inserted in the “case statements”. All IP-Symcon functions can be used. No output may be generated (echo), otherwise this is considered an error and will be displayed in the message log and the status of the “position” variable will not be updated.

SC_Move

```
bool SC_Move (int $InstanceID, int $Position)
```

PARAMETERS

InstanceID	ID of the shutter control instance
Position	0%-100%, 99% = leave a gap

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter for the device with ID **InstanceID** to **Position**.



Since some radio systems are unidirectional, it cannot be determined whether a function was successful. As a result, exact positioning cannot be guaranteed.

EXAMPLE

```
SC_Move(12345, 50); //Move to 50%
```

SC_MoveDown

```
bool SC_MoveDown (int $InstanceID, int $Duration)
```

PARAMETERS

InstanceID	ID of the shutter control instance
Duration	0 = limit switch, >1 = duration in ms

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter for the device with ID **InstanceID** down for **Duration** milliseconds.



The status variable is reset to 100 regardless of the time moved.

EXAMPLE

```
SC_MoveDown(12345, 0); //Move down
```

SC_MoveUp

```
bool SC_MoveUp (int $InstanceID, int $Duration)
```

PARAMETERS

InstanceID	ID of the shutter control instance
Duration	0 = limit switch, >1 = duration in ms

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Moves the shutter for the device with ID **InstanceID** up for **Duration** milliseconds.



The status variable is reset to 0 regardless of the time moved.

EXAMPLE

```
SC_MoveUp(12345, 0); //Move up
```

SC_Stop

```
bool SC_Stop (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the shutter control instance

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Stops a move operation for the device with ID InstanceID.
The status variable is reset to -1.

EXAMPLE

```
SC_Stop(12345); //Stop
```

USBMapper

Require: Symcon >= 5.5



This module is deprecated and should no longer be used.

The module automatically sets the correct USB port to the entered serial ports.

FUNCTION SCOPE

The module automatically maps entered SerialPorts to the respective correct USB port.
Several serial ports can be entered in the list.
It is checked at the start of IP-Symcon and every minute whether the SerialPorts are configured correctly.

REQUIREMENTS

Linux or Raspberry Pi

SOFTWARE INSTALLATION

Via the Module Store install the module USB-Mapper.

SETTING UP THE INSTANCES IN IP-SYMCON

Under "Add instance" the 'USBMapper' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Devices	List, which contains the SerialPorts to be mapped

USBM_FixPorts

Require: Symcon >= 5.5

```
bool USBM_FixPorts (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Checks for all SerialPorts in the list of the USBMapper with the InstanceID **InstanceID** if they still have the correct USB ports configured.

EXAMPLE

```
USBM_FixPorts(12345);
```

Web Graph

Require: Symcon >= 4.3
The module is used to make already existing diagrams available via WebHook. The WebHook can then be called both locally and via Connect Service. Stylistic configurations are also possible via parameters in the URL.

FUNCTION SCOPE

Creates a custom Webhook
Adding the allowed object IDs for diagrams via list
Ability to secure via username and password
Deployment via WebHook both locally or via Connect Service
Configuration of the display via URL parameters

PREREQUISITES

Active subscription when using the Connect Service

SOFTWARE INSTALLATION

Using the Module Store to install the Web Graph module.

SETTING UP INSTANCES IN IP-SYMCON

Under "Add Instance" the 'Web Graph' module can be found using the quick filter.
More information about adding instances in the Instances documentation

CONFIGURATION PAGE:

NAME	DESCRIPTION
Access List	The list contains the object IDs of diagram media which may be provided via WebHook
User Name/Password (Expert)	Restricts access using a user name and password
Test environment	Here you can try out different settings and read out the appropriate URL

After entry in the "Access List" the graphs are available with their ID.

EXAMPLE

Diagram media with the ID 12345 is entered into the "Access List". Thereupon this is visible under:

Local:
http://127.0.0.1:3777/hook/webgraph/?id=12345
Symcon Connect Service:
https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.ipmagic.de/hook/webgraph/?id=12345
Additional settings can be appended to the URL.
/hook/webgraph/?
id=12345&startTime=&timeSpan=3&isHighDensity=1&isExtrema=&isDynamic=1&isContinuous=

STATUS VARIABLES AND PROFILES

No additional status variables or profiles are created. Only the WebHook "/hook/webgraph/" is entered.

VISUALIZATION

No further configuration or display is possible via the Visualization.

Wunderground Weather

Require: Symcon >= 4.2



This module is deprecated and should no longer be used.

The module queries weather data via the Wunderground API.
You need to register at Wunderground to get an API key.
Current data, severe weather warnings, as well as hourly and daily forecasts can be queried.

FUNCTIONAL SCOPE

Enable/disable query of desired weather data.
Adjustability of the amount of severe weather, hourly and daily data.
Timer for automatic updating of data.

SOFTWARE INSTALLATION

Via the Module Store install the WundergroundWeather module.

SET UP THE INSTANCES IN IP-SYMCON

Under "Add instance" the 'WundergroundWeather' module can be found using the quick filter.
More information about adding instances in the Instances documentation.

CONFIGURATION PAGE:

NAME	DESCRIPTION
Location	Location from which the data should be taken. Whether a location is available can be tried on the wunderground.com page. If a location is not available, a nearest larger location should be chosen.
Country	The country must be entered here.
PWS ID	ID of the 'Personal Weatherstation' Here you can query a specific weather station
API Key	Wunderground API Key. Can be requested on the Wunderground homepage after registration. "More"->"Weather API for Developers".
Query Current Data	Enables the query of current weather data.
Hourly Forecast	Enables the query of the hourly forecast.
12-hourly forecast	Activates the query of the 12-hourly forecast.
Daily forecast	Activates the query of the daily forecast.

NAME	DESCRIPTION
Poll severe weather warning	Activates polling of the severe weather forecast.
number of forecasts (12-hourly)	The number of 12-hourly forecasts. Maximum value: 8
Number of forecasts (daily)	The number of daily forecasts. Maximum value: 4
number of predictions (hourly)	The number of hourly predictions. Maximum value: 24
Number of severe weather warnings	The number of severe weather forecasts. Maximum value: 6
Update Weather Data	Sets the timer in minutes how often the weather data should be updated. (current/hourly/12-hourly)
Update severe weather warnings	Sets the timer in minutes how often the severe weather warnings should be updated.
Update Weather button	Updates the weather data (current/hourly/12-hourly/daily). If all three queries are deactivated or the timer is set to 0 => Timer deactivated
Button Update severe weather warnings	Updates the severe weather warnings. Provided the severe weather warning query is deactivated or the timer is set to 0 => Timer deactivated

STATUS VARIABLES AND PROFILES

The status variables/categories are created automatically. Deleting individual ones can lead to malfunctions.

CURRENT WEATHER DATA

NAME	TYPE	DESCRIPTION
Air pressure	Float	Specification in hPa
Humidity	Float	Indication in %
Precipitation/h	Float	Indication in liters/m²
Precipitation day	Float	in liters/m²
Visibility	Float	Indication in km
Solar Radiation	Float	Specified in W/m²
Temperature	Float	in °C
Temperature felt	Float	Indication in °C
Temperature dew point	Float	value in °C
UV radiation	Integer	Information: UVIndex Explanation
Wind Gust	Float	Specification in km/h
Wind Speed	Float	Specification in km/h
wind direction	float	indication in cardinal points

HOURLY FORECAST

The variables are marked with 1..24h. (1 = forecast next full hour; 24 = forecast of the 24th full hour)

NAME	TYPE	DESCRIPTION
Condition	String	Describes the weather e.g. "Overcast", "Rain possible"
Humidity	Float	Specification in %
Air Pressure	Float	Indication in hPa
Rainfall Amount	Float	Indication in liters/m²
Probability of rain	Integer	Indication in %
Temperature	Float	in °C
Cloud cover	Integer	Specified in %
Wind Speed	Float	in km/h
wind direction	float	indication in cardinal directions

12-HOURLY-FORECAST

The variables are marked with 12, 24..96h (12 = forecast in 12 hours; 96 = forecast in 96 hours)
Note: the 12h-hourly forecast is no longer provided by Wunderground. The variables remain for compatibility reasons, but contain the values of the daily forecast.

NAME	TYPE	DESCRIPTION
maximum temperature	float	specification in °C
Lowest temperature	Float	Indication in °C

DAILY FORECAST

The variables are marked with 1..4d (1 = tomorrow; 2 = the day after tomorrow ...)

NAME	TYPE	DESCRIPTION
Condition	String	Describes the weather e.g. "Overcast", "Rain possible"
Highest Temperature	Float	Specifies in °C
Lowest Temperature	Float	Indication in °C
Humidity	Float	Indication in %
Rainfall Amount	Float	Indication in Liter/m²
Probability of rain	Integer	Specified in %
Wind Speed	Float	in km/h
Wind Direction	Float	Indication in Cardinal Points

WEATHERWARNING

NAME	TYPE	DESCRIPTION
Description	String	Describes the warning with possible additional information such as wind speeds or rain amounts.
Date	Integer	Specified in UnixTimeStamp. Date when the warning was issued.
Name	String	Written out type. e.g. thunderstorm
Type	String	3-letter abbreviation for the warning Overview

PROFILE:

NAME	TYPE
WGW.Rainfall	Float
WGW.Sunray	Float
WGW.Visibility	Float
WGW.WindSpeedkmh	Float
WGW.UVIndex	Integer
WGW.ProbabilityOfRain	Integer

WGW_UpdateStormWarningData

Require: Symcon >= 4.2

```
bool WGW_UpdateStormWarningData (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Updates the severe weather warnings of the Weatherground module with the InstanceID InstanceID.

EXAMPLE

```
WGW_UpdateStormWarningData(12345);
```

WGW_UpdateWeatherData

Require: Symcon >= 4.2

```
bool WGW_UpdateWeatherData (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the Instance

RETURNS

If the command could be executed successfully, it returns TRUE as result, otherwise FALSE.

DESCRIPTION

Updates the weather data (current/hourly/daily) of the Weatherground module with InstanceID InstanceID.

EXAMPLE

```
WGW_UpdateWeatherData(12345);
```

Command Reference

The Command Reference includes commands that are provided by IP-Symcon and are used for its own management.
This is a list of IP-Symcon specific commands.



For a list of module-specific commands, refer to the Module Reference.

PROCESS CONTROL

Functions in this group allow influencing the flow of a script.

IPS_AbortScript	aborts all running and queued executions of a flow script
IPS_Execute	Starts an external program
IPS_ExecuteEx	Starts an external program in the user context
IPS_RunAction	executes a single action
IPS_RunActionWait	executes a single action and waits for the result
IPS_RunScript	starts another IP-Symcon script
IPS_RunScriptEx	starts another IP-Symcon script and passes variables
IPS_RunScriptText	starts text as IP-Symcon script
IPS_RunScriptTextEx	starts a text as IP-Symcon script and passes variables
IPS_RunScriptTextWait	starts a text as IP-Symcon script and waits for its execution
IPS_RunScriptTextWaitEx	starts a text as IP-Symcon script, passes variables, and waits for the execution of the script
IPS_RunScriptWait	starts another IP-Symcon script and waits for the result
IPS_RunScriptWaitEx	starts another IP-Symcon script, passes variables, and waits for the result
IPS_SemaphoreEnter	allows the synchronization with other simultaneously running scripts
IPS_SemaphoreLeave	allows the synchronization with other simultaneously running scripts
IPS_Sleep	delays a script for a specified duration

MANAGEMENT OF PERMISSIONS

Functions in this group are used to manage users, roles and permissions.

Users	

IPS_AddPermissionToUser	adds a permission to a user
IPS_AddRoleToUser	assigns a role to a user
IPS_CreateUser	creates a new user
IPS_DeleteUser	deletes a user
IPS_GetUser	returns information about a user
IPS_GetUserList	returns the names of all users
IPS_RemovePermissionFromUser	removes a permission from a user
IPS_RemoveRoleFromUser	removes a role from a user
IPS_SetUserActive	activates/deactivates a user
IPS_SetUserBindDN	sets the BindDN of an LDAP user
IPS_SetUserEMail	sets the e-mail address of a user
IPS_SetUserFirstName	sets the first name of a user
IPS_SetUserLastName	sets the last name of a user
IPS_SetUserPassword	sets the password of a user
IPS_UserExists	checks whether a specific user exists
IPS_UserHasPermission	checks whether a user may perform an operation on an object
IPS_UserHasRole	checks whether a specific role is assigned to a user
IPS_UserIsAdministrator	checks whether a user is an administrator

IPS_GetLoggedInUser	returns the name of the user in whose context the current execution runs
IPS_HasPermission	checks whether the currently logged in user may perform an operation on an object
IPS_IsAdministrator	checks whether the currently logged in user is an administrator
IPS_IsUnknown	checks whether the current execution runs without a logged in user

Roles

IPS_AddPermissionToRole	adds a permission to a role
IPS_CreateRole	creates a new role
IPS_DeleteRole	deletes a role
IPS_GetRole	returns information about a role
IPS_GetRoleList	returns the names of all roles
IPS_RemovePermissionFromRole	removes a permission from a role
IPS_RoleExists	checks whether a role exists
IPS_RoleHasPermission	checks whether a role has a permission for an object

MANAGEMENT OF EVENTS

Functions in this group are used to manage events.

IPS_CreateEvent	create a new event
IPS_DeleteEvent	deletes an event
IPS_EventExists	checks if an event exists
IPS_GetEvent	returns extensive information about an event
IPS_GetEventIDByName	determines the ID of an event by its name
IPS_GetEventList	returns a list of all existing events
IPS_GetEventListByType	returns a list of all events of a given type
IPS_IsConditionPassing	checks if a condition is fulfilled
IPS_SetEventAction	set the executed action
IPS_SetEventActive	activates/deactivates an event
IPS_SetEventCondition	modifies the conditions of an event
IPS_SetEventConditionDateRule	modifies the rule of a condition of an event for date comparisons
IPS_SetEventConditionDayOfTheWeekRule	modifies the rule of a condition of an event for weekday comparisons
IPS_SetEventConditionTimeRule	modifies the rule of a condition of an event for time comparisons
IPS_SetEventConditionVariableDynamicRule	modifies a dynamic condition rule of an event for variable comparison
IPS_SetEventConditionVariableRule	modifies a condition rule of an event for variable comparison
IPS_SetEventConditionVariableStaticRule	modifies a static condition rule of an event for variable comparison
IPS_SetEventCyclic	configures a cyclic event
IPS_SetEventCyclicDateFrom	sets the start date
IPS_SetEventCyclicDateTo	sets the end date
IPS_SetEventCyclicTimeFrom	set the start time
IPS_SetEventCyclicTimeTo	sets the end time
IPS_SetEventLimit	limits the number of event executions
IPS_SetEventScheduleAction	modifies the action table of a schedule event and sets the action to "Execute PHP Code"
IPS_SetEventScheduleActionEx	modifies the action table of a schedule event
IPS_SetEventScheduleGroup	modifies the grouping of days for a schedule event
IPS_SetEventScheduleGroupPoint	modifies a switch point of the group of a schedule event
IPS_SetEventScript	sets the action to "Execute PHP Code" and define the code to be executed
IPS_SetEventTrigger	defines the trigger for a triggered event

IPS_SetEventTriggerSubsequentExecution	defines whether a triggered event is executed again on the same value
IPS_SetEventTriggerValue	set the value the trigger variable is compared to

MANAGEMENT OF INSTANCES

Functions in this group are used to manage instances.

Debug

IPS_DisableDebug	deactivates the forwarding of debug messages
IPS_DisableDebugFile	deactivates the writing of debug logs to a file
IPS_EnableDebug	activates the forwarding of debug messages for a specific duration
IPS_EnableDebugFile	activates the writing of debug logs to a file
IPS_SendDebug	sends a message to the debug output of an object

IPS_CreateInstance	creates an instance
IPS_DeleteInstance	deletes an instance
IPS_GetInstance	returns extensive information about a specific instance
IPS_GetInstanceIDByName	returns the ID of an instance by its name
IPS_GetInstanceList	returns a list of all existing instances
IPS_GetInstanceListByModuleID	returns all instances with a specific module ID
IPS_GetInstanceListByModuleType	returns a list of all instances with a specific type
IPS_InstanceExists	checks if a specific instance exists
IPS_RequestAction	calls the RequestAction method of an instance by ident
IPS_Translate	translates a text with the translations of the module of an instance

Configuration

IPS_ApplyChanges	applies a changed configuration
IPS_GetConfiguration	reads a configuration
IPS_GetConfigurationForm	reads a configuration form
IPS_GetConfigurationForParent	returns the configuration which an instance specifies for its parent instance
IPS_GetProperty	reads the current value of a property
IPS_GetVisualizationTile	reads the custom presentation of an instance
IPS_HasChanges	checks if the configuration was changed
IPS_ResetChanges	resets the modified configuration to the current configuration
IPS_SetConfiguration	sets a new configuration
IPS_SetProperty	sets a planned value for a property

References

IPS_GetReferenceList	returns the IDs of all references

Search

IPS_IsSearching	checks whether an instance is in search mode
IPS_StartSearch	starts the search mode of an instance
IPS_StopSearch	stops the search mode of an instance
IPS_SupportsSearching	checks whether an instance supports the search mode

Timers

IPS_GetTimer	returns information about a timer
IPS_GetTimerList	returns the IDs of all timers
IPS_TimerExists	checks whether a specific timer exists

Connection

IPS_ConnectInstance	creates a (data) connection between two instances
IPS_DisconnectInstance	disconnects a (data) connection between two instances
IPS_GetCompatibleInstances	returns compatible instances to an instance
IPS_GetCompatibleParents	returns the description of the compatible parent instances
IPS_IsInstanceCompatible	checks if a connection between two instances is possible

MANAGEMENT OF CATEGORIES

Functions in this group are used to manage categories.

IPS_CategoryExists	checks if a category already exists
IPS_CreateCategory	creates a new category
IPS_DeleteCategory	deletes an existing category
IPS_GetCategory	returns information about a category
IPS_GetCategoryIDByName	searches and returns the ID of an existing category by its name
IPS_GetCategoryList	returns a list of all existing categories

MANAGEMENT OF LINKS

Functions in this group are used to manage links.

IPS_CreateLink	creates a new link
IPS_DeleteLink	deletes an existing link
IPS_GetLink	returns extensive information about a link
IPS_GetLinkIDByName	returns the ID of a link by its name
IPS_GetLinkList	returns a list of all existing links
IPS_LinkExists	checks if a link exists already
IPS_SetLinkTargetID	sets the target of a link

MANAGEMENT OF MEDIA

Functions from this group are used to manage media

IPS_CreateMedia	creates a new media object in the media pool
IPS_DeleteMedia	removes a media object from the media pool
IPS_GetMedia	returns extensive information about a specific media object
IPS_GetMediaContent	returns the content of a media object
IPS_GetMediaIDByFile	returns the ID of a media object by its file
IPS_GetMediaIDByName	returns the ID of a media object by its name
IPS_GetMediaList	returns a list of all media objects in the media pool
IPS_GetMediaListByType	returns a list of all media objects of a specific type
IPS_MediaExists	checks if a specific media object is contained within the media pool already
IPS_SendMediaEvent	send a notification that a media object was changed
IPS_SetMediaCached	activates that a media object is only modified in the internal memory
IPS_SetMediaContent	set the content of a media object
IPS_SetMediaFile	bind a media file to a media object

MANAGEMENT OF MODULES

Functions from this group are used to manage modules and libraries

IPS_GetCompatibleModules	returns all modules that are compatible to a specific module
IPS_GetLibrary	returns extensive information about a library
IPS_GetLibraryList	returns a list of all libraries
IPS_GetLibraryModules	returns a list of modules in a library
IPS_GetModule	returns extensive information about a module
IPS_GetModuleList	returns a list of all modules
IPS_GetModuleListByType	returns a list of all modules of a given type
IPS_IsModuleCompatible	checks if two modules are compatible to each other
IPS_LibraryExists	Checks if the given library exists
IPS_ModuleExists	checks if the given module exists

MANAGEMENT OF OBJECTS

Functions from this group are used to manipulate objects.

IPS_FindObjectIDByIdent	searches the ID of an object by its ident without failing if nothing is found
IPS_FindObjectIDByName	searches the ID of an object by its name without failing if nothing is found
IPS_GetChildrenIDs	returns a list of all child objects
IPS_GetLocation	returns the complete path to an object
IPS_GetName	returns the name of the object
IPS_GetObject	returns extensive information about an object
IPS_GetObjectIDByIdent	returns the ID of an object by its ident
IPS_GetObjectIDByName	returns the ID of an object by its name
IPS_GetObjectList	returns a list of all existing objects
IPS_GetParent	returns the ID of the parent of the object
IPS_HasChildren	checks if an object has child objects
IPS_IsChild	checks if an object is a child of a parent object
IPS_ObjectExists	checks if a specific object exists
IPS_SetDisabled	sets an object as disabled for the visualizations
IPS_SetHidden	sets the visibility for an object
IPS_SetHiddenMaximize	sets the visibility of the maximize button of an object
IPS_SetHiddenTitle	sets the visibility of the title of an object
IPS_SetIcon	sets the icon of an object
IPS_SetIdent	sets the ident of an object
IPS_SetInfo	sets extended information of an object
IPS_SetName	sets the name of an object

IPS_SetParent	sets the parent of an object
IPS_SetPosition	sets the position of an object

PROGRAM INFORMATION

Functions from this group provide information about IP Symcon itself

IPS_FunctionExists	checks if a specific function exists
IPS_GetAvailableUpdates	returns an available update for the configured update channel
IPS_GetFunction	returns extensive information about a function
IPS_GetFunctionList	returns a list of all available functions
IPS_GetFunctionListByModuleID	returns a list of all available functions of a module
IPS_GetKernelArchitecture	returns the architecture that IP-Symcon was created for
IPS_GetKernelDate	returns the kernel creation date
IPS_GetKernelDir	returns the path to the IP Symcon program folder
IPS_GetKernelDirEx	returns the path to the IP Symcon installation folder
IPS_GetKernelDirSpace	returns the total/free/available disk space of the drive on which all data of IP-Symcon is stored
IPS_GetKernelPlatform	returns the operating system which IP-Symcon is running on
IPS_GetKernelRevision	returns a unique revision identifier
IPS_GetKernelRunlevel	returns the run level of the kernel
IPS_GetKernelStartTime	returns the point in time when IP-Symcon was started
IPS_GetKernelVersion	returns the version number of the used kernel
IPS_GetLiveUpdateVersion	returns a version identifier made of the build date and revision of the kernel
IPS_GetLogDir	returns the path to the IP Symcon log folder
IPS_GetSystemLanguage	returns the system language of the operating system on which IP-Symcon is running
IPS_GetUpdateChannel	returns the currently configured update channel
IPS_LogMessage	Writes a user defined message into the log file

MANAGEMENT OF SCRIPTS

Functions from this group are used to manage scripts.

Actions	Functions in this group are used to manage actions.

IPS_ActionExists	checks whether a specific action exists
IPS_GetActionForm	returns the dynamically generated configuration form of an action
IPS_GetActionReadableCode	returns an action with its parameters as readable PHP code
IPS_GetActions	returns all available actions as a JSON encoded string
IPS_GetActionsByEnvironment	returns the actions offered for a target object in an environment
IPS_UpdateFormField	changes the parameter of a form field in an action form

IPS_CreateScript	creates a script
IPS_DeleteScript	deletes a script
IPS_GetScript	returns extensive information about a script
IPS_GetScriptContent	returns the content of a script
IPS_GetScriptEventList	returns all events that are assigned to a script
IPS_GetScriptFile	returns the file name of a script
IPS_GetScriptIDByFile	returns the ID of a script by its file
IPS_GetScriptIDByName	returns the ID of a script by its name
IPS_GetScriptList	returns a list of all scripts
IPS_GetScriptTimer	returns the start value of the script timer
IPS_ScriptExists	checks if a script exists
IPS_SetScriptContent	sets the content of a script
IPS_SetScriptFile	sets the file name for a script
IPS_SetScriptTimer	sets the start value of a script timer

Threads

IPS_GetScriptThread	returns information about a thread of the script engine
IPS_GetScriptThreadList	returns the IDs of all threads of the script engine
IPS_ScriptThreadExists	checks whether a specific thread of the script engine exists

MANAGEMENT OF TRACES

Functions in this group are used to read traces.

IPS_GetTraceIDList	returns the IDs of the trace entries of an object
IPS_GetTraceList	returns a trace entry with its cause chain

MANAGEMENT OF VARIABLES

Functions in this group are used to manage variables and change their values

IPS_CreateVariable	creates a new variable
IPS_DeleteVariable	deletes a variable
IPS_GetVariable	returns extensive information about a variable
IPS_GetVariableEventList	returns a list of all events that are associated with a variable
IPS_GetVariableIDByName	returns the ID of a variable by its name
IPS_GetVariableList	returns a list of all existing variables
IPS_GetVariablePresentation	Returns all parameters of the currently used presentation
IPS_SetVariableCustomAction	sets a user-defined action script for a variable
IPS_SetVariableCustomPresentation	assigns a user-defined presentation to the variable
IPS_SetVariableCustomProfile	sets a user-defined profile for a variable
IPS_VariableExists	checks if a variable exists
Variable Presentation	Functions in this group are used to manage variable presentations

IPS_GetPresentation	provides comprehensive information about a specific presentation
IPS_GetPresentationForm	returns the configuration form of a presentation for a variable type
IPS_GetPresentationFormat	returns the format template of a presentation for a variable type
IPS_GetPresentations	provides all presentations of the system
IPS_GetPresentationValue	returns the preview value of a presentation
IPS_PresentationExists	Checks whether a specific presentation exists
Management of Templates	Functions in this group are used to manage templates

Variable Profiles

IPS_CreateVariableProfile	creates a variable profile
IPS_DeleteVariableProfile	deletes a variable profile
IPS_GetVariableProfile	returns extensive information about a variable profile
IPS_GetVariableProfileList	returns a list of all existing variable profiles
IPS_GetVariableProfileListByType	returns a list of all existing variable profiles of a specific type
IPS_SetVariableProfileAssociation	modifies the association table of a variable profile
IPS_SetVariableProfileDigits	sets the number of decimal places of a variable profile
IPS_SetVariableProfileIcon	sets the icon for a variable profile
IPS_SetVariableProfileText	sets the prefix and suffix of a variable profile
IPS_SetVariableProfileValues	sets the minimum value, maximum value, and step size of a variable profile
IPS_VariableProfileExists	checks if a variable profile exists

ACCESSING VARIABLES

Functions in this group are used to read or write the values of variables.

GetValue	returns the value of an IP-Symcon variable, independent of its type
GetValueBoolean	returns the value of an IP-Symcon variable of the type Boolean
GetValueColor	returns the value of a color variable as integer in the format 0xRRGGBB
GetValueFloat	returns the value of an IP-Symcon variable of the type Float
GetValueFormatted	returns the value of an IP-Symcon variable and formats it according to its profile
GetValueFormattedEx	returns a value formatted based on the profile of a variable
GetValueInteger	returns the value of an IP-Symcon variable of the type Integer
GetValueString	returns the value of an IP-Symcon variable of the type String
HasAction	checks whether an action is defined
RequestAction	executes the variable action of an IP-Symcon variable
RequestActionColor	executes the action of a color variable with a color value in the format 0xRRGGBB
RequestActionEx	executes the action of an IP-Symcon variable with individual sender
SetValue	sets the value of an IP-Symcon variable independent of its type
SetValueBoolean	sets the value of an IP-Symcon variable of the type Boolean
SetValueColor	sets the value of a color variable from a color value in the format 0xRRGGBB
SetValueFloat	sets the value of an IP-Symcon variable of the type Float
SetValueInteger	sets the value of an IP-Symcon variable of the type Integer

SetValueString	sets the value of an IP-Symcon variable of the type String

Process Control

IPS_AbortScript	aborts all running and queued executions of a flow script
IPS_Execute	Starts an external program
IPS_ExecuteEx	Starts an external program in the user context
IPS_RunAction	executes a single action
IPS_RunActionWait	executes a single action and waits for the result
IPS_RunScript	starts another IP-Symcon script
IPS_RunScriptEx	starts another IP-Symcon script and passes variables
IPS_RunScriptText	starts text as IP-Symcon script
IPS_RunScriptTextEx	starts a text as IP-Symcon script and passes variables
IPS_RunScriptTextWait	starts a text as IP-Symcon script and waits for its execution
IPS_RunScriptTextWaitEx	starts a text as IP-Symcon script, passes variables, and waits for the execution of the script
IPS_RunScriptWait	starts another IP-Symcon script and waits for the result
IPS_RunScriptWaitEx	starts another IP-Symcon script, passes variables, and waits for the result
IPS_SemaphoreEnter	allows the synchronization with other simultaneously running scripts
IPS_SemaphoreLeave	allows the synchronization with other simultaneously running scripts
IPS_Sleep	delays a script for a specified duration

IPS_AbortScript

```
bool IPS_AbortScript (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the flow script

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function aborts all running and all queued executions of the flow script with the ID **ScriptID**. The executions get the status "aborted by user". A step that is currently executed is not interrupted, but no further steps are executed after it has finished. Running wait actions (e.g. "Wait") end their waiting time early. If the flow script has no running or queued executions, nothing happens.
Only flow scripts can be aborted. For a PHP script, the function fails with "Aborting is only supported for flow scripts!", for an ID that does not exist with "Script #... does not exist".
IPS_GetScript returns the type of a script.
A flow script is started e.g. with IPS_RunScript or IPS_RunScriptWait. Within a flow script, the action "Abort Execution" is available for this purpose.

EXAMPLE

```
// Aborts all executions of the flow script
IPS_AbortScript(12345 /*[Close Shutters]*/);
```

IPS_Execute

```
string IPS_Execute (string $ProgramPath, string $Parameter, bool
$Dummy, bool $Wait)
```

PARAMETERS

ProgramPath	Full path to the program
Parameter	Parameter which is to be passed to the program (optional)
Dummy	Always specify true, parameter is not evaluated, this was only used in version 1.x.
Wait	Specifies whether to wait for the end of the program

RETURNS

The output of the program if the **Wait** parameter is **TRUE**: the output to stderr if present, otherwise the output to stdout. If **Wait** is **FALSE**, an empty string is returned. On failure (e.g. if the program does not exist or cannot be started), **FALSE** is returned.

DESCRIPTION

The function starts the external program with the path **ProgramPath**. The string **Parameter** is passed to the program as command line parameter. The Boolean value **Wait** determines if the end of the program should be awaited or the calling script should be resumed immediately. The **Dummy** parameter is not evaluated: IPS_Execute never shows a program window. If a window should be displayed, IPS_ExecuteEx must be used.
If the function waits for the program, the output of the program is returned as a string. The same rule applies on all operating systems:

If the program wrote anything to **stderr**, this output is returned, e.g. to not lose any error messages. The output to **stdout** is discarded in this case.

Otherwise the output to **stdout** is returned (an empty string if the program did not output anything).

If the function does not wait for the program, it always returns an empty string. If the program does not exist or cannot be started, a warning is issued and **FALSE** is returned.
The exit code of the program is not available. If the exit code or the outputs to **stdout** and **stderr** are needed separately, the PHP functions exec or proc_open should be used.
Differences between the operating systems:

Windows: The program must be specified with its full path. It is started with the command line `ProgramPath Parameter`, the working directory is the directory of the program.

Linux, macOS and SymBox: A program name without a path is searched in the directories of the PATH environment variable. With **Wait** = **TRUE**, the command is executed via the shell (`/bin/sh -c '"ProgramPath" Parameter'`), so **Parameter** may contain shell syntax such as quotes, variables, redirections or pipes. With **Wait** = **FALSE**, the program is started

directly. **Parameter** is split into individual arguments like the shell does (quotes and variables are evaluated), but redirections and pipes are not possible.



Caution: While waiting for the program to end, the script execution time is extended accordingly. Scripts have no maximum runtime, but while waiting the script occupies one of the limited threads in which scripts are executed. Long-running programs should therefore not be started with waiting enabled. This is especially important if the program expects user input.



Attention: If the program to be started is interactive (e.g. has a window), it will not be displayed/visible. It starts in the background as a system process and can only be stopped with the help of the Task Manager. If the program should be displayed on the desktop, the function IPS_ExecuteEx must be used.

EXAMPLE

```
//Start a batch file
IPS_Execute("C:/autoexec.bat", "", false, false);
```

IPS_ExecuteEx

```
string IPS_ExecuteEx (string $ProgramPath, string $Parameter, bool
$ShowWindow, bool $Wait, int $SessionID)
```

PARAMETERS

ProgramPath	Full path to the program
Parameter	Parameter which is to be passed to the program (optional)
ShowWindow	**True** if the window should be displayed; **False** if the window should not be visible
Wait	Specifies whether to wait for the end of the program
SessionID	The user session ID to be used (under XP from 0, 2003/Vista from 1), < 0 for automatic selection

RETURNS

The output of the program if the **Wait** parameter is **TRUE**: the output to stderr if present, otherwise the output to stdout. If **Wait** is **FALSE**, an empty string is returned. On failure, **FALSE** is returned.

DESCRIPTION

The function starts the external program with the path **ProgramPath** in the session of a logged-in user, so that the program can e.g. be displayed on the user's desktop. The string **Parameter** is passed to the program as command line parameter. The Boolean value **ShowWindow** determines whether the program window is displayed. In contrast to IPS_Execute, this parameter is evaluated here. The Boolean value **Wait** determines if the end of the program should be awaited or the calling script should be resumed immediately.
If the function waits for the program, the output of the program is returned as a string. If the program wrote anything to **stderr**, this output is returned, otherwise the output to **stdout**. If the function does not wait for the program, it always returns an empty string. If the program does not exist or cannot be started, a warning is issued and **FALSE** is returned. The exit code of the program is not available.
The **SessionID** parameter specifies in which user session the program should be started. (In Windows 2003, for example, multiple terminal sessions can run in parallel.) If the parameter is less than 0, IP-Symcon automatically determines the first active session and starts the program there. Starting a program in a user session requires IP-Symcon to run as a service.



This function is only available on Windows. On other operating systems, a warning is issued and **FALSE** is returned.



Caution: While waiting for the program to end, the script execution time is extended accordingly. Scripts have no maximum runtime, but while waiting the script occupies one of the limited threads in which scripts are executed. Long-running programs should therefore not be started with waiting enabled. This is especially important if the program expects user input.

EXAMPLE

```
// Start Notepad
IPS_ExecuteEx("C:\\Windows\\notepad.exe", "", true, false, -1);
```

IPS_RunAction

```
bool IPS_RunAction (string $ActionID, array $ActionParameters)
```

PARAMETERS

ActionID	GUID of the executed action
ActionParameters	List of Parameters for the executed action

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function causes the execution of the Action with the GUID **ActionID** and the parameters **ActionParameters**. The execution runs concurrently (at the same time) to the calling script. The action parameters need to include the following entries:

PARAMETER	DESCRIPTION
TARGET	ID of the target object of the action (-1 for general actions)
ENVIRONMENT	The execution environment of the action, see Environments
PARENT	ID of the object that executes the action, e.g. the PHP script or the Flow Script

EXAMPLE

```
// Execute Script with the ID 12345
IPS_RunAction("{7938A5A2-0981-5FE0-BE6C-8AA610D654EB}", ["TARGET" =>
12345, "ENVIRONMENT" => "Default", "PARENT" => $_IPS['SELF']]);

// Show all actions and their GUIDs
foreach(json_decode(IPS_GetActions(), true) as $action) {
echo $action['id'] . " -> " . $action['caption'] . PHP_EOL;
// var_dump($action); <-- Shows more details, for example: restrictions
}
```

IPS_RunActionWait

```
string IPS_RunActionWait (string $ActionID, array $ActionParameters)
```

PARAMETERS

ActionID	GUID of the executed action
ActionParameters	List of Parameters for the executed action

RETURNS

Error message of the execution

DESCRIPTION

This function causes the execution of the Action with the GUID **ActionID** and the parameters **ActionParameters**. The function waits for the end of the execution to return a possible error message. If the execution was without error, it returns an empty string.
The action parameters need to include the following entries:

PARAMETER	DESCRIPTION
TARGET	ID of the target object of the action (-1 for general actions)
ENVIRONMENT	The execution environment of the action, see Environments
PARENT	ID of the object that executes the action, e.g. the PHP script or the Flow Script

EXAMPLE

```
// Execute the script with the ID 12345
$error = IPS_RunActionWait("{7938A5A2-0981-5FE0-BE6C-8AA610D654EB}",
["TARGET" => 12345, "ENVIRONMENT" => "Default", "PARENT" =>
$_IPS['SELF']]);
if ($error === "") {
echo "Action was executed successfully";
}
else {
echo "There was an error during execution: " . $error;
}
```

IPS_RunScript

```
bool IPS_RunScript (int $ScriptID)
```

PARAMETERS

ScriptID	Unique ID of the script

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function starts the script with the ID **ScriptID**. The called script runs in parallel (simultaneously) to the calling script.



Scripts that are executed via IPS_RunScript use the following System Variables.

Comparison between IPS_RunScript and the PHP statement include:
include is a PHP statement, not an IPS function. The script included by **include** is executed as if it were placed instead of the **include** statement. It behaves as if it were part of the calling script. This means that all variables known to the calling script are also available to the included script. The runtime of the script is extended by the runtime of the included script. When using **IPS_RunScript** the called script is launched in a separate context. It therefore knows nothing of the variables of the calling script. Data can only be exchanged via external variables. The function is suitable for situations where a script must be executed in a timely manner, but no direct connection to the calling script is required. The runtime of the script does not change, since both scripts are running (almost) simultaneously.

EXAMPLE

```
IPS_RunScript(12345 /*[Garden lighting On]*/);
```

IPS_RunScriptEx

```
bool IPS_RunScriptEx (int $ScriptID, array $Parameter)
```

PARAMETERS

ScriptID	Unique ID of the script
Parameter	Key (string) => Value (variant) pairs that can be accessed in the newly executed script.

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function starts the script with the ID **ScriptID** and passes the array **Parameter** as argument. The called script runs in parallel (simultaneously) to the calling script. The notes described for IPS_RunScript apply here as well.



Scripts that are executed via IPS_RunScriptEx use the following System Variables .

The array **Parameter** can contain any number of **"Key => Value"** pairs (see PHP Arrays). Each pair is available in the called script in the global variable __$_IPS **at the index** Key **with its content** Value__.

EXAMPLE

```
//Script, that will launch another script with parameter passing.
IPS_RunScriptEx(12345 /*[Temp]*/, Array("Title" => "Temp.", "Tmin" =>
10.0));

//Script that was called. Parameters are available as individual
variables in the
//global variable $_IPS. The variable name corresponds to the array
index
//given name.
```

```
$Headline = $_IPS['Title']. "progress"; //Results: Temp.progress
$MaxTemp = $_IPS['Tmin'] + 30.0;
```

IPS_RunScriptText

```
bool IPS_RunScriptText (string $ScriptText)
```

PARAMETERS

ScriptText	String which is called as script.

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function starts a script with the content **ScriptText**. It is not necessary to state PHP tags. When using **IPS_RunScriptText**, the script is executed in its own context. Thus, it does not have access to the variables of the calling script. Data can only be exchanged via external variables. The function is suited for situations where a script needs to be executed soon but does not require the context of the calling script. The runtime of the calling script does not change as both scripts are executed simultaneously.

EXAMPLE

```
// Prints "Hello World" in the message view
IPS_RunScriptText("echo 'Hello World';");
```

IPS_RunScriptTextEx

```
bool IPS_RunScriptTextEx (string $ScriptText, array $Parameter)
```

PARAMETERS

ScriptText	String that is executed as script
Parameter	Key (string) => Value (variant) Pairs that can be accessed in the executed script

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function starts a script with the content **ScriptText**. It is not necessary to state PHP tags. When using **IPS_RunScriptTextEx**, the script is executed in its own context. Thus, it does not have access to the variables of the calling script except for the passed parameters. Additional data can only be exchanged via external variables. The function is suited for situations where a script needs to be executed soon but does not require the context of the calling script. The runtime of the calling script does not change as both scripts are executed simultaneously.
The array **Parameter** can contain any number of **"Key => Value"** pairs (see PHP Arrays). Each pair is available in the called script in the global variable __$_IPS **at the index** Key **with its content** Value__.

EXAMPLE

```
// Prints "Hello World" in the message view
IPS_RunScriptTextEx('echo $_IPS["start"] . " ". $_IPS["end"];',
Array("start" => "Hello", "end" => "World"));
```

IPS_RunScriptTextWait

```
string IPS_RunScriptTextWait (string $ScriptText)
```

PARAMETERS

ScriptText	String that is executed as script

RETURNS

Result of the executed script

DESCRIPTION

The function starts a script with the content **ScriptText** and waits for the execution of the script to return its result. It is not necessary to state PHP tags.

EXAMPLE

```
// Returns "Hello World"
echo IPS_RunScriptTextWait("echo 'Hello World';");
```

IPS_RunScriptTextWaitEx

```
string IPS_RunScriptTextWaitEx (string $ScriptText, array $Parameter)
```

PARAMETERS

ScriptText	String that is executed as script
Parameter	Key (string) => Value (variant) Pairs that can be accessed in the executed script

RETURNS

Result of the executed script

DESCRIPTION

The function starts a script with the content **ScriptText**, passes the array **Parameter** as argument and waits for the end of the script execution to return its result. It is not necessary to state PHP tags.
The array **Parameter** can contain any number of **"Key => Value"** pairs (see PHP Arrays). Each pair is available in the called script in the global variable __$_IPS **at the index** Key **with its content** Value__.

EXAMPLE

```
//Script that executes text as script and passes parameters
echo IPS_RunScriptTextWaitEx('echo $_IPS["title"]. "course" . PHP_EOL;
echo $_IPS["Tmin"] + 30.0;', Array("title" => "Temp.", "Tmin" =>
10.0));
```

IPS_RunScriptWait

```
string IPS_RunScriptWait (int $ScriptID)
```

PARAMETERS

ScriptID	Unique ID of the script

RETURNS

Result of the executed script

DESCRIPTION

The function starts the script with the ID **ScriptID** and waits for the end of the script execution to return its result. The notes described for IPS_RunScript apply here as well.

EXAMPLE

```
echo IPS_RunScriptWait(12345 /*[Script A]*/);
```

IPS_RunScriptWaitEx

```
string IPS_RunScriptWaitEx (int $ScriptID, array $Parameter)
```

PARAMETERS

ScriptID	Unique ID of the script
Parameter	Key (string) => Value (variant) pairs that can be accessed in the newly executed script.

RETURNS

Result of the executed script

DESCRIPTION

The function starts the script with the ID **ScriptID**, passes the array **Parameter** as argument and waits for the end of the script execution to return its result. The notes described for IPS_RunScript apply here as well.
The array **Parameter** can contain any number of **"Key => Value"** pairs (see PHP arrays). Each pair is available in the called script in the global variable __$_IPS **at the index** Key **with its content** Value__.

EXAMPLE

```
//Script, that will launch another script with parameter passing.
echo IPS_RunScriptWaitEx(12345 /*[Temp]*/, Array("Title" => "Temp.",
"Tmin" => 10.0));

//Script that was called. Parameters are available as individual
variables in the
//global variable $_IPS. The variable name corresponds to the array
index
//given name.
$Headline = $_IPS['Title']. "progress"; //Results: Temp.progress
echo $_IPS['Tmin'] + 30.0;
```

IPS_SemaphoreEnter

```
bool IPS_SemaphoreEnter (string $Name, int $WaitingTime)
```

PARAMETERS

Name	Name that describes the semaphore
WaitingTime	Milliseconds to wait until the function gives up

RETURNS

TRUE if the semaphore was entered. FALSE if the semaphore could not be entered before the waiting time expired.

DESCRIPTION

This function sets a Semaphore (a sign with signal effect), which can be used to execute a part of the script exclusively and to protect it against competing scripts.
In IP-Symcon, multiple scripts can generally run simultaneously. As long as a script does not require exclusive access to variables or system resources, this presents no problem. If exclusive access is imperative, a semaphore can be set with this function. The function first checks if the semaphore with the specified name is already set. If it does not exist yet, it is set by the function and TRUE is returned. All other scripts that subsequently try to set the same semaphore keep trying to set it for the duration of WaitingTime. If this is successful, TRUE is returned, otherwise, after the waiting time has expired, FALSE.
If you want to ensure that the data used by a script is not altered by other, concurrently running scripts during its execution, you can use this function in order to secure exclusive access.
When using this function, it is vital to reset the semaphore via IPS_SemaphoreLeave after the critical part has been processed.

EXAMPLE

```
if (IPS_SemaphoreEnter("CriticalPoint", 1000))
{
// ...Critical code section

//Release semaphore again!
IPS_SemaphoreLeave("CriticalPoint");
}
else
{
// ...No execution possible. Another script uses the "CriticalPoint"
```

```
// for more than 1 second, so our wait time is exceeded.
}
```

IPS_SemaphoreLeave

```
bool IPS_SemaphoreLeave (string $Name)
```

PARAMETERS

Name	Name that describes the semaphore.

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function releases the semaphore that was set with IPS_SemaphoreEnter.

EXAMPLE

```
//See IPS_SemaphoreEnter()
```

IPS_Sleep

```
bool IPS_Sleep (int $WaitingTime)
```

PARAMETERS

WaitingTime	Specifies the waiting time in milliseconds

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function suspends the execution of the script for **WaitingTime** milliseconds. After this time, the script continues. With this function **short**(!) waiting times can be realized. Longer waiting times should not be implemented using the Sleep function, but controlled through appropriate **Events**.
Scripts have no maximum runtime. However, during the **WaitingTime** the script still occupies one of the threads in which scripts are executed, and the number of these threads is limited. Longer waiting times in the range of several seconds or more should therefore be implemented using other methods.



IPS_Sleep has no influence on other concurrently running scripts.



Alternatively, the PHP functions sleep(int seconds) or usleep(int microseconds) can be used.

EXAMPLE

```
// Delayed data output to the COM port
COMPort_SendText($id, chr(0x1b)); // Send ESC to LC display
IPS_Sleep(200); // Wait 200ms
COMPort_SendText($id, "0"); // Clear LC display
IPS_Sleep(200); // Wait 200ms
COMPort_SendText($id, "Good day!"); // Output text
```

```
// Wait 2 seconds with PHP alternatives
sleep(2);
IPS_Sleep(2000);
usleep(2000000);
```

Management of Permissions

USERS

IPS_AddPermissionToUser	adds a permission to a user
IPS_AddRoleToUser	assigns a role to a user
IPS_CreateUser	creates a new user
IPS_DeleteUser	deletes a user
IPS_GetUser	returns information about a user
IPS_GetUserList	returns the names of all users
IPS_RemovePermissionFromUser	removes a permission from a user
IPS_RemoveRoleFromUser	removes a role from a user
IPS_SetUserActive	activates/deactivates a user
IPS_SetUserBindDN	sets the BindDN of an LDAP user
IPS_SetUserEMail	sets the e-mail address of a user
IPS_SetUserFirstName	sets the first name of a user
IPS_SetUserLastName	sets the last name of a user
IPS_SetUserPassword	sets the password of a user
IPS_UserExists	checks whether a specific user exists
IPS_UserHasPermission	checks whether a user may perform an operation on an object
IPS_UserHasRole	checks whether a specific role is assigned to a user
IPS_UserIsAdministrator	checks whether a user is an administrator

IPS_GETLOGGEDINUSER

returns the name of the user in whose context the current execution runs

IPS_HASPERMISSION

checks whether the currently logged in user may perform an operation on an object

IPS_ISADMINISTRATOR

checks whether the currently logged in user is an administrator

IPS_ISUNKNOWN

checks whether the current execution runs without a logged in user

ROLES

IPS_AddPermissionToRole	adds a permission to a role
IPS_CreateRole	creates a new role
IPS_DeleteRole	deletes a role
IPS_GetRole	returns information about a role
IPS_GetRoleList	returns the names of all roles
IPS_RemovePermissionFromRole	removes a permission from a role
IPS_RoleExists	checks whether a role exists
IPS_RoleHasPermission	checks whether a role has a permission for an object

Users

IPS_AddPermissionToUser	adds a permission to a user
IPS_AddRoleToUser	assigns a role to a user
IPS_CreateUser	creates a new user
IPS_DeleteUser	deletes a user
IPS_GetUser	returns information about a user
IPS_GetUserList	returns the names of all users
IPS_RemovePermissionFromUser	removes a permission from a user
IPS_RemoveRoleFromUser	removes a role from a user
IPS_SetUserActive	activates/deactivates a user
IPS_SetUserBindDN	sets the BindDN of an LDAP user
IPS_SetUserEMail	sets the e-mail address of a user
IPS_SetUserFirstName	sets the first name of a user
IPS_SetUserLastName	sets the last name of a user
IPS_SetUserPassword	sets the password of a user
IPS_UserExists	checks whether a specific user exists
IPS_UserHasPermission	checks whether a user may perform an operation on an object
IPS_UserHasRole	checks whether a specific role is assigned to a user
IPS_UserIsAdministrator	checks whether a user is an administrator

IPS_AddPermissionToUser

```
bool IPS_AddPermissionToUser (string $User, int $ObjectID, string
$Operation, bool $Recursive)
```

PARAMETERS

User	Name of the user (case-sensitive)
ObjectID	ID of the object the permission applies to
Operation	Name of the registered operation, e.g. VISUALIZATION
Recursive	TRUE if the permission also applies to all objects below ObjectID, otherwise FALSE

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function grants the user User the permission to perform the Operation on the object ObjectID. Alternatively, permissions can be granted via roles (see IPS_AddPermissionToRole). Access to a visualization can also be configured directly in its instance configuration under Security.
A recursive permission applies to the object itself and to all objects below it in the object tree (children, grandchildren, etc.). A recursive permission on the root object (ID 0) therefore applies to all objects. A non-recursive permission applies to exactly this object only. Links are not followed; only the position of the checked object in the object tree matters.
Operations are registered by modules. Currently the visualization registers the operation VISUALIZATION (access to a visualization with all its objects) as soon as a visualization instance (e.g. Tile Visualization or WebFront Visualization) exists. It is checked against the ID of the visualization instance. The operation name is case-sensitive. If the operation is not registered, the function fails with "Cannot verify unsupported operation".



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
Further checks in this order: If the operation is not registered, the function fails with "Cannot verify unsupported operation". If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!". If the object does not exist, it fails with "Object #… does not exist". If the user already has a permission with identical **ObjectID**, **Operation** and **Recursive**, it fails with "User already has this permission". A permission that only differs in **Recursive** counts as a separate entry.
See also: IPS_RemovePermissionFromUser, IPS_UserHasPermission

EXAMPLE

```
$VisuID = 12345; // ID of a tile visualization
IPS_AddPermissionToUser('anna', $VisuID, 'VISUALIZATION', false);
```

IPS_AddRoleToUser

```
bool IPS_AddRoleToUser (string $User, string $Role)
```

PARAMETERS

User	Name of the user (case-sensitive)
Role	Name of the role to be assigned (case-sensitive)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function assigns the role **Role** to the user **User**. The user thereby receives all permissions of the role. If the @admin role is assigned, the user becomes an administrator. Roles are created with IPS_CreateRole.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the role does not exist, the function fails with "Role with name '…' does not exist"; if the user does not exist, with "User with name '…' does not exist". If the role is already assigned, it fails with "User with name '…' already has Role '…' attached". As the @admin user implicitly has all roles, the function also fails with this message for it.
See also: IPS_RemoveRoleFromUser, IPS_UserHasRole

EXAMPLE

```
IPS_CreateRole('visu');
```

```
IPS_AddRoleToUser('anna', 'visu');
```

IPS_CreateUser

```
bool IPS_CreateUser (string $User, int $UserType)
```

PARAMETERS

User			Name of the new user (case-sensitive)
UserType			Type of the user, see the table in the description
	0	USER_TYPE_LOCAL	local account
	1	USER_TYPE_LDAP	LDAP

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function creates a new user with the name **User** and the type **UserType**.

USER TYPE	CONSTANT	DESCRIPTION
0	USER_TYPE_LOCAL	Local account, login with the password set via IPS_SetUserPassword
1	USER_TYPE_LDAP	LDAP, login via the LDAP server with the BindDN set via IPS_SetUserBindDN

Users are usually managed in the Permission Control.
A new user is **inactive** and has no password, no first and last name, no roles and no permissions. For the user to be able to log in, the user must be activated with IPS_SetUserActive and, for a local account, get a password with IPS_SetUserPassword or, for an LDAP account, a BindDN with IPS_SetUserBindDN.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". The new user counts against the user limit of the license (10 users by default, the @admin user is not counted). If the limit would be exceeded, the function fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
Further errors:
An empty name fails with "User cannot be empty".
The name "token" is reserved for logging in with temporary tokens and fails with "User 'token' is reserved and cannot be used".
A name that equals the licensee (e-mail address of the license) fails with "User cannot be equivalent to the licensee".
A user type other than 0 or 1 fails with "Unsupported User Type!".
If a user with this name already exists, the function fails with "User with name '…' already exists".
See also: IPS_DeleteUser, IPS_GetUser, IPS_AddRoleToUser

EXAMPLE

```
IPS_CreateUser('anna', USER_TYPE_LOCAL);
IPS_SetUserPassword('anna', 'secret');
IPS_SetUserFirstName('anna', 'Anna');
IPS_SetUserLastName('anna', 'Muster');
IPS_SetUserActive('anna', true);
```

IPS_DeleteUser

```
bool IPS_DeleteUser (string $User)
```

PARAMETERS

User	Name of the user (case-sensitive)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes the user **User** including the user's role assignments and permissions. The roles themselves are kept.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". The user limit of the license is not checked here, so users can be deleted even if there are more users than the license allows.

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".
See also: IPS_CreateUser, IPS_UserExists

EXAMPLE

```
if (IPS_UserExists('anna')) {
IPS_DeleteUser('anna');
}
```

IPS_GetUser

```
array IPS_GetUser (string $User)
```

PARAMETERS

User	Name of the user (case-sensitive)

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
UserName	string	Name of the user
UserType	integer	Type of the user 0 USER_TYPE_LOCAL Local account 1 USER_TYPE_LDAP LDAP
FirstName	string	First name
LastName	string	Last name
EMail	string	E-mail address
BindDN	string	BindDN for logging in at the LDAP server (only relevant for LDAP users)
IsActive	boolean	**TRUE** if the user is active and can log in
Roles	array	Names of the directly assigned roles
Permissions	array	Directly assigned permissions, see Permissions

PERMISSIONS

INDEX	TYPE	DESCRIPTION
ObjectID	integer	ID of the object the permission applies to
Operation	string	Name of the operation, e.g. VISUALIZATION
Recursive	boolean	**TRUE** if the permission also applies to all objects below the object

DESCRIPTION

The function returns an array with information about the user **User**. The password is not returned; it is only stored as a hash. **Roles** and **Permissions** only contain the entries assigned

directly to the user; permissions resulting from roles can be read with IPS_GetRole or checked with IPS_UserHasPermission.
For the @admin user the function returns the first name "Default" and the last name "Administrator" without roles and permissions, as this user has all permissions anyway.
If the user does not exist, the function fails with "User with name '…' does not exist". The function requires neither the RBAC feature in the license nor administrator permissions.
See also: IPS_GetUserList, IPS_UserExists

EXAMPLE

```
print_r(IPS_GetUser('anna'));

/* e.g. returns:
Array
(
[BindDN] =>
[EMail] => anna@example.com
[FirstName] => Anna
[IsActive] => 1
[LastName] => Muster
[Permissions] => Array
(
[0] => Array
(
[ObjectID] => 12345
[Operation] => VISUALIZATION
[Recursive] =>
)

)

[Roles] => Array
(
[0] => visu
)

[UserName] => anna
[UserType] => 0
)
*/
```

IPS_GetUserList

```
array IPS_GetUserList ()
```

RETURNS

An array with the names of all users, including the @admin user

DESCRIPTION

The function returns the names of all existing users. The @admin user is always included. The order of the entries is not guaranteed.
The function requires neither the RBAC feature in the license nor administrator permissions.
Detailed information about a user is returned by IPS_GetUser.

EXAMPLE

```
print_r(IPS_GetUserList());

/* e.g. returns:
Array
(
[0] => @admin
[1] => anna
)
*/
```

IPS_RemovePermissionFromUser

```
bool IPS_RemovePermissionFromUser (string $User, int $ObjectID, string
$Operation, bool $Recursive)
```

PARAMETERS

User	Name of the user (case-sensitive)
ObjectID	ID of the object the permission applies to
Operation	Name of the registered operation, e.g. VISUALIZATION
Recursive	Must match the value the permission was added with

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function removes the permission for the **Operation** on the object **ObjectID** from the user **User**. The combination of **ObjectID**, **Operation** and **Recursive** must exactly match a permission assigned directly to the user (see **Permissions** in IPS_GetUser). Permissions the user receives via roles are not affected.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
Further checks in this order: If the operation is not registered, the function fails with "Cannot verify unsupported operation". If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!". If the object does not exist, it fails with "Object #… does not exist". If the user does not have the permission, it fails with "User does not have this permission".
See also: IPS_AddPermissionToUser, IPS_RemovePermissionFromRole

EXAMPLE

```
$VisuID = 12345; // ID of a tile visualization
IPS_RemovePermissionFromUser('anna', $VisuID, 'VISUALIZATION', false);
```

IPS_RemoveRoleFromUser

```
bool IPS_RemoveRoleFromUser (string $User, string $Role)
```

PARAMETERS

User	Name of the user (case-sensitive)
Role	Name of the role to be removed (case-sensitive)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function removes the role **Role** from the user **User**. The permissions of the role no longer apply to the user afterwards; the role itself is kept.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". If the role is not assigned to the user, it fails with "User with name '…' does not have Role '…' attached". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".
See also: IPS_AddRoleToUser, IPS_DeleteRole

EXAMPLE

```
if (IPS_UserHasRole('anna', 'visu')) {
IPS_RemoveRoleFromUser('anna', 'visu');
```

```
}
```

IPS_SetUserActive

```
bool IPS_SetUserActive (string $User, bool $Active)
```

PARAMETERS

User	Name of the user (case-sensitive)
Active	Specifies whether the user is to be activated (**TRUE**) / deactivated (**FALSE**)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function activates or deactivates the user **User**. Only active users can log in. This way a user can be blocked temporarily without deleting the user. Newly created users are initially inactive (see IPS_CreateUser). Roles and permissions of the user are kept when deactivating.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".

EXAMPLE

```
IPS_SetUserActive('anna', true); // Activates the user
```

IPS_SetUserBindDN

```
bool IPS_SetUserBindDN (string $User, string $BindDN)
```

PARAMETERS

User	Name of the user (case-sensitive)
BindDN	Distinguished name the user logs in with at the LDAP server

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the BindDN (distinguished name) of the user **User**. When an LDAP user (user type 1, **USER_TYPE_LDAP**) logs in, a bind with this BindDN and the entered password is performed against the LDAP server configured in the Permission Control. The LDAP synchronization of the Permission Control sets the BindDN automatically.
The value can be set for every user type, but it is only evaluated for LDAP users.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".

EXAMPLE

```
IPS_CreateUser('max', USER_TYPE_LDAP);
IPS_SetUserBindDN('max', 'uid=max,cn=users,dc=symcon');
```

```
IPS_SetUserActive('max', true);
```

IPS_SetUserEMail

```
bool IPS_SetUserEMail (string $User, string $EMail)
```

PARAMETERS

User	Name of the user (case-sensitive)
EMail	New e-mail address of the user

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the e-mail address of the user **User**. The value is not validated. The current value can be read with IPS_GetUser.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".

EXAMPLE

```
IPS_SetUserEMail('anna', 'anna@example.com');
```

IPS_SetUserFirstName

```
bool IPS_SetUserFirstName (string $User, string $FirstName)
```

PARAMETERS

User	Name of the user (case-sensitive)
FirstName	New first name of the user

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the first name of the user **User**. First and last name are displayed e.g. in the visualization. The current value can be read with IPS_GetUser.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".

EXAMPLE

```
IPS_SetUserFirstName('anna', 'Anna');
```

IPS_SetUserLastName

```
bool IPS_SetUserLastName (string $User, string $LastName)
```

PARAMETERS

User	Name of the user (case-sensitive)
LastName	New last name of the user

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the last name of the user **User**. First and last name are displayed e.g. in the visualization. The current value can be read with IPS_GetUser.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".

EXAMPLE

```
IPS_SetUserLastName('anna', 'Muster');
```

IPS_SetUserPassword

```
bool IPS_SetUserPassword (string $User, string $Password)
```

PARAMETERS

User	Name of the user (case-sensitive)
Password	New password of the user

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function sets the password of the user **User**. The password is not stored in plain text but only as a salted hash (PBKDF2-HMAC-SHA256) and therefore cannot be read back.
The password is only used for local accounts (user type 0, USER_TYPE_LOCAL). As long as no password has been set for a local account, logging in is not possible. LDAP users (user type 1, USER_TYPE_LDAP) log in via the LDAP server with their BindDN instead (see IPS_SetUserBindDN); the password set here is not evaluated for them.



This function is only available for licenses that include the RBAC feature (Permission Control). Otherwise it fails with "This function is only available for licenses with the RBAC feature enabled!". If there are more users than the license allows (10 users by default, the @admin user is not counted), it fails with "Your license is currently limited to 10 users. Please consider upgrading the license to more users!" (the number is the limit of the license).

The currently logged in user (see IPS_GetLoggedInUser) must be an administrator, i.e. the @admin user or a user with the @admin role. Otherwise the function fails with "Administrator permissions are required for this action!".
If the user does not exist, the function fails with "User with name '…' does not exist". The @admin user cannot be modified; trying to do so fails with "User @admin cannot be modified!".

EXAMPLE

```
IPS_SetUserPassword('anna', 'secret');
```

IPS_UserExists

```
bool IPS_UserExists (string $User)
```

PARAMETERS

User	Name of the user to be checked (case-sensitive)

RETURNS

If the user exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether a user with the name **User** exists. The check is case-sensitive.
The @admin user always exists, whereas the special user @unknown is not an actual user, so the function returns **FALSE** for it.
The function requires neither the RBAC feature in the license nor administrator permissions.

EXAMPLE

```
if (IPS_UserExists('anna')) {
echo 'User exists!';
}
```

IPS_UserHasPermission

```
bool IPS_UserHasPermission (string $User, int $ObjectID, string
$Operation)
```

PARAMETERS

User	Name of the user (case-sensitive)
ObjectID	ID of the object to be checked
Operation	Name of the registered operation, e.g. VISUALIZATION

RETURNS

If the user has the permission, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the user **User** may perform the **Operation** on the object **ObjectID**. The permissions of the user and the permissions of all the user's roles are taken into account. Whether the user is active does not matter for this check.
The check is performed in this order: If the operation is not registered, the function fails with "Cannot verify unsupported operation"; if the object does not exist, with "Object #… does not exist". The @admin user always has all permissions, the @unknown user never has any. For all other users: If the user does not exist, the function fails with "User with name '…' does not exist". If the user has the @admin role, **TRUE** is returned. Otherwise **TRUE** is returned if a permission of the user or of one of the user's roles grants the operation on the object.
A recursive permission applies to the object itself and to all objects below it in the object tree (children, grandchildren, etc.). A recursive permission on the root object (ID 0) therefore applies to all objects. A non-recursive permission applies to exactly this object only. Links are not followed; only the position of the checked object in the object tree matters.
Operations are registered by modules. Currently the visualization registers the operation **VISUALIZATION** (access to a visualization with all its objects) as soon as a visualization instance (e.g. Tile Visualization or WebFront Visualization) exists. It is checked against the ID of the visualization instance. The operation name is case-sensitive. If the operation is not registered, the function fails with "Cannot verify unsupported operation".
The function requires neither the RBAC feature in the license nor administrator permissions. For the currently logged in user, IPS_HasPermission can be used.

EXAMPLE

```
$VisuID = 12345; // ID of a tile visualization
IPS_AddPermissionToUser('anna', 0, 'VISUALIZATION', true); // recursive
from the root object
var_dump(IPS_UserHasPermission('anna', $VisuID, 'VISUALIZATION'));

/* returns:
bool(true)
*/
```

IPS_UserHasRole

```
bool IPS_UserHasRole (string $User, string $Role)
```

PARAMETERS

User	Name of the user (case-sensitive)
Role	Name of the role to be checked (case-sensitive)

RETURNS

If the role is assigned to the user, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the role **Role** is assigned to the user **User**. The role itself does not need to exist.
For the @admin user the function returns **TRUE** for every role, for the special user @unknown always **FALSE**. If the user does not exist, the function fails with "User with name '…' does not exist". The function requires neither the RBAC feature in the license nor administrator permissions.
See also: IPS_AddRoleToUser, IPS_UserIsAdministrator

EXAMPLE

```
if (IPS_UserHasRole('anna', 'visu')) {
echo 'Anna has the role visu';
}
```

IPS_UserIsAdministrator

```
bool IPS_UserIsAdministrator (string $User)
```

PARAMETERS

User	Name of the user (case-sensitive)

RETURNS

If the user is an administrator, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the user **User** is an administrator. Administrators are the @admin user and every user that has been assigned the @admin role (see IPS_AddRoleToUser). For the special user @unknown the function returns **FALSE**.
If the user does not exist, the function fails with "User with name '…' does not exist". The function requires neither the RBAC feature in the license nor administrator permissions.
For the currently logged in user, IPS_IsAdministrator can be used.

EXAMPLE

```
var_dump(IPS_UserIsAdministrator('anna'));

/* e.g. returns:
bool(false)
*/
```

IPS_GetLoggedInUser

```
string IPS_GetLoggedInUser ()
```

RETURNS

Name of the currently logged in user, e.g. "@admin" or "@unknown"

DESCRIPTION

The function returns the name of the user in whose context the current execution (e.g. a script) runs. A script inherits the user of the context that started it. If a script is started e.g. by an action in a visualization where a user is logged in, the function returns the name of this user.
The following special users exist:

USER	DESCRIPTION
@admin	Administrator with all permissions. Logging in is done with the license (e-mail address) and the remote access password, e.g. in the console.
@unknown	No user is logged in. This user has no permissions at all.

The function does not require the RBAC feature in the license. The executing user is also shown in the traces as **ExecutionUser**, see IPS_GetTraces.
See also: IPS_IsAdministrator, IPS_IsUnknown, IPS_HasPermission

EXAMPLE

```
echo IPS_GetLoggedInUser();

/* e.g. returns:
@admin
*/
```

IPS_HasPermission

```
bool IPS_HasPermission (int $ObjectID, string $Operation)
```

PARAMETERS

ObjectID	ID of the object to be checked
Operation	Name of the registered operation, e.g. VISUALIZATION

RETURNS

If the currently logged in user has the permission, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the user in whose context the current execution runs (see IPS_GetLoggedInUser) may perform the **Operation** on the object **ObjectID**. The permissions of the user and the permissions of all the user's roles are taken into account. It corresponds to IPS_UserHasPermission for the current user.
The check is performed in this order: If the operation is not registered, the function fails with "Cannot verify unsupported operation"; if the object does not exist, with "Object #… does not exist". The @admin user always has all permissions, the @unknown user never has any. For all other users: If the user does not exist, the function fails with "User with name '…' does not exist". If the user has the @admin role, **TRUE** is returned. Otherwise **TRUE** is returned if a permission of the user or of one of the user's roles grants the operation on the object.
A recursive permission applies to the object itself and to all objects below it in the object tree (children, grandchildren, etc.). A recursive permission on the root object (ID 0) therefore applies to all objects. A non-recursive permission applies to exactly this object only. Links are not followed; only the position of the checked object in the object tree matters.
Operations are registered by modules. Currently the visualization registers the operation **VISUALIZATION** (access to a visualization with all its objects) as soon as a visualization instance (e.g. Tile Visualization or WebFront Visualization) exists. It is checked against the ID of the visualization instance. The operation name is case-sensitive. If the operation is not registered, the function fails with "Cannot verify unsupported operation".
The function does not require the RBAC feature in the license.

EXAMPLE

```
$VisuID = 12345; // ID of a tile visualization
if (IPS_HasPermission($VisuID, 'VISUALIZATION')) {
```

```
echo 'Access granted';
}
```

IPS_IsAdministrator

```
bool IPS_IsAdministrator ()
```

RETURNS

If the currently logged in user is an administrator, TRUE is returned, otherwise FALSE.

DESCRIPTION

The function checks whether the user in whose context the current execution runs (see IPS_GetLoggedInUser) is an administrator. Administrators are the @admin user and every user that has been assigned the @admin role. For the @unknown user the function always returns FALSE.
Administrator permissions are required e.g. for all functions that modify users and roles. The function does not require the RBAC feature in the license.
To check an arbitrary user, IPS_UserIsAdministrator can be used.

EXAMPLE

```
if (IPS_IsAdministrator()) {
echo 'The script runs with administrator permissions';
}
```

IPS_IsUnknown

```
bool IPS_IsUnknown ()
```

RETURNS

If the current user is @unknown, TRUE is returned, otherwise FALSE.

DESCRIPTION

The function checks whether the current execution runs without a logged in user, i.e. whether IPS_GetLoggedInUser returns the special user @unknown. This is the case if no user is logged in in the triggering context or no login succeeded. The @unknown user has no permissions at all and is not an administrator.
The function does not require the RBAC feature in the license.

EXAMPLE

```
if (IPS_IsUnknown()) {
echo 'No user logged in';
} else {
echo 'Logged in as ' . IPS_GetLoggedInUser();
}
```

Roles

IPS_AddPermissionToRole	adds a permission to a role
IPS_CreateRole	creates a new role
IPS_DeleteRole	deletes a role
IPS_GetRole	returns information about a role
IPS_GetRoleList	returns the names of all roles
IPS_RemovePermissionFromRole	removes a permission from a role
IPS_RoleExists	checks whether a role exists
IPS_RoleHasPermission	checks whether a role has a permission for an object

IPS_AddPermissionToRole

```
bool IPS_AddPermissionToRole (string $Role, int $ObjectID, string
$Operation, bool $Recursive)
```

PARAMETERS

Role	Name of the role
ObjectID	ID of the object the permission applies to
Operation	Name of the operation, e.g. VISUALIZATION
Recursive	Specifies whether the permission also applies to all objects below **ObjectID** (**TRUE**) or only to the object itself (**FALSE**)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function grants the role **Role** the permission to perform **Operation** on the object **ObjectID**. All users the role is assigned to thereby receive this permission.
If **Recursive** is enabled, the permission applies to the object and all objects below it – a recursive permission on the root object (ID 0) therefore applies to all objects. A permission is identified by **ObjectID**, **Operation** and **Recursive** together; the same **ObjectID** and **Operation** can therefore be added once recursively and once non-recursively.
Operation must be an operation registered in the system, otherwise the function fails with "Cannot verify unsupported operation". The operation is case-sensitive. Currently there is the operation `VISUALIZATION` (access to a visualization with all its objects). It is registered as soon as a visualization instance (e.g. Tile Visualization) exists and is checked against the ID of the visualization instance.
The checks are performed in this order: operation, role ("Role with name '…' does not exist"), object ("Object #… does not exist"). If the same permission already exists, the function fails with "Role already has this permission".
The executing user must be an administrator (see IPS_IsAdministrator), otherwise the function fails with "Administrator permissions are required for this action!".



Role management is only available with a license that includes the RBAC feature. Otherwise the function fails with "This function is only available for licenses with the RBAC feature enabled!". If more users exist than the license allows, the function fails with "Your license is currently limited to … users. Please consider upgrading the license to more users!".

EXAMPLE

```
// The role may open the visualization with the ID 12345
IPS_AddPermissionToRole('Residents', 12345, 'VISUALIZATION', false);
```

IPS_CreateRole

```
bool IPS_CreateRole (string $Role)
```

PARAMETERS

Role	Name of the role to be created

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function creates a new role with the name **Role**. A new role has no permissions yet. They are added with IPS_AddPermissionToRole. Afterwards the role can be assigned to any number of users with IPS_AddRoleToUser, who thereby receive all permissions of the role. Roles can also be managed in the Permission Control.
The name must not be empty ("Role cannot be empty") and must be unique, otherwise the function fails with "Role with name '…' already exists". The name is case-sensitive. The role @admin always exists and grants all permissions.
The executing user must be an administrator (see IPS_IsAdministrator), otherwise the function fails with "Administrator permissions are required for this action!".



Role management is only available with a license that includes the RBAC feature. Otherwise the function fails with "This function is only available for licenses with the RBAC feature enabled!". If more users exist than the license allows, the function fails with "Your license is currently limited to … users. Please consider upgrading the license to more users!".

EXAMPLE

```
if (!IPS_RoleExists('Residents')) {
IPS_CreateRole('Residents');
}

// Allow access to the visualization with the ID 12345
IPS_AddPermissionToRole('Residents', 12345, 'VISUALIZATION', false);
```

```
// Assign the role to a user
IPS_AddRoleToUser('anna', 'Residents');
```

IPS_DeleteRole

```
bool IPS_DeleteRole (string $Role)
```

PARAMETERS

Role	Name of the role to be deleted

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes the role with the name **Role** together with its permissions. The role is automatically removed from all users it was assigned to – they lose all permissions they only had through this role.
If the role does not exist, the function fails with "Role with name '…' does not exist". The role @admin cannot be deleted ("Role @admin cannot be deleted!").
The executing user must be an administrator (see IPS_IsAdministrator), otherwise the function fails with "Administrator permissions are required for this action!".



Role management is only available with a license that includes the RBAC feature. Otherwise the function fails with "This function is only available for licenses with the RBAC feature enabled!". If more users exist than the license allows, the function fails with "Your license is currently limited to … users. Please consider upgrading the license to more users!".

EXAMPLE

```
if (IPS_RoleExists('Residents')) {
IPS_DeleteRole('Residents');
}
```

IPS_GetRole

```
array IPS_GetRole (string $Role)
```

PARAMETERS

Role	Name of the role

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
RoleName	string	Name of the role
Permissions	array	List of the permissions of the role (see permission table)

Permission table

INDEX	TYPE	DESCRIPTION
ObjectID	integer	ID of the object the permission applies to
Operation	string	Name of the operation, e.g. VISUALIZATION
Recursive	boolean	The permission also applies to all objects below ObjectID

DESCRIPTION

The function returns the name and the permissions of the role with the name **Role**. If the role does not exist, the function fails with "Role with name '...' does not exist".
The role @admin has no entries in **Permissions**, but still grants all permissions. Whether a role has a specific permission is checked by IPS_RoleHasPermission. The function requires neither administrator permissions nor the RBAC feature in the license.

EXAMPLE

```
print_r(IPS_GetRole('Residents'));

/* e.g. returns:
Array
(
[Permissions] => Array
(
```

```
[0] => Array
(
[ObjectID] => 12345
[Operation] => VISUALIZATION
[Recursive] =>
)

)

[RoleName] => Residents
)
*/
```

IPS_GetRoleList

```
array IPS_GetRoleList ()
```

RETURNS

An array with the names of all existing roles.

DESCRIPTION

The function returns the names of all roles, including the always existing role @admin. The order of the entries is not guaranteed. Details of a role are returned by IPS_GetRole. The function requires neither administrator permissions nor the RBAC feature in the license.

EXAMPLE

```
print_r(IPS_GetRoleList());

/* e.g. returns:
Array
(
[0] => @admin
[1] => Residents
)
*/
```

IPS_RemovePermissionFromRole

```
bool IPS_RemovePermissionFromRole (string $Role, int $ObjectID, string $Operation, bool $Recursive)
```

PARAMETERS

Role	Name of the role
ObjectID	ID of the object the permission applies to
Operation	Name of the operation, e.g. VISUALIZATION
Recursive	Specifies whether the permission also applies to all objects below **ObjectID** (**TRUE**) or only to the object itself (**FALSE**)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function removes the permission to perform **Operation** on the object **ObjectID** from the role **Role**. **ObjectID**, **Operation** and **Recursive** must exactly match the permission added with IPS_AddPermissionToRole (see IPS_GetRole), otherwise the function fails with "Role does not have this permission".
Operation must be an operation registered in the system, otherwise the function fails with "Cannot verify unsupported operation". The operation is case-sensitive. Currently there is the operation `VISUALIZATION` (access to a visualization with all its objects). It is registered as soon as a visualization instance (e.g. Tile Visualization) exists and is checked against the ID of the visualization instance.
The checks are performed in this order: operation, role ("Role with name '…' does not exist"), object ("Object #… does not exist").
The executing user must be an administrator (see IPS_IsAdministrator), otherwise the function fails with "Administrator permissions are required for this action!".



Role management is only available with a license that includes the RBAC feature. Otherwise the function fails with "This function is only available for licenses with the RBAC feature enabled!". If more users exist than the license allows, the function fails with "Your license is currently limited to … users. Please consider upgrading the license to more users!".

EXAMPLE

```
// The role may no longer open the visualization with the ID 12345
IPS_RemovePermissionFromRole('Residents', 12345, 'VISUALIZATION',
false);
```

IPS_RoleExists

```
bool IPS_RoleExists (string $Role)
```

PARAMETERS

Role	Name of the role to be checked

RETURNS

If the role exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether a role with the name **Role** exists. The name is case-sensitive. The role @admin always exists. The function requires neither administrator permissions nor the RBAC feature in the license.

EXAMPLE

```
if (IPS_RoleExists('Residents')) {
echo 'Role already exists!';
}
```

IPS_RoleHasPermission

```
bool IPS_RoleHasPermission (string $Role, int $ObjectID, string
$Operation)
```

PARAMETERS

Role	Name of the role
ObjectID	ID of the object to be checked
Operation	Name of the operation, e.g. VISUALIZATION

RETURNS

If the role has the permission, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the role **Role** may perform **Operation** on the object **ObjectID**. This is the case if the role has a permission with this operation directly for **ObjectID** or a recursive permission for **ObjectID** or one of its parent objects (up to the root object with the ID 0). The role @admin always has all permissions.
Only the permissions of the role are taken into account. Whether a user is permitted – by their own permissions or one of their roles – is checked by IPS_UserHasPermission.
The checks are performed in this order: operation ("Cannot verify unsupported operation"), object ("Object #… does not exist"), role ("Role with name '…' does not exist"). The operation must be registered in the system; currently there is the operation `VISUALIZATION` as soon as a visualization instance exists. The function requires neither administrator permissions nor the RBAC feature in the license.

EXAMPLE

```
if (IPS_RoleHasPermission('Residents', 12345, 'VISUALIZATION')) {
echo 'The role may open the visualization';
}
```

Management of Events

IPS_CreateEvent	create a new event
IPS_DeleteEvent	deletes an event
IPS_EventExists	checks if an event exists
IPS_GetEvent	returns extensive information about an event
IPS_GetEventIDByName	determines the ID of an event by its name
IPS_GetEventList	returns a list of all existing events
IPS_GetEventListByType	returns a list of all events of a given type
IPS_IsConditionPassing	checks if a condition is fulfilled
IPS_SetEventAction	set the executed action
IPS_SetEventActive	activates/deactivates an event
IPS_SetEventCondition	modifies the conditions of an event
IPS_SetEventConditionDateRule	modifies the rule of a condition of an event for date comparisons
IPS_SetEventConditionDayOfTheWeekRule	modifies the rule of a condition of an event for weekday comparisons
IPS_SetEventConditionTimeRule	modifies the rule of a condition of an event for time comparisons
IPS_SetEventConditionVariableDynamicRule	modifies a dynamic condition rule of an event for variable comparison
IPS_SetEventConditionVariableRule	modifies a condition rule of an event for variable comparison
IPS_SetEventConditionVariableStaticRule	modifies a static condition rule of an event for variable comparison
IPS_SetEventCyclic	configures a cyclic event
IPS_SetEventCyclicDateFrom	sets the start date
IPS_SetEventCyclicDateTo	sets the end date
IPS_SetEventCyclicTimeFrom	set the start time
IPS_SetEventCyclicTimeTo	sets the end time
IPS_SetEventLimit	limits the number of event executions
IPS_SetEventScheduleAction	modifies the action table of a schedule event and sets the action to "Execute PHP Code"
IPS_SetEventScheduleActionEx	modifies the action table of a schedule event
IPS_SetEventScheduleGroup	modifies the grouping of days for a schedule event
IPS_SetEventScheduleGroupPoint	modifies a switch point of the group of a schedule event
IPS_SetEventScript	sets the action to "Execute PHP Code" and define the code to be executed
IPS_SetEventTrigger	defines the trigger for a triggered event

IPS_SetEventTriggerSubsequentExecution	defines whether a triggered event is executed again on the same value
IPS_SetEventTriggerValue	set the value the trigger variable is compared to

IPS_CreateEvent

```
int IPS_CreateEvent (int $EventType)
```

PARAMETERS

EventType

VALUE	CONSTANT	DESCRIPTION
0	EVENTTYPE_TRIGGER	creates a "triggered" event
1	EVENTTYPE_CYCLIC	creates a "cyclical" event
2	EVENTTYPE_SCHEDULE	creates a "schedule" event

RETURNS

ID of the newly created event

DESCRIPTION

The function creates an event object of the type **EventType**.
The function returns an ID that can be used to uniquely identify the created event object. After creation, the object is still unconfigured and must be configured with the functions IPS_SetEventActive, IPS_SetEventAction and the corresponding IPS_SetEventCyclic / IPS_SetEventTrigger.

EXAMPLE

```
$eid = IPS_CreateEvent(EVENTTYPE_TRIGGER); //Triggered event
IPS_SetEventTrigger($eid, EVENTTRIGGERTYPE_ONCHANGE, 15754); //On
change of variable with ID 15754
IPS_SetParent($eid, $_IPS['SELF']); //Assign the event
IPS_SetEventActive($eid, true); //Activate the event

$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC); //Cyclic event
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_DAY, 1, 0, 0,
EVENTCYCLICTIMETYPE_HOUR, 6); //Every day, every 6 hours
IPS_SetEventCyclicDateFrom($eid, 1, 12, date("Y")); //From December 1st
of this year
IPS_SetEventCyclicDateTo($eid, 31, 12, date("Y")); //Until December
31st of this year

IPS_SetEventCyclicTimeFrom($eid, 15, 0, 0); //From 15:00
```

```
IPS_SetEventCyclicTimeTo($eid, 23, 30, 0); //Until 23:30

IPS_SetParent($eid, $_IPS['SELF']); //Assign the event
IPS_SetEventActive($eid, true); //Activate the event

// Required since IP-Symcon 6.0, if the event shall execute an
automation (e.g. a PHP script)
IPS_SetEventAction($eid, '{7938A5A2-0981-5FE0-BE6C-8AA610D654EB}', []);
```

IPS_DeleteEvent

```
bool IPS_DeleteEvent (int $EventID)
```

PARAMETERS

EventID	ID of the event to be deleted

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes an existing event with the ID **EventID**.



The event can only be deleted if it has no child objects.

EXAMPLE

```
IPS_DeleteEvent($EventID);
```

IPS_EventExists

```
bool IPS_EventExists (int $EventID)
```

PARAMETERS

EventID	ID of the event to be tested

RETURNS

If the EventID exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether an event with the ID **EventID** exists.

EXAMPLE

```
if (IPS_EventExists(34881))
echo "An event with this ID exists!";
```

IPS_GetEvent

Require: Symcon >= 3.1

```
array IPS_GetEvent (int $EventID)
```

PARAMETERS

EventID	ID of the event

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
CyclicDateType	integer	Date type. See IPS_SetEventCyclic
CyclicDateValue	integer	Date interval. See IPS_SetEventCyclic
CyclicDateDay	integer	Day of date. See IPS_SetEventCyclic
CyclicDateDayValue	integer	Day of date, interval. See IPS_SetEventCyclic
CyclicDateFrom	array	Start date as array of day, month, year; 0/0/0 = Always. See IPS_SetEventCyclicDateFrom
CyclicDateTo	array	End date as array of day, month, year; 0/0/0 = No end date. See IPS_SetEventCyclicDateTo
CyclicTimeType	integer	Time type. See IPS_SetEventCyclic
CyclicTimeValue	integer	Time interval. See IPS_SetEventCyclic
CyclicTimeFrom	array	Start time as array of hour, minute, second; 0/0/0 = Midnight. See IPS_SetEventCyclicTimeFrom
CyclicTimeTo	array	End time as array of hour, minute, second; 0/0/0 = No end time (until 23:59:59). See IPS_SetEventCyclicTimeTo
EventActionID	string	ID of the executed Action
EventActionParameters	array	Parameters of the executed Action
EventConditions	array	Array of conditions. See IPS_SetEventCondition, IPS_SetEventConditionDateRule, IPS_SetEventConditionDayOfTheWeekRule, IPS_SetEventConditionTimeRule and IPS_SetEventConditionVariableRule
EventID	integer	ID of the event
EventLimit	integer	Remaining number of executions. 0 = No Limit

INDEX	TYPE	DESCRIPTION
EventScript	string	Only present for compatibility reasons. Contains the PHP code if the action was set via IPS_SetEventScript, otherwise an empty string. The execution is always controlled via the action, which is configured via IPS_SetEventAction.
EventActive	boolean	TRUE if the event is active
EventType	integer	Event type 0 EVENTTYPE_TRIGGER trigger 1 EVENTTYPE_CYCLIC cyclic 2 EVENTTYPE_SCHEDULE schedule
LastRun	integer	Unix timestamp of the last run, 0 = Never
NextRun	integer	Unix timestamp of the next run, 0 = Never
LastActionID	integer	ID of the schedule action of the last execution, 0 = No action was executed (also for other event types)
NextActionID	integer	ID of the schedule action of the next execution, 0 = No action will be executed next time (also for other event types)
ScheduleActions	array	Array of actions of the schedule. See IPS_SetEventScheduleActionEx
ScheduleGroups	array	Array of groups with switch points of the schedule. See IPS_SetEventScheduleGroup and IPS_SetEventScheduleGroupPoint
TriggerSubsequentExecution	boolean	Allow repeated execution when triggered without a value change
TriggerType	integer	Value for the trigger type: See IPS_SetEventTrigger
TriggerValue	variant	Value that is used for the trigger check, depending on the used trigger type
TriggerVariableID	integer	VariableID to be used as a trigger

DESCRIPTION

The function returns an array containing information about the event with the ID **EventID**.



Since IP-Symcon 3.1, the fields CyclicDateFrom, CyclicDateTo, CyclicTimeFrom, and CyclicTimeTo are arrays. If you need compatibility with version 3.0 and older, you can use the IPS_GetEventCompatibility function available since IP-Symcon 3.1, which restores the old functionality.



LastActionID/NextActionID have been available since version 8.1

EXAMPLE

```
$EventID = 46413;
$EventInfo = IPS_GetEvent($EventID);
print_r($EventInfo);

/* returns e.g.:
Array
(
[EventID] => 41227
[EventType] => 2
[EventActive] => 1
[EventLimit] => 0
[EventConditions] => Array
(
[0] => Array
(
[ID] => 0
[ParentID] => 0
[VariableRules] => Array
(
[0] => Array
(
[ID] => 1
[VariableID] => 29025
[Comparison] => 4
[Type] => 0
[Value] => 500
)

)

[DateRules] => Array
(
[0] => Array
(
[ID] => 0
[Comparison] => 0
[Value] => Array
(
[Day] => 8
[Month] => 8
[Year] => 2001
)
)

[1] => Array
(
[ID] => 1
[Comparison] => 4
```

```
[Value] => Array
(
[Day] => 1
[Month] => 1
[Year] => 2021
)
)
)

[TimeRules] => Array
(
[0] => Array
(
[ID] => 1
[Comparison] => 3
[Value] => Array
(
[Hour] => 9
[Minute] => 0
[Second] => 0
)

)

[1] => Array
(
[ID] => 2
[Comparison] => 5
[Value] => Array
(
[Hour] => 17
[Minute] => 0
[Second] => 0
)

)

)
[DayOfTheWeekRules] => Array
(
[0] => Array
(
[ID] => 0
[Comparison] => 0
[Value] => 3
)

)

[Operation] => 1
)
)

[TriggerType] => 0
[TriggerVariableID] => 0
[TriggerValue] =>
[TriggerSubsequentExecution] =>
[CyclicDateType] => 0
[CyclicDateValue] => 0
```

```
[CyclicDateDay] => 0
[CyclicDateDayValue] => 0
[CyclicDateFrom] => Array
(
[Day] => 0
[Month] => 0
[Year] => 0
)

[CyclicDateTo] => Array
(
[Day] => 0
[Month] => 0
[Year] => 0
)

[CyclicTimeType] => 0
[CyclicTimeValue] => 0
[CyclicTimeFrom] => Array
(
[Hour] => 0
[Minute] => 0
[Second] => 0
)

[CyclicTimeTo] => Array
(
[Hour] => 0
[Minute] => 0
[Second] => 0
)

[ScheduleActions] => Array
(
[0] => Array
(
[ID] => 1
[Name] => Kalt
[Color] => 255
[ScriptText] =>
[ActionID] => {3644F802-C152-464A-868A-242C2A3DEC5C}
[ActionParameters] => Array
(
[VALUE] => 2
)
)

[1] => Array
(
[ID] => 2
[Name] => Warm
[Color] => 16711680
[ScriptText] =>
[ActionID] => {3644F802-C152-464A-868A-242C2A3DEC5C}
[ActionParameters] => Array
(
[VALUE] => 0
)
)
```

```
)

[ScheduleGroups] => Array
(
[0] => Array
(
[ID] => 0
[Days] => 127
[Points] => Array
(
[0] => Array
(
[ID] => 13
[Start] => Array
(
[Hour] => 0
[Minute] => 0
[Second] => 0
)

[ActionID] => 2
)

[1] => Array
(
[ID] => 14
[Start] => Array
(
[Hour] => 12
[Minute] => 0
[Second] => 0
)

[ActionID] => 1
)

)

)

)

[EventScript] =>
[EventActionID] =>
[EventActionParameters] => Array
(
)

[LastRun] => 0
[NextRun] => 0
[LastActionID] => 0
[NextActionID] => 0
)
*/
```

IPS_GetEventIDByName

```
int IPS_GetEventIDByName (string $EventName, int $ParentID)
```

PARAMETERS

EventName	Name of the event
ParentID	Object whose child objects are searched for the event

RETURNS

ID of the found event, otherwise FALSE.

DESCRIPTION

The function is trying to determine the ID of an event with the name **EventName** and the parent with the ID **ParentID**. It will return the ID of the first found event whose name matches with **EventName**. If such an event is found, the function returns its ID and FALSE otherwise.



Caution: Names in IP-Symcon can be chosen freely, so they are **not unique**. For this reason, the function can return a wrong ID.



This function generates a **warning** if the name was not found. Use the @ operator, if you want to suppress this message. The exact function of this operator is described in the PHP manual.

EXAMPLE

```
$EventID = @IPS_GetEventIDByName("TimerABC", $ParentID);
if ($EventID === false)
echo "Event not found!";
else
echo "The Event ID is: ". $EventID;
```

IPS_GetEventList

```
array IPS_GetEventList ()
```

RETURNS

An array of integer values of all IDs of the events in IP-Symcon. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all registered events in IP-Symcon. The IDs are listed in an array. If no event exists, the array is empty.

EXAMPLE

```
$allEvents = IPS_GetEventList();
print_r($allEvents);

/* returns e.g.:
Array
(
[0] => 37659
[1] => 18326
etc. ...
)
*/
```

IPS_GetEventListByType

```
array IPS_GetEventListByType (int $EventType)
```

PARAMETERS

EventType

VALUE	CONSTANT	DESCRIPTION
0	EVENTTYPE_TRIGGER	triggered event
1	EVENTTYPE_CYCLIC	cyclical event
2	EVENTTYPE_SCHEDULE	schedule event

RETURNS

An array of integer values of all IDs of the events of the type **EventType** in IP-Symcon. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all events of a specific **EventType**. The IDs are listed in an array. If no event of the specific type exists, the array is empty.
The list only contains events of the event type **EventType**. If all events are wanted, the function IPS_GetEventList can be used.

EXAMPLE

```
$allEvents = IPS_GetEventListByType(EVENTTYPE_CYCLIC); // list only
cyclical events
print_r($allEvents);

/* returns e.g.:
Array
(
[0] => 37659
[1] => 18326
etc. …
)
*/
```

IPS_IsConditionPassing

Require: Symcon >= 6.1

```
bool IPS_IsConditionPassing (string $Condition)
```

PARAMETERS

Condition	A list of JSON encoded conditions in the format of SelectCondition

RETURNS

TRUE, if the condition is fulfilled, otherwise FALSE

DESCRIPTION

This function checks if the JSON encoded condition **Condition** is fulfilled. It is typically used in conjunction with a module that uses the form field SelectCondition.

EXAMPLE

```
if (IPS_IsConditionPassing($Condition)) {
// Only do something if the condition is fulfilled
};
```

IPS_SetEventAction

Require: Symcon >= 6.0

```
bool IPS_SetEventAction (int $EventID, string $ActionID, array
$ActionParameters)
```

PARAMETERS

EventID	ID of the event to be changed
ActionID	ID of the executed action
ActionParameters	List of Parameters for the executed action

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The Action of the Event with the ID **EventID** is set to the action with the ID **ActionID**. The parameters of that action are defined as **ActionParameters**. The action ID is not a numeric ID but a GUID in the format {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}.



The special parameters TARGET, ENVIRONMENT, and PARENT are not required here and are handled by the event

MIGRATION FROM IP-SYMCON 5.5 OR OLDER

If the event should start the parent script (the default in IP-Symcon 5.5 or older) this connection must be set accordingly since IP-Symcon 6.0. This requires the following line, which assigns the "Run Automation" action to the event.

```
IPS_SetEventAction($EventID, "{7938A5A2-0981-5FE0-BE6C-8AA610D654EB}",
[]);
```

EXAMPLE

```
// Multiply the target variable of the event by 3
IPS_SetEventAction($EventID, "{A3153696-013A-41B1-A001-5E8085D95465}",
["FACTOR" => 3]);

// Execute the target automation of the event
IPS_SetEventAction($EventID, "{7938A5A2-0981-5FE0-BE6C-8AA610D654EB}",
[]);
```

IPS_SetEventActive

```
bool IPS_SetEventActive (int $EventID, bool $Active)
```

PARAMETERS

EventID	ID of the event to be changed
Active	Indicates whether the event should be activated (TRUE) or deactivated (FALSE)

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

After the activation / deactivation, the new status can be read via IPS_GetEvent. It is possible to determine the next execution date for active cyclic and schedule events. For a deactivated event the NextRun value is zero.

EXAMPLE

```
IPS_SetEventActive($EventID, true); // Activates the event
```

IPS_SetEventCondition

Require: Symcon >= 4.4

```
bool IPS_SetEventCondition (int $EventID, int $ConditionID, int
$ParentID, int $Operation)
```

PARAMETERS

EventID	ID of the event to be changed
ConditionID	Unique ID for this condition. The root condition needs to have the ID 0. The ID only needs to be unique for this event.
ParentID	ID of the parent condition. On creation of the root condition, this parameter needs to be 0.

Operation

VALUE	CONSTANT	DESCRIPTION
0	EVENT_CONDITION_OPERATION_AND	AND
1	EVENT_CONDITION_OPERATION_OR	OR
2		NAND (not implemented yet)
3		NOR (not implemented yet)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function modifies the condition with the ID **ConditionID** for the event with the ID **EventID**. The condition will become a child of the condition with the ID **ParentID** and use the operation **Operation**. If no condition with the ID **ConditionID** exists yet, it is created.
If **ParentID** is negative, the condition with the ID **ConditionID** is deleted. A condition cannot be deleted if it has child conditions. When a condition is deleted, all its rules are deleted as well.



A condition evaluates by applying its operation to the set of all its rules and child conditions. This can be used to create nested conditions.

EXAMPLE

```
// Create root condition with the operation OR for event with ID 12345
IPS_SetEventCondition(12345, 0, 0, EVENT_CONDITION_OPERATION_OR);

// Create a child condition with the operation AND
IPS_SetEventCondition(12345, 1, 0, EVENT_CONDITION_OPERATION_AND);

// Delete child condition
IPS_SetEventCondition(12345, 1, -1, EVENT_CONDITION_OPERATION_AND);
```

IPS_SetEventConditionDateRule

Require: Symcon >= 4.4

```
bool IPS_SetEventConditionDateRule (int $EventID, int $ConditionID, int $RuleID, int $Comparison, int $Day, int $Month, int $Year)
```

PARAMETERS

EventID	ID of the event to be changed
ConditionID	ID of the condition that contains the rule
RuleID	Unique ID for this rule. IDs need to be unique for this condition only.
Comparison	

VALUE	CONSTANT	DESCRIPTION
0	EVENTCONDITIONCOMPARISON_EQUAL	==
1	EVENTCONDITIONCOMPARISON_NOTEQUAL	!=
2	EVENTCONDITIONCOMPARISON_GREATER	>
3	EVENTCONDITIONCOMPARISON_GREATEROREQUAL	>=
4	EVENTCONDITIONCOMPARISON_SMALLER	<
5	EVENTCONDITIONCOMPARISON_SMALLEROREQUAL	<=

Day	Day with which the current date is compared
Month	Month with which the current date is compared
Year	Year with which the current date is compared

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function modifies the rule with the ID **RuleID** of the condition with the ID **ConditionID** of the event with the ID **EventID**. The rule compares the current date with the **Day**, **Month** and **Year**, according to the comparison operation **Comparison**. If there is no rule with the ID **RuleID**, one will be created.
If one of the values is outside of the valid range (**Day** < 0 or > 31, **Month** < 0 or > 12, **Year** < 0), the rule with the ID **RuleID** will be deleted. If no rule with this ID exists, an error is raised.



Day, **Month** and **Year** can be set to 0 to be used as a wildcard



Which RuleID contains which rule can be looked up via IPS_GetEvent.

EXAMPLE

```
//Add an AND condition
IPS_SetEventCondition(12345, 0, 0, EVENT_CONDITION_OPERATION_AND);

// Adds / modifies the rule with the ID 2 for the event with the
ObjectID 12345
// It is the first half of the year -> current date < 01.07.2018
IPS_SetEventConditionDateRule(12345, 0, 2,
EVENTCONDITIONCOMPARISON_SMALLER, 1, 7, 2018);

// Delete the rule with ID 1
IPS_SetEventConditionDateRule(12345, 0, 1, 0, -1, 0, 0);
```

IPS_SetEventConditionDayOfTheWeekRule

Require: Symcon >= 5.2

```
bool IPS_SetEventConditionDayOfTheWeekRule (int $EventID, int $ConditionID, int $RuleID, int $Comparison, int $Weekday)
```

PARAMETERS

EventID	ID of the event to be changed
ConditionID	ID of the condition that contains the rule
RuleID	Unique ID for this rule. IDs need to be unique for this condition only.

Comparison

VALUE	CONSTANT	DESCRIPTION
0	EVENTCONDITIONCOMPARISON_EQUAL	==
1	EVENTCONDITIONCOMPARISON_NOTEQUAL	!=
2	EVENTCONDITIONCOMPARISON_GREATER	>
3	EVENTCONDITIONCOMPARISON_GREATEROREQUAL	>=
4	EVENTCONDITIONCOMPARISON_SMALLER	<
5	EVENTCONDITIONCOMPARISON_SMALLEROREQUAL	<=

Weekday

VALUE	DESCRIPTION
0	Delete
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Sunday

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function modifies the rule with the ID **RuleID** of the condition with the ID **ConditionID** of the event with the ID **EventID**. The rule compares the current day of the week with the **Weekday**, according to the comparison operation **Comparison**. If there is no rule with the ID **RuleID**, one will be created.
If the value **Weekday** is 0, the rule with the ID **RuleID** will be deleted.



Which rule ID contains which rule can be looked up via IPS_GetEvent.

EXAMPLE

```
//Add an AND condition
IPS_SetEventCondition(12345, 0, 0, EVENT_CONDITION_OPERATION_AND);

// Adds / modifies the rule with the ID 2 for the event with the object
ID 12345
// It is greater than or equal to Thursday i.e.: Thursday to Sunday
included
IPS_SetEventConditionDayOfTheWeekRule(12345, 0, 2,
EVENTCONDITIONCOMPARISON_GREATEROREQUAL, 4);

// Adds / modifies the rule with the ID 3 for the event with the object
ID 12345
// It is unequal to Sunday
IPS_SetEventConditionDayOfTheWeekRule(12345, 0, 3,
EVENTCONDITIONCOMPARISON_NOTEQUAL, 7);

// Delete the rule with ID 2
IPS_SetEventConditionDayOfTheWeekRule(12345, 0, 2, 0, 0);
```

IPS_SetEventConditionTimeRule

Require: Symcon >= 4.4

```
bool IPS_SetEventConditionTimeRule (int $EventID, int $ConditionID, int $RuleID, int $Comparison, int $Hour, int $Minute, int $Second)
```

PARAMETERS

EventID	ID of the event to be changed
ConditionID	ID of the condition that contains the rule
RuleID	Unique ID for this rule. IDs need to be unique for this condition only.
Comparison	

VALUE	CONSTANT	DESCRIPTION
0	EVENTCONDITIONCOMPARISON_EQUAL	==
1	EVENTCONDITIONCOMPARISON_NOTEQUAL	!=
2	EVENTCONDITIONCOMPARISON_GREATER	>
3	EVENTCONDITIONCOMPARISON_GREATEROREQUAL	>=
4	EVENTCONDITIONCOMPARISON_SMALLER	<
5	EVENTCONDITIONCOMPARISON_SMALLEROREQUAL	<=

Hour	Hour value (0..23) with which the current time is compared
Minute	Minute value (0..59) with which the current time is compared
Second	Second value (0..59) with which the current time is compared

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function modifies the rule with the ID **RuleID** of the condition with the ID **ConditionID** of the event with the ID **EventID**. The rule compares the current time with **Hour**, **Minute** and **Second**, according to the comparison operation **Comparison**. If there is no rule with the ID **RuleID**, one will be created.
If one of the values is outside of the valid range (**Hour** < 0 or > 23, **Minute** < 0 or > 59, **Second** < 0 or > 59), the rule with the ID **RuleID** will be deleted. If no rule with this ID exists, an error is raised.



Which rule ID contains which rule can be looked up via IPS_GetEvent.

EXAMPLE

```
// Create a root condition
IPS_SetEventCondition(12345, 0, 0, EVENT_CONDITION_OPERATION_AND);

// Adds/modifies the rule with the ID 2 for the event with the object
ID 12345
// It has to be exactly 10:30 p.m. for this rule to be met.
IPS_SetEventConditionTimeRule(12345, 0, 2,
EVENTCONDITIONCOMPARISON_EQUAL, 22, 30, 0);

// Delete the rule with ID 1
IPS_SetEventConditionTimeRule(12345, 0, 1, 0, -1, 0, 0);
```

IPS_SetEventConditionVariableDynamicRule

Require: Symcon >= 6.1

```
bool IPS_SetEventConditionVariableDynamicRule (int $EventID, int $ConditionID, int $RuleID, int $VariableID, int $Comparison, int $CompareVariableID)
```

PARAMETERS

EventID	ID of the event to be changed
ConditionID	ID of the condition that contains the rule
RuleID	Unique ID for this rule. IDs only need to be unique for this condition.
VariableID	ID of the checked variable
Comparison	

VALUE	CONSTANT	DESCRIPTION
0	EVENTCONDITIONCOMPARISON_EQUAL	==
1	EVENTCONDITIONCOMPARISON_NOTEQUAL	!=
2	EVENTCONDITIONCOMPARISON_GREATER	>
3	EVENTCONDITIONCOMPARISON_GREATEROREQUAL	>=
4	EVENTCONDITIONCOMPARISON_SMALLER	<
5	EVENTCONDITIONCOMPARISON_SMALLEROREQUAL	<=

CompareVariableID	ID of the variable that the value should be compared to

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function modifies the rule with the ID **RuleID** for the condition with the ID **ConditionID** of the event with the ID **EventID**. The rule compares the variable with the ID **VariableID** to

the value of the variable with the ID **CompareVariableID** according to the comparison operation **Comparison**. If no rule with the ID **RuleID** exists yet, it is created.
If **VariableID** is 0, the rule with the ID **RuleID** is deleted.



The variable types Boolean and String only support the first two comparison operations. This applies to both **VariableID** and **CompareVariableID**.

EXAMPLE

```
// Create root condition for event with ID 12345
IPS_SetEventCondition(12345, 0, 0, EVENT_CONDITION_OPERATION_AND);

// Add a rule with ID 2: Variable 23456 < Variable 34567
IPS_SetEventConditionVariableDynamicRule(12345, 0, 2, 23456,
EVENTCONDITIONCOMPARISON_SMALLER, 34567);

// Delete rule with ID 2
IPS_SetEventConditionVariableDynamicRule(12345, 0, 2, 0, 0, 0);
```

IPS_SetEventConditionVariableRule

Require: Symcon >= 4.4

```
bool IPS_SetEventConditionVariableRule (int $EventID, int $ConditionID,
int $RuleID, int $VariableID, int $Comparison, mixed $Value)
```

PARAMETERS

EventID	ID of the event to be changed
ConditionID	ID of the condition that contains the rule
RuleID	Unique ID for this rule. IDs only need to be unique for this condition.
VariableID	ID of the checked variable
Comparison	

VALUE	CONSTANT	DESCRIPTION
0	EVENTCONDITIONCOMPARISON_EQUAL	==
1	EVENTCONDITIONCOMPARISON_NOTEQUAL	!=
2	EVENTCONDITIONCOMPARISON_GREATER	>
3	EVENTCONDITIONCOMPARISON_GREATEROREQUAL	>=
4	EVENTCONDITIONCOMPARISON_SMALLER	<
5	EVENTCONDITIONCOMPARISON_SMALLEROREQUAL	<=

Value	Value the checked variable is compared to

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function is deprecated. Please use the newer IPS_SetEventConditionVariableStaticRule function, which has the same parameters and a better naming

This function modifies the rule with the ID **RuleID** for the condition with the ID **ConditionID** of the event with the ID **EventID**. The rule compares the variable with the ID **VariableID** to the value **Value** according to the comparison operation **Comparison**. If no rule with the ID **RuleID** exists yet, it is created.
If **VariableID** is 0, the rule with the ID **RuleID** is deleted.

The variable types Boolean and String only support the first two comparison operations.

The value in **Value** must match the type of the variable with the ID **VariableID**.

EXAMPLE

```
// Create root condition for event with ID 12345
IPS_SetEventCondition(12345, 0, 0, EVENT_CONDITION_OPERATION_AND);

// Add a rule with ID 2: Variable 23456 < 75
IPS_SetEventConditionVariableRule(12345, 0, 2, 23456,
EVENTCONDITIONCOMPARISON_SMALLER, 75);

// Delete rule with ID 2
IPS_SetEventConditionVariableRule(12345, 0, 2, 0, 0, 0);
```

IPS_SetEventConditionVariableStaticRule

Require: Symcon >= 6.1

```
bool IPS_SetEventConditionVariableStaticRule (int $EventID, int $ConditionID, int $RuleID, int $VariableID, int $Comparison, mixed $Value)
```

PARAMETERS

EventID	ID of the event to be changed
ConditionID	ID of the condition that contains the rule
RuleID	Unique ID for this rule. IDs only need to be unique for this condition.
VariableID	ID of the checked variable
Comparison	
Value	Value the checked variable is compared to

VALUE	CONSTANT	DESCRIPTION
0	EVENTCONDITIONCOMPARISON_EQUAL	==
1	EVENTCONDITIONCOMPARISON_NOTEQUAL	!=
2	EVENTCONDITIONCOMPARISON_GREATER	>
3	EVENTCONDITIONCOMPARISON_GREATEROREQUAL	>=
4	EVENTCONDITIONCOMPARISON_SMALLER	<
5	EVENTCONDITIONCOMPARISON_SMALLEROREQUAL	<=

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function modifies the rule with the ID **RuleID** for the condition with the ID **ConditionID** of the event with the ID **EventID**. The rule compares the variable with the ID **VariableID** to

the value **Value** according to the comparison operation **Comparison**. If no rule with the ID **RuleID** exists yet, it is created.
If **VariableID** is 0, the rule with the ID **RuleID** is deleted.



The variable types Boolean and String only support the first two comparison operations.



The value in **Value** must match the type of the variable with the ID **VariableID**.

EXAMPLE

```
// Create root condition for event with ID 12345
IPS_SetEventCondition(12345, 0, 0, EVENT_CONDITION_OPERATION_AND);

// Add a rule with ID 2: Variable 23456 < 75
IPS_SetEventConditionVariableStaticRule(12345, 0, 2, 23456,
EVENTCONDITIONCOMPARISON_SMALLER, 75);

// Delete rule with ID 2
IPS_SetEventConditionVariableStaticRule(12345, 0, 2, 0, 0, 0);
```

IPS_SetEventCyclic

```
bool IPS_SetEventCyclic (int $EventID, int $DateType, int
$DateInterval, int $DateDay, int $DateDayInterval, int $TimeType, int
$TimeInterval)
```

PARAMETERS

EventID	ID of the event to be changed

DateType

VALUE	CONSTANT	DESCRIPTION
0	EVENTCYCLICDATETYPE_NONE	No date type. Daily execution.
1	EVENTCYCLICDATETYPE_ONCE	Once. IPS_SetEventCyclicDateFrom for target date
2	EVENTCYCLICDATETYPE_DAY	Daily
3	EVENTCYCLICDATETYPE_WEEK	Weekly
4	EVENTCYCLICDATETYPE_MONTH	Monthly
5	EVENTCYCLICDATETYPE_YEAR	Annually

DateInterval

VALUE	CONSTANT	DESCRIPTION
0	EVENTCYCLICDATETYPE_NONE	0 (No Evaluation)
1	EVENTCYCLICDATETYPE_ONCE	0 (No Evaluation)
2	EVENTCYCLICDATETYPE_DAY	Every X days
3	EVENTCYCLICDATETYPE_WEEK	Every X weeks
4	EVENTCYCLICDATETYPE_MONTH	Every X months
5	EVENTCYCLICDATETYPE_YEAR	1 = January ... 12 = December

VALUE	CONSTANT	DESCRIPTION
0	EVENTCYCLICDATETYPE_NONE	0 (No Evaluation)
1	EVENTCYCLICDATETYPE_ONCE	0 (No Evaluation)
2	EVENTCYCLICDATETYPE_DAY	0 (No Evaluation)
3	EVENTCYCLICDATETYPE_WEEK	see table: Daily values (desired daily values must be added)
4	EVENTCYCLICDATETYPE_MONTH	see table: Daily values, 0 for every **Xth** day of the month
5	EVENTCYCLICDATETYPE_YEAR	0 (No Evaluation)

DateDay

Table: Daily values

DAY	VALUE	CONSTANT
Monday	1	EVENT_DAY_MONDAY
Tuesday	2	EVENT_DAY_TUESDAY
Wednesday	4	EVENT_DAY_WEDNESDAY
Thursday	8	EVENT_DAY_THURSDAY
Friday	16	EVENT_DAY_FRIDAY
Saturday	32	EVENT_DAY_SATURDAY
Sunday	64	EVENT_DAY_SUNDAY
All days	127	EVENT_DAY_ALL

DateDayInterval

VALUE	CONSTANT	DESCRIPTION
0	`EVENTCYCLICDATETYPE_NONE`	0 (No Evaluation)
1	`EVENTCYCLICDATETYPE_ONCE`	0 (No Evaluation)
2	`EVENTCYCLICDATETYPE_DAY`	0 (No Evaluation)
3	`EVENTCYCLICDATETYPE_WEEK`	0 (No Evaluation)
4	`EVENTCYCLICDATETYPE_MONTH`	Every **Xth DateDay** of the month or every **Xth** day of the month if **DateDay** = 0
5	`EVENTCYCLICDATETYPE_YEAR`	Every **Xth** day of the specified month

TimeType

VALUE	CONSTANT	DESCRIPTION
0	`EVENTCYCLICTIMETYPE_ONCE`	Once. IPS_SetEventCyclicTimeFrom for target time
1	`EVENTCYCLICTIMETYPE_SECOND`	every second
2	`EVENTCYCLICTIMETYPE_MINUTE`	every minute
3	`EVENTCYCLICTIMETYPE_HOUR`	hourly

TimeInterval

VALUE	CONSTANT	DESCRIPTION
0	`EVENTCYCLICTIMETYPE_ONCE`	0 (No Evaluation)
1	`EVENTCYCLICTIMETYPE_SECOND`	Every **X** seconds
2	`EVENTCYCLICTIMETYPE_MINUTE`	Every **X** minutes
3	`EVENTCYCLICTIMETYPE_HOUR`	Every **X** hours

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The configuration of a cyclic event determines in which intervals the event should run. This includes a **DateInterval** that specifies the interval in a day, week, month, or year rhythm and a time interval that defines second, minute, or hour-based intervals for each day.
If the **DateType** "Weekly" is selected, the days of the week on which the event should be started can be specified with the parameter **DateDay**. For a monthly or annual **DateType**, the day of the month is specified with the parameter **DateDayInterval**. For all other types the parameter is not used and can simply be set to the placeholder 0.



Remarks:

(Monthly) In the variant "On the Xth day every Y months", choosing X > 31 signals that the event is run on the last day of every month. When choosing X = 31, the event only triggers in months with 31 days.
(Monthly) In the variant "On the Xth weekday every Y months", choosing X > 5 signals that the event is run on the last defined weekday of every month. Choosing X = 5 can lead to an event not being run, as, e.g., not every month includes a 5th Friday.
(Annually) In the variant "The Xth day of a specific month", a timer is executed only if X is a valid day in the month, otherwise the timer is not running.

EXAMPLE

```
//Execute something every 8 weeks at the current point in time
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclicTimeFrom($eid, (int)date("H"), (int)date("i"),
(int)date("s"));
IPS_SetEventCyclicDateFrom($eid, (int)date("d"), (int)date("m"),
(int)date("Y"));
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_DAY, 8 * 7 /* Every 8
weeks */, 0, 0, EVENTCYCLICTIMETYPE_ONCE, 0);
IPS_SetEventActive($eid, true);

//Every day, every 6 hours starting at 0:00
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_DAY, 1 /* Every day */, 0,
0, EVENTCYCLICTIMETYPE_HOUR, 6 /* Every 6 hours */);

//Every 2 minutes from 0:00 until 12:00
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_NONE /* No date check */,
0, 0, 0, EVENTCYCLICTIMETYPE_MINUTE, 2 /* Every 2 minutes */);
IPS_SetEventCyclicTimeTo($eid, 12, 0, 0);

//Every 2 minutes between 8:00 and 10:00
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_NONE /* No date check */,
0, 0, 0, EVENTCYCLICTIMETYPE_MINUTE, 2 /* Every 2 minutes */);
IPS_SetEventCyclicTimeFrom($eid, 8, 0, 0);
IPS_SetEventCyclicTimeTo($eid, 10, 0, 0);

//Every 2 weeks on Monday+Wednesday, once at exactly 0:00
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
```

```
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_WEEK, 2 /* Every 2 weeks
*/, EVENT_DAY_MONDAY + EVENT_DAY_WEDNESDAY, 0,
EVENTCYCLICTIMETYPE_ONCE, 0);

//Daily at 15:00
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclicTimeFrom($eid, 15, 0, 0);

//On the first day of the month at 0:00
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_MONTH, 1 /* Every month
*/, 0 /* Any day */, 1 /* First day of the month */,
EVENTCYCLICTIMETYPE_ONCE, 0);

//On the first Friday of the month at 0:00
$eid = IPS_CreateEvent(EVENTTYPE_CYCLIC);
IPS_SetParent($eid, $_IPS['SELF']);
IPS_SetEventCyclic($eid, EVENTCYCLICDATETYPE_MONTH, 1 /* Every month
*/, EVENT_DAY_FRIDAY, 1 /* First Friday of the month */,
EVENTCYCLICTIMETYPE_ONCE, 0);
```

IPS_SetEventCyclicDateFrom

Require: Symcon >= 3.1

```
bool IPS_SetEventCyclicDateFrom (int $EventID, int $Day, int $Month,
int $Year)
```

PARAMETERS

EventID	ID of the cyclic event to be changed
Day	Start day (0, 1..31)
Month	Start month (0, 1..12)
Year	Start year (0, 1970..2038)

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function sets the start date for a cyclic event.
If the arguments for day, month, and year are 0, the event does not have a specific start date and starts immediately.



The start date is especially relevant for events that are not executed daily. For example, when an event is run every 2 weeks, the start date determines the reference point of the first execution. It can be used to control if the event should run during odd or even calendar weeks.

EXAMPLE

```
//No start date for the event
IPS_SetEventCyclicDateFrom($EventID, 0, 0, 0);
```

IPS_SetEventCyclicDateTo

Require: Symcon >= 3.1

```
bool IPS_SetEventCyclicDateTo (int $EventID, int $Day, int $Month, int $Year)
```

PARAMETERS

EventID	ID of the cyclic event to be changed
Day	End day (0, 1..31)
Month	End month (0, 1..12)
Year	End year (0, 1970..2038)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the end date for a cyclic event.
If the arguments for day, month, and year are 0, the event does not have a specific end date and is run indefinitely.



The event is run until the end date, including the end date itself

EXAMPLE

```
//Set February 1st, 2020 as end date for the event
IPS_SetEventCyclicDateTo($EventID, 1, 2, 2020);
```

IPS_SetEventCyclicTimeFrom

Require: Symcon >= 3.1

```
bool IPS_SetEventCyclicTimeFrom (int $EventID, int $Hour, int $Minute,
int $Second)
```

PARAMETERS

EventID	ID of the cyclic event to be changed
Hour	Start hour (0..23)
Minute	Start minute (0..59)
Second	Start second (0..59)

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function defines the start time of a cyclic event. The start time defines the first execution of the event.



The start time is especially relevant when an event should run slightly shifted. This is useful to prevent all events from running at the same time at the full minute. For example, when the start time 00:00:05 is set and the event runs every minute, the script is always run five seconds after a full minute.

EXAMPLE

```
//Set 7:15:00 as start time
IPS_SetEventCyclicTimeFrom($EventID, 7, 15, 0);
```

IPS_SetEventCyclicTimeTo

Require: Symcon >= 3.1

```
bool IPS_SetEventCyclicTimeTo (int $EventID, int $Hour, int $Minute,
int $Second)
```

PARAMETERS

EventID	ID of the cyclic event to be changed
Hour	End hour (0..23)
Minute	End minute (0..59)
Second	End second (0..59)

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function defines the end time for a cyclic event. The end time describes the last time of execution.
If the arguments for hour, minute, and second are 0, no end time is set and the last execution is at 23:59:59 at latest.

EXAMPLE

```
//No end time for the event
IPS_SetEventCyclicTimeTo($EventID, 0, 0, 0);
```

IPS_SetEventLimit

```
bool IPS_SetEventLimit (int $EventID, int $Number)
```

PARAMETERS

EventID	ID of the event to be changed
Number	The number of executions before the event is deactivated. 0 = No Limit

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function limits the number of executions of the event with the ID **EventID**. The limitation to **Number** causes the event to run only **Number** times. After running **Number** times, the event is deactivated. Setting **Number** = 0 removes the limitation.

EXAMPLE

```
IPS_SetEventLimit($EventID, 0); //No limitation
```

IPS_SetEventScheduleAction

Require: Symcon >= 3.2

```
bool IPS_SetEventScheduleAction (int $EventID, int $ScheduleActionID,
string $Name, int $Color, string $ScriptContent)
```

PARAMETERS

EventID	ID of the event to be changed
ScheduleActionID	Unique ID for this schedule action. The ID is used for sorting. IDs only need to be unique for this event. Must be greater than 0.
Name	Name of the given schedule action
Color	Color value as HTML color code, e.g., 0x0000FF for blue
ScriptContent	PHP code without PHP tags (<? … ?>)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the **Name**, the **Color**, and the executed Action to "Execute PHP Code" with the code **ScriptContent** for the schedule action with the ID **ScheduleActionID** of a schedule event with the ID **EventID**. The system variable $_IPS['ACTION'] is defined. System Variables can be used during the execution of the script.
If the argument **Name** is empty, the schedule action with the given **ScheduleActionID** is deleted. This is only possible if no switch point references that **ScheduleActionID**.



The **ScheduleActionID** must be greater than 0, otherwise an error is raised. In IPS_GetEvent, the ID 0 in **LastActionID/NextActionID** means "no action". Existing schedule actions with an ID less than or equal to 0 from older installations can still be deleted in order to recreate them with an ID of 1 or higher.

EXAMPLE

```
//Creation of actions
IPS_SetEventScheduleAction($EventID, 1, "Warm", 0xFF0000,
"FHT_SetTemperature(\$_IPS['TARGET'], 22.5);");
IPS_SetEventScheduleAction($EventID, 2, "Cold", 0x0000FF,
"FHT_SetTemperature(\$_IPS['TARGET'], 17);");
IPS_SetEventScheduleAction($EventID, 3, "Eco", 0x00FF00,
"FHT_SetTemperature(\$_IPS['TARGET'], 20);");

//Delete action with ID 3
IPS_SetEventScheduleAction($EventID, 3, "", 0, "");
```

IPS_SetEventScheduleActionEx

Require: Symcon >= 6.0

```
bool IPS_SetEventScheduleActionEx (int $EventID, int $ScheduleActionID,
string $Name, int $Color, string $ActionID, array $ActionParameters)
```

PARAMETERS

EventID	ID of the event to be changed
ScheduleActionID	Unique ID for this schedule action. The ID is used for sorting. IDs only need to be unique for this event. Must be greater than 0.
Name	Name of the given schedule action
Color	Color value as HTML color code, e.g., 0x0000FF for blue
ActionID	ID of the executed action
ActionParameters	List of Parameters for the executed action

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the **Name**, the **Color**, and the executed Action with the ID **ActionID** with the parameters **ActionParameters** for the schedule action with the ID **ScheduleActionID** of a schedule event with the ID **EventID**. If the action executes a PHP Script, the system variable $_IPS['ACTION'] is defined. System Variables can be used during the execution of the script. If the argument **Name** is empty, the schedule action with the given **ScheduleActionID** is deleted. This is only possible if no switch point references that **ScheduleActionID**.



The special parameters TARGET, ENVIRONMENT, and PARENT are not required here and are handled by the event



The **ScheduleActionID** must be greater than 0, otherwise an error is raised. In IPS_GetEvent, the ID 0 in **LastActionID/NextActionID** means "no action". Existing schedule actions with an ID less than or equal to 0 from older installations can still be deleted in order to recreate them with an ID of 1 or higher.

EXAMPLE

```
//Creating schedule actions
IPS_SetEventScheduleActionEx($EventID, 1, "Warm", 0xFF0000, "{3644F802-
C152-464A-868A-242C2A3DEC5C}", ["VALUE" => 22.5]);
IPS_SetEventScheduleActionEx($EventID, 2, "Cold", 0x0000FF, "{3644F802-
C152-464A-868A-242C2A3DEC5C}", ["VALUE" => 17]);
IPS_SetEventScheduleActionEx($EventID, 3, "Eco", 0x00FF00, "{3644F802-
C152-464A-868A-242C2A3DEC5C}", ["VALUE" => 20]);

//Delete schedule action with the ID 3
IPS_SetEventScheduleActionEx($EventID, 3, "", 0, "", []);
```

IPS_SetEventScheduleGroup

Require: Symcon >= 3.2

```
bool IPS_SetEventScheduleGroup (int $EventID, int $GroupID, int $Days)
```

PARAMETERS

EventID	ID of the event to be changed
GroupID	Unique ID for this group. The ID is used for sorting. IDs only need to be unique for this event.
Days	Sum of the day values that belong to the group. The day values are found in the table in IPS_SetEventCyclic.

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the bit mask **Days** for the **GroupID** of a schedule event with the ID **EventID**. Any week days can be combined in a group. However, a specific day can only be contained in one group. It is possible to put all days (**Days** = 127, EVENT_DAY_ALL) or only a single day into one group, e.g., 1 (EVENT_DAY_MONDAY) for Monday. If not all days are separated into groups, it is not possible to create switch points for the missing days. In this case, the last executed switch point remains in place.

DAY	VALUE	CONSTANT
Monday	1	EVENT_DAY_MONDAY
Tuesday	2	EVENT_DAY_TUESDAY
Wednesday	4	EVENT_DAY_WEDNESDAY
Thursday	8	EVENT_DAY_THURSDAY
Friday	16	EVENT_DAY_FRIDAY
Saturday	32	EVENT_DAY_SATURDAY
Sunday	64	EVENT_DAY_SUNDAY
All days	127	EVENT_DAY_ALL

If the value of **Days** is 0, the group with the given **GroupID** is deleted. This also removes all switch points of that group.

EXAMPLE

```
//Create schedule event
$EventID = IPS_CreateEvent(EVENTTYPE_SCHEDULE);

//Create groups
IPS_SetEventScheduleGroup($EventID, 0, EVENT_DAY_MONDAY +
EVENT_DAY_TUESDAY + EVENT_DAY_WEDNESDAY + EVENT_DAY_THURSDAY +
EVENT_DAY_FRIDAY); //Mo - Fr (31)
IPS_SetEventScheduleGroup($EventID, 1, EVENT_DAY_SATURDAY +
EVENT_DAY_SUNDAY); //Sa + Su (96)

//Delete group with ID 1
IPS_SetEventScheduleGroup($EventID, 1, 0);
```

IPS_SetEventScheduleGroupPoint

Require: Symcon >= 3.2

```
bool IPS_SetEventScheduleGroupPoint (int $EventID, int $GroupID, int $PointID, int $StartHour, int $StartMinute, int $StartSecond, int $ActionID)
```

PARAMETERS

EventID	ID of the event to be changed
GroupID	ID of the group the switch point belongs to. The group must already exist.
PointID	Unique ID for this switch point. IDs only need to be unique for this group.
StartHour	Start hour (0..23)
StartMinute	Start minute (0..59)
StartSecond	Start second (0..59)
ActionID	ID of the schedule action which should be executed at the start time. The ID needs to be valid.

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the switch point with the start time according to **StartHour**, **StartMinute**, and **StartSecond** and the ID **PointID** to the group with the ID **GroupID** of a schedule event with the ID **EventID**.
If **StartHour**, **StartMinute**, or **StartSecond** are outside of the valid range (**StartHour** < 0 or > 23, **StartMinute** < 0 or > 59, **StartSecond** < 0 or > 59, e.g., -1), the switch point with the given **PointID** is deleted. If no switch point with this ID exists, an error is raised.



Within a group, only a single switch point may have the same combination of **StartHour**, **StartMinute**, and **StartSecond**.

EXAMPLE

```
//Create switch points for the group with ID 0 (= Mo - Fr)
IPS_SetEventScheduleGroupPoint($EventID, 0, 0, 8, 0, 0, 1); //Call
action with ID 1 (=Warm) at 8:00:00
IPS_SetEventScheduleGroupPoint($EventID, 0, 1, 22, 30, 0, 3); //Call
action with ID 3 (=Eco) at 22:30:00

//Delete switch point with ID 1 from group with ID 0 (= Mo - Fr)
IPS_SetEventScheduleGroupPoint($EventID, 0, 1, -1, -1, -1, 1);
```

IPS_SetEventScript

```
bool IPS_SetEventScript (int $EventID, string $EventScript)
```

PARAMETERS

EventID	ID of the event to be changed
EventScript	PHP code without PHP tags (<? ... ?>)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The executed Action of the Event is set to "Execute PHP Code" and the code to **EventScript**.
The PHP code will be given directly without the PHP tags (<? ...?>).
The list of the System Variables shows what specific variables are available.
If **EventScript** is empty and the parent object is a script, the action "Run Automation" is set instead, which executes the parent script.

EXAMPLE

```
//Assign the script content to the event
IPS_SetEventScript($EventID, "echo 'Linked object: ' .
\$_IPS['TARGET'];");
```

IPS_SetEventTrigger

```
bool IPS_SetEventTrigger (int $EventID, int $TriggerType, int
$TriggerVariableID)
```

PARAMETERS

EventID	ID of the event to be changed
TriggerType	
TriggerVariableID	VariableID whose change or update should be responded to

VALUE	CONSTANT	DESCRIPTION
0	EVENTTRIGGERTYPE_ONUPDATE	On Variable Update
1	EVENTTRIGGERTYPE_ONCHANGE	On Variable Change
2	EVENTTRIGGERTYPE_ONLIMITEXCEED	On Limit Exceed - Value is determined by IPS_SetEventTriggerValue
3	EVENTTRIGGERTYPE_ONLIMITDROP	On Limit Drop - Value is determined by IPS_SetEventTriggerValue
4	EVENTTRIGGERTYPE_ONVALUE	On specific Value - Value is determined by IPS_SetEventTriggerValue

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function configures a triggering event. It defines the observed variable **TriggerVariableID** and which **TriggerType** the event should react to.

EXAMPLE

```
IPS_SetEventTrigger($EventID, EVENTTRIGGERTYPE_ONUPDATE, 12345); // On
update of variable 12345
```

IPS_SetEventTriggerSubsequentExecution

```
bool IPS_SetEventTriggerSubsequentExecution (int $EventID, bool
$AllowSubsequentExecutions)
```

PARAMETERS

EventID	ID of the event to be changed
AllowSubsequentExecutions	TRUE if permitted, otherwise FALSE

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function defines whether the event should be run again if it is triggered for the second time on a limit exceed/limit drop. Otherwise, the event is only executed again if the variable value was within the limit range and exceeds it again. For the trigger type "On specific Value", the event is otherwise only executed if the variable value changed to that value.

EXAMPLE

```
IPS_SetEventTriggerSubsequentExecution($EventID, true); //Allow
```

IPS_SetEventTriggerValue

```
bool IPS_SetEventTriggerValue (int $EventID, mixed $TriggerValue)
```

PARAMETERS

EventID	ID of the event to be changed
TriggerValue	Value/ type, depending on TriggerVariableID

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Indicates the value/threshold at which the event should be run.
The **TriggerType** of the event must be set to:
On limit exceed
On limit drop
On specific value



The limit must match the type of the trigger variable (for float variables, an integer is accepted as well). Otherwise the function fails. Therefore, the trigger variable must be set via IPS_SetEventTrigger beforehand.

EXAMPLE

```
IPS_SetEventTriggerValue($EventID, true); //Trigger only for TRUE
values
```

Management of Instances

DEBUG

IPS_DisableDebug	deactivates the forwarding of debug messages
IPS_DisableDebugFile	deactivates the writing of debug logs to a file
IPS_EnableDebug	activates the forwarding of debug messages for a specific duration
IPS_EnableDebugFile	activates the writing of debug logs to a file
IPS_SendDebug	sends a message to the debug output of an object

IPS_CREATEINSTANCE

creates an instance

IPS_DELETEINSTANCE

deletes an instance

IPS_GETINSTANCE

returns extensive information about a specific instance

IPS_GETINSTANCEIDBYNAME

returns the ID of an instance by its name

IPS_GETINSTANCELIST

returns a list of all existing instances

IPS_GETINSTANCELISTBYMODULEID

returns all instances with a specific module ID

IPS_GETINSTANCELISTBYMODULETYPE

returns a list of all instances with a specific type

IPS_INSTANCEEXISTS

checks if a specific instance exists

IPS_REQUESTACTION

calls the RequestAction method of an instance by ident

IPS_TRANSLATE

translates a text with the translations of the module of an instance

CONFIGURATION

IPS_ApplyChanges	applies a changed configuration
IPS_GetConfiguration	reads a configuration
IPS_GetConfigurationForm	reads a configuration form
IPS_GetConfigurationForParent	returns the configuration which an instance specifies for its parent instance
IPS_GetProperty	reads the current value of a property
IPS_GetVisualizationTile	reads the custom presentation of an instance
IPS_HasChanges	checks if the configuration was changed
IPS_ResetChanges	resets the modified configuration to the current configuration
IPS_SetConfiguration	sets a new configuration
IPS_SetProperty	sets a planned value for a property

REFERENCES

IPS_GetReferenceList	returns the IDs of all references

SEARCH

IPS_IsSearching	checks whether an instance is in search mode
IPS_StartSearch	starts the search mode of an instance
IPS_StopSearch	stops the search mode of an instance
IPS_SupportsSearching	checks whether an instance supports the search mode

TIMERS

IPS_GetTimer	returns information about a timer
IPS_GetTimerList	returns the IDs of all timers
IPS_TimerExists	checks whether a specific timer exists

CONNECTION

IPS_ConnectInstance	creates a (data) connection between two instances
IPS_DisconnectInstance	disconnects a (data) connection between two instances
IPS_GetCompatibleInstances	returns compatible instances to an instance
IPS_GetCompatibleParents	returns the description of the compatible parent instances
IPS_IsInstanceCompatible	checks if a connection between two instances is possible

Debug

IPS_DisableDebug	deactivates the forwarding of debug messages
IPS_DisableDebugFile	deactivates the writing of debug logs to a file
IPS_EnableDebug	activates the forwarding of debug messages for a specific duration
IPS_EnableDebugFile	activates the writing of debug logs to a file
IPS_SendDebug	sends a message to the debug output of an object

IPS_DisableDebug

```
bool IPS_DisableDebug (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function ends the forwarding of debug messages of the instance with the ID **InstanceID** activated with IPS_EnableDebug before the duration has expired. If the forwarding is not active, the function has no effect. Writing to a file via IPS_EnableDebugFile is not affected by this, use IPS_DisableDebugFile for that.
If **InstanceID** is an object mirrored via Sync Remote, the function reports the error "Remote instances cannot be debugged". Debug messages of mirrored instances can only be displayed directly on the remote system.

EXAMPLE

```
// End the forwarding of debug messages of instance 12345
IPS_DisableDebug(12345);
```

IPS_DisableDebugFile

Require: Symcon >= 5.2

```
bool IPS_DisableDebugFile (int $InstanceID)
```

PARAMETERS

InstanceID	Instance ID

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deactivates the writing of debug logs to a file for the instance with the ID **InstanceID**, after it was previously activated via IPS_EnableDebugFile.
If **InstanceID** is an object mirrored via Sync Remote, the function reports the error "Remote instances cannot be debugged". Debug messages of mirrored instances can only be displayed directly on the remote system.

EXAMPLE

```
// Deactivates the writing of debug logs for instance 12345
IPS_DisableDebugFile(12345);
```

IPS_EnableDebug

```
bool IPS_EnableDebug (int $InstanceID, int $Duration)
```

PARAMETERS

InstanceID	ID of the instance
Duration	Duration in seconds for which the debug messages are forwarded

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function activates the forwarding of debug messages for the instance with the ID **InstanceID** for **Duration** seconds. Debug messages which the instance sends during this time (e.g. via SendDebug) are distributed as message KL_DEBUG (10206). The management console uses the function for the display in the debug window of the instance.
Calling the function again restarts the duration from the current point in time. After the duration has expired, debug messages are no longer forwarded. The forwarding can be ended early with IPS_DisableDebug. Independently of this, debug messages can be written to a file with IPS_EnableDebugFile.
If **InstanceID** is an object mirrored via Sync Remote, the function reports the error "Remote instances cannot be debugged". Debug messages of mirrored instances can only be displayed directly on the remote system.



This option is temporary. After restarting the IP-Symcon service, the forwarding of debug messages is automatically deactivated again.

EXAMPLE

```
// Activate debug messages of instance 12345 for 5 minutes
IPS_EnableDebug(12345, 300);
```

IPS_EnableDebugFile

Require: Symcon >= 5.2

```
bool IPS_EnableDebugFile (int $InstanceID)
```

PARAMETERS

InstanceID	Instance ID

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function activates the writing of debug logs to a file for the instance with the ID **InstanceID**.
The debug file is located in the logs folder (e.g. /var/log/symcon/debug_<InstanceID>.log).
If **InstanceID** is an object mirrored via Sync Remote, the function reports the error "Remote instances cannot be debugged". Debug messages of mirrored instances can only be displayed directly on the remote system.



This option is temporary. After restarting the IP-Symcon service, debug logging is automatically deactivated again.

EXAMPLE

```
// Activates the writing of debug logs for instance 12345
IPS_EnableDebugFile(12345);

// Filename: debug_12345.log
```

IPS_SendDebug

```
bool IPS_SendDebug (int $ObjectID, string $MessageName, string $Data,
int $Format)
```

PARAMETERS

ObjectID	ID of the object in whose debug output the message should appear
MessageName	Name/title of the debug message
Data	Content of the debug message
Format	Default for the display (0 = DEBUG_FORMAT_TEXT, 1 = DEBUG_FORMAT_BINARY)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sends the debug message **Data** with the name **MessageName** to the debug output of the object with the ID **ObjectID**. This allows, for example, scripts to show their own messages in the debug window of an instance. **Format** specifies whether the data is displayed as text or as hex with automatic display:

VALUE	CONSTANT	DISPLAY
0	DEBUG_FORMAT_TEXT	Text
1	DEBUG_FORMAT_BINARY	Binary/Hex

The message is only passed on if the forwarding of debug messages is currently active for the object (e.g. because the debug window is open in the management console or IPS_EnableDebug was called) or writing to a file is activated via IPS_EnableDebugFile. Otherwise the message is discarded. In the debug file, the data is always written both as text and as hex, regardless of **Format**.
Within a PHP module, the module method SendDebug is used instead, which automatically sends the message to the debug output of its own instance. IPS_SendDebug, on the other hand, can send a message to any object from any script.
If **ObjectID** is an object mirrored via Sync Remote, the function reports the error "Remote instances cannot be debugged". Debug messages of mirrored instances can only be displayed directly on the remote system.

EXAMPLE

```
// Output a text message in the debug output of instance 12345
IPS_SendDebug(12345, "Script", "Value was updated", DEBUG_FORMAT_TEXT);

// Binary data which is displayed as hex in the debug window
IPS_SendDebug(12345, "Packet", "\x01\x02\xFF", DEBUG_FORMAT_BINARY);
```

IPS_CreateInstance

```
int IPS_CreateInstance (string $ModuleID)
```

PARAMETERS

ModuleID	ModuleID of the object to create

RETURNS

ID of the newly created instance

DESCRIPTION

The function creates an unconfigured instance of the module ID **ModuleID**. Valid values for **ModuleID** can be determined by the function IPS_GetModuleList. The ModuleID is a GUID in the format {00000000-0000-0000-0000-000000000000}.
The function returns an ID which identifies the created instance. After using the function, the created object is still unconfigured and may need to be configured with the function IPS_SetProperty. The configuration needs to be applied with the function IPS_ApplyChanges. Otherwise, the new configuration will not affect the instance.

EXAMPLE

```
//Create FS20 instance with the name "Standing lamp"
$InsID = IPS_CreateInstance("{48FCFDC1-11A5-4309-BB0B-A0DB8042A969}");

IPS_SetName($InsID, "Standing lamp"); // Name instance
IPS_SetParent($InsID, 12345); // Sort instance under the object with ID
"12345"

//Configuration
IPS_SetProperty($InsID, "HomeCode", "12345678"); //Change property
"HomeCode"
IPS_ApplyChanges($InsID); //Apply changes -> The instance uses the
changed HomeCode
```

IPS_DeleteInstance

```
bool IPS_DeleteInstance (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance to delete

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes an existing instance with the ID **InstanceID**.



An instance can only be deleted if it has no child objects.

EXAMPLE

```
// Delete the Instance 47788
IPS_DeleteInstance(47788);
```

IPS_GetInstance

```
array IPS_GetInstance (int $InstanceID)
```

PARAMETERS

InstanceID	InstanceID to be returned

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
InstanceID	integer	InstanceID
ConnectionID	integer	ID of the connected instance
InstanceStatus	integer	Status code of the instance
InstanceSupportsSearching	boolean	Does the instance support the search mode
InstanceIsSearching	boolean	Instance is in search mode
InstanceVisualizationType	integer	Visualization type of the instance 0 INSTANCE_VISUALIZATION_TYPE_NONE None 1 INSTANCE_VISUALIZATION_TYPE_HTML HTML 2 INSTANCE_VISUALIZATION_TYPE_HTML_FULLSCREEN HTML fullscreen 3 INSTANCE_VISUALIZATION_TYPE_FORM Form
InstanceChanged	integer	Unix timestamp of the last time the configuration was applied
ModuleInfo	array	Module Information

STATUS OF INSTANCE

CODE	CONSTANT	STATUS
101	`IS_CREATING`	Instance will be created
102	`IS_ACTIVE`	Instance is active
103	`IS_DELETING`	Instance will be deleted
104	`IS_INACTIVE`	Instance is inactive
105	`IS_NOTCREATED`	Instance was not created
106	`IS_STANDBY`	Instance is in standby
>=200	`IS_EBASE`	Instance is faulty

MODULE INFORMATION

INDEX	TYPE	DESCRIPTION	
ModuleID	string	ModuleID of Instance	
ModuleName	string	ModuleName of Instance	
ModuleType	integer	ModuleType of Instance	
		0 `MODULETYPE_CORE`	Core
		1 `MODULETYPE_IO`	I/O
		2 `MODULETYPE_SPLITTER`	Splitter
		3 `MODULETYPE_DEVICE`	Device
		4 `MODULETYPE_CONFIGURATOR`	Configurator
		5 `MODULETYPE_DISCOVERY`	Discovery
		6 `MODULETYPE_VISUALIZATION`	Visualization

DESCRIPTION

The function returns an array containing information about the instance with the ID **InstanceID**. If the instance does not exist, a warning is generated.

EXAMPLE

```
print_r(IPS_GetInstance(19668));

/* returns e.g.:
Array
(
[InstanceID] => 19668
[ConnectionID] => 29416
[InstanceStatus] => 102
[InstanceSupportsSearching] =>
[InstanceIsSearching] =>
[InstanceVisualizationType] => 0
[InstanceChanged] => 1735686000
```

```
[ModuleInfo] => Array
(
[ModuleID] => {48FCFDC1-11A5-4309-BB0B-A0DB8042A969}
[ModuleName] => FS20
[ModuleType] => 3
)
)
*/
```

IPS_GetInstanceIDByName

```
int IPS_GetInstanceIDByName (string $InstanceName, int $ParentID)
```

PARAMETERS

InstanceName	Name of the searched instance
ParentID	Object whose child objects are searched

RETURNS

ID of the found instance, otherwise **FALSE**

DESCRIPTION

The function tries to determine the ID of the instance named **InstanceName**, whose parent has the ID **ParentID**. The ID of the first instance found whose name matches **InstanceName** is used. If such an instance is found, its ID is returned, otherwise **FALSE**.



Caution: Names in IP-Symcon can be chosen freely, so they are **not unique**. For this reason, the function can return a wrong ID.



This function generates a **warning** if the name was not found. Use the @ operator if you want to suppress this message. The exact behavior of this operator is described in the PHP manual.

EXAMPLE

```
$InstanceID = @IPS_GetInstanceIDByName("Rainfall", $ParentID);
if ($InstanceID === false)
echo "Instance not found!";
else
echo "The Instance ID is: ". $InstanceID;
```

IPS_GetInstanceList

```
array IPS_GetInstanceList ()
```

RETURNS

An array of integer values of all IDs of the instances in IP-Symcon. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all registered IPS instances in IP-Symcon. The IDs are listed in an array. If no instance exists, the array is empty.

EXAMPLE

```
$allInstances = IPS_GetInstanceList();
print_r($allInstances);

/* returns e.g.:
Array
(
[0] => 37659
[1] => 18326
etc. ...
*/
```

IPS_GetInstanceListByModuleID

```
array IPS_GetInstanceListByModuleID (string $ModuleID)
```

PARAMETERS

ModuleID	ModuleID of the instances to be returned

RETURNS

An array of integer values of all found IDs. The order of the entries is not guaranteed.

DESCRIPTION

The function returns all instances that match the specified **ModuleID** in an array.



The GUIDs can be looked up using IPS_GetModuleList.

EXAMPLE

```
//FS20 Transmitter
$guid = "{48FCFDC1-11A5-4309-BB0B-A0DB8042A969}";
//List
print_r(IPS_GetInstanceListByModuleID($guid));
```

IPS_GetInstanceListByModuleType

```
array IPS_GetInstanceListByModuleType (int $ModuleType)
```

PARAMETERS

ModuleType

VALUE	CONSTANT	DESCRIPTION
0	MODULETYPE_CORE	Core
1	MODULETYPE_IO	I/O
2	MODULETYPE_SPLITTER	Splitter
3	MODULETYPE_DEVICE	Device
4	MODULETYPE_CONFIGURATOR	Configurator
5	MODULETYPE_DISCOVERY	Discovery
6	MODULETYPE_VISUALIZATION	Visualization

RETURNS

An array of integer values of all IDs of the instances of the type **ModuleType** in IP-Symcon.
The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all instances of the given **ModuleType**. The IDs are listed in an array. If there is no instance of the requested type, the array is empty.
The list only contains objects of the type **ModuleType**. The function IPS_GetInstanceList can be used to list all existing instances.

EXAMPLE

```
$allInstances = IPS_GetInstanceListByModuleType(MODULETYPE_IO); // list
only I/O instances
print_r($allInstances);

/* returns e.g.:
Array
(
[0] => 37659
```

```
[1] => 18326
etc. ...
*/
```

IPS_InstanceExists

```
bool IPS_InstanceExists (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance to be checked

RETURNS

If the InstanceID exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the instance with the ID **InstanceID** exists.

EXAMPLE

```
if (IPS_InstanceExists(45724))
echo "Instance already exists!";
```

IPS_RequestAction

```
bool IPS_RequestAction (int $InstanceID, string $VariableIdent, mixed
$Value)
```

PARAMETERS

InstanceID	ID of the instance
VariableIdent	Ident of the status variable
Value	Value for the action

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



To switch a variable, RequestAction with the ID of the variable is recommended. RequestAction determines the responsible action by itself: it takes custom actions (action scripts) into account and also works for variables whose action is not provided by the parent instance. IPS_RequestAction bypasses both. The exception are PHP modules: there, IPS_RequestAction calls the module's own RequestAction method, e.g. from the configuration form (onClick/onChange), with an ident that does not have to be a variable.

The function directly calls the RequestAction function of the module of the instance with the ID **InstanceID** and passes **VariableIdent** and **Value**. A custom action of the variable is not executed.
For built-in modules, an object with the ident **VariableIdent** must exist below the instance, otherwise the function fails with "Object with ident ... could not be found". If the module does not support an action for this variable, it fails with "No action for VariableIdent '...' found".
For PHP modules, **VariableIdent** is passed to the RequestAction method of the module without any check; whether a variable with this ident exists is up to the module. Errors raised by the module are returned as errors of the function. While Symcon is starting up or shutting down, the method is not called.

EXAMPLE

```
// Trigger the action via the instance and the ident
IPS_RequestAction(12345, "Status", true);

// Recommended: trigger the action via the ID of the variable
RequestAction(IPS_GetObjectIDByIdent("Status", 12345), true);
```

IPS_Translate

```
string IPS_Translate (int $InstanceID, string $Text)
```

PARAMETERS

InstanceID	ID of the instance whose module provides the translations
Text	Text to translate (usually the original English text)

RETURNS

Translated text, or **Text** unchanged if no translation exists

DESCRIPTION

The function translates **Text** with the translations of the module of the instance with the ID **InstanceID** into the system language (IPS_GetSystemLanguage). It corresponds to calling Translate inside a module, but can be used from outside, e.g. to output texts of a module in a script in the same language.
It looks for an entry whose key matches **Text** exactly. For PHP modules, the translations come from the locale.json of the module, for built-in modules from the translations shipped with the module. For the system language, the full locale (e.g. de_AT) is searched first, then the language code used as country code (de_DE) and finally only the language code (de).
If there is no locale.json, no translation for the system language or no matching entry, **Text** is returned unchanged. If the locale.json of a PHP module is not valid JSON, the function fails with "Cannot parse locale.json". Placeholders like %s are not replaced; this can be done after the translation, e.g. with sprintf.

EXAMPLE

```
echo IPS_Translate(12345, "Users");
// e.g. returns: Benutzer (on a German system)
// (without a matching translation, "Users" is returned unchanged)

printf(IPS_Translate(12345, "The current time is %s"), date("H:i"));
```

Configuration

IPS_ApplyChanges	applies a changed configuration
IPS_GetConfiguration	reads a configuration
IPS_GetConfigurationForm	reads a configuration form
IPS_GetConfigurationForParent	returns the configuration which an instance specifies for its parent instance
IPS_GetProperty	reads the current value of a property
IPS_GetVisualizationTile	reads the custom presentation of an instance
IPS_HasChanges	checks if the configuration was changed
IPS_ResetChanges	resets the modified configuration to the current configuration
IPS_SetConfiguration	sets a new configuration
IPS_SetProperty	sets a planned value for a property

IPS_ApplyChanges

```
bool IPS_ApplyChanges (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function applies the planned configuration as the current actual configuration and reinitializes appropriate module components if necessary, e.g., for a socket module, the network connection is dropped and rebuilt.

EXAMPLE

```
if(IPS_HasChanges(12345))
{
IPS_ApplyChanges(12345);
}
```

IPS_GetConfiguration

Require: Symcon >= 2.7

```
string IPS_GetConfiguration (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function is executed successfully, it returns the configuration of the instance as a string.

DESCRIPTION

Returns the configuration of the instance with the ID **InstanceID** as a string.

EXAMPLE

```
// Read the configuration of instance 12345 and output it via "echo".
$config = IPS_GetConfiguration(12345);
echo $config;

// Example output of a ModBus instance:
//
{"DataType":3,"WriteAddress":0,"ReadAddress":0,"Poller":3600000,"ReadOnly
```

IPS_GetConfigurationForm

Require: Symcon >= 2.7

```
string IPS_GetConfigurationForm (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function is executed successfully, it returns the configuration form of the instance as a JSON-encoded string.

DESCRIPTION

Returns the configuration form of the instance with the ID **InstanceID** as a JSON-encoded string.

EXAMPLE

```
// Read the configuration form of instance 12345 and output its
elements
$configpage = json_decode(IPS_GetConfigurationForm(12345));
var_dump ($configpage->elements);

/* Example output:
array(2) {
[0]=>
object(stdClass)#2 (2) {
["type"]=>
string(5) "Label"
["label"]=>
string(28) "Minimum needed daily changes"
}
[1]=>
object(stdClass)#3 (3) {
["type"]=>
string(13) "NumberSpinner"
["name"]=>
string(19) "RequiredSwitchCount"
["caption"]=>
string(5) "Count"
```

```
}
}
*/
```

IPS_GetConfigurationForParent

```
string IPS_GetConfigurationForParent (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

JSON encoded object with the properties which the instance specifies for its parent instance. If the instance specifies nothing, "{}" is returned.

DESCRIPTION

The function returns the configuration which the instance with the ID **InstanceID** specifies for its parent instance. The result is a JSON encoded object whose keys are the names of the properties of the parent instance and whose values are the required values.
The management console uses this information to set the values when connecting or creating the parent instance. Properties specified this way can no longer be changed in the configuration form of the parent instance (see IPS_GetConfigurationsFromChildren).
For PHP modules, the result of the module's function GetConfigurationForParent is returned. If it is not implemented or returns an empty string, "{}" is returned. If the output does not start with "{" (e.g. on an error message from PHP), the function fails with this output as error. The specification may depend on the current configuration, e.g. on the type of the currently connected parent instance.
If the instance does not exist, the function fails with "Instance #... does not exist".

EXAMPLE

```
// Determine the configuration which instance 12345 (e.g. an IRTrans
Gateway connected to a Client Socket) specifies for its parent instance
echo IPS_GetConfigurationForParent(12345);

/* e.g. returns:
{"Port": 21000}
*/
```

IPS_GetProperty

Require: Symcon >= 2.7

```
mixed IPS_GetProperty (int $InstanceID, string $Property)
```

PARAMETERS

InstanceID	ID of the instance
Property	Name of the property

RETURNS

Current value of the property

DESCRIPTION

The function reads the current value of the **Property** for **InstanceID**. An exact list of all possible properties for every instance is shown at the description of the function IPS_SetProperty.
It is possible to set a new planned value via IPS_SetProperty and apply it via IPS_ApplyChanges.

EXAMPLE

```
echo IPS_GetProperty($id, "Open"); // Is the I/O instance active?
```

IPS_GetVisualizationTile

```
string IPS_GetVisualizationTile (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function is executed successfully, it returns the HTML content of the custom presentation of the instance. If the instance has no custom presentation, the string can be empty.

DESCRIPTION

Returns the HTML content of the custom presentation of the instance with the ID **InstanceID**, as displayed by the tile visualization for the tile of the instance.
Modules provide this content via the HTML-SDK by overriding the module function GetVisualizationTile. IPS_GetVisualizationTile calls this function for the given instance. For PHP modules, a <script> block is prepended to the return value that contains the JavaScript functions of the HTML-SDK as well as the translations from the locale.json of the module. This block is also present if the module does not override GetVisualizationTile. For the modules built into IP-Symcon the function returns an empty string.
The following errors are possible:
If the instance does not exist, the function fails with "Instance #... does not exist".
If **InstanceID** is outside of the valid range, it fails with "Parameter for InstanceID is not inside of the specified bounds (0.....)".
If the instance could not be loaded, e.g. because the associated module is not available, it fails with "InstanceInterface is not available".

EXAMPLE

```
// Read the custom presentation of instance 12345 and save it as an
HTML file
$html = IPS_GetVisualizationTile(12345);
if ($html === '') {
echo "The instance has no custom presentation";
} else {
file_put_contents(IPS_GetKernelDir() . 'tile.html', $html);
echo strlen($html) . " bytes saved";
}
```

IPS_HasChanges

```
bool IPS_HasChanges (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

TRUE if there are changes, otherwise FALSE

DESCRIPTION

The function checks whether any changes were made to the configuration that need to be applied.

EXAMPLE

```
if(IPS_HasChanges(12345))
IPS_ApplyChanges(12345);
```

IPS_ResetChanges

```
bool IPS_ResetChanges (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function overwrites the new planned configuration with the current actual configuration.

EXAMPLE

```
if(IPS_HasChanges(12345))
{
IPS_ResetChanges(12345);
}
```

IPS_SetConfiguration

Require: Symcon >= 2.7

```
bool IPS_SetConfiguration (int $InstanceID, string $Configuration)
```

PARAMETERS

InstanceID	ID of the instance
Configuration	The configuration as a string

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the planned configuration for **InstanceID** to **Configuration**.
It needs to be applied via IPS_ApplyChanges to affect the instance.

EXAMPLE

```
// Set the configuration for the ModBus instance with the ID 12345
IPS_SetConfiguration(12345,
'{"DataType":3,"WriteAddress":1,"ReadAddress":0,"Poller":3600000,"ReadOnl
IPS_ApplyChanges(12345); // Apply new configuration
```

IPS_SetProperty

Require: Symcon >= 2.7

```
bool IPS_SetProperty (int $InstanceID, string $Property, mixed $Value)
```

PARAMETERS

InstanceID	ID of the instance
Property	Name of the property
Value	New planned value of the property

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the planned value of the property **Property** for **InstanceID** to **Value**.
A planned value needs to be applied via IPS_ApplyChanges to affect the instance.



If information is missing in the following list, e.g., which values and associations belong to certain properties, they can be looked up in the "/forms" folder in the Symcon installation for the respective instances.

ALL2x

PROPERTY	TYPE	DEFAULT VALUE
DeviceNum	integer	0
DeviceType	integer	0
Interval	integer	0
IPAddress	string	

ALL3690

PROPERTY	TYPE	DEFAULT VALUE
Interval	integer	0
IPAddress	string	

ALL3691

PROPERTY	TYPE	DEFAULT VALUE
Interval	integer	0
IPAddress	string	

ALL4000

PROPERTY	TYPE	DEFAULT VALUE
Devices	string	[0,0,0,

BACnet Client Object

PROPERTY	TYPE	DEFAULT VALUE
COV	boolean	true
InstanceNumber	integer	0
ObjectType	integer	-1
Priority	integer	8

BACnet Server Device

PROPERTY	TYPE	DEFAULT VALUE
CompatibilityMode	boolean	false
Configuration	string	[]
ForeignDeviceHost	string	
ForeignDevicePort	integer	47808
ForeignDeviceRegistration	boolean	false
ForeignDeviceTTL	integer	300
InstanceNumber	integer	0
ObjectType	integer	8

Calendar Control

PROPERTY	TYPE	DEFAULT VALUE
Items	string	[]

Catan Channel DOI

PROPERTY	TYPE	DEFAULT VALUE
AnalogDefaultVal	double	0
Channel	integer	1
CounterScale	double	1
CovIncrement	double	0.1
DigitalDefaultVal	boolean	false
Enabled	boolean	true
FalseText	string	Aus
Inverted	boolean	false
LocalSetupEnabled	boolean	true
MinSendTimeMs	integer	100
Name	string	
NameSource	integer	0
OverrideCancelable	boolean	false

PROPERTY	TYPE	DEFAULT VALUE
OverrideEnabled	boolean	false
ShowErrorCode	boolean	false
ShowPointStatus	boolean	false
ShowRawCounterValue	boolean	false
TrueText	string	An
Type	integer	1
Unit	string	
UseDefaultValOnFailure	boolean	false
UseDefaultValOnStart	boolean	false

Catan Channel DOR

PROPERTY	TYPE	DEFAULT VALUE
Channel	integer	1
CounterScale	double	1
DigitalDefaultVal	boolean	false
Enabled	boolean	true
FalseText	string	Aus
Inverted	boolean	false
LocalSetupEnabled	boolean	true
Name	string	
NameSource	integer	0
OverrideCancelable	boolean	false
OverrideEnabled	boolean	false
ShowErrorCode	boolean	false
ShowPointStatus	boolean	false
ShowRawCounterValue	boolean	false
TrueText	string	An
UseDefaultValOnFailure	boolean	false
UseDefaultValOnStart	boolean	false

Catan Channel UI

PROPERTY	TYPE	DEFAULT VALUE
Channel	integer	1
CounterScale	double	1
CovIncrement	double	0.1
Enabled	boolean	true
FalseText	string	Aus

PROPERTY	TYPE	DEFAULT VALUE
Inverted	boolean	false
LocalSetupEnabled	boolean	true
MinSendTimeMs	integer	100
Name	string	
NameSource	integer	0
OpenContact	boolean	false
OverrideCancelable	boolean	false
OverrideEnabled	boolean	false
ShowErrorCode	boolean	false
ShowPointStatus	boolean	false
ShowRawCounterValue	boolean	false
TrueText	string	An
Type	integer	1
Unit	string	

Catan Channel UOI

PROPERTY	TYPE	DEFAULT VALUE
AnalogDefaultVal	double	0
Channel	integer	1
CounterScale	double	1
CovIncrement	double	0.1
DigitalDefaultVal	boolean	false
Enabled	boolean	true
FalseText	string	Aus
Inverted	boolean	false
LocalSetupEnabled	boolean	true
MinSendTimeMs	integer	100
Name	string	
NameSource	integer	0
OverrideCancelable	boolean	false
OverrideEnabled	boolean	false
ShowErrorCode	boolean	false
ShowPointStatus	boolean	false
ShowRawCounterValue	boolean	false
TrueText	string	An
Type	integer	1
Unit	string	

PROPERTY	TYPE	DEFAULT VALUE
UseDefaultValOnFailure	boolean	false
UseDefaultValOnStart	boolean	false

Catan Configurator

PROPERTY	TYPE	DEFAULT VALUE
DisplayName	string	Catan C1
ModelNumber	string	1886438
RS485_C1_Biasing	boolean	false
RS485_C1_Termination	boolean	false
RS485_C2_Biasing	boolean	false
RS485_C2_Termination	boolean	false

Catan Device

PROPERTY	TYPE	DEFAULT VALUE
IPAddress	string	
SerialNumber	string	

Catan Discovery

This module has no properties.

Catan Splitter

This module has no properties.

Client Socket

PROPERTY	TYPE	DEFAULT VALUE
Certificate	string	
CertificateAuthority	string	
Host	string	
Open	boolean	false
Password	string	
Port	integer	0
PrivateKey	string	
UseCertificate	boolean	false
UseSSL	boolean	false
VerifyHost	boolean	true
VerifyPeer	boolean	true

CMI

PROPERTY	TYPE	DEFAULT VALUE
Device	integer	0

PROPERTY	TYPE	DEFAULT VALUE
Interval	integer	0
IPAddress	string	
NodeNumber	integer	0
RequestDate	boolean	false
RequestDLBus	boolean	false
RequestElectricalPower	boolean	false
RequestGeneral	boolean	false
RequestInputs	boolean	true
RequestKNX	boolean	false
RequestLoggingAnalog	boolean	false
RequestLoggingDigital	boolean	false
RequestMBus	boolean	false
RequestModbus	boolean	false
RequestNetworkAnalog	boolean	false
RequestNetworkDigital	boolean	false
RequestOutputs	boolean	true
RequestSun	boolean	false
RequestTime	boolean	false

Connect Control

PROPERTY	TYPE	DEFAULT VALUE
Active	boolean	true

Cutter

PROPERTY	TYPE	DEFAULT VALUE
DeleteCutChars	boolean	false
InputLength	integer	0
LeftCutChar	string	
LeftCutCharAsHex	boolean	false
ParseType	integer	0
RightCutChar	string	
RightCutCharAsHex	boolean	false
SyncChar	string	
SyncCharAsHex	boolean	false
Timeout	integer	0

DMX Channel

PROPERTY	TYPE	DEFAULT VALUE
ChannelBase	integer	1
ChannelCount	integer	1
ChannelSize	integer	1

DMX Gateway

PROPERTY	TYPE	DEFAULT VALUE
GatewayMode	integer	0
OnlySendOnChanges	boolean	true
Universe	integer	0

DMX RGB

PROPERTY	TYPE	DEFAULT VALUE
ChannelB	integer	3
ChannelG	integer	2
ChannelR	integer	1
ChannelSize	integer	1

DNSSD Control

PROPERTY	TYPE	DEFAULT VALUE
Services	string	[]

DoorIP

PROPERTY	TYPE	DEFAULT VALUE
Actions	string	[]
Active	boolean	false
AuthID	string	
Domain	string	
EnforceTransport	string	
Mode	integer	0
Password	string	
TrackMediaID	integer	0
Tracks	integer	0
Username	string	symcon_{InstanceID}

dS Apartment

PROPERTY	TYPE	DEFAULT VALUE
GroupID	integer	0

dS Configurator
This module has no properties.
dS Expert
This module has no properties.
dS Joker

PROPERTY	TYPE	DEFAULT VALUE
BinaryInputsInterval	integer	0
dSID	string	
SensorInputsInterval	integer	0

dS Light

PROPERTY	TYPE	DEFAULT VALUE
dSID	string	

dS Shutter

PROPERTY	TYPE	DEFAULT VALUE
dSID	string	

dS Splitter

PROPERTY	TYPE	DEFAULT VALUE
IPAddress	string	dss.local
Token	string	

dS Zone

PROPERTY	TYPE	DEFAULT VALUE
GroupID	integer	0
ZoneID	integer	0

Dummy Module
This module has no properties.
EHZ Device

PROPERTY	TYPE	DEFAULT VALUE
DeviceAddress	string	
EnableHistoricalData	boolean	false
Interval	integer	600
Mode	integer	0

PROPERTY	TYPE	DEFAULT VALUE
OBISList	string	[]
Password	string	(00000000)
RequirePassword	boolean	false

EHZ Discovery

This module has no properties.

EHZ Gateway

PROPERTY	TYPE	DEFAULT VALUE
GatewayMode	integer	0

ekey Fingerprint

PROPERTY	TYPE	DEFAULT VALUE
Protocol	integer	0
Separator	string	_

Energy Distribution

PROPERTY	TYPE	DEFAULT VALUE
Calculation	boolean	false
Config	string	[]
EnableLimits	boolean	false
Restrictions	string	[]

EnOcean alphaEOS SENSE TF-H

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean Configurator

This module has no properties.

EnOcean D20500

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	
EmulateStatus	boolean	true

EnOcean D20501

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	

PROPERTY	TYPE	DEFAULT VALUE
EmulateStatus	boolean	true

EnOcean D20502

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	
EmulateStatus	boolean	true

EnOcean Discovery
This module has no properties.
EnOcean EEP A50201RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50202RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50203RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50204RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50205RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50206RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50207RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50208RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50209RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5020ARX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5020BRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50210RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50211RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50212RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50213RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50214RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50215RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50216RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50217RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50218RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50219RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5021ARX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5021BRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50220RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50230RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50401RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50402RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50403RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50601RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50602RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50603RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50701RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50702RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50703RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50801RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50802RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50803RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50901RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50902RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50904RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50905RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50906RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50907RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A50908RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51001RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51002RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51003RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51004RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51005RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51006RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51007RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51008RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51009RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5100ARX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5100BRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5100CRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5100DRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51010RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51011RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51012RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51013RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51014RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51015RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51016RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51017RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51018RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51019RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5101ARX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5101BRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5101CRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5101DRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5101FRX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51020RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51021RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51022RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51023RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51101RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51102RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51103RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51104RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51200RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51201RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51202RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51203RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51301RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51302RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51303RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51304RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51305RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51401RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51402RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51403RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51404RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51405RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51406RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51407RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51408RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A51409RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A5140ARX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A52006

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
ReturnID	string	00000000
SPMode	integer	0

EnOcean EEP A52012RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A52012TX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0

EnOcean EEP A53001RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A53002RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A53003RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A53004RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A53701RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A53808RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP A53809RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP D20100

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20101

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20102

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20103

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20104

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20105

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20106

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20107

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20108

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20109

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D2010A

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D2010B

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D2010C

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D2010D

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D2010E

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D2010F

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20110

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20111

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20112

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20113

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20114

PROPERTY	TYPE	DEFAULT VALUE
ChannelID	integer	0
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20300

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D2030A

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D20310

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D20601

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D20700

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D21101

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21102

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21103

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21104

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21105

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21106

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21107

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21108

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	

EnOcean EEP D21430

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D21441

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D21500

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D23200

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D23201

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D23202

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	

EnOcean EEP D2A001

PROPERTY	TYPE	DEFAULT VALUE
DestinationID	string	
EmulateStatus	boolean	true

EnOcean EEP D50001RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60101RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60201RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60202RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60203RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60204RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60301RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60302RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60401RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60402RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60500RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60501RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F60502RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F61000RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EEP F61001RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoDimmer

PROPERTY	TYPE	DEFAULT VALUE
ButtonMode	integer	0
DeviceID	integer	0
DualStatus	boolean	false
EmulateStatus	boolean	true
Mode	integer	0
ReturnID	string	00000000
UseLastDimValueForOn	boolean	false

EnOcean EltakoFABH130

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFABH65S

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFAFT60

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFAH60

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFHK14

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
Mode	integer	0
ReturnID	string	00000000
ThermostatID	string	00000000

EnOcean EltakoFIUS65

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
EmulateStatus	boolean	true
ReturnID	string	00000000

EnOcean EltakoFKC

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFRW

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFSS12

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFTKE

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFTR65

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFUTH

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
ReturnID	string	00000000

EnOcean EltakoFWRB

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFWS61

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoFZS

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean EltakoRGBW

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
EmulateStatus	boolean	true
ReturnID	string	00000000

EnOcean EltakoShutter

PROPERTY	TYPE	DEFAULT VALUE
AutomaticReversal	boolean	false
AutomaticReversalDuration	integer	500
ButtonMode	integer	0
DeviceID	integer	0
EmulateStatus	boolean	true

PROPERTY	TYPE	DEFAULT VALUE
ReturnID	string	00000000

EnOcean EltakoSwitch

PROPERTY	TYPE	DEFAULT VALUE
ButtonMode	integer	0
DeviceID	integer	0
DualStatus	boolean	false
EmulateStatus	boolean	true
Mode	integer	0
ReturnID	string	00000000

EnOcean EltakoSwitchEx

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
EmulateStatus	boolean	true
ReturnID	string	00000000

EnOcean EltakoWWKW

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
EmulateStatus	boolean	true
ReturnID	string	00000000

EnOcean Gateway

PROPERTY	TYPE	DEFAULT VALUE
BaseID	string	0000A000
DelayTime	integer	50
GatewayMode	integer	2
NoiseThreshold	integer	-1

EnOcean Hoppe

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean Opus

PROPERTY	TYPE	DEFAULT VALUE
ButtonMode	integer	0
ButtonType	integer	0
DeviceID	integer	0

EnOcean PTM200

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean PTM200RX

PROPERTY	TYPE	DEFAULT VALUE
ButtonMode	integer	0
DeviceID	string	00000000

EnOcean RCM100

PROPERTY	TYPE	DEFAULT VALUE
ButtonMode	integer	0
ButtonType	integer	0
DeviceID	integer	0

EnOcean Shutter

PROPERTY	TYPE	DEFAULT VALUE
ButtonMode	integer	0
DeviceID	integer	0

EnOcean SmartDrive MX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
ReturnID	string	00000000
WUC	integer	0

EnOcean STM100

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean STM250

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000

EnOcean Thermokon SR0x

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000
TempOffset	double	0
TempRange	double	40

EnOcean Thermokon Thanos

PROPERTY	TYPE	DEFAULT VALUE
BaseSetPoint1	integer	22
DeviceID	integer	0
LowerSetPoint1	integer	17
ReturnID	string	00000000
StepSetPoint1	double	0.5
UpperSetPoint1	integer	30

EnOcean Valve Actuator

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
ReturnID	string	00000000
SPMode	integer	0

Event Control

PROPERTY	TYPE	DEFAULT VALUE
CustomStrategy	string	[]
DefaultStrategy	integer	2
ShutdownScript	integer	0
StartupScript	integer	0
StatusEvents	string	[]
WatchdogScript	integer	0

FHT

PROPERTY	TYPE	DEFAULT VALUE
Address	string	0000
EmulateStatus	boolean	false

FHZ
This module has no properties.
FS10

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
DeviceType	integer	0

FS10 Gateway
This module has no properties.
FS20

PROPERTY	TYPE	DEFAULT VALUE
Address	string	11
EnableReceive	boolean	true
EnableTimer	boolean	true
HomeCode	string	11111111
Mapping	string	[]
SubAddress	string	11

FS20EX

PROPERTY	TYPE	DEFAULT VALUE
DeviceList	string	[]
HomeCode	string	11111111

Heating Control

PROPERTY	TYPE	DEFAULT VALUE
Hysteresis	double	2
Mode	integer	0
OverrideVariables	string	[]
Range	double	3
ResendInterval	integer	15
SetBack	double	2
SetBackVariables	string	[]
SourceID	integer	0
TargetVariables	string	[]
Type	integer	0

Heating Control Legacy

PROPERTY	TYPE	DEFAULT VALUE
HandlerScript	integer	0
Hysteresis	double	0
OverrideVariables	string	[]
ResendInterval	integer	15
SetBack	double	0
SetBackVariables	string	[]
SourceVariable	integer	0
TransmitDevices	string	[]

HID

PROPERTY	TYPE	DEFAULT VALUE
DeviceProductID	integer	0
DeviceSerial	string	
DeviceVendorID	integer	0
Open	boolean	false

HMS

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
DeviceType	integer	0

HomeMatic CCU Configurator

This module has no properties.

HomeMatic CCU Device

PROPERTY	TYPE	DEFAULT VALUE
Address	string	
EmulateStatus	boolean	true
Protocol	integer	0

HomeMatic CCU Discovery

This module has no properties.

HomeMatic CCU Socket

PROPERTY	TYPE	DEFAULT VALUE
GRKeepAliveTimeout	integer	30
GROpen	boolean	false
GRPort	integer	9292
GRSSLPort	integer	49292
Host	string	

PROPERTY	TYPE	DEFAULT VALUE
HSOpen	boolean	true
HSPort	integer	8181
HSSSLPort	integer	48181
IPAddress	string	{Local IP Address}
IPKeepAliveTimeout	integer	30
IPOpen	boolean	true
IPPort	integer	2010
IPSSLPort	integer	42010
Open	boolean	false
Password	string	
Port	integer	5544
RFKeepAliveTimeout	integer	30
RFOpen	boolean	false
RFPort	integer	2001
RFSSLPort	integer	42001
Username	string	Admin
UseSSL	boolean	false
VerifyHost	boolean	true
VerifyPeer	boolean	true
WaitTimeConfigurator	integer	25000
WaitTimeDevice	integer	7000
WaitTimePing	integer	5000
WaitTimeServiceMessages	integer	10000
WRKeepAliveTimeout	integer	30
WROpen	boolean	false
WRPort	integer	2000
WRSSLPort	integer	42000

HomeMatic HCU Configurator

This module has no properties.

HomeMatic HCU Device

PROPERTY	TYPE	DEFAULT VALUE
DatapointCreationMode	integer	0
DatapointWhitelistTree	string	[]
DeviceID	string	

HomeMatic HCU Discovery

This module has no properties.

HomeMatic HCU Gateway

PROPERTY	TYPE	DEFAULT VALUE
IPAddress	string	

HomeMatic HCU Group

PROPERTY	TYPE	DEFAULT VALUE
DatapointCreationMode	integer	0
DatapointWhitelist	string	[]
GroupID	string	

HTTP Client

PROPERTY	TYPE	DEFAULT VALUE
Active	boolean	true
AuthPass	string	
AuthType	integer	0
AuthUser	string	
Headers	string	[]
Interval	integer	0
URL	string	
UseBasicAuth	boolean	false
VerifyHost	boolean	true
VerifyPeer	boolean	true

Image Grabber

PROPERTY	TYPE	DEFAULT VALUE
AuthPass	string	
AuthType	integer	0
AuthUser	string	
ImageAddress	string	http://localhost/cam.jpg
ImageType	integer	0
Interval	integer	300
UseBasicAuth	boolean	false
UseCaching	boolean	true
VerifyHost	boolean	true
VerifyPeer	boolean	true

IMAP

PROPERTY	TYPE	DEFAULT VALUE
CacheInterval	integer	300

PROPERTY	TYPE	DEFAULT VALUE
CacheSize	integer	10
Host	string	imap.gmail.com
Password	string	
Port	integer	993
UseAuthentication	boolean	true
Username	string	mail@gmail.com
UseSSL	boolean	true
VerifyHost	boolean	true
VerifyPeer	boolean	true

IPS-868 Accelerometer

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0

IPS-868 AD

PROPERTY	TYPE	DEFAULT VALUE
Channel	integer	1
DeviceID	integer	184
Interval	integer	0

IPS-868 AirQuality

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	176
Interval	integer	0
LimitRed	integer	1500
LimitYellow	integer	1000

IPS-868 Configurator

This module has no properties.

IPS-868 Counter

PROPERTY	TYPE	DEFAULT VALUE
CounterID	integer	1
DeviceID	integer	128
Impulses	integer	1000
Interval	integer	0
Mode	integer	1

IPS-868 DA

PROPERTY	TYPE	DEFAULT VALUE
Channel	integer	1
DeviceID	integer	184

IPS-868 Data

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	184
Request	string	

IPS-868 Discovery

This module has no properties.

IPS-868 Display

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0

IPS-868 Display Input

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0

IPS-868 Display Output

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
OutputID	integer	0

IPS-868 Gateway

PROPERTY	TYPE	DEFAULT VALUE
GatewayMode	integer	0

IPS-868 Input

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
InputID	integer	1

IPS-868 Level

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	166
Interval	integer	0

IPS-868 Servo

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	144

IPS-868 Servo Input

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	1

IPS-868 Stripe

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	144
Mode	integer	0

IPS-868 Stripe Input

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	1

IPS-868 Thermo

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	166
Interval	integer	0
Mode	integer	1

IPS-868 Tracker

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
TrackerID	integer	1

IPS-868 WatchDogTimer

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	165

IRTrans Device

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0

IRTrans Gateway

This module has no properties.

JSON Decoder

PROPERTY	TYPE	DEFAULT VALUE
Locked	boolean	false

KNX Configurator

PROPERTY	TYPE	DEFAULT VALUE
EnableStatusDetection	boolean	true
EnableStatusRemoval	boolean	false
ExportMode	integer	1
OPCExport	string	
StatusDetectionKeywords	string	[{"Keyword": " status"},{"Keyword": ".status"}, {"Keyword": "-status"},{"Keyword": " rm"}, {"Keyword": ".rm"},{"Keyword": "-rm"}]
StatusDetectionMatching	integer	0
StatusRemovalKeywords	string	[{"Keyword": " ea"},{"Keyword": " e/a"}]
TTLExport	string	
XMLExport	string	

KNX Device

PROPERTY	TYPE	DEFAULT VALUE
GroupAddresses	string	[]
TTLID	string	

KNX Discovery
This module has no properties.
KNX DPT 1

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 10

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 11

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 12

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 13

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 14

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 15

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 16

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 17

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 18

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 19

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 2

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 20

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 200

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 201

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 202

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 203

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 204

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 205

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 206

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 207

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 209

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 21

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 210

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 211

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 212

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 213

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 214

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 215

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 216

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 217

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 218

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 219

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 22

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 220

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 221

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 222

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 223

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 224

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 225

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 229

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 23

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 230

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 231

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 232

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 234

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 235

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 236

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 237

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 238

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 239

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 240

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 241

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 242

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 249

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 25

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 251

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 255

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 26

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 27

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 273

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 275

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 28

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 29

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 3

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 30

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 31

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 4

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 5

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 6

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 7

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 8

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX DPT 9

PROPERTY	TYPE	DEFAULT VALUE
Address1	integer	0
Address2	integer	0
Address3	integer	0
CapabilityRead	boolean	false
CapabilityReceive	boolean	true
CapabilityTransmit	boolean	false
CapabilityWrite	boolean	true
Dimension	integer	0
EmulateStatus	boolean	true
Mapping	string	[]

KNX EIS Group

PROPERTY	TYPE	DEFAULT VALUE
GroupAddress1	integer	0
GroupAddress2	integer	0
GroupAddress3	integer	0
GroupCapabilityRead	boolean	false
GroupCapabilityReceive	boolean	true
GroupCapabilityTransmit	boolean	false
GroupCapabilityWrite	boolean	true
GroupEmulateStatus	boolean	true
GroupFunction	string	Switch
GroupInterpretation	string	Standard
GroupMapping	string	[]

KNX Gateway

PROPERTY	TYPE	DEFAULT VALUE
GatewayMode	integer	1
KeyringExport	string	
KeyringPassword	string	
SendLimit	integer	6
TunnelAddress	string	

KNX RGB

PROPERTY	TYPE	DEFAULT VALUE
GroupAddress1	integer	0
GroupAddress2	integer	0
GroupAddress3	integer	0
GroupCapabilityRead	boolean	false
GroupCapabilityReceive	boolean	true
GroupCapabilityTransmit	boolean	false
GroupCapabilityWrite	boolean	true
GroupEmulateStatus	boolean	true
GroupMapping	string	[]

KNX RGBW

PROPERTY	TYPE	DEFAULT VALUE
GroupAddress1	integer	0
GroupAddress2	integer	0
GroupAddress3	integer	0

PROPERTY	TYPE	DEFAULT VALUE
GroupCapabilityRead	boolean	false
GroupCapabilityReceive	boolean	true
GroupCapabilityTransmit	boolean	false
GroupCapabilityWrite	boolean	true
GroupEmulateStatus	boolean	true
GroupMapping	string	[]

KNX Shutter

PROPERTY	TYPE	DEFAULT VALUE
EnableBladePosition	boolean	false
EnablePosition	boolean	false
EnableStep	boolean	false
GroupBladePositionAddress1	integer	0
GroupBladePositionAddress2	integer	0
GroupBladePositionAddress3	integer	0
GroupBladePositionMapping	string	[]
GroupMoveAddress1	integer	0
GroupMoveAddress2	integer	0
GroupMoveAddress3	integer	0
GroupMoveMapping	string	[]
GroupPositionAddress1	integer	0
GroupPositionAddress2	integer	0
GroupPositionAddress3	integer	0
GroupPositionMapping	string	[]
GroupStepAddress1	integer	0
GroupStepAddress2	integer	0
GroupStepAddress3	integer	0
GroupStepMapping	string	[]
GroupStopAddress1	integer	0
GroupStopAddress2	integer	0
GroupStopAddress3	integer	0
GroupStopMapping	string	[]

KS300
This module has no properties.
LCN Configurator
This module has no properties.
LCN Data

PROPERTY	TYPE	DEFAULT VALUE
DataType	integer	0
Threshold11ValueType	integer	13
Threshold12ValueType	integer	13
Threshold13ValueType	integer	13
Threshold14ValueType	integer	13
Threshold1ValueType	integer	13
Threshold21ValueType	integer	13
Threshold22ValueType	integer	13
Threshold23ValueType	integer	13
Threshold24ValueType	integer	13
Threshold2ValueType	integer	13
Threshold31ValueType	integer	13
Threshold32ValueType	integer	13
Threshold33ValueType	integer	13
Threshold34ValueType	integer	13
Threshold3ValueType	integer	13
Threshold41ValueType	integer	13
Threshold42ValueType	integer	13
Threshold43ValueType	integer	13
Threshold44ValueType	integer	13
Threshold4ValueType	integer	13
Threshold5ValueType	integer	13
Threshold6ValueType	integer	13

LCN Display

This module has no properties.

LCN Gateway

PROPERTY	TYPE	DEFAULT VALUE
OutputMode	integer	2
Password	string	
RequireAcknowledge	boolean	true
Username	string	

LCN Module

PROPERTY	TYPE	DEFAULT VALUE
Segment	integer	0
Target	integer	0

LCN RGBW

This module has no properties.

LCN Shutter

PROPERTY	TYPE	DEFAULT VALUE
DirectionChannel	integer	1
DirectionInvert	boolean	false
MoveChannel	integer	1
Unit	integer	0

LCN ShutterMotor

PROPERTY	TYPE	DEFAULT VALUE
Interval	integer	0
Motor	integer	1

LCN Unit

PROPERTY	TYPE	DEFAULT VALUE
Channel	integer	0
Unit	integer	0

LCN Value

PROPERTY	TYPE	DEFAULT VALUE
A10ValueEnabled	boolean	false
A10ValueType	integer	0
A11ValueEnabled	boolean	false
A11ValueType	integer	0
A12ValueEnabled	boolean	false
A12ValueType	integer	0
A1ValueEnabled	boolean	false
A1ValueType	integer	0
A2ValueEnabled	boolean	false
A2ValueType	integer	0
A3ValueEnabled	boolean	false
A3ValueType	integer	0
A4ValueEnabled	boolean	false
A4ValueType	integer	0
A5ValueEnabled	boolean	false
A5ValueType	integer	0
A6ValueEnabled	boolean	false

PROPERTY	TYPE	DEFAULT VALUE
A6ValueType	integer	0
A7ValueEnabled	boolean	false
A7ValueType	integer	0
A8ValueEnabled	boolean	false
A8ValueType	integer	0
A9ValueEnabled	boolean	false
A9ValueType	integer	0
C1CounterEnabled	boolean	false
C2CounterEnabled	boolean	false
C3CounterEnabled	boolean	false
C4CounterEnabled	boolean	false
EmulateStatus	boolean	false
Interval	integer	0
R1TargetEnabled	boolean	false
R1ValueEnabled	boolean	false
R1ValueType	integer	0
R2TargetEnabled	boolean	false
R2ValueEnabled	boolean	false
R2ValueType	integer	0
S1TargetEnabled	boolean	false
S2TargetEnabled	boolean	false
TShiftEnabled	boolean	false
TValueEnabled	boolean	false
TValueType	integer	0

LevelJet

This module has no properties.

Location Control

PROPERTY	TYPE	DEFAULT VALUE
IntervalPosition	integer	300
IsDayEnd	integer	1
IsDayStart	integer	2
Location	string	{"latitude": 0, "longitude": 0}
Offsets	string	[]

M-Bus Configurator

PROPERTY	TYPE	DEFAULT VALUE
Address	integer	1
DefaultAddressing	integer	-1
DefaultInterval	integer	5
DefaultLimit	integer	0
SearchPause	integer	1000

M-Bus Device

PROPERTY	TYPE	DEFAULT VALUE
Address	integer	1
ExtendedRequests	string	[]
Interval	integer	5
Limit	integer	0
SecondaryAddress	string	00000000
UseSecondaryAddress	boolean	false

M-Bus Discovery

This module has no properties.

M-Bus Gateway

PROPERTY	TYPE	DEFAULT VALUE
GatewayMode	integer	1
ResetMode	integer	1
TimeoutTime	integer	1000
TimeoutTimeLongFrame	integer	2000

M-Bus Wireless Configurator

This module has no properties.

M-Bus Wireless Device

PROPERTY	TYPE	DEFAULT VALUE
ID	string	00000000
Key	string	00000000000000000000000000000000

M-Bus Wireless Discovery

This module has no properties.

M-Bus Wireless Gateway

PROPERTY	TYPE	DEFAULT VALUE
GatewayMode	integer	1
Keys	string	[]
LinkMode	integer	2

Matter Configurator

This module has no properties.

Matter Controller

This module has no properties.

Matter Device

PROPERTY	TYPE	DEFAULT VALUE
EndpointId	integer	1
NodeId	integer	0

Media Player

PROPERTY	TYPE	DEFAULT VALUE
DeviceDriver	string	
DeviceName	string	
DeviceNum	integer	0
UpdateInterval	integer	0

ModBus Address

PROPERTY	TYPE	DEFAULT VALUE
ByteOrder	integer	0
DataType	integer	0
EmulateStatus	boolean	true
Factor	double	0
Length	integer	0
Poller	integer	5000
ReadAddress	integer	0
ReadFunctionCode	integer	1
WriteAddress	integer	0
WriteFunctionCode	integer	5

ModBus Configurator

This module has no properties.

ModBus Device

PROPERTY	TYPE	DEFAULT VALUE
Addresses	string	[]
ByteOrder	integer	0
EmulateStatus	boolean	true
ErrorHandling	integer	0
PollDataBlocks	string	[]
Poller	integer	5000

PROPERTY	TYPE	DEFAULT VALUE
PollType	integer	0
VirtualAddresses	string	[]

ModBus Discovery
This module has no properties.
ModBus Gateway

PROPERTY	TYPE	DEFAULT VALUE
DataBlocks	string	[]
DelayTimeASCII	integer	30
DelayTimeRTU	integer	20
DelayTimeTCP	integer	10
DeviceID	integer	1
GatewayMode	integer	0
SwapWords	boolean	false
TimeoutTime	integer	5

Module Control
This module has no properties.
MQTT Client

PROPERTY	TYPE	DEFAULT VALUE
ClientID	string	{generated from the license}
KeepAliveInterval	integer	60
Password	string	
Subscriptions	string	[{"Topic":"#", "QoS": 0}]
UserName	string	

MQTT Client Configurator
This module has no properties.
MQTT Client Device

PROPERTY	TYPE	DEFAULT VALUE
Locked	boolean	false
Retain	boolean	false
SendTopic	string	
Topic	string	
Type	integer	3
UseSendTopic	boolean	false

MQTT Server

PROPERTY	TYPE	DEFAULT VALUE
Password	string	
RetainedMessageExpiration	integer	604800
SessionQueueLimit	integer	500
SessionTimeout	integer	3600
UserName	string	

MQTT Server Configurator

This module has no properties.

MQTT Server Device

PROPERTY	TYPE	DEFAULT VALUE
Locked	boolean	false
Retain	boolean	false
SendTopic	string	
Topic	string	
Type	integer	3
UseSendTopic	boolean	false

Multicast Socket

PROPERTY	TYPE	DEFAULT VALUE
BindIP	string	{Local IP Address}
BindPort	integer	0
EnableBroadcast	boolean	false
EnableLoopback	boolean	false
EnableReuseAddress	boolean	false
Host	string	
MulticastIP	string	239.255.255.250
Open	boolean	false
Port	integer	0

Notification Control

PROPERTY	TYPE	DEFAULT VALUE
Active	boolean	true

OneWire Configurator

This module has no properties.

OneWire Discovery

This module has no properties.

OneWire F05

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60
Invert	boolean	false
PortSettings	integer	0

OneWire F10

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60

OneWire F12

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60
Invert0	boolean	false
Invert1	boolean	false
PortSettings0	integer	0
PortSettings1	integer	0

OneWire F1D

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60

OneWire F20

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60
PortType0	integer	0
PortType1	integer	0
PortType2	integer	0
PortType3	integer	0
Resolution0	integer	12
Resolution1	integer	12
Resolution2	integer	12
Resolution3	integer	12
Voltage0	integer	0

PROPERTY	TYPE	DEFAULT VALUE
Voltage1	integer	0
Voltage2	integer	0
Voltage3	integer	0

OneWire F26

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60

OneWire F28

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60
Precision	integer	0

OneWire F29

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60
Invert0	boolean	false
Invert1	boolean	false
Invert2	boolean	false
Invert3	boolean	false
Invert4	boolean	false
Invert5	boolean	false
Invert6	boolean	false
Invert7	boolean	false
PortSettings0	integer	0
PortSettings1	integer	0
PortSettings2	integer	0
PortSettings3	integer	0
PortSettings4	integer	0
PortSettings5	integer	0
PortSettings6	integer	0
PortSettings7	integer	0

OneWire F2C

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60
UseCPC	boolean	false

OneWire F3A

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	
Interval	integer	60
Invert0	boolean	false
Invert1	boolean	false
PortSettings0	integer	0
PortSettings1	integer	0

OneWire Gateway

PROPERTY	TYPE	DEFAULT VALUE
DelayTime	integer	10
GatewayMode	integer	0

OPCUA Client

PROPERTY	TYPE	DEFAULT VALUE
Certificate	string	
Endpoint	string	
Password	string	
PrivateKey	string	
SecurityMode	integer	1
SecurityPolicy	string	None
TokenType	integer	0
Username	string	
ValidateEndpoint	boolean	false

OPCUA Configurator

PROPERTY	TYPE	DEFAULT VALUE
RequestedMaxReferencesPerNode	integer	100

OPCUA Node

PROPERTY	TYPE	DEFAULT VALUE
NodeId	string	

OZW Configurator
This module has no properties.
OZW DataPoint

PROPERTY	TYPE	DEFAULT VALUE
Id	integer	0
Interval	integer	60
WriteAccess	boolean	false

OZW Device

PROPERTY	TYPE	DEFAULT VALUE
SerialNumber	string	

OZW Gateway

PROPERTY	TYPE	DEFAULT VALUE
IPAddress	string	
Password	string	Password
Username	string	Administrator
UseSSL	boolean	false

Permission Control

PROPERTY	TYPE	DEFAULT VALUE
AuthMechanism	string	
AutomaticSync	boolean	false
BaseDN	string	cn=users,dc=symcon
BindDN	string	uid=root,cn=users,dc=symcon
Filter	string	(objectClass=*)
Interval	integer	24
Password	string	
RoleMapping	string	[{"MemberOf":"cn=visu,cn=groups,dc=symcon","Role":"visu"}]
URI	string	ldap://192.168.1.250
UserCaseSensitive	boolean	true
UserMapping	string	[{"Attribute":"uid","Key":"userName"}, {"Attribute":"givenName","Key":"firstName"}, {"Attribute":"sn","Key":"lastName"}, {"Attribute":"mail","Key":"eMail"}]
VerifyCertificate	boolean	true

POP3

PROPERTY	TYPE	DEFAULT VALUE
CacheInterval	integer	300
CacheSize	integer	10
Host	string	pop.gmail.com
Password	string	
Port	integer	995
UseAuthentication	boolean	true
Username	string	mail@gmail.com
UseSSL	boolean	true
VerifyHost	boolean	true
VerifyPeer	boolean	true

Popup Module

This module has no properties.

Presence Control

PROPERTY	TYPE	DEFAULT VALUE
LockTime	integer	300
TriggerVariables	string	[]
ValueVariables	string	[]

Register Variable

PROPERTY	TYPE	DEFAULT VALUE
RXObjectID	integer	0

RS232 Discovery

This module has no properties.

Serial Port

PROPERTY	TYPE	DEFAULT VALUE
BaudRate	string	9600
DataBits	string	8
Open	boolean	false
Parity	string	None
Port	string	
StopBits	string	1

Server Socket

PROPERTY	TYPE	DEFAULT VALUE
Certificate	string	
CertificateAuthority	string	
Limit	integer	0
LogConnections	boolean	false
Open	boolean	false
Password	string	
Port	integer	1024
PrivateKey	string	
SilenceErrors	boolean	true
UseSSL	boolean	false

Shutter Control

PROPERTY	TYPE	DEFAULT VALUE
DriveDownTimeClose	double	0
DriveDownTimeDown	double	0
DriveDownTimeHalf	double	0
DriveUpTimeDown	double	0
DriveUpTimeHalf	double	0
DriveUpTimeOpen	double	0
LearnID	integer	0
MotorDelay	integer	0
ShutterType	integer	0
TargetID	integer	0

Shutter Control Legacy

PROPERTY	TYPE	DEFAULT VALUE
DriveDownTimeClose	double	0
DriveDownTimeDown	double	0
DriveDownTimeHalf	double	0
DriveUpTimeDown	double	0
DriveUpTimeHalf	double	0
DriveUpTimeOpen	double	0
HandlerScript	integer	0
MotorDelay	integer	0
ShutterType	integer	0
TransmitDevice	integer	0
TransmitDevice2	integer	0

Siemens Configurator

PROPERTY	TYPE	DEFAULT VALUE
DefaultInterval	integer	5000
PLCExport	string	
Type	integer	1

Siemens Device

PROPERTY	TYPE	DEFAULT VALUE
Address	integer	0
Area	integer	7
AreaAddress	integer	0
Bit	integer	0
DataType	integer	1
EmulateStatus	boolean	true
Factor	double	0
Length	integer	0
Poller	integer	0
ReadOnly	boolean	false

Siemens Gateway

PROPERTY	TYPE	DEFAULT VALUE
ClientTSAPHi	integer	1
ClientTSAPLo	integer	2
ServerTSAPHi	integer	1
ServerTSAPLo	integer	2

Skin Control

PROPERTY	TYPE	DEFAULT VALUE
DefaultSkin	string	

SMS

PROPERTY	TYPE	DEFAULT VALUE
APIID	integer	0
Password	string	
Sender	string	
Username	string	

SMS REST

PROPERTY	TYPE	DEFAULT VALUE
APIKey	string	

SMTP

PROPERTY	TYPE	DEFAULT VALUE
Host	string	smtp.gmail.com
Password	string	
Port	integer	587
Recipient	string	
SenderAddress	string	
SenderName	string	
UseAuthentication	boolean	true
Username	string	mail@gmail.com
UseSSL	boolean	true
VerifyHost	boolean	true
VerifyPeer	boolean	true

SSDP Control

This module has no properties.

SSE Client

PROPERTY	TYPE	DEFAULT VALUE
Active	boolean	false
Headers	string	[]
URL	string	
VerifyHost	boolean	true
VerifyPeer	boolean	true

Store Control

This module has no properties.

Sync Remote

PROPERTY	TYPE	DEFAULT VALUE
Active	boolean	false
BurstUpdateCount	integer	10
BurstUpdateInterval	integer	2
DownloadTotalID	integer	0
Host	string	
Interval	integer	60
Password	string	
Port	integer	3777

PROPERTY	TYPE	DEFAULT VALUE
StrategyProfile	integer	0
Timeout	integer	10000
Username	string	
UseSnapshotDiff	boolean	false
UseSSL	boolean	false
VerifyHost	boolean	true
VerifyPeer	boolean	true

Text Parser

PROPERTY	TYPE	DEFAULT VALUE
Rules	string	[]

Text To Speech

PROPERTY	TYPE	DEFAULT VALUE
TTSAudioOutput	string	
TTSEngine	string	

ThermoJet

This module has no properties.

Tile Visualization

PROPERTY	TYPE	DEFAULT VALUE
Access	integer	0
AllowPasswordlessOnWAN	boolean	false
AuthorizedUsersOnly	boolean	false
Automations	string	[]
AutoStart	string	[]
BaseID	integer	0
CategoryDisplay	string	Buttons
CustomCategories	string	[]
CustomDesign	string	
DashboardGauge	boolean	false

PROPERTY	TYPE	DEFAULT VALUE
DashboardTypes	string	[{"TypeID": "{F00618CD-385B-B69F-F6EC-5F27383EA870}","TypeName" : "Elektrizität","TypeBaseColor" : 16562691, "TypeProfile" : "~Power"},{ "TypeID": "{3A33E327-BB88-CCDD-E672-B008A9354FA7}", "TypeName" : "Wasser","TypeBaseColor" : 202748,"TypeProfile" : "~Water"},{"TypeID": "{B23EE5C8-60DC-6358-8ECC-23F9939CDA9A}","TypeName" : "Gas","TypeBaseColor" : 11338748,"TypeProfile" : "~Gas"}]
DateFormat	string	short
EnergyDashboard	string	[]
ExperimentalCharts	boolean	true
Favorites	string	[]
GraphContinuousSpan	boolean	false
GraphDynamicScaling	boolean	false
GraphHighDensity	boolean	false
GraphLegendType	integer	0
GraphRawDensity	boolean	false
GraphResolution	integer	0
GraphShowExtrema	boolean	false
GraphShowLegend	boolean	false
GraphTimeSpan	integer	1
GreetingName	string	
GreetingVariableID	integer	0
GridConfigurationSynchronization	integer	0
IgnorePasswordOnLAN	boolean	false
InfoWidgets	string	[]
LoginRoles	string	[]
LoginUsers	string	[]
OverrideDesign	boolean	false
Password	string	
ShowAutomations	boolean	true
ShowCalendar	boolean	true
ShowDateTime	boolean	false
ShowEdit	boolean	true
ShowEnergyDashboard	boolean	true
ShowFavorites	boolean	true
ShowGreeting	integer	0

PROPERTY	TYPE	DEFAULT VALUE
ShowNotifications	boolean	true
ShowSearch	boolean	true
ShowSettings	boolean	true
ShowShare	boolean	true
TimeFormat	string	short
VirtualHost	string	[]
VisuDisplayMode	integer	0

TMEX

PROPERTY	TYPE	DEFAULT VALUE
Open	boolean	false
Port	integer	1

Translation Control

PROPERTY	TYPE	DEFAULT VALUE
DeepLAuthenticationKey	string	

UDP Socket

PROPERTY	TYPE	DEFAULT VALUE
BindIP	string	{Local IP Address}
BindPort	integer	0
EnableBroadcast	boolean	false
EnableReuseAddress	boolean	false
Host	string	
Open	boolean	false
Port	integer	0

Universal RX

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	string	00000000
RORG	integer	165
ScriptID	integer	0

Util Control

PROPERTY	TYPE	DEFAULT VALUE
EnableRequestLicenseData	boolean	false

PROPERTY	TYPE	DEFAULT VALUE
EnableSendUsageData	boolean	false
LastCheckedVersion	string	

UVR1611

PROPERTY	TYPE	DEFAULT VALUE
DeviceCount	integer	1
Interval	integer	0

Velleman USB

This module has no properties.

Virtual IO

PROPERTY	TYPE	DEFAULT VALUE
Open	boolean	false

VoIP

PROPERTY	TYPE	DEFAULT VALUE
AcceptList	string	[]
Active	boolean	false
AuthID	string	
DefaultAction	integer	0
Domain	string	
EnforceTransport	string	
HandlerScript	integer	0
Password	string	
Username	string	

WebFront Visualization

PROPERTY	TYPE	DEFAULT VALUE
Access	integer	0
AllowPasswordlessOnWAN	boolean	false
AuthorizedUsersOnly	boolean	false
AutoStart	string	[]
BooleanCheckBox	boolean	true
EnableMobile	boolean	true
GraphContinuousSpan	boolean	false
GraphDynamicScaling	boolean	false
GraphHighDensity	boolean	false

PROPERTY	TYPE	DEFAULT VALUE
GraphLegendType	integer	0
GraphRawDensity	boolean	false
GraphShowExtrema	boolean	false
GraphShowLegend	boolean	false
GraphTimeSpan	integer	1
IgnorePasswordOnLAN	boolean	false
Items	string	
LoginRoles	string	[]
LoginUsers	string	[]
MobileID	integer	0
Nested	boolean	true
Password	string	
Skin	string	
VirtualHost	string	[]

WebHook Control

PROPERTY	TYPE	DEFAULT VALUE
Hooks	string	[]

WebOAuth Control

PROPERTY	TYPE	DEFAULT VALUE
ClientIDs	string	[]

WebServer

PROPERTY	TYPE	DEFAULT VALUE
Active	boolean	false
AuthPassword	string	
AuthUsername	string	
Certificate	string	
CertificateAuthority	string	
DHParameters	string	
EnableBasicAuth	boolean	false
EnableLogFile	boolean	false
EnableSSL	boolean	true
HomeDir	string	webfront
Password	string	
Port	integer	443

PROPERTY	TYPE	DEFAULT VALUE
PrivateKey	string	
Server	string	0.0.0.0

WinLIRC

This module has no properties.

WMRS200

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0
DeviceType	integer	0

WMRS200 Gateway

This module has no properties.

WS Client

PROPERTY	TYPE	DEFAULT VALUE
Active	boolean	false
Headers	string	[]
Type	integer	0
URL	string	
VerifyCertificate	boolean	true

WuT Counter

PROPERTY	TYPE	DEFAULT VALUE
Impulses	integer	1000
IONum	integer	0

WuT Gateway

PROPERTY	TYPE	DEFAULT VALUE
Interval	integer	0
IPAddress	string	
Password	string	

WuT Input

PROPERTY	TYPE	DEFAULT VALUE
IONum	integer	0

WuT Output

PROPERTY	TYPE	DEFAULT VALUE
IONum	integer	0

WuT ThermoHygro

PROPERTY	TYPE	DEFAULT VALUE
Interval	integer	0
IPAddress	string	

XBee Gateway

This module has no properties.

XBee Splitter

PROPERTY	TYPE	DEFAULT VALUE
DeviceID	integer	0

xComfort Binary Input

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0

xComfort Configurator

PROPERTY	TYPE	DEFAULT VALUE
DPData	string	

xComfort Dimmer

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
EmulateStatus	boolean	false

xComfort Energy

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
EnergyType	integer	0

xComfort Gateway

PROPERTY	TYPE	DEFAULT VALUE
GatewayMode	integer	0

xComfort Heating

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
TemperatureVariableID	integer	0

xComfort HRV

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
TemperatureVariableID	integer	0

xComfort Humidity

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0

xComfort Impulse

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
Impulses	integer	1000

xComfort One Channel Heating

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
TemperatureVariableID	integer	0

xComfort Remote

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0

xComfort Room Control

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0

xComfort Shutter

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
EmulateStatus	boolean	false

xComfort Switch

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
EmulateStatus	boolean	false

xComfort Temperature

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0

xComfort Value RX

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
ValueType	integer	0

xComfort Value TX

PROPERTY	TYPE	DEFAULT VALUE
DataPoint	integer	0
ValueType	integer	0

Z-Wave Configurator

PROPERTY	TYPE	DEFAULT VALUE
SupportSecurity	boolean	true
UseNetworkWide	boolean	true
UseNormalPower	boolean	true

Z-Wave Discovery

This module has no properties.

Z-Wave Gateway

PROPERTY	TYPE	DEFAULT VALUE
FailedThreshold	integer	10
GatewayMode	integer	1
OptimizeSchedule	integer	1
SendRetryCount	integer	3
SendRetryWait	integer	500
SendTimeout	integer	1

Z-Wave Module

PROPERTY	TYPE	DEFAULT VALUE
DisableOptimize	boolean	false

PROPERTY	TYPE	DEFAULT VALUE
DisableTransmitQueue	boolean	false
EnforceBasicClass	boolean	false
EnforceSimpleAssociations	boolean	false
Interval	integer	0
LockVariables	boolean	false
MultiInstanceID	integer	0
NodeID	integer	0

EXAMPLE

```
IPS_SetProperty($id, "Open", true); // I/O instance should be active
IPS_ApplyChanges($id); // Apply new configuration
```

References

IPS_GetReferenceList	returns the IDs of all references

IPS_GetReferenceList

```
array IPS_GetReferenceList (int $InstanceID)
```

PARAMETERS

InstanceID	Instance ID

RETURNS

An array of integer values of all IDs of the referenced objects. The order of the entries is not guaranteed.

DESCRIPTION

This function returns an array of the IDs of all referenced objects of the instance with the ID **InstanceID**. If there is no reference, the array is empty. (e.g. the "Archive Control" references all variables that are logged there)

EXAMPLE

```
IPS_GetReferenceList(12345);
```

Search

IPS_IsSearching	checks whether an instance is in search mode
IPS_StartSearch	starts the search mode of an instance
IPS_StopSearch	stops the search mode of an instance
IPS_SupportsSearching	checks whether an instance supports the search mode

IPS_IsSearching

```
bool IPS_IsSearching (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance to be checked

RETURNS

If the instance is in search mode, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the instance with the ID **InstanceID** is in search mode, i.e. whether the search mode started with IPS_StartSearch has neither expired nor been stopped with IPS_StopSearch. For modules without a search mode, it returns **FALSE**. For an instance that does not exist, it fails with "Instance #... does not exist".
Some device modules support a search mode, e.g. the instances for FS20, HMS, EnOcean or HomeMatic. In search mode, the instance reports every received device with its address, together with an existing instance for this address. The management console shows these results in the search dialog, so a device can be found by triggering it and then be assigned to the instance. The search mode ends automatically after 5 minutes.
The entry **InstanceIsSearching** of IPS_GetInstance contains the same information.

EXAMPLE

```
if (IPS_IsSearching(12345)) {
echo 'The instance is currently searching for devices';
}
```

IPS_StartSearch

```
bool IPS_StartSearch (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function starts the search mode of the instance with the ID **InstanceID** for 5 minutes. If the instance is already in search mode, the search is extended to 5 minutes from now.
Some device modules support a search mode, e.g. the instances for FS20, HMS, EnOcean or HomeMatic. In search mode, the instance reports every received device with its address, together with an existing instance for this address. The management console shows these results in the search dialog, so a device can be found by triggering it and then be assigned to the instance. The search mode ends automatically after 5 minutes. IPS_StopSearch ends the search early.
Start and end of the search are reported with the messages IM_SEARCHSTART and IM_SEARCHSTOP.
If the module does not support a search mode, the function fails with "This module does not support searching", for an instance that does not exist with "Instance #... does not exist".
IPS_SupportsSearching checks whether an instance supports the search mode.

EXAMPLE

```
if (IPS_SupportsSearching(12345)) {
IPS_StartSearch(12345);
}
```

IPS_StopSearch

```
bool IPS_StopSearch (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function stops the search mode of the instance with the ID **InstanceID** that was started with IPS_StartSearch. If the instance is not in search mode, nothing happens.
Some device modules support a search mode, e.g. the instances for FS20, HMS, EnOcean or HomeMatic. In search mode, the instance reports every received device with its address, together with an existing instance for this address. The management console shows these results in the search dialog, so a device can be found by triggering it and then be assigned to the instance. The search mode ends automatically after 5 minutes.
Start and end of the search are reported with the messages IM_SEARCHSTART and IM_SEARCHSTOP.
If the module does not support a search mode, the function fails with "This module does not support searching", for an instance that does not exist with "Instance #... does not exist".

EXAMPLE

```
if (IPS_IsSearching(12345)) {
IPS_StopSearch(12345);
}
```

IPS_SupportsSearching

```
bool IPS_SupportsSearching (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance to be checked

RETURNS

If the instance supports the search mode, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the module of the instance with the ID **InstanceID** supports a search mode that can be started with IPS_StartSearch. For an instance that does not exist, it fails with "Instance #... does not exist".
Some device modules support a search mode, e.g. the instances for FS20, HMS, EnOcean or HomeMatic. In search mode, the instance reports every received device with its address, together with an existing instance for this address. The management console shows these results in the search dialog, so a device can be found by triggering it and then be assigned to the instance. The search mode ends automatically after 5 minutes.
The entry **InstanceSupportsSearching** of IPS_GetInstance contains the same information.

EXAMPLE

```
if (IPS_SupportsSearching(12345)) {
echo 'The instance supports the search mode';
}
```

Timers

IPS_GetTimer	returns information about a timer
IPS_GetTimerList	returns the IDs of all timers
IPS_TimerExists	checks whether a specific timer exists

IPS_GetTimer

```
array IPS_GetTimer (int $TimerID)
```

PARAMETERS

TimerID	ID of the timer

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
TimerID	integer	ID of the timer
InstanceID	integer	ID of the instance that registered the timer (1 for internal timers of the kernel)
Name	string	Name of the timer as given on registration
Interval	integer	Interval in milliseconds, 0 if the timer is disabled
LastRun	integer	Unix timestamp of the last execution, 0 if the timer has never been executed
NextRun	integer	Unix timestamp of the next scheduled execution, 0 if no execution is scheduled
Running	boolean	TRUE while the timer is being executed
RunOnce	boolean	TRUE for a one-time timer that removes itself after its execution

DESCRIPTION

The function returns information about the timer with the ID **TimerID**. If the timer does not exist, the function fails with "Timer #... does not exist".
Timers are registered by instances, e.g. by PHP modules with RegisterTimer and SetTimerInterval or as one-time timers with RegisterOnceTimer. The built-in modules use timers as well, e.g. "SearchExpires" for the end of the search mode. The TimerIDs are assigned at runtime and are not persistent, one-time timers disappear after their execution.
Script timers (IPS_SetScriptTimer) are cyclic events and do not belong to these timers.
The next execution is scheduled after the previous one has finished, i.e. the interval counts from the end of the last execution.
IPS_GetTimerList returns the IDs of all timers, IPS_GetTimers returns the information of several timers at once.

EXAMPLE

```
print_r(IPS_GetTimer(7));

/* e.g. returns:
Array
(
[TimerID] => 7
[InstanceID] => 12345
[Name] => Update
[Interval] => 60000
[LastRun] => 1758880000
[NextRun] => 1758880060
[Running] =>
[RunOnce] =>
)
*/
```

IPS_GetTimerList

```
array IPS_GetTimerList ()
```

RETURNS

An array of integer values with the IDs of all timers. The order of the entries is not guaranteed.

DESCRIPTION

The function returns the IDs of all timers that are currently registered by instances. Timers are registered by instances, e.g. by PHP modules with RegisterTimer and SetTimerInterval or as one-time timers with RegisterOnceTimer. The built-in modules use timers as well, e.g. "SearchExpires" for the end of the search mode. The TimerIDs are assigned at runtime and are not persistent, one-time timers disappear after their execution. Script timers (IPS_SetScriptTimer) are cyclic events and do not belong to these timers. IPS_GetTimer returns information about a timer.

EXAMPLE

```
print_r(IPS_GetTimerList());

/* e.g. returns:
Array
(
[0] => 1
[1] => 2
[2] => 3
etc. ...
)
*/
```

IPS_TimerExists

```
bool IPS_TimerExists (int $TimerID)
```

PARAMETERS

TimerID	ID of the timer to be checked

RETURNS

If the timer exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the timer with the ID **TimerID** exists.
Timers are registered by instances, e.g. by PHP modules with RegisterTimer and SetTimerInterval or as one-time timers with RegisterOnceTimer. The built-in modules use timers as well, e.g. "SearchExpires" for the end of the search mode. The TimerIDs are assigned at runtime and are not persistent, one-time timers disappear after their execution. Script timers (IPS_SetScriptTimer) are cyclic events and do not belong to these timers.

EXAMPLE

```
if (IPS_TimerExists(7)) {
print_r(IPS_GetTimer(7));
}
```

Connection

IPS_ConnectInstance	creates a (data) connection between two instances
IPS_DisconnectInstance	disconnects a (data) connection between two instances
IPS_GetCompatibleInstances	returns compatible instances to an instance
IPS_GetCompatibleParents	returns the description of the compatible parent instances
IPS_IsInstanceCompatible	checks if a connection between two instances is possible

IPS_ConnectInstance

```
bool IPS_ConnectInstance (int $InstanceID, int $ParentID)
```

PARAMETERS

InstanceID	ID of the instance
ParentID	ID of the parent instance to be connected

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function connects the instance **InstanceID** with the instance **ParentID**, from the hardware (physical) point of view, so that data can be exchanged. An example is an FS20 device that is connected to an FHZ splitter.



Each instance can have only one parent instance.

EXAMPLE

```
IPS_ConnectInstance(12345, 23456);
```

IPS_DisconnectInstance

```
bool IPS_DisconnectInstance (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function disconnects a connected instance **InstanceID** from the parent instance. The specification of the parent instance is not necessary because an instance can only have one parent.

EXAMPLE

```
IPS_DisconnectInstance(12345);
```

IPS_GetCompatibleInstances

```
array IPS_GetCompatibleInstances (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

Array of integer values of all InstanceIDs compatible with **InstanceID**. The order of the entries is not guaranteed.

DESCRIPTION

The function returns the IDs of all available instances in IP-Symcon that are compatible as a parent instance with **InstanceID** (see IPS_IsInstanceCompatible). The IDs are listed in an array.

EXAMPLE

```
print_r(IPS_GetCompatibleInstances(12345));

/* returns e.g.:
Array
(
[0] => 22222
[1] => 33333
[2] => 44444
[3] => 55555
etc. ...
)
*/
```

IPS_GetCompatibleParents

```
string IPS_GetCompatibleParents (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

JSON encoded object which describes the compatible parent instances. The structure (type, moduleIDs or modules) is described at GetCompatibleParents. If the instance does not need a parent instance, "{}" is returned.

DESCRIPTION

The function returns a JSON encoded object which describes the parent instances the instance with the ID **InstanceID** can be connected to. The management console uses this information to suggest suitable parent instances when creating or editing the instance. With "require", a new instance or an instance without other child instances is needed, with "connect", existing compatible instances are suggested as well.
For PHP modules, the result of the module's function GetCompatibleParents is returned. If it is not implemented, a specification made with ConnectParent, RequireParent or ForceParent (like RequireParent) is used. If there is none either, the specification is determined based on the module type:

MODULE TYPE	RESULT
Core, I/O, Visualization	"{}"
Device, Configurator	type "connect" with all modules compatible according to the data flow (IPS_GetCompatibleModules)
Splitter, Discovery	type "require" with all modules compatible according to the data flow (IPS_GetCompatibleModules)

If there are no compatible modules, "{}" is returned as well. If the instance does not exist, the function fails with "Instance #... does not exist". The IDs of already existing compatible instances are returned by IPS_GetCompatibleInstances.

EXAMPLE

```
// Determine the compatible parent instances of instance 12345 (e.g. a
ThermoJet)
$parents = json_decode(IPS_GetCompatibleParents(12345), true);
print_r($parents);

/* e.g. returns:
Array
(
[moduleIDs] => Array
(
[0] => {6DC3D946-0D31-450F-A8C6-C42DB8D7D4F1}
)

[type] => require
)
*/
```

IPS_IsInstanceCompatible

```
bool IPS_IsInstanceCompatible (int $InstanceID, int $ParentID)
```

PARAMETERS

InstanceID	ID of the instance
ParentID	ID of the potential parent instance

RETURNS

TRUE if the instances are compatible, otherwise FALSE

DESCRIPTION

The function determines whether the instance with the ID ParentID is a valid parent instance for the instance with the ID InstanceID.

EXAMPLE

```
if (IPS_IsInstanceCompatible(12345, 23456))
echo "Instance is compatible with the other instance!";
```

Management of Categories

IPS_CategoryExists	checks if a category already exists
IPS_CreateCategory	creates a new category
IPS_DeleteCategory	deletes an existing category
IPS_GetCategory	returns information about a category
IPS_GetCategoryIDByName	searches and returns the ID of an existing category by its name
IPS_GetCategoryList	returns a list of all existing categories

IPS_CategoryExists

```
bool IPS_CategoryExists (int $CategoryID)
```

PARAMETERS

CategoryID	ID of the category to be tested

RETURNS

If the **CategoryID** exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the category with the ID **CategoryID** exists.

EXAMPLE

```
if (IPS_CategoryExists(45724))
echo "Category already exists!";
```

IPS_CreateCategory

```
int IPS_CreateCategory ()
```

RETURNS

ID of the newly created category

DESCRIPTION

This function creates a new category. No parameters are required. After the function has been executed, a new category appears in the category tree of IP-Symcon. The category is initially named "Unnamed Object (ID: 12345)" or similar, depending on its ID. The function IPS_SetName can be used to give the new category a meaningful name. The name should not be used for identification. The ID is responsible for that.
If the category is meant to be a subcategory, it can be moved to its designated parent by using the function IPS_SetParent.
The function returns an ID that uniquely identifies the created category.

EXAMPLE

```
// Create a new category called "Rain sensing"
$CatID = IPS_CreateCategory(); // Create the category
IPS_SetName($CatID, "Rain sensing"); // Name the category
IPS_SetParent($CatID, 12345); // Move the category below the object
with the ID "12345"
```

IPS_DeleteCategory

```
bool IPS_DeleteCategory (int $CategoryID)
```

PARAMETERS

CategoryID	ID of the category to be deleted

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes the existing category with the ID **CategoryID**.



The category can only be deleted if it has no child objects. Otherwise, the function fails with "Cannot delete category if it has child objects".

EXAMPLE

```
// Delete the category 47788
IPS_DeleteCategory(47788);
```

IPS_GetCategory

```
array IPS_GetCategory (int $CategoryID)
```

PARAMETERS

CategoryID	ID of the category

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
CategoryID	integer	ID of the category

DESCRIPTION

The function returns an array with information about the category with the ID **CategoryID**. Apart from their ID, categories have no properties of their own, so the array only contains the entry **CategoryID**. This function is documented for completeness only.
If no category with the ID **CategoryID** exists, the function fails with "Category #... does not exist". This also applies if the ID belongs to a different object type (e.g. a variable).



The useful information about a category, such as its name, ident, parent, position or visibility, are general object properties and can be read with IPS_GetObject. To only check whether a category exists, IPS_CategoryExists can be used.

EXAMPLE

```
print_r(IPS_GetCategory(12345));

/* e.g. returns:
Array
(
[CategoryID] => 12345
)
*/

// Name and parent are available via IPS_GetObject
```

```
$object = IPS_GetObject(12345);
echo $object['ObjectName'];
```

IPS_GetCategoryIDByName

```
int IPS_GetCategoryIDByName (string $CategoryName, int $ParentID)
```

PARAMETERS

CategoryName	Name of the searched category
ParentID	Object whose direct child objects are searched

RETURNS

ID of the found category, otherwise **FALSE**

DESCRIPTION

The function tries to determine the ID of the category called **CategoryName** whose parent has the ID **ParentID**. The ID of the first found category whose name matches **CategoryName** is used. If such a category was found, its ID is returned, otherwise **FALSE**.



Caution: Names in IP-Symcon can be chosen freely, so they are **not unique**. For this reason, the function can return a wrong ID.



This function generates a warning if the name was not found. The @ operator can be used to suppress this message. The exact functionality of this operator is explained in the PHP manual.

EXAMPLE

```
$CatID = @IPS_GetCategoryIDByName("Rain sensing", $ParentID);
if ($CatID === false)
echo "Category not found!";
else
echo "The category ID is: ". $CatID;
```

IPS_GetCategoryList

```
array IPS_GetCategoryList ()
```

RETURNS

An array that contains all IDs of the categories in IP-Symcon as integer values. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all categories within IP-Symcon. The IDs are listed in an array. If no category exists, the array is empty.

EXAMPLE

```
$allCategories = IPS_GetCategoryList();
print_r($allCategories);

/* returns e.g.:
Array
(
[0] => 0
[1] => 37659
[2] => 18326
etc. ...
)
*/
```

Management of Links

IPS_CreateLink	creates a new link
IPS_DeleteLink	deletes an existing link
IPS_GetLink	returns extensive information about a link
IPS_GetLinkIDByName	returns the ID of a link by its name
IPS_GetLinkList	returns a list of all existing links
IPS_LinkExists	checks if a link exists already
IPS_SetLinkTargetID	sets the target of a link

IPS_CreateLink

Require: Symcon >= 2.1

```
int IPS_CreateLink ()
```

RETURNS

ID of the newly created link

DESCRIPTION

The function creates a new link. It requires no parameters. After the execution a new link named "Unnamed Object (ID: 48490)" or similar, depending on the ID, appears in the category tree of IP-Symcon. The function IPS_SetName can be used to give a meaningful name to the new link. The name is irrelevant for identification. The ID is always responsible for that.
Furthermore, the link should be linked to another object. This can be done via the function IPS_SetLinkTargetID.
The function returns an ID that uniquely identifies the created link.

EXAMPLE

```
//Create a new link called "Rain sensing"
$LinkID = IPS_CreateLink(); //Create link
IPS_SetName($LinkID, "Rain sensing"); //Name the link
IPS_SetParent($LinkID, 12345); //Set a parent
IPS_SetLinkTargetID($LinkID, 54321); //Attach the link
```

IPS_DeleteLink

Require: Symcon >= 2.1

```
bool IPS_DeleteLink (int $LinkID)
```

PARAMETERS

LinkID	ID of the link to be deleted

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes the existing link with the ID **LinkID**.



The link can only be deleted if it has no child objects. Otherwise, the function fails with "Cannot delete link if it has child objects".

EXAMPLE

```
//Delete the link 47788
IPS_DeleteLink(47788);
```

IPS_GetLink

Require: Symcon >= 2.1

```
array IPS_GetLink (int $LinkID)
```

PARAMETERS

LinkID	ID of the link

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
LinkID	integer	ObjectID
LinkChildID(until 3.4)	integer	ID of the object with which this link is connected. See: IPS_SetLinkTargetID (since 4.0 permanently replaced by TargetID)
TargetID(since 2.6)	integer	ID of the object with which this link is connected. See: IPS_SetLinkTargetID

DESCRIPTION

The function returns an array containing information about the link with the ID **LinkID**. If the link does not exist, a warning is generated.

EXAMPLE

```
//Since Version 4.0
print_r(IPS_GetLink(19668));

/* returns e.g.:
Array
(
[LinkID] => 19668
[TargetID] => 14444
)
*/

//Up to Version 3.4
print_r(IPS_GetLink(19668));
```

```
/* returns e.g.:
Array
(
[LinkID] => 19668
[LinkChildID] => 14444
[TargetID] => 14444
)
*/

//Up to Version 2.5
print_r(IPS_GetLink(19668));

/* returns e.g.:
Array
(
[LinkID] => 19668
[LinkChildID] => 14444
)
*/
```

IPS_GetLinkIDByName

Require: Symcon >= 2.1

```
int IPS_GetLinkIDByName (string $LinkName, int $ParentID)
```

PARAMETERS

LinkName	Name of the link
ParentID	Object whose direct child objects are searched

RETURNS

ID of the found link, otherwise **FALSE**

DESCRIPTION

The function tries to determine the ID of the link called **LinkName** whose parent has the ID **ParentID**. The ID of the first found link whose name matches **LinkName** is used. If such a link was found, its ID is returned, otherwise **FALSE**.



Caution: Names in IP-Symcon can be chosen freely, so they are **not unique**. For this reason, the function can return a wrong ID.



This function generates a **warning** if the name was not found. Use the @ operator if you want to suppress this message. The exact functionality of this operator is explained in the PHP manual.

EXAMPLE

```
$LinkID = @IPS_GetLinkIDByName("Linked rain sensing", $ParentID);
if ($LinkID === false)
echo "Link not found!";
else
echo "The link ID is: ". $LinkID;
```

IPS_GetLinkList

Require: Symcon >= 2.1

```
array IPS_GetLinkList ()
```

RETURNS

An array that contains all IDs of the links in IP-Symcon as integer values. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all available links in IP-Symcon. The IDs are listed in an array. If no link exists, the array is empty.

EXAMPLE

```
$allLinks = IPS_GetLinkList();
print_r($allLinks);

/* returns e.g.:
Array
(
[0] => 37659
[1] => 18326
etc. ...
)
*/
```

IPS_LinkExists

Require: Symcon >= 2.1

```
bool IPS_LinkExists (int $LinkID)
```

PARAMETERS

LinkID	ID of the link to be checked

RETURNS

If the LinkID exists in the system, TRUE is returned, otherwise FALSE.

DESCRIPTION

The function checks whether the link with the ID LinkID exists.

EXAMPLE

```
if (IPS_LinkExists(45724))
echo "Link already exists!";
```

IPS_SetLinkTargetID

Require: Symcon >= 2.6

```
bool IPS_SetLinkTargetID (int $LinkID, int $LinkedObject)
```

PARAMETERS

LinkID	ID of the link to be changed
LinkedObject	ID of the target object for the link

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function links the object **LinkedObject** to the link **LinkID**, so that the link refers to this object. The object must exist and must not be a link itself, otherwise the function fails ("TargetID must not be another Link").

EXAMPLE

```
IPS_SetLinkTargetID($LinkID, 12345); //Refer to object 12345
```

Management of Media

IPS_CreateMedia	creates a new media object in the media pool
IPS_DeleteMedia	removes a media object from the media pool
IPS_GetMedia	returns extensive information about a specific media object
IPS_GetMediaContent	returns the content of a media object
IPS_GetMediaIDByFile	returns the ID of a media object by its file
IPS_GetMediaIDByName	returns the ID of a media object by its name
IPS_GetMediaList	returns a list of all media objects in the media pool
IPS_GetMediaListByType	returns a list of all media objects of a specific type
IPS_MediaExists	checks if a specific media object is contained within the media pool already
IPS_SendMediaEvent	send a notification that a media object was changed
IPS_SetMediaCached	activates that a media object is only modified in the internal memory
IPS_SetMediaContent	set the content of a media object
IPS_SetMediaFile	bind a media file to a media object

IPS_CreateMedia

```
int IPS_CreateMedia (int $MediaType)
```

PARAMETERS

	VALUE	CONSTANT	DESCRIPTION
MediaType	0	MEDIATYPE_IPSVIEW	creates an IPSView Form
	1	MEDIATYPE_IMAGE	creates an Image Object
	2	MEDIATYPE_SOUND	creates a Sound Object
	3	MEDIATYPE_STREAM	creates a Stream Object
	4	MEDIATYPE_CHART	creates a Chart Object
	5	MEDIATYPE_ DOCUMENT	creates a Document Object

RETURNS

ID of the newly created media object

DESCRIPTION

The function creates an empty media object of the type MediaType.
The function returns an ID that uniquely identifies the created media object. After creation, the object is still empty and needs to be connected to a real media file using the function IPS_SetMediaFile.

EXAMPLE

```
// Create a new media object with the name "Alarm"
$ImageFile = "media/Alarm_symbol.png"; // Image file (relative to the
IP-Symcon main folder)
$MediaID = IPS_CreateMedia(MEDIATYPE_IMAGE); // create the image in the
media pool
IPS_SetMediaFile($MediaID, $ImageFile, true); // connect the image in
the media pool with image file
IPS_SetName($MediaID, "Alarm"); // Name media object
```

```
IPS_SetParent($MediaID, 12345); // Move media object below the object
with the ID "12345"
```

IPS_DeleteMedia

```
bool IPS_DeleteMedia (int $MediaID, bool $DeleteFile)
```

PARAMETERS

MediaID	ID of the media object to be deleted
DeleteFile	TRUE if the file should be deleted, FALSE if the file should be moved to the 'deleted' folder.

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function removes the media object with ID MediaID from the media pool.



The media object can only be deleted if it has no children.

EXAMPLE

```
IPS_DeleteMedia($MediaID, true);
```

IPS_GetMedia

```
array IPS_GetMedia (int $MediaID)
```

PARAMETERS

MediaID	The ID of the media object

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
IsAvailable(up to 3.4)	boolean	TRUE if media file is available (replaced by MediaIsAvailable since version 4.0)
IsLinked(up to 3.4)	boolean	TRUE if the file is located outside the Media folder (removed since version 4.0)
LastUpdate(up to 3.4)	integer	Unix timestamp of last update (replaced by MediaUpdated since version 4.0)
MediaCRC	string	CRC32 of the file
MediaFile	string	Path to the file
MediaID	integer	MediaID
MediaIsAvailable(since 4.0)	boolean	TRUE if media file is available
MediaIsCached(since 4.0)	boolean	TRUE if the media object should be kept in memory. See IPS_SetMediaCached
MediaSize	integer	Size in bytes
MediaType	integer	Media type 0 `MEDIATYPE_FORM` Form 1 `MEDIATYPE_IMAGE` Image 2 `MEDIATYPE_SOUND` Sound 3 `MEDIATYPE_STREAM` Stream 4 `MEDIATYPE_CHART` Chart 5 `MEDIATYPE_DOCUMENT` Document
MediaUpdated(since 4.0)	integer	Unix timestamp of last update
SendEvent(up to 3.4)	boolean	TRUE if automatic file changes should be sent (removed since version 4.0)

DESCRIPTION

The function returns an array with information about the media object with the ID **MediaID**.



The previous form of the function that was used up to version 3.4 is accessible as "array IPS_GetMediaCompatibility(integer $MediaID)" since version 4.0 if the corresponding Special Switch is set.

EXAMPLE

```
// Since Version 4.0
print_r(IPS_GetMedia(45699));
/* returns e.g.:
Array
(
[MediaCRC] => E2D2C1D1
[MediaFile] => media\45699.bin
[MediaID] => 45699
[MediaIsAvailable] => 1
[MediaIsCached] =>
[MediaSize] => 8192
[MediaType] => 0
[MediaUpdated] => 1214421546
)
*/

print_r(IPS_GetMediaCompatibility(45699));
/* returns e.g.:
Array
(
[IsAvailable] => 1
[IsLinked] =>
[LastUpdate] => 1214421546
[MediaCRC] => E2D2C1D1
[MediaFile] => media\45699.bin
[MediaID] => 45699
[MediaSize] => 8192
[MediaType] => 0
[SendEvent] =>
)
*/



// Up to Version 3.4
print_r(IPS_GetMedia(45699));
/* returns e.g.:
Array
(
[IsAvailable] => 1
[IsLinked] =>
```

```
[LastUpdate] => 1214421546
[MediaCRC] => E2D2C1D1
[MediaFile] => media\45699.bin
[MediaID] => 45699
[MediaSize] => 8192
[MediaType] => 0
[SendEvent] =>
)
*/
```

IPS_GetMediaContent

Require: Symcon >= 3.1

```
string IPS_GetMediaContent (int $MediaID)
```

PARAMETERS

MediaID	ID of the media object whose content should be returned

RETURNS

Returns the content of the media object coded in Base64

DESCRIPTION

This function returns the Base64 coded content of the media object with the ID **MediaID**. The content is read from the file that is defined in the media object.



A useful example for this function is to provide the content to an external application that cannot access the file system via the JSON-RPC API.

EXAMPLE

```
$Content = base64_decode(IPS_GetMediaContent(12345));
```

IPS_GetMediaIDByFile

```
int IPS_GetMediaIDByFile (string $MediaPath)
```

PARAMETERS

MediaPath	Relative media path as seen from main program path

RETURNS

ID of the found media object, otherwise **FALSE** and a warning

DESCRIPTION

The function returns the ID of the media object with the path **MediaPath**. If no such object exists, the function returns **FALSE**.



This function generates a **warning** if the file was not found. The @ operator can be used to suppress these messages. The exact functionality of this operator is found in the PHP manual.

EXAMPLE

```
$MediaID = @IPS_GetMediaIDByFile("media/help.png");
if ($MediaID === false)
echo "Picture not found!";
else
echo "The Media ID is: ". $MediaID;
```

IPS_GetMediaIDByName

```
int IPS_GetMediaIDByName (string $MediaName, int $ParentID)
```

PARAMETERS

MediaName	Name of the searched media
ParentID	ID of the object whose children are searched for the media object

RETURNS

ID of the found media object, otherwise FALSE

DESCRIPTION

The function tries to determine the ID of the media object called **MediaName**, whose parent has the ID **ParentID**. It will use the ID of the first found media object whose name matches **MediaName**. If such a media object was found, its ID is returned, otherwise the function returns FALSE.
This function utilizes an additional criterion (ParentID) which restricts the search to the children of one object. Thus, the probability to find the right ID is increased considerably. However:



As names within IP-Symcon are **not unique** it is possible that the function returns a wrong ID.



This function generates a **warning** if the name was not found. The @ operator can be used to suppress these messages. The exact functionality of this operator is found in the PHP manual.

EXAMPLE

```
$MediaID = @IPS_GetMediaIDByName("MyPicture", 12345);
if ($MediaID === false)
echo "Picture not found!";
```

```
else
echo "The Media ID is: ". $MediaID;
```

IPS_GetMediaList

```
array IPS_GetMediaList ()
```

RETURNS

An array that contains all IDs of the media objects in IP Symcon as integers. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all registered media objects in the media pool. The IDs are listed in an array. If no media object exists, the array is empty.
The objects of all media types are listed. To restrict the array by a media type, the function IPS_GetMediaListByType can be used.

EXAMPLE

```
$allMediaObjects = IPS_GetMediaList();
print_r($allMediaObjects);

/* returns e.g.:
Array
(
[0] => 37659
[1] => 18326
etc. ...
*/
```

IPS_GetMediaListByType

```
array IPS_GetMediaListByType (int $MediaType)
```

PARAMETERS

MediaType

VALUE	CONSTANT	DESCRIPTION
0	MEDIATYPE_IPSVIEW	IPSView Form
1	MEDIATYPE_IMAGE	Image object
2	MEDIATYPE_SOUND	Sound object
3	MEDIATYPE_STREAM	Stream object
4	MEDIATYPE_CHART	Chart object
5	MEDIATYPE_DOCUMENT	Document object

RETURNS

An array of integer values of all IDs of the media objects with the type **MediaType** in IP Symcon. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all registered media objects in the media pool of the type **MediaType**. The IDs are listed in an array. If no corresponding media object exists, the array is empty.
The array only contains objects of the media type **MediaType**. To list all media objects, the function IPS_GetMediaList can be used.

EXAMPLE

```
$allImageObjects = IPS_GetMediaListByType(MEDIATYPE_IMAGE); // list
only Image objects
print_r($allImageObjects);

/* returns e.g.:
Array
(
[0] => 37659
[1] => 18326
```

```
etc. ...
)
*/
```

IPS_MediaExists

```
bool IPS_MediaExists (int $MediaID)
```

PARAMETERS

MediaID	ID of the media object

RETURNS

If the MediaID exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks if the media object with the ID **MediaID** exists.

EXAMPLE

```
if (IPS_MediaExists(34881))
echo "A media object with this ID exists!";
```

IPS_SendMediaEvent

```
bool IPS_SendMediaEvent (int $MediaID)
```

PARAMETERS

MediaID	ID of the media object

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sends a message that the media object with ID **MediaID** has changed. Size, CRC, time of the last update and availability of the media object are determined anew. This message is used for different purposes, e.g., to update an image/form. The function is only needed if the file was modified outside of IP-Symcon. When using IPS_SetMediaFile or IPS_SetMediaContent, the notification is sent automatically.

EXAMPLE

```
IPS_SendMediaEvent(12345);
```

IPS_SetMediaCached

```
bool IPS_SetMediaCached (int $MediaID, bool $CacheActivated)
```

PARAMETERS

MediaID	ID of the media object
CacheActivated	TRUE, if the media object should only be modified in the cache, otherwise FALSE

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function activates or deactivates the caching for a media object with the ID MediaID. If activated, the media object is only modified within the internal memory when the functions IPS_SetMediaContent or IPS_GetMediaContent are used. When IP-Symcon shuts down, the media object is automatically written to the disk and thus becomes persistent. When caching is deactivated, the content held in memory is first written to the file of the media object, so no changes are lost.



This function can be used to decrease the burden to SSDs or flash memories.

EXAMPLE

```
$MediaID = IPS_CreateMedia(MEDIATYPE_IMAGE); // Create image within the
media pool
IPS_SetMediaCached($MediaID, true);
// Caching is activated for the media object
// The object is read from the hard disk on the first access
// and is only processed in the internal memory afterwards.
```

IPS_SetMediaContent

Require: Symcon >= 3.1

```
bool IPS_SetMediaContent (int $MediaID, string $Content)
```

PARAMETERS

MediaID	ID of the media object whose content is set
Content	Base64 coded content

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the content of the media object with the ID **MediaID**. It uses the filename that was previously defined by IPS_SetMediaFile.
An error message is returned if the file name is not set or invalid or if the content is not Base64 coded.



This function should be used carefully as it overwrites the content without warning.

EXAMPLE

```
$MediaID = IPS_CreateMedia(MEDIATYPE_IMAGE); //Create image
IPS_SetMediaFile($MediaID, "BlackPixel.gif", False);
IPS_SetMediaContent($MediaID,
"R0lGODlhAQABAIAAAAUEBAAAACwAAAAAAQABAAACAkQBADs=");
```

IPS_SetMediaFile

```
bool IPS_SetMediaFile (int $MediaID, string $FileName, bool
$FileMustExist)
```

PARAMETERS

MediaID	ID of the media object
FileName	Filename/path of the file to link to
FileMustExist	TRUE, if existence should be checked, otherwise FALSE

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function binds the file name **FileName** to the media object with the ID **MediaID**. If the parameter **FileMustExist** is set, IP-Symcon checks if the given file exists. The allocated media file must match the media type of the media object.



The file name should be unique. It is not checked whether **FileName** is already bound to another media object.



It is absolutely necessary that these files are stored in the main folder "IP-Symcon" or one of its subfolders. Other directories are not possible due to security reasons.

EXAMPLE

```
$ImageFile = "Alarm symbol.png"; // Image file
$MediaID = IPS_CreateMedia(MEDIATYPE_IMAGE); // create the image in the
media pool
IPS_SetMediaFile($MediaID, $ImageFile, true); // bind the image in the
media pool with image file
```

Management of Modules

IPS_GetCompatibleModules	returns all modules that are compatible to a specific module
IPS_GetLibrary	returns extensive information about a library
IPS_GetLibraryList	returns a list of all libraries
IPS_GetLibraryModules	returns a list of modules in a library
IPS_GetModule	returns extensive information about a module
IPS_GetModuleList	returns a list of all modules
IPS_GetModuleListByType	returns a list of all modules of a given type
IPS_IsModuleCompatible	checks if two modules are compatible to each other
IPS_LibraryExists	Checks if the given library exists
IPS_ModuleExists	checks if the given module exists

IPS_GetCompatibleModules

```
array IPS_GetCompatibleModules (string $ModuleID)
```

PARAMETERS

ModuleID	ID of the module

RETURNS

An array of all IDs of modules that are valid parents to **ModuleID**, given as strings. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all modules in IP-Symcon that are valid parents to the module with the ID **ModuleID** (see IPS_IsModuleCompatible). The IDs are listed in an array of string values.

EXAMPLE

```
print_r(IPS_GetCompatibleModules("{57040540-4432-4220-8D2D-
4676B57E223D}"));

/* returns e.g.:
Array
(
[0] => {AC6C6E74-C797-40B3-BA82-F135D941D1A2}
[1] => {6179ED6A-FC31-413C-BB8E-1204150CF376}
[2] => {3CFF0FD9-E306-41DB-9B5A-9D06D38576C3}
[3] => {82347F20-F541-41E1-AC5B-A636FD3AE2D8}
etc. ...
)
*/
```

IPS_GetLibrary

```
array IPS_GetLibrary (string $LibraryID)
```

PARAMETERS

LibraryID	ID of the library

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
Author	string	Name of author
Build	integer	Serial number for versioning
Date	integer	Unix timestamp
LibraryID	string	LibraryID
Name	string	Name of library
URL	string	Internet address of the library
Version	string	Version of the library, e.g. "1.1"

DESCRIPTION

The function returns an array containing extensive information about the library with the GUID **LibraryID**.

EXAMPLE

```
print_r(IPS_GetLibrary("{4414D465-0A32-49AC-929A-D948CBCDE06E}"));

/* returns e.g.:
Array
(
[LibraryID] => {4414D465-0A32-49AC-929A-D948CBCDE06E}
[Author] => Symcon GmbH
[URL] => www.symcon.de
[Name] => SzenenSteuerung
[Version] => 1.1
[Build] => 0
[Date] => 0
```

```
)
*/
```

IPS_GetLibraryList

```
array IPS_GetLibraryList ()
```

RETURNS

An array of all the GUIDs of the libraries in IP-Symcon given as strings. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all available libraries in IP-Symcon. The IDs are listed in an array of strings.

EXAMPLE

```
print_r(IPS_GetLibraryList());

/* returns e.g.:
Array
(
[0] => {FF95B199-B3BD-424C-9AEF-3F004BC672B6}
[1] => {6EC74E99-C6FD-4E03-9195-E7BD90E6C07E}
[2] => {2CE84600-0A3C-438C-AC22-86439A1E5DF0}
etc. ...
)
*/
```

IPS_GetLibraryModules

```
array IPS_GetLibraryModules (string $LibraryID)
```

PARAMETERS

LibraryID	ID of the shown library

RETURNS

An array of all the GUIDs of the modules of a library in IP-Symcon, given as strings. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all modules in the library with the ID **LibraryID**. The IDs are listed in an array of strings.

EXAMPLE

```
print_r(IPS_GetLibraryModules("{7DC57F9A-C095-4CDE-A6F0-
2CB35A29A8FE}"));

/* returns e.g.:
Array
(
[0] => {57040540-4432-4220-8D2D-4676B57E223D}
[1] => {48FCFDC1-11A5-4309-BB0B-A0DB8042A969}
[2] => {56800073-A809-4513-9618-1C593EE1240C}
[3] => {2FD7576A-D2AD-47EE-9779-A502F23CABB3}
etc. ...
)
*/
```

IPS_GetModule

```
array IPS_GetModule (string $ModuleID)
```

PARAMETERS

ModuleID	ID of the module

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
Aliases	array	Array of alternative names (string) for the module
ChildRequirements	array	Array of GUIDs (string), which are expected by child objects
ParentRequirements	array	Array of GUIDs (string), which are expected by parent objects
Implemented	array	Array of GUIDs (string) offered by the module
LibraryID	string	LibraryID in which the module is included
Limit	integer	Maximum number of instances of this module (0: unlimited, -1: depends on the license)
ModuleID	string	ID of the module
ModuleName	string	Name of module
ModuleType	integer	Type of module 0 `MODULETYPE_CORE` Core 1 `MODULETYPE_IO` I/O 2 `MODULETYPE_SPLITTER` Splitter 3 `MODULETYPE_DEVICE` Device 4 `MODULETYPE_CONFIGURATOR` Configurator 5 `MODULETYPE_DISCOVERY` Discovery 6 `MODULETYPE_VISUALIZATION` Visualization
Prefix	string	Prefix of the module for calling the corresponding PHP functions (since Version 6.1)
Translation	array	Translations for ModuleName and Aliases (since Version 7.0)
URL	string	URL to the documentation website
Vendor	string	System/Manufacturer designation

DESCRIPTION

The function returns an array containing extensive information about the module with the ID ModuleID.

EXAMPLE

```
print_r(IPS_GetModule("{BAEA5454-4256-48AA-982B-538201A374D4}"));

/* returns e.g.:
Array
(
[ParentRequirements] => Array
(
[0] => {42DFD4E4-5831-4A27-91B9-6FF1B2960260}
)

[ChildRequirements] => Array
(
)

[Implemented] => Array
(
[0] => {8A4D3B17-F8D7-4905-877F-9E69CEC3D579}
)

[Vendor] => KNX
[Aliases] => Array
(
[0] => DPT 013.x - 4-Byte Signed Value
)

[Translation] => Array
(
[de] => Array
(
[KNX DPT 13] => KNX DPT 13
[DPT 013.x - 4-Byte Signed Value] => DPT 013.x - 4-Byte
vorzeichenbehaftet Wert
)
)

[URL] => https://www.symcon.de/service/dokumentation/modulreferenz/knx/
[LibraryID] => {0945206A-47AA-4FDD-9093-99051E410E82}
[ModuleID] => {BAEA5454-4256-48AA-982B-538201A374D4}
[ModuleName] => KNX DPT 13
[ModuleType] => 3
)
*/
```

IPS_GetModuleList

```
array IPS_GetModuleList ()
```

RETURNS

An array of all the GUIDs of the modules in IP-Symcon given as strings. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the IDs of all available modules in IP-Symcon. The IDs are listed in an array of strings.

EXAMPLE

```
print_r(IPS_GetModuleList());

/* returns e.g.:
Array
(
[0] => {2D8B0172-166C-45B0-B979-61A787809E22}
[1] => {CCC52C33-D77A-4AE9-A6CF-462F152532A0}
[2] => {11842D31-3CA4-4F1B-BBA9-E4A0FE1873AF}
[3] => {4CB91589-CE01-4700-906F-26320EFCF6C4}
etc. ...
)
*/

//Output of all module names with GUID
foreach(IPS_GetModuleList() as $guid)
{
$module = IPS_GetModule($guid);
// Limit modules to Core library
if($module['LibraryID'] == "{0945206A-47AA-4FDD-9093-99051E410E82}") {
$pair[$module['ModuleName']] = $guid;
}
}
ksort($pair);
foreach($pair as $key=>$guid)
{
echo $key." = ".$guid."\n";
}

/*
ALL2x = {D805EC4C-7D17-4E84-98D3-A441AA71ACA3}
```

```
ALL3690 = {BBD04875-8CC5-412A-B848-B2AB6F08C425}
ALL3691 = {07CAE61E-01FA-4086-A2D1-55C442D8C8BF}
ALL4000 = {2D8B0172-166C-45B0-B979-61A787809E22}
ALLUniversal = {FB6B8DD0-CD8E-4CE7-8613-D7BD6436720A}
Archive Control = {43192F0B-135B-4CE7-A0A7-1475603F3060}
CMI = {AAAB8DD0-CD8E-4CE7-8613-D913371701AE}
Calendar Control = {1B9A5AF2-F2D2-4827-A13D-FFA6AB350371}
Client Socket = {3CFF0FD9-E306-41DB-9B5A-9D06D38576C3}
Connect Control = {9486D575-BE8C-4ED8-B5B5-20930E26DE6F}
Cutter = {AC6C6E74-C797-40B3-BA82-F135D941D1A2}
DMX Gateway = {B1E43BF6-770A-4FD7-B4FE-6D265F93746B}
DMX Output = {E19C2E41-7347-4A3B-B7D9-A9A88E0D133E}
DMX RGB = {1AF36760-644E-4DA9-8776-01BA6DF6178E}
DNSSD Control = {780B2D48-916C-4D59-AD35-5A429B2355A5}
Dummy Module = {485D0419-BE97-4548-AA9C-C083EB82E61E}
EnOcean Configurator = {9BA823CF-83C6-4A5F-DBC2-295CA971833E}
EnOcean D20500 = {8DFBE8BC-FC2D-61F0-375C-C8C2A13CDC1A}
EnOcean D20501 = {927296DC-B74F-F0B8-2C4E-6B98236EDD94}
EnOcean D20502 = {B4D91880-71C1-626B-94D7-521565495264}
EnOcean Discovery = {29E1D618-F3CB-6868-59C6-4DCD02F8C78D}
EnOcean EEP A50201RX = {432FF87E-4497-48D6-8ED9-EE7104A50201}
EnOcean EEP A50202RX = {432FF87E-4497-48D6-8ED9-EE7104A50202}
EnOcean EEP A50203RX = {432FF87E-4497-48D6-8ED9-EE7104A50203}
EnOcean EEP A50204RX = {432FF87E-4497-48D6-8ED9-EE7104A50204}
EnOcean EEP A50205RX = {432FF87E-4497-48D6-8ED9-EE7104A50205}
EnOcean EEP A50206RX = {432FF87E-4497-48D6-8ED9-EE7104A50206}
EnOcean EEP A50207RX = {432FF87E-4497-48D6-8ED9-EE7104A50207}
EnOcean EEP A50208RX = {432FF87E-4497-48D6-8ED9-EE7104A50208}
EnOcean EEP A50209RX = {432FF87E-4497-48D6-8ED9-EE7104A50209}
EnOcean EEP A5020ARX = {432FF87E-4497-48D6-8ED9-EE7104A5020A}
EnOcean EEP A5020BRX = {432FF87E-4497-48D6-8ED9-EE7104A5020B}
EnOcean EEP A50210RX = {432FF87E-4497-48D6-8ED9-EE7104A50210}
EnOcean EEP A50211RX = {432FF87E-4497-48D6-8ED9-EE7104A50211}
EnOcean EEP A50212RX = {432FF87E-4497-48D6-8ED9-EE7104A50212}
EnOcean EEP A50213RX = {432FF87E-4497-48D6-8ED9-EE7104A50213}
EnOcean EEP A50214RX = {432FF87E-4497-48D6-8ED9-EE7104A50214}
EnOcean EEP A50215RX = {432FF87E-4497-48D6-8ED9-EE7104A50215}
EnOcean EEP A50216RX = {432FF87E-4497-48D6-8ED9-EE7104A50216}
EnOcean EEP A50217RX = {432FF87E-4497-48D6-8ED9-EE7104A50217}
EnOcean EEP A50218RX = {432FF87E-4497-48D6-8ED9-EE7104A50218}
EnOcean EEP A50219RX = {432FF87E-4497-48D6-8ED9-EE7104A50219}
EnOcean EEP A5021ARX = {432FF87E-4497-48D6-8ED9-EE7104A5021A}
EnOcean EEP A5021BRX = {432FF87E-4497-48D6-8ED9-EE7104A5021B}
EnOcean EEP A50220RX = {432FF87E-4497-48D6-8ED9-EE7104A50220}
EnOcean EEP A50230RX = {432FF87E-4497-48D6-8ED9-EE7104A50230}
EnOcean EEP A50401RX = {432FF87E-4497-48D6-8ED9-EE7104A50401}
EnOcean EEP A50402RX = {432FF87E-4497-48D6-8ED9-EE7104A50402}
EnOcean EEP A50403RX = {432FF87E-4497-48D6-8ED9-EE7104A50403}
EnOcean EEP A50601RX = {432FF87E-4497-48D6-8ED9-EE7104A50601}
EnOcean EEP A50602RX = {432FF87E-4497-48D6-8ED9-EE7104A50602}
EnOcean EEP A50603RX = {432FF87E-4497-48D6-8ED9-EE7104A50603}
EnOcean EEP A50701RX = {432FF87E-4497-48D6-8ED9-EE7104A50701}
EnOcean EEP A50702RX = {432FF87E-4497-48D6-8ED9-EE7104A50702}
EnOcean EEP A50703RX = {432FF87E-4497-48D6-8ED9-EE7104A50703}
EnOcean EEP A50801RX = {432FF87E-4497-48D6-8ED9-EE7104A50801}
EnOcean EEP A50802RX = {432FF87E-4497-48D6-8ED9-EE7104A50802}
EnOcean EEP A50803RX = {432FF87E-4497-48D6-8ED9-EE7104A50803}
EnOcean EEP A50901RX = {432FF87E-4497-48D6-8ED9-EE7104A50901}
EnOcean EEP A50902RX = {432FF87E-4497-48D6-8ED9-EE7104A50902}
```

```
EnOcean EEP A50904RX = {432FF87E-4497-48D6-8ED9-EE7104A50904}
EnOcean EEP A50905RX = {432FF87E-4497-48D6-8ED9-EE7104A50905}
EnOcean EEP A50906RX = {432FF87E-4497-48D6-8ED9-EE7104A50906}
EnOcean EEP A50907RX = {432FF87E-4497-48D6-8ED9-EE7104A50907}
EnOcean EEP A50908RX = {432FF87E-4497-48D6-8ED9-EE7104A50908}
EnOcean EEP A51001RX = {432FF87E-4497-48D6-8ED9-EE7104A51001}
EnOcean EEP A51002RX = {432FF87E-4497-48D6-8ED9-EE7104A51002}
EnOcean EEP A51003RX = {432FF87E-4497-48D6-8ED9-EE7104A51003}
EnOcean EEP A51004RX = {432FF87E-4497-48D6-8ED9-EE7104A51004}
EnOcean EEP A51005RX = {432FF87E-4497-48D6-8ED9-EE7104A51005}
EnOcean EEP A51006RX = {432FF87E-4497-48D6-8ED9-EE7104A51006}
EnOcean EEP A51007RX = {432FF87E-4497-48D6-8ED9-EE7104A51007}
EnOcean EEP A51008RX = {432FF87E-4497-48D6-8ED9-EE7104A51008}
EnOcean EEP A51009RX = {432FF87E-4497-48D6-8ED9-EE7104A51009}
EnOcean EEP A5100ARX = {432FF87E-4497-48D6-8ED9-EE7104A5100A}
EnOcean EEP A5100BRX = {432FF87E-4497-48D6-8ED9-EE7104A5100B}
EnOcean EEP A5100CRX = {432FF87E-4497-48D6-8ED9-EE7104A5100C}
EnOcean EEP A5100DRX = {432FF87E-4497-48D6-8ED9-EE7104A5100D}
EnOcean EEP A51010RX = {432FF87E-4497-48D6-8ED9-EE7104A51010}
EnOcean EEP A51011RX = {432FF87E-4497-48D6-8ED9-EE7104A51011}
EnOcean EEP A51012RX = {432FF87E-4497-48D6-8ED9-EE7104A51012}
EnOcean EEP A51013RX = {432FF87E-4497-48D6-8ED9-EE7104A51013}
EnOcean EEP A51014RX = {432FF87E-4497-48D6-8ED9-EE7104A51014}
EnOcean EEP A51015RX = {432FF87E-4497-48D6-8ED9-EE7104A51015}
EnOcean EEP A51016RX = {432FF87E-4497-48D6-8ED9-EE7104A51016}
EnOcean EEP A51017RX = {432FF87E-4497-48D6-8ED9-EE7104A51017}
EnOcean EEP A51018RX = {432FF87E-4497-48D6-8ED9-EE7104A51018}
EnOcean EEP A51019RX = {432FF87E-4497-48D6-8ED9-EE7104A51019}
EnOcean EEP A5101ARX = {432FF87E-4497-48D6-8ED9-EE7104A5101A}
EnOcean EEP A5101BRX = {432FF87E-4497-48D6-8ED9-EE7104A5101B}
EnOcean EEP A5101CRX = {432FF87E-4497-48D6-8ED9-EE7104A5101C}
EnOcean EEP A5101DRX = {432FF87E-4497-48D6-8ED9-EE7104A5101D}
EnOcean EEP A5101FRX = {432FF87E-4497-48D6-8ED9-EE7104A5101F}
EnOcean EEP A51020RX = {432FF87E-4497-48D6-8ED9-EE7104A51020}
EnOcean EEP A51021RX = {432FF87E-4497-48D6-8ED9-EE7104A51021}
EnOcean EEP A51022RX = {BBE179DE-1241-CB89-6803-492049D5546B}
EnOcean EEP A51023RX = {9880FC1A-AD89-5749-12A0-AF6C99D50B28}
EnOcean EEP A51101RX = {432FF87E-4497-48D6-8ED9-EE7104A51101}
EnOcean EEP A51102RX = {432FF87E-4497-48D6-8ED9-EE7104A51102}
EnOcean EEP A51103RX = {432FF87E-4497-48D6-8ED9-EE7104A51103}
EnOcean EEP A51104RX = {432FF87E-4497-48D6-8ED9-EE7104A51104}
EnOcean EEP A51200RX = {432FF87E-4497-48D6-8ED9-EE7104A51200}
EnOcean EEP A51201RX = {432FF87E-4497-48D6-8ED9-EE7104A51201}
EnOcean EEP A51202RX = {432FF87E-4497-48D6-8ED9-EE7104A51202}
EnOcean EEP A51203RX = {432FF87E-4497-48D6-8ED9-EE7104A51203}
EnOcean EEP A51301RX = {432FF87E-4497-48D6-8ED9-EE7104A51301}
EnOcean EEP A51302RX = {432FF87E-4497-48D6-8ED9-EE7104A51302}
EnOcean EEP A51303RX = {432FF87E-4497-48D6-8ED9-EE7104A51303}
EnOcean EEP A51304RX = {432FF87E-4497-48D6-8ED9-EE7104A51304}
EnOcean EEP A51305RX = {432FF87E-4497-48D6-8ED9-EE7104A51305}
EnOcean EEP A51401RX = {432FF87E-4497-48D6-8ED9-EE7104A51401}
EnOcean EEP A51402RX = {432FF87E-4497-48D6-8ED9-EE7104A51402}
EnOcean EEP A51403RX = {432FF87E-4497-48D6-8ED9-EE7104A51403}
EnOcean EEP A51404RX = {432FF87E-4497-48D6-8ED9-EE7104A51404}
EnOcean EEP A51405RX = {432FF87E-4497-48D6-8ED9-EE7104A51405}
EnOcean EEP A51406RX = {432FF87E-4497-48D6-8ED9-EE7104A51406}
EnOcean EEP A51407RX = {B3A8A26C-FF2F-491E-9279-E981959293C0}
EnOcean EEP A51408RX = {2C9E6E3A-319F-7ADF-024C-E8DEA7209A52}
EnOcean EEP A51409RX = {29CBDDA5-6CA6-4C17-9360-1979DCBD1E43}
```

```
EnOcean EEP A5140ARX = {1F3E63B6-F354-44FF-98DE-BEB63F36310C}
EnOcean EEP A52006 = {D25790E7-DC98-7EE3-4D74-2F0484392795}
EnOcean EEP A52012RX = {432FF87E-4497-48D6-8ED9-EE7104A52012}
EnOcean EEP A52012TX = {432FF87E-4497-48D6-8ED9-AA7104A52012}
EnOcean EEP A53001RX = {432FF87E-4497-48D6-8ED9-EE7104A53001}
EnOcean EEP A53002RX = {432FF87E-4497-48D6-8ED9-EE7104A53002}
EnOcean EEP A53003RX = {432FF87E-4497-48D6-8ED9-EE7104A53003}
EnOcean EEP A53004RX = {432FF87E-4497-48D6-8ED9-EE7104A53004}
EnOcean EEP A53701RX = {432FF87E-4497-48D6-8ED9-EE7104A53701}
EnOcean EEP A53808RX = {432FF87E-4497-48D6-8ED9-EE7104A53808}
EnOcean EEP A53809RX = {432FF87E-4497-48D6-8ED9-EE7104A53809}
EnOcean EEP D20100 = {8328F257-35B7-4034-AA23-20B0A30CAF11}
EnOcean EEP D20101 = {5F30BDCD-487F-44EE-9795-82C5DD429F2D}
EnOcean EEP D20102 = {C3825266-A6DE-44B6-97B4-C0BCFB6BD52B}
EnOcean EEP D20103 = {EAF7026C-A0DB-4B25-8DFD-D4FAB7A3EFCF}
EnOcean EEP D20104 = {5B38D44B-A570-4381-B719-8BFB58387AA1}
EnOcean EEP D20105 = {19FDCECC-0735-4ADA-8304-3AF53C919668}
EnOcean EEP D20106 = {1881B19B-A03C-437D-934D-28C598F13C39}
EnOcean EEP D20107 = {06DFB7CC-9945-4E1D-B08D-C505F4A7D2B4}
EnOcean EEP D20108 = {C885AB2C-7D83-4274-9420-EFEA5B8B0908}
EnOcean EEP D20109 = {BFDADCB0-BAB5-45BC-BF7A-1D19EA0FF31D}
EnOcean EEP D2010A = {6A37AC89-EFF0-40D8-88A1-8B24478FEE76}
EnOcean EEP D2010B = {F054B2EB-D812-47ED-B73C-22FBC0ADA667}
EnOcean EEP D2010C = {0389DA40-1B2A-446C-9448-03DED7977DC0}
EnOcean EEP D2010D = {744C3ACE-4680-4E1D-966F-CC959B25DFD6}
EnOcean EEP D2010E = {7E11E659-98E1-45A1-A9DB-29D337423C92}
EnOcean EEP D2010F = {AE62BF01-6C90-4FC2-8E1E-75EB5516BCF0}
EnOcean EEP D20110 = {699264EC-BEF1-4426-B501-687EA30C03B1}
EnOcean EEP D20111 = {21DDC756-E3CD-4B2D-B2F4-86092918A832}
EnOcean EEP D20112 = {47804DFF-5534-4256-8938-2C5D018736A3}
EnOcean EEP D20113 = {783B1089-4A50-4973-A522-C5288B9E1909}
EnOcean EEP D20114 = {A08C2716-CF6A-472C-A303-5C37C57A65BB}
EnOcean EEP D20300 = {8DA41914-45A6-8F0C-C935-178F76A36C42}
EnOcean EEP D2030A = {B24CF984-5CD7-B0D7-9661-41E6F2CEA0E7}
EnOcean EEP D20310 = {C14DBE59-E9B6-C9B7-7DEF-D98B8ADBA6F3}
EnOcean EEP D20601 = {6E5F79FE-3F0B-4CCE-8786-E8DD8893185B}
EnOcean EEP D20700 = {832ABD55-1173-B194-EB3F-D23681113A01}
EnOcean EEP D21101 = {3B1E2E8A-7C09-F58C-2841-3EF2D7C67F7A}
EnOcean EEP D21102 = {ED6AB19B-5A4A-80FD-BC4B-C07EE7F9915A}
EnOcean EEP D21103 = {7973060A-0241-E1E5-6E4B-CACBB08ACA76}
EnOcean EEP D21104 = {CC1482D1-6AD9-2199-C5E5-AD4EB6926455}
EnOcean EEP D21105 = {360F32BD-D9D5-5233-966A-63EAF54F671C}
EnOcean EEP D21106 = {8B2F1F85-A51C-0606-7CF3-8F980FEADB12}
EnOcean EEP D21107 = {F84B8E1B-B0DF-9383-1F6E-56EFAFB405A8}
EnOcean EEP D21108 = {1828E2A3-D83B-1433-46BF-AD0F11FF07DC}
EnOcean EEP D21430 = {1858BB3E-78BD-B090-7798-F1C0C26D912E}
EnOcean EEP D21500 = {75CA6030-43B9-8CF6-5901-E2DEF86E5249}
EnOcean EEP D23200 = {F60551AF-5D2B-4560-8BDD-7D1474C7A490}
EnOcean EEP D23201 = {C3B79AA9-54DB-4AE4-8ED4-3F954C723465}
EnOcean EEP D23202 = {0A03908E-694E-4CB1-BD35-AF7777504BFE}
EnOcean EEP D2A001 = {EA335B84-7E0A-F982-32CE-B50B60FC8E39}
EnOcean EEP D50001RX = {432FF87E-4497-48D6-8ED9-EE7104D50001}
EnOcean EEP F60101RX = {B9ED3CC4-A354-0420-A7D3-9AFC6C721FB3}
EnOcean EEP F60201RX = {432FF87E-4497-48D6-8ED9-EE7104F60201}
EnOcean EEP F60202RX = {432FF87E-4497-48D6-8ED9-EE7104F60202}
EnOcean EEP F60203RX = {432FF87E-4497-48D6-8ED9-EE7104F60203}
EnOcean EEP F60204RX = {432FF87E-4497-48D6-8ED9-EE7104F60204}
EnOcean EEP F60301RX = {432FF87E-4497-48D6-8ED9-EE7104F60301}
EnOcean EEP F60302RX = {432FF87E-4497-48D6-8ED9-EE7104F60302}
```

```
EnOcean EEP F60401RX = {432FF87E-4497-48D6-8ED9-EE7104F60401}
EnOcean EEP F60402RX = {432FF87E-4497-48D6-8ED9-EE7104F60402}
EnOcean EEP F60500RX = {74DB7CEE-18F9-478D-9FBC-CA857F807C50}
EnOcean EEP F60501RX = {432FF87E-4497-48D6-8ED9-EE7104F60501}
EnOcean EEP F60502RX = {8D42EFEE-119C-4CA8-BED2-66D18921749E}
EnOcean EEP F61000RX = {432FF87E-4497-48D6-8ED9-EE7104F61000}
EnOcean EEP F61001RX = {C31A7F85-4E76-E9C9-F920-2953DE737560}
EnOcean EltakoDimmer = {48909406-A2B9-4990-934F-28B9A80CD079}
EnOcean EltakoFABH130 = {03AEBCA9-1140-969E-7C72-728A1F30CD55}
EnOcean EltakoFABH65S = {0E8E3267-5CFA-4A52-AB3E-899170D16116}
EnOcean EltakoFAFT60 = {0BE195DC-6002-4D99-A566-3B9B0B57FAD6}
EnOcean EltakoFAH60 = {AF827EB8-08A3-434D-9690-424AFF06C698}
EnOcean EltakoFHK14 = {7C25F5A6-ED34-4FB4-8A6D-D49DFE636CDC}
EnOcean EltakoFKC = {40C99CC9-EC04-49C8-BB9B-73E21B6FDDDD}
EnOcean EltakoFRW = {40C99CC9-EC04-49C8-BB9B-73E21B6FEEEE}
EnOcean EltakoFSS12 = {7124C1BC-B260-4C5E-BF00-B38D3C7B5CB7}
EnOcean EltakoFTR65 = {40C99CC9-EC04-49C8-BB9B-73E21B6FFF65}
EnOcean EltakoFWS61 = {9E4572C0-C306-4F00-B536-E75B4950F094}
EnOcean EltakoFZS = {40C99CC9-EC04-49C8-BB9B-73E21B6FCCCC}
EnOcean EltakoRGBW = {71072C70-7ADF-6CC5-8D6F-7368132584F8}
EnOcean EltakoShutter = {1463CAE7-C7D5-4623-8539-DD7ADA6E92A9}
EnOcean EltakoSwitch = {FD46DA33-724B-489E-A931-C00BFD0166C9}
EnOcean Gateway = {A52FEFE9-7858-4B8E-A96E-26E15CB944F7}
EnOcean Hoppe = {1C8D7E80-3ED1-4117-BB53-9C5F61B1BEF3}
EnOcean Opus = {9B1F32CD-CD74-409A-9820-E5FFF064449A}
EnOcean PTM200 = {40C99CC9-EC04-49C8-BB9B-73E21B6FA265}
EnOcean PTM200RX = {63484585-F8AD-4780-BAFD-3C0353641046}
EnOcean RCM100 = {8492CEAF-ED62-4634-8A2F-B09A7CEDDE5B}
EnOcean STM100 = {FA1479DE-C0C1-433D-98BC-EA7C298D1AA5}
EnOcean STM250 = {B01DE819-EA69-4FC1-91AB-4D9FF8D55370}
EnOcean Shutter = {1463CAE7-C7D5-4623-8539-DD7ADA6E92AA}
EnOcean SmartDrive MX = {004A19B5-6E60-41A3-813C-7653ABAB6116}
EnOcean Thermokon SR0x = {B4249BC6-5BA8-45E3-B506-86680935D4EE}
EnOcean Thermokon Thanos = {54E114A3-F9B6-4747-A07E-D2A064629627}
EnOcean Valve Actuator = {004A19B5-6E60-41A3-813C-A11DC00B6116}
EnOcean alphaEOS SENSE TF-H = {F98CD462-0147-4A79-A0DF-448580E01629}
Event Control = {ED573B53-8991-4866-B28C-CBE44C59A2DA}
FHT = {A89F8DFA-A439-4BF1-B7CB-43D047208DDD}
FHZ = {57040540-4432-4220-8D2D-4676B57E223D}
FS10 = {6D508C91-F197-44A9-A1AB-A27F97A18A5F}
FS10 Gateway = {753E7267-7558-49D3-ACFB-86755C28318D}
FS20 = {48FCFDC1-11A5-4309-BB0B-A0DB8042A969}
FS20EX = {56800073-A809-4513-9618-1C593EE1240C}
HID = {E6D7692A-7F4C-441D-827B-64062CFE1C02}
HMS = {2FD7576A-D2AD-47EE-9779-A502F23CABB3}
HTTP Client = {4CB91589-CE01-4700-906F-26320EFCF6C4}
Heating Control = {FF7AF0F4-D616-4EF5-B73A-D49C56CF4C92}
Heating Control Legacy = {3F52FA69-77F5-4DE6-8B2A-347452AC5F8F}
HomeMatic Configurator = {5214C3C6-91BC-4FE1-A2D9-A3920261DA74}
HomeMatic Device = {EE4A81C6-5C90-4DB7-AD2F-F6BBD521412E}
HomeMatic Discovery = {3718244C-71A2-B20D-F754-DF5C79340AB4}
HomeMatic Socket = {A151ECE9-D733-4FB9-AA15-7F7DD10C58AF}
IMAP = {CABFCCA1-FBFF-4AB7-B11B-9879E67E152F}
IPS-868 AD = {134A1CFF-3CAA-4D2E-8517-F9BFDE7555C9}
IPS-868 Accelerometer = {304B6E9A-A518-43C6-80E8-D066879E7FD1}
IPS-868 AirQuality = {2B2842ED-0BB9-45FE-934C-1DFE07E1A724}
IPS-868 Configurator = {0A3CCE10-7613-1C54-0721-57023C98EFDA}
IPS-868 Counter = {134A1CFF-3CAA-4D2E-8517-F9BFDE7544C9}
IPS-868 DA = {134A1CFF-3CAA-4D2E-8517-F9BFDE7566C9}
```

```
IPS-868 Data = {984FFE86-71BE-48D3-943E-45277BECB52D}
IPS-868 Discovery = {F2B8A368-F9F2-7919-B299-79DC5B64B8A4}
IPS-868 Display = {9542B617-C603-4E8A-BFDE-DD69663F0226}
IPS-868 Display Input = {FBF6F86C-4901-4957-8D08-2B2CBC1DF71A}
IPS-868 Display Output = {C8CF38E9-BEAA-4149-A9ED-C8AAE04E8A6D}
IPS-868 Gateway = {995946C3-7995-48A5-86E1-6FB16C3A0F8A}
IPS-868 Input = {5509B26C-F2F7-428F-973B-FD5D07C80111}
IPS-868 Level = {C7A069AB-E2BE-4AD2-BB51-A7B52D6FAC4E}
IPS-868 Servo = {D2C92D2D-FC07-4545-AEBA-4294FD9D3035}
IPS-868 Servo Input = {7D07671F-65FF-4D9A-B521-9F0964DD9399}
IPS-868 Stripe = {7AF52E55-4994-44AE-9DB8-752895F1EB76}
IPS-868 Stripe Input = {26B22717-924E-4D58-B091-C98E49812238}
IPS-868 Thermo = {95D715AA-AB5A-4423-AD7B-5081EB542971}
IPS-868 Tracker = {50460BD9-FA93-4DE0-976A-55208C90DF15}
IPS-868 WatchDogTimer = {9F88A462-BFDA-47C5-990C-B9E20D2F861E}
IRTrans Device = {899DBC68-0147-4528-B95D-D43C568D4E56}
IRTrans Gateway = {0F0F74EF-2304-4C23-840F-EC1C5B9A9A82}
Image Grabber = {5A5D5DBD-53AB-4826-8B09-71E9E4E981E5}
JSON Decoder = {9F976F04-7DC4-BB2D-38CA-67C1AFC84C7A}
KNX Configurator = {33765ABB-CFA5-40AA-89C0-A7CEA89CFE7A}
KNX DPT 1 = {F3058AB2-AFDC-4479-A053-5F4599DF6F5B}
KNX DPT 10 = {2D26EF69-C3BC-4738-879D-FB493DD166BF}
KNX DPT 11 = {4A5AE804-59E2-41EA-ABCE-3BC80F33543C}
KNX DPT 12 = {AD992707-B7DC-4255-96DE-B70057CEC989}
KNX DPT 13 = {BAEA5454-4256-48AA-982B-538201A374D4}
KNX DPT 14 = {E8C3C9ED-A6FA-45FF-A942-A206FEEE360C}
KNX DPT 15 = {16B72DE7-3C8B-4709-8819-6A260E890A12}
KNX DPT 16 = {2146FF76-FAE6-46B6-9630-73E5F3FAE190}
KNX DPT 17 = {E8A0352C-B13F-4691-AAF5-586DE502569E}
KNX DPT 18 = {8B2A23B9-6B96-4F76-9EC8-5365BA7E9441}
KNX DPT 19 = {8456D139-6771-4821-8A37-9304BEAA1BFD}
KNX DPT 2 = {434A6354-235F-4484-8883-9FF30616E3FB}
KNX DPT 20 = {C9304609-D1CD-4F07-BE7C-E5546A91A5DF}
KNX DPT 200 = {04ED3286-C982-436F-A01E-8A03C2E976A8}
KNX DPT 201 = {5F14B9D5-297D-4C25-9E45-92BF7B7C3F02}
KNX DPT 202 = {22726542-A2B6-4FBB-BB98-BA5927932C23}
KNX DPT 203 = {3243B56D-DE90-4BED-8E07-E4DEE777C6AF}
KNX DPT 204 = {06E5F74A-BC1F-4F69-BBE6-8AD663058CD9}
KNX DPT 205 = {28837D84-085F-4701-ADA0-CF9A8F49F591}
KNX DPT 206 = {F5E3C9F5-A0E8-4919-BB38-0551AE5886A7}
KNX DPT 207 = {EE68B239-8E3A-4F54-9501-96CB4B2700A4}
KNX DPT 209 = {D9A125D0-8D82-4808-A024-0FC5E82799D6}
KNX DPT 21 = {8941E278-1901-478D-AFC8-807E3515DD28}
KNX DPT 210 = {68E7C9CE-724E-4CBE-8206-A61AD9A6A328}
KNX DPT 211 = {0636647E-EB02-4EE3-857E-0B69C0E8AA3D}
KNX DPT 212 = {4D8AD2DA-5669-4BEB-B9C7-983DA48A7777}
KNX DPT 213 = {DEA88D19-78B6-4D51-89FF-E3958151863D}
KNX DPT 214 = {AC33D286-FBBC-40BE-91CC-C470FB896C04}
KNX DPT 215 = {619D8815-1E90-4A7A-B51D-14DCB2FCE0E0}
KNX DPT 216 = {F5339A58-E891-4A22-AF29-3C801D3092EE}
KNX DPT 217 = {9CF216A5-370F-4BD9-92F0-B7BF87933A76}
KNX DPT 218 = {DB54CF66-4ED5-4457-A1EF-A3DF52EC135F}
KNX DPT 219 = {7EA00021-4300-4B9F-A274-863E1DD41695}
KNX DPT 22 = {DD0F4EF5-3C71-44F1-A346-397D34809D30}
KNX DPT 220 = {14BBE7DF-8442-4ED8-9D7D-C00B3E1B892A}
KNX DPT 221 = {3C16BD4B-45BF-435A-84DA-D7908F2D6140}
KNX DPT 222 = {3AD960E1-E129-4D96-BFAC-696AC149C629}
KNX DPT 223 = {EA6A0F52-ECEF-46D2-A942-96DDED471732}
KNX DPT 224 = {7245A626-1864-4834-A181-EE4FB4AD50DC}
```

```
KNX DPT 225 = {CC6C64D0-2F48-4497-A6A6-2120BAE5CD52}
KNX DPT 229 = {3F20B758-71DD-4C0B-A169-D4B59B023555}
KNX DPT 23 = {7CA94B7E-6460-4D68-8568-EE4B0912C5BF}
KNX DPT 230 = {88546CFE-1265-494C-B80C-1A3A6E1E671D}
KNX DPT 231 = {44A18680-880B-4ACC-9E84-8FB79FB9D727}
KNX DPT 232 = {DE2E3FBF-1588-4293-B76C-C90446D9E5F1}
KNX DPT 234 = {C667FC98-9FE4-4953-87ED-2B75F59704A4}
KNX DPT 235 = {960A48AD-36AB-4174-A298-F31E99014738}
KNX DPT 236 = {8A57A40D-AE55-42D5-BF33-4F1E0DDA1F8C}
KNX DPT 237 = {E3E1F95B-3356-4F6A-A419-9486125E023A}
KNX DPT 238 = {117D49A5-7D8F-4509-8669-F44972F072DD}
KNX DPT 239 = {F8AE906F-76E2-4464-96DA-B9A6D19049C9}
KNX DPT 240 = {CC5F383D-975B-44E9-BC18-C64FA2698CC0}
KNX DPT 241 = {A26BC82A-5953-4385-9FAA-5A5D3814D2B5}
KNX DPT 242 = {F565EF9F-30A5-4E1C-BA10-7BC15AB4EFB4}
KNX DPT 249 = {34B83F38-03F2-4D15-9031-70197CE9456E}
KNX DPT 25 = {9FB4BD67-5E67-4E94-B8C8-71D609048BBB}
KNX DPT 251 = {9165E3E2-8ED2-4152-BEFF-97793823FE4D}
KNX DPT 26 = {135F8FCF-DAFC-46AA-85EC-94C911FD21DF}
KNX DPT 27 = {EDE5D3E1-FB5A-48EE-A944-0A4ED0FBACC1}
KNX DPT 28 = {3E175751-4B84-4E8E-98C1-BF0D12835F2C}
KNX DPT 29 = {AA9ED268-3B37-4962-85BB-CBC73CB11618}
KNX DPT 3 = {EB931B1A-3A4A-47E9-AC67-D2A0D285B60C}
KNX DPT 30 = {2E7AADF5-F1DF-476D-B207-22D204C4AD28}
KNX DPT 31 = {5AD8B9C3-5A87-4426-B9A1-C7935022A680}
KNX DPT 4 = {6891595A-BF7C-4F4C-8A3B-7EA7346C8657}
KNX DPT 5 = {EBD0EE8D-DFA5-449F-BBE4-49CFC5F0EEB4}
KNX DPT 6 = {BAC22632-D6ED-4D88-82EA-726D700C4F2A}
KNX DPT 7 = {DCAFED37-B20B-4EEB-AA44-799C9B804557}
KNX DPT 8 = {7852DBDE-C68A-4A3E-88AF-155B99605371}
KNX DPT 9 = {95CDE37D-72CE-4360-BB91-573757C274A0}
KNX Discovery = {3519D734-79C9-4E7A-9E9B-91B6374D1C10}
KNX EIS Group = {D62B95D3-0C5E-406E-B1D9-8D102E50F64B}
KNX Gateway = {1C902193-B044-43B8-9433-419F09C641B8}
KNX RGB = {4D7F7548-0979-4ABD-9BB3-81F9477C0903}
KNX RGBW = {81F54858-72B1-4C2C-8CE3-7E00A3168378}
KNX Shutter = {24A9D68D-7B98-4D74-9BAE-3645D435A9EF}
KS300 = {9D21F700-6F67-4FBB-ACC2-AA42420A0486}
LCN Configurator = {0F64973F-4669-4272-BDB3-6338D0350269}
LCN Data = {A26E7E5A-A7C5-4063-8BE0-ED8BB26F8411}
LCN Display = {A26E7E5A-A7C5-4063-8BE0-ED8BB26F8522}
LCN Gateway = {9BDFC391-DEFF-4B71-A76B-604DBA80F207}
LCN Module = {0E31FED6-E465-4621-95D4-AAF2683C41EC}
LCN RGBW = {839D1208-D4BB-4FA7-86FA-AA9FBF96ED17}
LCN Shutter = {C81E019F-6341-4748-8644-1C29D99B813E}
LCN ShutterMotor = {C81E019F-6341-4748-8644-1C29D99B813F}
LCN Unit = {2D871359-14D8-493F-9B01-26432E3A710F}
LCN Value = {0102BDC9-3B85-4A11-968D-7D314DA07C06}
LevelJet = {D64B904C-5312-443C-A2F3-03201ED9811C}
Location Control = {45E97A63-F870-408A-B259-2933F7EABF74}
M-Bus Configurator = {73336CB8-B5E5-187E-8BB8-CC4F32BF127D}
M-Bus Device = {B53BE2E5-892D-4537-94AC-EAC68A469188}
M-Bus Discovery = {6BED8C28-002F-66B7-07B5-4145D4408F34}
M-Bus Gateway = {301AB802-23CD-4DE2-91D1-6E3BC9BF03FC}
MQTT Client = {F7A0DD2E-7684-95C0-64C2-D2A9DC47577B}
MQTT Client Configurator = {2408C0E0-E672-7FE5-414B-F78C9B9244E4}
MQTT Client Device = {91D174F2-AE0F-B8D8-5EF4-6232B9083CCF}
MQTT Server = {C6D2AEB3-6E1F-4B2E-8E69-3A1A00246850}
MQTT Server Configurator = {CC4F15B1-81C2-4F45-8D53-972F0C9C8103}
```

```
MQTT Server Device = {01C00ADD-D04E-452E-B66A-D253278743FE}
Media Player = {2999EBBB-5D36-407E-A52B-E9142A45F19C}
ModBus Address = {CB197E50-273D-4535-8C91-BB35273E3CA5}
ModBus Configurator = {8C1E6AC6-4C45-067A-3696-6B4284B75C4B}
ModBus Gateway = {A5F663AB-C400-4FE5-B207-4D67CC030564}
Module Control = {B8A5067A-AFC2-3798-FEDC-BCD02A45615E}
Multicast Socket = {BAB408E0-0A0F-48C3-B14E-9FB2FA81F66A}
Notification Control = {D4B231D6-8141-4B9E-9B32-82DA3AEEAB78}
OZW Configurator = {62BDA2F6-6206-4DC3-9431-4328B5890836}
OZW DataPoint = {422A36CD-5565-4CDC-9FC4-242D3C810901}
OZW Device = {CE54AFB0-625A-4F37-B5C1-EAF1FD2A1DA6}
OZW Gateway = {1D60A51E-FF04-4D82-9E0D-04B3A11EF13F}
OneWire Configurator = {F462BFF3-6772-4720-8450-49E6E2820643}
OneWire Discovery = {1739B440-4A46-F5FF-8640-F9DF3D14C06B}
OneWire F05 = {F1B54BB1-DC7D-42D9-A973-6CA4789E358F}
OneWire F10 = {685D4911-57CC-43AF-BA36-183EF2C8518F}
OneWire F12 = {50DB3978-CF6A-4BCE-87DF-5BB45D900628}
OneWire F1D = {88A63FF8-832B-48E3-B989-A416C7908E6A}
OneWire F20 = {5DF182B0-01DF-4D8A-82CD-E646FD9BF0B2}
OneWire F26 = {301FA314-65F8-4317-8BF5-729CF8664F54}
OneWire F28 = {766E337F-A707-48CC-A323-70D7E77E3F8C}
OneWire F29 = {6A75828A-25CD-4CF3-83EA-DAAB914030A7}
OneWire F2C = {E02955B3-49E4-47A9-A9ED-2C71401F6D6E}
OneWire F3A = {BD0F2622-F67C-4248-9A04-316DF13914C3}
OneWire Gateway = {CED1D815-2477-4B05-8F65-0E4475913063}
POP3 = {69CA7DBF-5FCE-4FDF-9F36-C05E0136ECFD}
Popup Module = {5EA439B8-FB5C-4B81-AA35-1D14F4EA9821}
Presence Control = {B263AFBC-950F-462C-95B1-E3ACACE0B9B0}
Register Variable = {F3855B3C-7CD6-47CA-97AB-E66D346C037F}
SMS = {96102E00-FD8C-4DD3-A3C2-376A44895AB1}
SMS REST = {96102E00-FD8C-4DD3-A3C2-376A44895AC2}
SMTP = {375EAF21-35EF-4BC4-83B3-C780FD8BD88A}
SSDP Control = {FFFFA648-B296-E785-96ED-065F7CEE6F29}
SSE Client = {2FADB4B7-FDAB-3C64-3E2C-068A4809849A}
Serial Port = {6DC3D946-0D31-450F-A8C6-C42DB8D7D4F1}
Server Socket = {8062CF2B-600E-41D6-AD4B-1BA66C32D6ED}
Shutter Control = {9512FC2C-7999-4BD0-B322-B6C49BD09127}
Shutter Control Legacy = {542CC907-CA63-4E7A-A8C7-92F74639FA4C}
Siemens Configurator = {691B4B06-C774-BA52-6C63-B88D5EDD79EB}
Siemens Device = {932076B1-B18E-4AB6-AB6D-275ED30B62DB}
Siemens Gateway = {1B0A36F7-343F-42F3-8181-0748819FB324}
Skin Control = {E8C1043A-E2C2-4CC8-88DC-CBF02D45E15E}
Store Control = {F45B5D1F-56AE-4C61-9AB2-C87C63149EC3}
TMEX = {ABEE57FB-E243-461D-B2F7-4D176ACB7C7C}
Text Parser = {4B00C7F7-1A6D-4795-A2D2-08151854D259}
Text To Speech = {684CC410-6777-46DD-A33F-C18AC615BB94}
ThermoJet = {C3380B6E-2351-4A19-9668-A17960F97E51}
UDP Socket = {82347F20-F541-41E1-AC5B-A636FD3AE2D8}
UVR1611 = {0A8F9D69-E78B-4EAB-8E26-C8A4ACF7FA25}
Universal RX = {23D1E4CD-77E9-4F4B-B8B3-53AE81222AF6}
Util Control = {B69010EA-96D5-46DF-B885-24821B8C8DBD}
Velleman USB = {8CE70CD0-6674-4907-B3CE-F6E5235C9938}
Virtual IO = {6179ED6A-FC31-413C-BB8E-1204150CF376}
VoIP = {A4224A63-49EA-445F-8422-22EF99D8F624}
WMRS200 = {DD2A4676-82C6-4154-9AAD-DE30668D53B0}
WMRS200 Gateway = {E4FDC411-95D5-453C-B731-0CEB0483E663}
WS Client = {D68FD31F-0E90-7019-F16C-1949BD3079EF}
WebFront Visualization = {3565B1F2-8F7B-4311-A4B6-1BF1D868F39E}
WebHook Control = {015A6EB8-D6E5-4B93-B496-0D3F77AE9FE1}
```

```
WebOAuth Control = {F99BF07D-CECA-438B-A497-E4B55F139D37}
WebServer = {D83E9CCF-9869-420F-8306-2B043E9BA180}
WinLIRC = {19E51FC2-064B-4A51-8995-11FEFF7F129A}
WuT Counter = {5F1C5261-07A2-4A7E-9AC4-88AF9ED29420}
WuT Gateway = {01CA6888-C833-484E-A3F3-806535421CB7}
WuT Input = {C3D0F82C-CD07-4AA8-AE5C-7AD983FE91F3}
WuT Output = {E85C40B3-C1E9-4A60-85C7-6CDDA3D8D7BF}
WuT ThermoHygro = {2EF634A4-D96D-4018-BD90-94E487A89D49}
XBee Gateway = {7FA47C08-E31B-44A6-9E50-20C4DDD3E081}
XBee Splitter = {9D5DCE79-1A97-4531-9D10-68839F4BEAAC}
Z-Wave Configurator = {2D7CA355-2C51-4430-8F67-4E397EAAEA19}
Z-Wave Discovery = {5754F842-9D3E-BF8D-E3FC-8B1237AEC7E5}
Z-Wave Gateway = {4EF72D56-BF9F-4347-8F0A-2035D241116F}
Z-Wave Module = {101352E1-88C7-4F16-998B-E20D50779AF6}
dS Apartment = {555C65F5-F436-467F-A2B1-DAA1BB02CA3B}
dS Configurator = {7CADC358-60B0-4D91-9825-27E9029B62C4}
dS Expert = {1822D865-2093-32A1-6999-280C82E669EB}
dS Joker = {DB54D0EC-64C4-4941-9B64-2337183C9C2E}
dS Light = {DB54D0DB-64C3-4930-9B53-2337183C9C1D}
dS Shutter = {3DDA1E2B-B807-4680-AB6D-E7E8FBD6093A}
dS Splitter = {8D7872F4-CAC3-409D-926B-CCF1BA9E937B}
dS Zone = {4E0FE8A0-1A1E-460A-9446-2D18B3635310}
ekey Fingerprint = {9AB0DD38-A396-4D35-81A1-2FB6A3992332}
xComfort Binary Input = {3040A77D-3E9C-42D4-A1B6-329EFE8086DB}
xComfort Configurator = {5DD921D4-4712-443F-B89F-03434A4DBF94}
xComfort Dimmer = {8050FEEC-C875-4BDD-9143-D15134B89D35}
xComfort Energy = {814067F0-EACB-43C3-99BD-5CB9B2F8FB9E}
xComfort Gateway = {D2DCE381-19A7-4D14-B819-49C0539BC350}
xComfort HRV = {E4693C3F-95F1-48B6-9443-4A6B3EE0FACA}
xComfort Heating = {586D59EE-04A8-4896-B49D-63B4DD9618EF}
xComfort Humidity = {3EBA1AB7-72CA-48D2-8F89-813E085D41BB}
xComfort Impulse = {A374DCF0-CEDE-4EB7-B6A8-E92787E19B25}
xComfort One Channel Heating = {62444FDF-FBF0-8F98-50B3-B1E85ADD86E2}
xComfort Remote = {DCBD8143-83AB-4068-8FC0-0C92A93AA8A8}
xComfort Room Control = {1A1C4C67-C99D-4D3E-8A34-23581CE8CCAA}
xComfort Shutter = {1B7B5B7D-CAA9-4AB5-B9D8-EC805EC955AD}
xComfort Switch = {27DD9788-802E-45B7-BA54-FB97141398F7}
xComfort Temperature = {591B4A05-E5BF-4EEA-BC34-36E6F1CC9D56}
xComfort Value RX = {DA2FCC12-2DE1-404A-8A5E-1C6AF05F96A2}
xComfort Value TX = {ED6A1E00-81C7-416F-9F97-1F2CC8F45B15}
*/
```

IPS_GetModuleListByType

Require: Symcon >= 3.3

```
array IPS_GetModuleListByType (int $ModuleType)
```

PARAMETERS

ModuleType

VALUE	CONSTANT	DESCRIPTION
0	MODULETYPE_CORE	Core
1	MODULETYPE_IO	I/O
2	MODULETYPE_SPLITTER	Splitter
3	MODULETYPE_DEVICE	Device
4	MODULETYPE_CONFIGURATOR	Configurator
5	MODULETYPE_DISCOVERY	Discovery
6	MODULETYPE_VISUALIZATION	Visualization

RETURNS

An array of all GUIDs of modules in IP-Symcon of the type **ModuleType**, given as strings. The order of the entries is not guaranteed.

DESCRIPTION

The function determines the GUIDs of all modules within IP-Symcon of the type **ModuleType**. The IDs are listed in an array of strings.

EXAMPLE

```
print_r(IPS_GetModuleListByType(MODULETYPE_CORE));

/* returns e.g.:
Array
(
[0] => {2D8B0172-166C-45B0-B979-61A787809E22}
[1] => {CCC52C33-D77A-4AE9-A6CF-462F152532A0}
[2] => {11842D31-3CA4-4F1B-BBA9-E4A0FE1873AF}
```

```
[3] => {4CB91589-CE01-4700-906F-26320EFCF6C4}
etc. ...
)
*/
```

IPS_IsModuleCompatible

```
bool IPS_IsModuleCompatible (string $ModuleID, string $ParentModuleID)
```

PARAMETERS

ModuleID	ID of the potential child module
ParentModuleID	ID of the potential parent module

RETURNS

TRUE if the modules are compatible, otherwise FALSE.

DESCRIPTION

The function determines if the module with the ID ParentModuleID is a valid parent to the module with the ID ModuleID.

EXAMPLE

```
if (IPS_IsModuleCompatible("{48FCFDC1-11A5-4309-BB0B-A0DB8042A969}",
"{57040540-4432-4220-8D2D-4676B57E223D}"))
echo "FS20 module is compatible with the FHZ module!";
```

IPS_LibraryExists

```
bool IPS_LibraryExists (string $LibraryID)
```

PARAMETERS

LibraryID	ID of the library

RETURNS

If the LibraryID exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks if the library with ID **LibraryID** exists.

EXAMPLE

```
if (IPS_LibraryExists("{7DC57F9A-C095-4CDE-A6F0-2CB35A29A8FE}"))
echo "ELV Library exists!";
```

IPS_ModuleExists

```
bool IPS_ModuleExists (string $ModuleID)
```

PARAMETERS

ModuleID	ID of the module

RETURNS

If the ModuleID exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks if the module with the ID **ModuleID** exists.

EXAMPLE

```
if (IPS_ModuleExists("{48FCFDC1-11A5-4309-BB0B-A0DB8042A969}"))
echo "FS20 Module exists!";
```

Management of Objects

IPS_FindObjectIDByIdent	searches the ID of an object by its ident without failing if nothing is found
IPS_FindObjectIDByName	searches the ID of an object by its name without failing if nothing is found
IPS_GetChildrenIDs	returns a list of all child objects
IPS_GetLocation	returns the complete path to an object
IPS_GetName	returns the name of the object
IPS_GetObject	returns extensive information about an object
IPS_GetObjectIDByIdent	returns the ID of an object by its ident
IPS_GetObjectIDByName	returns the ID of an object by its name
IPS_GetObjectList	returns a list of all existing objects
IPS_GetParent	returns the ID of the parent of the object
IPS_HasChildren	checks if an object has child objects
IPS_IsChild	checks if an object is a child of a parent object
IPS_ObjectExists	checks if a specific object exists
IPS_SetDisabled	sets an object as disabled for the visualizations
IPS_SetHidden	sets the visibility for an object
IPS_SetHiddenMaximize	sets the visibility of the maximize button of an object
IPS_SetHiddenTitle	sets the visibility of the title of an object
IPS_SetIcon	sets the icon of an object
IPS_SetIdent	sets the ident of an object
IPS_SetInfo	sets extended information of an object
IPS_SetName	sets the name of an object
IPS_SetParent	sets the parent of an object
IPS_SetPosition	sets the position of an object

IPS_FindObjectIDByIdent

```
int IPS_FindObjectIDByIdent (string $Ident, int $ParentID)
```

PARAMETERS

Ident	Object ident to search for
ParentID	ID of the object whose direct children are searched

RETURNS

ID of the found object, otherwise 1 (invalid ID)

DESCRIPTION

The function searches the **direct** children of the object with the ID **ParentID** for the object with the ident **Ident** and returns its ID. Deeper nested objects are not searched. The comparison is exact (case-sensitive). 0 (root) is also allowed as **ParentID**.
Unlike IPS_GetObjectIDByIdent, the function does not fail if no matching object exists, but returns the invalid ID **1**. It is therefore suited to check whether an object with a certain ident already exists without having to suppress a warning.



If no object is found, the result is **1**, not **FALSE** or 0. Valid object IDs are always greater than or equal to 10000, so the result should be compared with 1 explicitly.

If **Ident** is an empty string, the function fails with "Ident cannot be empty". If the object **ParentID** does not exist, it fails with "Object #... does not exist".
See also: IPS_FindObjectIDByName, IPS_SetIdent

EXAMPLE

```
$id = IPS_FindObjectIDByIdent("Temperature", 12345);
if ($id === 1) {
echo "No object with the ident 'Temperature' found";
} else {
echo "Found: #" . $id;
}
```

IPS_FindObjectIDByName

```
int IPS_FindObjectIDByName (string $Name, int $ParentID)
```

PARAMETERS

Name	Object name to search for
ParentID	ID of the object whose direct children are searched

RETURNS

ID of the first found object, otherwise 1 (invalid ID)

DESCRIPTION

The function searches the **direct** children of the object with the ID **ParentID** for an object with the name **Name** and returns its ID. Deeper nested objects are not searched. The comparison is exact (case-sensitive), the object type does not matter. 0 (root) is also allowed as **ParentID**.
Unlike IPS_GetObjectIDByName, the function does not fail if no matching object exists, but returns the invalid ID **1**.



If no object is found, the result is **1**, not **FALSE** or 0. Valid object IDs are always greater than or equal to 10000, so the result should be compared with 1 explicitly.



As names are not unique, the ID of the first found object is returned if several objects share the same name. For a unique search, IPS_FindObjectIDByIdent should be used.

If **Name** is an empty string, the function fails with "Name cannot be empty". If the object **ParentID** does not exist, it fails with "Object #... does not exist".

EXAMPLE

```
$id = IPS_FindObjectIDByName("Temperature", 12345);
if ($id === 1) {
echo "No object with the name 'Temperature' found";
} else {
```

```
echo "Found: #" . $id;
}
```

IPS_GetChildrenIDs

```
array IPS_GetChildrenIDs (int $ObjectID)
```

PARAMETERS

ObjectID	ID of the object

RETURNS

An array of integer values that contains the IDs of all child objects of the object **ObjectID**. The order of the entries is not guaranteed. If the order matters, sort the IDs by **ObjectPosition** from IPS_GetObject.

DESCRIPTION

This function determines the IDs of all child objects of the object with the ID **ObjectID**. The IDs of the children are listed in an array.

EXAMPLE

```
print_r(IPS_GetChildrenIDs(32102));

/* returns, e.g.,:
Array
(
[0] => 11650
[1] => 25578
[2] => 30202
etc. ...
)
*/
```

IPS_GetLocation

```
string IPS_GetLocation (int $ObjectID)
```

PARAMETERS

ObjectID	ID of the object

RETURNS

Path/ name of the object within the object tree

DESCRIPTION

This function returns the complete path including the name of the object with the ID **ObjectID**.



Since version 4.0: If the target is an event without a name (Unnamed Object), the describing name is returned



This function should only be used as a quick visualization of the path. The return value is not unique, because names are not unique. In addition, the name of an object can contain backslashes "\". Thus, handling the return value with functions like explode is not recommended.

EXAMPLE

```
echo IPS_GetLocation(47359);
//returns, e.g., "FHT8b\LowBattery"
```

IPS_GetName

```
string IPS_GetName (int $ObjectID)
```

PARAMETERS

ObjectID	ID for which the name should be returned

RETURNS

Name of the object in the logical object tree

DESCRIPTION

The function returns the name of an object.



Since version 4.0: If the object is an event and has no name (Unnamed object), the descriptive name is returned

EXAMPLE

```
echo IPS_GetName(47359);
// Returns e.g.: "My Table Lamp"

// Special case for unnamed events (since version 4.0)
echo IPS_GetName(48864 /*[TestScript\Unnamed Object (ID: 48864)]*/);
// Returns e.g.: "On Variable Change for Variable "Test Folder\IRTrans
USB/LAN/WLAN\Remote Control"
```

IPS_GetObject

```
array IPS_GetObject (int $ObjectID)
```

PARAMETERS

ObjectID	The ID of the object

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
ChildrenIDs	array	Object IDs of the children. See: IPS_GetChildrenIDs
HasChildren	boolean	TRUE if the object has child objects. See: IPS_HasChildren
ObjectID	integer	ID of the object
ObjectIcon	string	File name of the icon without extension. See: IPS_SetIcon
ObjectIdent	string	Identifier of the Object. See: IPS_SetIdent
ObjectInfo	string	Description that can be provided by the user. See: IPS_SetInfo
ObjectIsDisabled (since 4.0)	boolean	TRUE if the object is disabled. See IPS_SetDisabled
ObjectIsHidden	boolean	TRUE if the object is hidden in the visualization. See IPS_SetHidden
ObjectIsHiddenMaximize (since 9.1)	boolean	TRUE if the maximize button of the object should be hidden in the visualization. See IPS_SetHiddenMaximize
ObjectIsHiddenTitle (since 9.1)	boolean	TRUE if the title of the object should be hidden in the visualization. See IPS_SetHiddenTitle
ObjectIsLocked	boolean	Reserved
ObjectIsReadOnly	boolean	TRUE if the object is read-only. (Currently only used for state variables)
ObjectName	string	Name of the Object. See: IPS_SetName
ObjectPosition	integer	Position of the object in the visualization. This value does not have to be unique. See: IPS_SetPosition

INDEX	TYPE	DESCRIPTION
ObjectType	integer	Type of the object 0 OBJECTTYPE_CATEGORY Category 1 OBJECTTYPE_INSTANCE Instance 2 OBJECTTYPE_VARIABLE Variable 3 OBJECTTYPE_SCRIPT Script 4 OBJECTTYPE_EVENT Event 5 OBJECTTYPE_MEDIA Media 6 OBJECTTYPE_LINK Link
ObjectSummary	string	Short description of an object, which is generated by the module if needed
ParentID	integer	Parent object. 0 = No parent object. See: IPS_SetParent

DESCRIPTION

This function returns an array containing information about the object with the ID **ObjectID**. If the object is not available, a warning is generated.

EXAMPLE

```
print_r(IPS_GetObject(43502));

/* returns, e.g.,:
Array
(
[ParentID] => 26691
[ObjectID] => 43502
[ObjectType] => 1
[ObjectIdent] =>
[ObjectName] => MX FESLIM (ID: 1)
[ObjectInfo] => This is a device instance
[ObjectIcon] => IPS
[ObjectSummary] =>
[ObjectPosition] => 1
[ObjectIsReadOnly] =>
[ObjectIsHidden] =>
[ObjectIsHiddenTitle] =>
[ObjectIsHiddenMaximize] =>
[ObjectIsDisabled] =>
[ObjectIsLocked] =>
[HasChildren] => 1
[ChildrenIDs] => Array
(
[0] => 43430
[1] => 55892
[2] => 36597
[3] => 18993
[4] => 34524
```

```
[5] => 12269
[6] => 44450
[7] => 23394
[8] => 11693
[9] => 13654
)
)
*/
```

IPS_GetObjectIDByIdent

Require: Symcon >= 2.5

```
int IPS_GetObjectIDByIdent (string $ObjectIdent, int $ParentID)
```

PARAMETERS

ObjectIdent	Object ident to search for
ParentID	ID of the object whose direct children are searched

RETURNS

ID of the found object, otherwise **FALSE**

DESCRIPTION

This function tries to detect the object with the ident **ObjectIdent** whose parent object has the ID **ParentID**. If such an object is found, its ID is returned. Otherwise, this function returns FALSE.



The parameter **ObjectIdent** must not be an empty string ("").



__Difference to IPS_GetObjectIDByName:__ As idents of objects are unique within an individual level, either a single ID is returned or a warning is generated.



This function generates a **warning** if the ident is not found. The @ operator can be used to suppress this message. Please see the PHP manual for the exact functionality of this operator.

EXAMPLE

```
// This function can be used to replace IPS_StatusVariableExists
// Example for IPS_StatusVariableExists($id, "StatusVariable");
```

```
echo !(@IPS_GetObjectIDByIdent("StatusVariable", $id) === false);

// In addition, the function can be used to replace
IPS_GetStatusVariableID
// Example for IPS_GetStatusVariableID($id, "StatusVariable");
echo IPS_GetObjectIDByIdent("StatusVariable", $id);


$id = IPS_GetObjectIDByIdent($VariableIdent, $InstanceID);
$v = IPS_GetVariable($id);

// Array since 4.0
$sv = Array(
"VariableID" => $id,
"VariableIdent" => $VariableIdent,
"VariableName" => "N/A",
"VariablePosition" => 0,
"VariableProfile" => $v['VariableProfile'],
"VariableType" => $v['VariableType'],
"VariableHasAction" => ($v['VariableAction'] > 0),
"VariableUseAction" => ($v['VariableAction'] > 0)
);

// Array until 3.4
$sv = Array(
"VariableID" => $id,
"VariableIdent" => $VariableIdent,
"VariableName" => "N/A",
"VariablePosition" => 0,
"VariableProfile" => $v['VariableProfile'],
"VariableType" => $v['VariableValue']['ValueType'],
"VariableHasAction" => ($v['VariableAction'] > 0),
"VariableUseAction" => ($v['VariableAction'] > 0)
);
```

IPS_GetObjectIDByName

```
int IPS_GetObjectIDByName (string $ObjectName, int $ParentID)
```

PARAMETERS

ObjectName	Object name to search for
ParentID	ID of the object whose direct children are searched

RETURNS

ID of the found object, otherwise **FALSE**

DESCRIPTION

This function tries to detect the object with the name **ObjectName** whose parent object has the ID **ParentID**. It will return the ID of the first object whose name matches **ObjectName**. If no such object is found, this function returns FALSE.



The parameter **ObjectName** must not be an empty string ("").



Caution: Names in IP Symcon can be chosen freely, so they are **not unique**. For this reason, the function can return the ID of another object.



This function generates a **warning** if the name is not found. The @ operator can be used to suppress this message. Please see the PHP manual for the exact functionality of this operator.

EXAMPLE

```
$ObjectID = @IPS_GetObjectIDByName("Rain sensing", $ParentID);
if ($ObjectID === false)
echo "Object not found!";
```

```
else
echo "The Object ID is: ". $ObjectID;
```

IPS_GetObjectList

```
array IPS_GetObjectList ()
```

RETURNS

An array of integer values that contains the IDs of all objects within IP Symcon. The order of the entries is not guaranteed.

DESCRIPTION

This function determines the IDs of all objects registered within IP Symcon. The IDs are listed in an array.

EXAMPLE

```
print_r(IPS_GetObjectList());

/* returns, e.g.,:
Array
(
[0] => 0
[1] => 10573
[2] => 11363
[3] => 11650
[4] => 14114
etc. ...
)
*/
```

IPS_GetParent

```
int IPS_GetParent (int $ObjectID)
```

PARAMETERS

ObjectID	The ID of the object

RETURNS

The ID of the parent of the object

DESCRIPTION

This function returns the ID of the parent of the object with the ID **ObjectID**.

EXAMPLE

```
echo IPS_GetParent(47359);
//returns, e.g.,: 0 for the main category
```

IPS_HasChildren

```
bool IPS_HasChildren (int $ObjectID)
```

PARAMETERS

ObjectID	The ID of the object

RETURNS

TRUE if the object has child objects, otherwise FALSE

DESCRIPTION

This function checks if the object with the ID **ObjectID** has child objects.

EXAMPLE

```
if (IPS_HasChildren(0))
echo "The root object has child objects!";
```

IPS_IsChild

```
bool IPS_IsChild (int $ObjectID, int $ParentID, bool $Recursive)
```

PARAMETERS

ObjectID	The ID of the object
ParentID	The ID of the potential parent object
Recursive	FALSE if only a direct parent-child relationship should be checked; TRUE if a relationship over multiple levels should also be checked

RETURNS

TRUE if the object is a child of the object with the ID **ParentID**, otherwise **FALSE**

DESCRIPTION

This function checks if the object with the ID **ObjectID** is a child of the object with the ID **ParentID**. If the parameter **Recursive** is set to TRUE, all descendants of the parent node are checked and the function also returns TRUE if the object is multiple levels below the parent node, e.g., it is its grandchild. Otherwise, only the direct child objects of the parent are checked.

EXAMPLE

```
if (IPS_IsChild(12345, 0, true))
echo "The object 12345 is a descendant of the root object!";
```

IPS_ObjectExists

```
bool IPS_ObjectExists (int $ObjectID)
```

PARAMETERS

ObjectID	The ID of the object

RETURNS

If the object exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

This function checks if an object with the ID **ObjectID** exists.

EXAMPLE

```
if (IPS_ObjectExists(34881))
echo "An object with that ID exists!";
```

IPS_SetDisabled

Require: Symcon >= 4.0

```
bool IPS_SetDisabled (int $ObjectID, bool $Disabled)
```

PARAMETERS

ObjectID	The ID of the object
Disabled	TRUE if the object should be disabled; FALSE if it should be enabled

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the object with the ID **ObjectID** to disabled, i.e., it may be visible in the visualizations but it is not clickable or usable.

EXAMPLE

```
IPS_SetDisabled(47381, true); // The object becomes disabled
```

IPS_SetHidden

Require: Symcon >= 2.1

```
bool IPS_SetHidden (int $ObjectID, bool $Hidden)
```

PARAMETERS

ObjectID	The ID of the object
Hidden	TRUE if invisible

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the visibility of the object with the ID **ObjectID**. Invisible objects and all their descendants are excluded from the visualization. Hidden objects are displayed in gray in the management console.

EXAMPLE

```
IPS_SetHidden(47381, true); //hide object
```

IPS_SetHiddenMaximize

Require: Symcon >= 9.1

```
bool IPS_SetHiddenMaximize (int $ObjectID, bool $Hidden)
```

PARAMETERS

ObjectID	The ID of the object
Hidden	TRUE if maximize button is invisible

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the visibility of the maximize button of the object's tile with the ID **ObjectID**. If the maximize button is not shown, the corresponding space of the tile remains empty and the tile cannot be maximized.

EXAMPLE

```
IPS_SetHiddenMaximize(47381, true); //hide maximize button
```

IPS_SetHiddenTitle

Require: Symcon >= 9.1

```
bool IPS_SetHiddenTitle (int $ObjectID, bool $Hidden)
```

PARAMETERS

ObjectID	The ID of the object
Hidden	TRUE if title is invisible

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the visibility of the title of the object's tile with the ID **ObjectID**. If the title is not shown, the corresponding space of the tile remains empty. If the title is hidden, the tile automatically uses the full available space.

EXAMPLE

```
IPS_SetHiddenTitle(47381, true); //hide title
```

IPS_SetIcon

Require: Symcon >= 2.1

```
bool IPS_SetIcon (int $ObjectID, string $Icon)
```

PARAMETERS

ObjectID	The ID of the object
Icon	Filename of the icon without path and extension

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function assigns an icon to the object with the ID **ObjectID**. The name of the icon is the filename without the extension or path.



A list of available icons can be found under Icons. In the tile visualization the name of every Font Awesome icon can be used (light style and brand icons), e.g. "lightbulb". The classic icon names, as returned by UC_GetIconList, are shown there as a similar Font Awesome icon.

EXAMPLE

```
IPS_SetIcon(47381, "weather");
```

IPS_SetIdent

Require: Symcon >= 2.5

```
bool IPS_SetIdent (int $ObjectID, string $Ident)
```

PARAMETERS

ObjectID	ID of the object to be changed
Ident	New identifier for the object

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function assigns the identifier **Ident** to the object with ID **ObjectID**. The identifier is a string that is invisible to the end user. Allowed characters are "a..z", "A..Z", "0..9" and "_". It can be used to find objects within a level without using the unique and system dependent ID. For example, all state variables of an instance can be addressed by identifiers and can be handled equivalently for copies of that instance. The function can be used on all the objects that have an ID for identification.



Within a single level, an identifier is unique. Thus, the function IPS_GetObjectIDByIdent returns the intended object.

EXAMPLE

```
IPS_SetIdent(47381, "TEMPERATURE");
```

IPS_SetInfo

```
bool IPS_SetInfo (int $ObjectID, string $Info)
```

PARAMETERS

ObjectID	The ID of the object
Info	New description text for the object

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function assigns the description **Info** to the object with the ID **ObjectID**. These can be used to provide information about specific configurations or other special information on each object.

EXAMPLE

```
IPS_SetInfo(47381, "USB sound card - Port A, basement - Green cable");
```

IPS_SetName

```
bool IPS_SetName (int $ObjectID, string $Name)
```

PARAMETERS

ObjectID	The ID of the object
Name	New name for the object

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function assigns the name **Name** to the object with the ID **ObjectID**. The name can be seen in the object tree and visualizations and can contribute to readability. The function can be applied to all objects that have an ID for identification.

EXAMPLE

```
IPS_SetName(47381, "Safety precautions");
```

IPS_SetParent

```
bool IPS_SetParent (int $ObjectID, int $ParentID)
```

PARAMETERS

ObjectID	The ID of the object
ParentID	ID of the new parent object

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function moves the object with the ID **ObjectID** below the object with the ID **ParentID** within the object tree. The object is visually subordinated to the parent object. The new assignment is visible in the logical treeview. By moving objects, the object tree can be modified to properly reflect the unity and/or functionality of various components.
The assignment of an object to a new parent object does not change the existing physical connections. It only provides a better representation.
If the object has an ident, no object with the same ident may exist at the new target position.

EXAMPLE

```
IPS_SetParent(47381, 15361);
```

IPS_SetPosition

Require: Symcon >= 2.1

```
bool IPS_SetPosition (int $ObjectID, int $Position)
```

PARAMETERS

ObjectID	The ID of the object
Position	Position value

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function assigns the position value **Position** to the object with the ID **ObjectID**. The position will be used to display the objects in the desired order. The default value is 0. The position value does not have to be unique. If two positions are equal, the objects are primarily sorted by type. For equal types, the name is used as secondary sorting criterion.

EXAMPLE

```
IPS_SetPosition(47381, 5);
```

Program Information

IPS_FunctionExists	checks if a specific function exists
IPS_GetAvailableUpdates	returns an available update for the configured update channel
IPS_GetFunction	returns extensive information about a function
IPS_GetFunctionList	returns a list of all available functions
IPS_GetFunctionListByModuleID	returns a list of all available functions of a module
IPS_GetKernelArchitecture	returns the architecture that IP-Symcon was created for
IPS_GetKernelDate	returns the kernel creation date
IPS_GetKernelDir	returns the path to the IP Symcon program folder
IPS_GetKernelDirEx	returns the path to the IP Symcon installation folder
IPS_GetKernelDirSpace	returns the total/free/available disk space of the drive on which all data of IP-Symcon is stored
IPS_GetKernelPlatform	returns the operating system which IP-Symcon is running on
IPS_GetKernelRevision	returns a unique revision identifier
IPS_GetKernelRunlevel	returns the run level of the kernel
IPS_GetKernelStartTime	returns the point in time when IP-Symcon was started
IPS_GetKernelVersion	returns the version number of the used kernel
IPS_GetLiveUpdateVersion	returns a version identifier made of the build date and revision of the kernel
IPS_GetLogDir	returns the path to the IP Symcon log folder
IPS_GetSystemLanguage	returns the system language of the operating system on which IP-Symcon is running
IPS_GetUpdateChannel	returns the currently configured update channel
IPS_LogMessage	Writes a user defined message into the log file

IPS_FunctionExists

Require: Symcon >= 2.6

```
bool IPS_FunctionExists (string $FunctionName)
```

PARAMETERS

FunctionName	Name of the checked function

RETURNS

If the function exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks if the function named **FunctionName** is registered within IP-Symcon.

EXAMPLE

```
if (IPS_FunctionExists("GetValue"))
echo "The function GetValue exists!";
```

IPS_GetAvailableUpdates

```
array IPS_GetAvailableUpdates ()
```

RETURNS

Array with at most one entry. If no newer version is available, the array is empty. Each entry contains the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Build	integer	Build number of the available version
Channel	string	Update channel in which the version was found (stable, beta, testing)
Date	integer	Unix timestamp of the release
Version	string	Version number of the available version, e.g. "8.1"

DESCRIPTION

The function checks whether a newer version of Symcon is available for the currently configured update channel (IPS_GetUpdateChannel). For this, the package list of the own platform (IPS_GetKernelPlatform) and architecture is fetched from the update server (apt.symcon.de); an internet connection is therefore required.
Only the newest published version of the channel is considered. It counts as available if its version number (major/minor compared numerically) is greater than the installed one (IPS_GetKernelVersion) or, for the same version number, if its build number is greater. In this case the result contains exactly one entry, otherwise it is empty.
The platforms SymBox, Windows, Ubuntu, Raspberry Pi, Mac, Catan C1 and Docker are supported. On other platforms, the function fails with "Updates cannot be detected for this platform!". On a development version (platform "Unknown"), an empty array is always returned. If the update server cannot be reached or the package list cannot be read, the function fails with a corresponding error.

EXAMPLE

```
$updates = IPS_GetAvailableUpdates();
if (count($updates) == 0) {
echo "Symcon is up to date";
} else {
print_r($updates);
}

/* e.g. returns:
Array
(
```

```
[0] => Array
(
[Build] => 1234
[Channel] => stable
[Date] => 1757412000
[Version] => 8.1
)
)
*/
```

IPS_GetFunction

Require: Symcon >= 2.6

```
array IPS_GetFunction (string $FunctionName)
```

PARAMETERS

FunctionName	Name of the inspected function

RETURNS

An array with the following **key => value** pairs

INDEX	TYPE	DESCRIPTION
FunctionName	string	Function name
Parameters	array	See Table "Parameter Information"
Result	array	See Table "Result Information"

Table: Parameter Information

INDEX	TYPE	DESCRIPTION
Description	string	Name of the parameter
Enumeration	array	Named presentation of possible integer values
Type_	integer	Variable Type of parameter/result: 0=Boolean, 1=Integer, 2=Float, 3=String, 4=Variant, 5=Array

Table: Result Information

INDEX	TYPE	DESCRIPTION
Description	string	Name of the result value
Enumeration	array	Named presentation of possible integer values
Type_	integer	Variable Type of parameter/result: 0=Boolean, 1=Integer, 2=Float, 3=String, 4=Variant, 5=Array

DESCRIPTION

This function provides additional information about a function registered within IP-Symcon. The parameter **FunctionName** provides the name of the inspected function.

EXAMPLE

```
// Request information about the functions IPS_GetFunction and
IPS_CreateMedia
var_dump(IPS_GetFunction("IPS_GetFunction"));
var_dump(IPS_GetFunction("IPS_CreateMedia"));

// Result "IPS_GetFunction"
/*
array(3) {
["FunctionName"]=>
string(15) "IPS_GetFunction"
["Result"]=>
array(3) {
["Type_"]=>
int(5)
["Description"]=>
string(6) "Result"
["Enumeration"]=>
array(0) {
}
}
["Parameters"]=>
array(1) {
[0]=>
array(3) {
["Type_"]=>
int(3)
["Description"]=>
string(12) "FunctionName"
["Enumeration"]=>
array(0) {
}
}
}
*/

// Result "IPS_CreateMedia"
/*
array(3) {
["FunctionName"]=>
string(15) "IPS_CreateMedia"
["Result"]=>
array(3) {
["Type_"]=>
int(1)
["Description"]=>
string(6) "Result"
["Enumeration"]=>
array(0) {
}
}
["Parameters"]=>
array(1) {
[0]=>
array(3) {
```

```
["Type_"]=>
int(1)
["Description"]=>
string(9) "MediaType"
["Enumeration"]=>
array(6) {
[0]=>
string(9) "mtIPSView"
[1]=>
string(7) "mtImage"
[2]=>
string(7) "mtSound"
[3]=>
string(8) "mtStream"
[4]=>
string(7) "mtChart"
[5]=>
string(10) "mtDocument"
}
}
}
}
*/
```

IPS_GetFunctionList

Require: Symcon >= 2.6

```
array IPS_GetFunctionList (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance whose functions are returned, 0 for all functions

RETURNS

An array of string values of all function names. The order of the entries is not guaranteed.

DESCRIPTION

The function determines all available IPS functions that are available within the current configuration of IP Symcon. The names of the functions are listed in an array. The number of functions depends on the current configuration of IP Symcon. E.g., if the TTS (Text-To-Speech) module is available or a library was installed via the Module Store, its functions are listed as well.
The parameter **InstanceID** can be used to specify a filter. In that case, only functions of the specified instance are shown in the array. If the InstanceID is specified as 0, the filtering mechanism is not active and all available functions within IP Symcon will be returned.
Details about the individual functions can be determined by IPS_GetFunction.

EXAMPLE

```
$allFunctions = IPS_GetFunctionList(0);
print_r(IPS_GetFunction($allFunctions[48])); // output only function 48
(IPS_CreateScript)

/* returns:
Array
(
[FunctionName] => IPS_CreateScript
[Parameters] => Array
(
[0] => Array
(
[Description] => ScriptType
[Type_] => 1
)
```

```
)
[Result] => Array
(
[Description] => Result
[Type_] => 1
)
)
*/

//Export all IP Symcon functions with a parameter list
$instanceid = 0; //0 = All functions, otherwise filter on InstanceID
$fs = IPS_GetFunctionList($instanceid);
asort($fs);
$typestr = Array("boolean", "integer", "float", "string", "variant",
"array");
foreach($fs as $f) {
$f = IPS_GetFunction($f);
echo sprintf("[%7s]", $typestr[$f['Result']['Type_']]) . " -
".$f['FunctionName']."(";
$a = Array();
foreach($f['Parameters'] as $p) {
if(isset($p['Enumeration']) && sizeof($p['Enumeration']) > 0) {
$b=Array();
foreach($p['Enumeration'] as $k => $v) {
$b[] = $k."=".$v;
}
$type = "integer/enum[".implode(", ", $b)."]";
} else {
$type = $typestr[$p['Type_']];
}
$a[]=$type." $".$p['Description'];
}
echo implode(", ", $a).");\n";
}
```

IPS_GetFunctionListByModuleID

```
array IPS_GetFunctionListByModuleID (string $ModuleID)
```

PARAMETERS

ModuleID	GUID of the module

RETURNS

An array of string values of all functions. The order of the entries is not guaranteed.

DESCRIPTION

The function determines all available IPS functions of the module with the GUID **ModuleID**.
The names of the functions are listed in an array.
Details about the individual module can be determined by IPS_GetModule.
Details about the individual functions can be determined by IPS_GetFunction.

EXAMPLE

```
// All functions of "DPT 200.x - Binary Value & Status" from KNX
print_r(IPS_GetFunctionListByModuleID("{04ED3286-C982-436F-A01E-
8A03C2E976A8}"));

/* returns:
Array
(
[0] => IPS_RequestAction
[1] => IPS_StopSearch
[2] => IPS_StartSearch
[3] => IPS_IsSearching
[4] => KNX_WriteDPT200
[5] => IPS_GetConfigurationForm
[6] => IPS_SupportsSearching
[7] => KNX_RequestStatus
[8] => IPS_GetConfiguration
[9] => IPS_GetProperty
[10] => IPS_SetProperty
[11] => IPS_SetConfiguration
[12] => IPS_GetConfigurationForParent
[13] => IPS_HasChanges
[14] => IPS_ResetChanges
[15] => IPS_ApplyChanges
```

```
[16] => IPS_GetReferenceList
[17] => IPS_Translate
[18] => KNX_RenameVariables
)
*/
```

IPS_GetKernelArchitecture

Require: Symcon >= 6.0

```
string IPS_GetKernelArchitecture ()
```

RETURNS

The architecture that IP-Symcon was created for

DESCRIPTION

The function returns a string with the architecture that IP-Symcon was created for.

EXAMPLE

```
echo IPS_GetKernelArchitecture ();
// Sample output:
/*
// MacOS, Windows 32Bit
i386

// Linux, Windows 64Bit
amd64

// Linux, Raspberry Pi
armhf

// Linux, Raspberry Pi
arm64
*/
```

IPS_GetKernelDate

Require: Symcon >= 4.4

```
int IPS_GetKernelDate ()
```

RETURNS

The creation date as Unix timestamp

DESCRIPTION

The function returns an integer value containing the creation date of the kernel as Unix timestamp.

EXAMPLE

```
echo IPS_GetKernelDate ();
// Sample output:
// 1511046903
```

IPS_GetKernelDir

```
string IPS_GetKernelDir ()
```

RETURNS

The full path to the IP Symcon program folder

DESCRIPTION

The function returns a string containing the full path to the IP Symcon program folder (Settings, Scripts, Media, ...). Depending on the operating system, the path contains a trailing slash ("/") or backslash ("\").

EXAMPLE

```
echo IPS_GetKernelDir();
// Sample output:
/*
// Windows
// since Version 5.3
C:\ProgramData\Symcon\
// until Version 5.2
C:\Program Files\IP-Symcon\

// Linux, RaspberryPi
/var/lib/symcon/

// MacOS
/Library/Application Support/Symcon/
*/
```

IPS_GetKernelDirEx

Require: Symcon >= 4.0

```
string IPS_GetKernelDirEx ()
```

RETURNS

The full path to the IP Symcon installation folder

DESCRIPTION

The function returns a string containing the full path to the IP Symcon installation folder. Depending on the operating system, the path contains a trailing slash ("/") or backslash ("\").

EXAMPLE

```
echo IPS_GetKernelDirEx();

// Sample output:
/*
// Windows
C:\Program Files\Symcon\

// Linux, RaspberryPi
/usr/share/symcon/

// MacOS
/Applications/Symcon.app/Contents/Service/
*/
```

IPS_GetKernelDirSpace

Require: Symcon >= 6.2

```
array IPS_GetKernelDirSpace ()
```

RETURNS

Array with the total/free/available disk space in bytes

DESCRIPTION

The function returns an array containing the keys Total/Free/Available.
Those contain the total/free/available disk space in bytes of the drive on which all data of IP-Symcon is stored.

EXAMPLE

```
print_r(IPS_GetKernelDirSpace());
// Sample output:
/*
Array
(
[Total] => 511582613504
[Free] => 96859312128
[Available] => 96859312128
)
*/
```

IPS_GetKernelPlatform

Require: Symcon >= 4.4

```
string IPS_GetKernelPlatform ()
```

RETURNS

The operating system IP-Symcon is running on

DESCRIPTION

The function returns a string with the operating system the IP-Symcon server is running on.

EXAMPLE

```
echo IPS_GetKernelPlatform();
// Sample output:
/*
// SymBox
SymBox

// Windows
Windows

// Linux
Ubuntu
Ubuntu (Docker)

// RaspberryPi
Raspberry Pi
Raspberry Pi (Docker)

// MacOS
Mac
*/
```

IPS_GetKernelRevision

Require: Symcon >= 4.4

```
string IPS_GetKernelRevision ()
```

RETURNS

Unique revision identifier

DESCRIPTION

The function returns a string with a unique revision identifier.

EXAMPLE

```
echo IPS_GetKernelRevision();
// Sample output:
// 8edcb65b27506d687dd6a4d1cee470f86e2d89e1
```

IPS_GetKernelRunlevel

Require: Symcon >= 4.0

```
int IPS_GetKernelRunlevel ()
```

RETURNS

Run level of the kernel

DESCRIPTION

This function returns the run level of the system.

RUN LEVEL	RETURN VALUE	DESCRIPTION
KR_CREATE	10101	Kernel is being created
KR_INIT	10102	Kernel is being initialized, e.g., modules are loaded and instances created
KR_READY	10103	Kernel is ready and running
KR_UNINIT	10104	Kernel is being shut down and everything is quit cleanly
KR_SHUTDOWN	10105	Kernel was shut down completely

EXAMPLE

```
echo IPS_GetKernelRunlevel(); // returns, e.g., 10103, if the run level
is KR_READY
```

IPS_GetKernelStartTime

Require: Symcon >= 4.0

```
int IPS_GetKernelStartTime ()
```

RETURNS

Point in time of the most recent start of IP-Symcon

DESCRIPTION

The function returns the date and time of the start of IP-Symcon as Unix timestamp.

EXAMPLE

```
// returns, e.g., 1204675012
echo IPS_GetKernelStartTime();
// returns, e.g., "2008.03.04 23:56:52"
echo date("Y.m.d H:i:s", IPS_GetKernelStartTime());
```

IPS_GetKernelVersion

```
string IPS_GetKernelVersion ()
```

RETURNS

Program version. Major/minor version separated by a dot (".")

DESCRIPTION

The function returns a string with the number of the IP Symcon program version. This information can be used to decide if a script can be run under the reported version.

EXAMPLE

```
echo IPS_GetKernelVersion(); // returns, e.g., "2.00"
```

IPS_GetLiveUpdateVersion

```
string IPS_GetLiveUpdateVersion ()
```

RETURNS

Build date and the first 12 characters of the revision identifier, separated by ", "; on a development version the revision identifier itself ("Unknown")

DESCRIPTION

The function returns a readable identifier of the installed kernel build. It consists of the build date of the kernel (IPS_GetKernelDate) as a date in the format of the system settings and the first 12 characters of the revision identifier (IPS_GetKernelRevision), separated by ", ".
If no build date is available, as with a self-built development version, the complete revision identifier is returned instead - in this case "Unknown".
The version number is returned by IPS_GetKernelVersion; whether a newer update is available is returned by IPS_GetAvailableUpdates.

EXAMPLE

```
echo IPS_GetLiveUpdateVersion();
// e.g. returns: 09/15/26, 8edcb65b2750
// on a development version: Unknown
```

IPS_GetLogDir

Require: Symcon >= 4.0

```
string IPS_GetLogDir ()
```

RETURNS

The full path to the IP Symcon log folder

DESCRIPTION

The function returns a string containing the full path to the IP Symcon log folder. Depending on the operating system, the path contains a trailing slash ("/") or backslash ("\").

EXAMPLE

```
echo IPS_GetLogDir();
// Sample output:
/*
//Windows
"C:\ProgramData\Symcon\logs\"

// Linux, RaspberryPi
"/var/log/symcon/"

// MacOS
"/Library/Logs/Symcon/"
*/
```

IPS_GetSystemLanguage

Require: Symcon >= 6.1

```
string IPS_GetSystemLanguage ()
```

RETURNS

System language of the operating system on which IP-Symcon is running

DESCRIPTION

The function returns a string with the system language of the operating system on which IP-Symcon is running.

EXAMPLE

```
echo IPS_GetSystemLanguage();
// Sample output:
/*
// German
de_DE

// English
en_US
*/
```

IPS_GetUpdateChannel

Require: Symcon >= 6.1

```
string IPS_GetUpdateChannel ()
```

RETURNS

Currently configured update channel

DESCRIPTION

The function returns a string with the currently configured update channel.

EXAMPLE

```
echo IPS_GetUpdateChannel(); //returns e.g.: stable, beta or testing
```

IPS_LogMessage

```
bool IPS_LogMessage (string $Sender, string $Message)
```

PARAMETERS

Sender	The displayed sender of the message
Message	The message to be displayed

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function writes the message **Message** with the sender **Sender** into the IP-Symcon log file. The string **Sender** should indicate the originator of the message, e.g., the name of the script. The message can be observed live in the message window as "Custom Message".

EXAMPLE

```
$ScriptStart = microtime(true);
// calculate a lot...
$ScriptRuntime = microtime(true) - $ScriptStart;
IPS_LogMessage($_IPS['SELF'], "Runtime is ". $ScriptRuntime. "sec");
```

Management of Scripts

ACTIONS

Functions in this group are used to manage actions.

IPS_ActionExists	checks whether a specific action exists
IPS_GetActionForm	returns the dynamically generated configuration form of an action
IPS_GetActionReadableCode	returns an action with its parameters as readable PHP code
IPS_GetActions	returns all available actions as a JSON encoded string
IPS_GetActionsByEnvironment	returns the actions offered for a target object in an environment
IPS_UpdateFormField	changes the parameter of a form field in an action form

IPS_CREATESCRIPT

creates a script

IPS_DELETESCRIPT

deletes a script

IPS_GETSCRIPT

returns extensive information about a script

IPS_GETSCRIPTCONTENT

returns the content of a script

IPS_GETSCRIPTEVENTLIST

returns all events that are assigned to a script

IPS_GETSCRIPTFILE

returns the file name of a script

IPS_GETSCRIPTIDBYFILE

returns the ID of a script by its file

IPS_GETSCRIPTIDBYNAME

returns the ID of a script by its name

IPS_GETSCRIPTLIST

returns a list of all scripts

IPS_GETSCRIPTTIMER

returns the start value of the script timer

IPS_SCRIPTEXISTS

checks if a script exists

IPS_SETSCRIPTCONTENT

sets the content of a script

IPS_SETSCRIPTFILE

sets the file name for a script

IPS_SETSCRIPTTIMER

sets the start value of a script timer

THREADS

IPS_GetScriptThread	returns information about a thread of the script engine
IPS_GetScriptThreadList	returns the IDs of all threads of the script engine
IPS_ScriptThreadExists	checks whether a specific thread of the script engine exists

Actions

IPS_ActionExists	checks whether a specific action exists
IPS_GetActionForm	returns the dynamically generated configuration form of an action
IPS_GetActionReadableCode	returns an action with its parameters as readable PHP code
IPS_GetActions	returns all available actions as a JSON encoded string
IPS_GetActionsByEnvironment	returns the actions offered for a target object in an environment
IPS_UpdateFormField	changes the parameter of a form field in an action form

IPS_ActionExists

```
bool IPS_ActionExists (string $ActionID)
```

PARAMETERS

ActionID	GUID of the action to be checked

RETURNS

If the action exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the action with the GUID **ActionID** exists. Actions of libraries only exist while the library is installed. An invalid GUID does not cause an error but returns **FALSE**.
IPS_GetActions returns all available actions.

EXAMPLE

```
if (IPS_ActionExists('{346AA8C1-30E0-1663-78EF-93EFADFAC650}')) {
echo 'The action "Execute PHP Code" is available!';
}
```

IPS_GetActionForm

```
string IPS_GetActionForm (string $ActionID, array $Parameters)
```

PARAMETERS

ActionID	GUID of the action
Parameters	Parameters of the action, at least TARGET, PARENT and ENVIRONMENT

RETURNS

A JSON encoded string with the array of form elements.

DESCRIPTION

The function generates the configuration form of the action with the GUID **ActionID**. The form of an action (entry **form**, see IPS_GetActions) is either static, i.e. an array of form elements, or dynamic as PHP code. Only dynamic forms can be generated with this function. The PHP code is run with the **Parameters** as system variables (e.g. `$_IPS['TARGET']`), and the returned array is returned JSON encoded. This way, the form can depend on the target object or on parameters that were already selected. The function waits until the execution has finished.
The **Parameters** need to include the following entries:

PARAMETER	DESCRIPTION
TARGET	ID of the target object of the action (-1 for general actions)
PARENT	ID of the object that executes the action, e.g. the PHP script or the flow script
ENVIRONMENT	The environment of the action, see Environments

Further entries, e.g. parameters that were already selected, are available to the PHP code as well. Only Boolean, Integer, Float and String are possible as values.
The checks are done in this order:
If **ActionID** is not a valid GUID, the function fails with "ActionID ... is not a valid GUID".
If the action does not exist, it fails with "Action with ID ... not found!".
If a required parameter is missing, it fails with "TARGET is missing in Parameters", "PARENT is missing in Parameters" or "ENVIRONMENT is missing in Parameters".
For a static form, it fails with "GetActionForm should only be called for dynamic forms (string, array of strings)!".
The captions in the form are not translated, the translations are in the entry **locale** of the action.

EXAMPLE

```
$form = IPS_GetActionForm('{46B65CA6-3098-4982-9A6C-B89DADBB0A96}', [
'TARGET' => 12345, // Boolean variable
'PARENT' => $_IPS['SELF'],
'ENVIRONMENT' => 'Default'
]);
print_r(json_decode($form, true));

/* e.g. returns:
Array
(
[0] => Array
(
[type] => Select
[name] => VALUE
[caption] => Value
[options] => Array
(
[0] => Array
(
[value] =>
[caption] => Off
)

[1] => Array
(
[value] => 1
[caption] => On
)

)

)

)
*/
```

IPS_GetActionReadableCode

```
string IPS_GetActionReadableCode (string $ActionID, array $Parameters)
```

PARAMETERS

ActionID	GUID of the action
Parameters	Parameters of the action, at least TARGET, PARENT and ENVIRONMENT

RETURNS

PHP code that executes the action with the given parameters.

DESCRIPTION

The function returns the action with the GUID **ActionID** and the parameters **Parameters** as readable PHP code, which can e.g. be inserted into a script. The management console uses this in the "Add command" dialog of the script editor.

If the action has no entry **readable**, the PHP code of the action (entry **action**) is returned, with every `$_IPS['NAME']` replaced by the value of the parameter NAME: Boolean as true/false, Integer and Float as number, String in double quotes. Parameters that are not passed remain as `$_IPS['...']`. If the action is defined as an array of code lines, each line ends with a line break.

If the action has an entry **readable**, this PHP code is run with the parameters as system variables and its output is returned. The function waits until the execution has finished.

The **Parameters** need to include the following entries:

PARAMETER	DESCRIPTION
TARGET	ID of the target object of the action (-1 for general actions)
PARENT	ID of the object that executes the action, e.g. the PHP script or the flow script
ENVIRONMENT	The environment of the action, see Environments

Only Boolean, Integer, Float and String are possible as values. If the action does not exist, the function fails with "Action with ID ... not found!", then the required parameters are checked ("TARGET is missing in Parameters", "PARENT is missing in Parameters", "ENVIRONMENT is missing in Parameters"). If the entry **readable** is invalid, it fails with "Readable Field of Action ... is invalid!".

An action is executed with IPS_RunAction or IPS_RunActionWait.

EXAMPLE

```
echo IPS_GetActionReadableCode('{46B65CA6-3098-4982-9A6C-
B89DADBB0A96}', [
'TARGET' => 12345, // Boolean variable
'PARENT' => $_IPS['SELF'],
'ENVIRONMENT' => 'Default',
'VALUE' => true
]);

/* returns:
RequestAction(12345, true);
*/
```

IPS_GetActions

```
string IPS_GetActions ()
```

RETURNS

A JSON encoded string with an array of all actions. After decoding with `json_decode($json, true)`, each action is an array with the following entries:

INDEX	TYPE	DESCRIPTION
id	string	GUID of the action (ActionID)
caption	string	Caption of the action (English, the translations are in **locale**)
form	array/string	Configuration form of the action: an array of form elements (static form) or PHP code as a string or an array of code lines (dynamic form, see IPS_GetActionForm)
action	array/string	PHP code that is run when the action is executed, as a string or an array of code lines
restrictions	array	(optional) Restrictions under which the action is offered, see restrictions
locale	array	(optional) Translations per language, e.g. **de** => [English text => translation]
category	string	(optional) Category of the action (target, math, set, switch, expert or other)
description	string	(optional) Description of the action for the action selection
format	string	(optional) Short format of the action with its parameters in ICU syntax, e.g. "Switch to {VALUE, valueFormatted}"
priority	integer	(optional) Priority by which the actions are sorted in a selection (higher values first)
readable	array/string	(optional) PHP code that returns the action as readable code, see IPS_GetActionReadableCode
deprecated	array/string	(optional) Marks the action as deprecated. Contains PHP code that converts the parameters into a successor action (returns **actionID** and **parameters**)
actionList	string	(optional) Name of the parameter that holds a nested list of actions (e.g. "ACTIONS" for "If ..., then" in flow scripts)

The entries are exactly those of the action definition. Optional entries are missing if the definition does not specify them.

DESCRIPTION

The function returns all actions known to the system. These are the built-in actions and the actions from the "actions" folder of installed libraries, see Actions. Actions are predefined operations that are executed by events, flow scripts, variable actions as well as IPS_RunAction and IPS_RunActionWait.
The result is a JSON encoded string, not a PHP array, and needs to be decoded with `json_decode`. The order of the entries is not guaranteed. The list is not filtered and also contains deprecated actions (**deprecated**). The texts (**caption**, **description**, **format**, form) are English, the translations are in the entry **locale**.
The actions offered for a specific target object in a specific environment are returned by IPS_GetActionsByEnvironment. IPS_ActionExists checks whether an action exists.

EXAMPLE

```
$actions = json_decode(IPS_GetActions(), true);
foreach ($actions as $action) {
echo $action['id'] . ' -> ' . $action['caption'] . PHP_EOL;
}

/* e.g. returns:
{04603AA4-F1B8-3E28-8F31-8BC98C75A04B} -> Multiply with Value of other
Variable
{0792E8F6-07B1-C12F-D8C0-7CB048CABFB3} -> Wait for Value
{07DEB4D2-D32C-4226-8371-E8CD1A507D99} -> Switch to Value
...
*/
```

IPS_GetActionsByEnvironment

```
string IPS_GetActionsByEnvironment (int $TargetID, string $Environment,
bool $IncludeDefault)
```

PARAMETERS

TargetID	ID of the target object of the actions or -1 for general actions without a target object
Environment	Environment in which the actions are offered, e.g. "Default", "EventTrigger" or "FlowScript"
IncludeDefault	If **TRUE**, the actions of the environment "Default" are returned as well

RETURNS

A JSON encoded string with an array of the matching actions. Each action has the same structure as in IPS_GetActions.

DESCRIPTION

The function returns the actions that are offered for the target object with the ID **TargetID** in the environment **Environment**. The management console shows the same selection, e.g. in events or flow scripts. The result is a JSON encoded string and needs to be decoded with `json_decode`.
The environment is an arbitrary string that is compared with the restrictions **includeEnvironments** and **excludeEnvironments** of the actions. The management console uses the following environments, see also Environments:

ENVIRONMENT	USAGE
Default	Default environment in which most actions are defined
EventTrigger	Action selection in a triggered event
EventCyclic	Action selection in a cyclic event
EventSchedule	Action selection in a weekly schedule
FlowScript	Action selection in a flow script
ScriptEditor	Action selection in the "Add command" dialog of the script editor
TestCommands	Action selection in the "Test commands" dialog of an instance

The following rules apply to each action:
An action is not offered if **Environment** is listed in **excludeEnvironments**.

An action is offered if **Environment** is listed in **includeEnvironments**, or if **IncludeDefault** is **TRUE** and **includeEnvironments** contains "Default". Without **includeEnvironments**, the action counts as an action of the environment "Default".
If **TargetID** is -1, only general actions without a target object are returned (restriction **general**, e.g. "Wait").
For any other **TargetID**, the object has to exist, otherwise the function fails with "Object #... does not exist". General actions are not returned in this case. Actions without restrictions are only returned if **IncludeDefault** is **TRUE** or **Environment** is "Default". All further restrictions (object type, variable type, presentation, profile, variable action, read-only, module, idents, script type, event type) are checked against the target object, see restrictions.
The order of the entries is not guaranteed. The list also contains deprecated actions (**deprecated**). The management console sorts by **priority** and hides deprecated actions.

EXAMPLE

```
// Actions for a Boolean variable with a variable action
$actions = json_decode(IPS_GetActionsByEnvironment(12345 /*[Living Room
Light]*/, 'Default', true), true);
foreach ($actions as $action) {
echo $action['id'] . ' -> ' . $action['caption'] . PHP_EOL;
}

/* e.g. returns (excerpt):
{46B65CA6-3098-4982-9A6C-B89DADBB0A96} -> Switch to Value
{A7A928A3-86AB-4FFA-AECA-39C0A2E4933D} -> Set Visibility
{EF078D00-FBD5-8857-F5DC-F5B54B107DD3} -> Toggle Value
...
*/

// General actions without a target object, as offered by a flow script
$general = json_decode(IPS_GetActionsByEnvironment(-1, 'FlowScript',
true), true);
```

IPS_UpdateFormField

```
bool IPS_UpdateFormField (string $Field, string $Parameter, mixed
$Value, string $SessionID)
```

PARAMETERS

Field	Name of the form field to be changed
Parameter	Name of the parameter to be changed
Value	New value of the parameter
SessionID	SessionID of the open action form

RETURNS

If the function is executed successfully, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the parameter **Parameter** of the form field named **Field** in the open form of an action to the value **Value**. The form is identified by the **SessionID**. Within the form fields of an action, e.g. in "onChange", the SessionID is available in the variable $id.
The function is the counterpart of the module function UpdateFormField (`$this->UpdateFormField`), which changes form fields in the open configuration forms of an instance. As the form of an action does not belong to an instance, it is addressed via the SessionID instead.
The change is sent as a message to the connected clients, e.g. the console. It is not checked whether a form with this SessionID is open or whether the form field exists. In this case the call has no effect, the function still returns **TRUE**.



It is not possible to change the parameters **name** and **type** of form fields. Other parameters that cannot be changed are listed in the respective documentation.



Only Boolean, Integer, Float and String are possible as **Value**. For complex parameters, i.e. lists or objects, the new value must be passed JSON-encoded.

EXAMPLE

```
// Dynamic form of an action (entry "form"): If another status variable
is selected,
// the target of the field "ACTION" is adjusted in the open action
form. $id contains the SessionID
return [
[
'type' => 'Select',
'name' => 'IDENT',
'caption' => 'Status Variable',
'options' => $options,
'onChange' => 'IPS_UpdateFormField("ACTION", "targetID",
IPS_GetObjectIDByIdent($IDENT, ' . $_IPS['TARGET'] . '), $id);'
],
[
'type' => 'SelectAction',
'name' => 'ACTION',
'caption' => 'Action'
]
];
```

IPS_CreateScript

```
int IPS_CreateScript (int $ScriptType)
```

PARAMETERS

	VALUE	CONSTANT	DESCRIPTION
ScriptType	0	SCRIPTTYPE_PHP	generates a PHP script
	1	SCRIPTTYPE_FLOW	generates a Flow Script
	2	SCRIPTTYPE_ IPSWORKFLOW	generates an IPSWorkflow

RETURNS

ID of the newly created script

DESCRIPTION

This function creates a script of the type **ScriptType**.
The function returns an ID that can help to uniquely identify the generated script. The function automatically generates a corresponding script file, usually named ScriptID.ips.php, e.g., "43954.ips.php". It is possible to use another file with the function IPS_SetScriptFile. In this case, it is recommended to delete the previous script file to avoid unnecessary script files.

EXAMPLE

```
$ScriptID = IPS_CreateScript(SCRIPTTYPE_PHP);
echo"The Script ID is: ". $ScriptID;
```

IPS_DeleteScript

```
bool IPS_DeleteScript (int $ScriptID, bool $DeleteFile)
```

PARAMETERS

ScriptID	ID of the script
DeleteFile	TRUE if the file should be deleted. FALSE if the file should be moved to the 'deleted' folder.

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function removes the script with the ID **ScriptID** from the internal list of scripts. The script no longer appears in the object tree and cannot be edited.
If the script file is not permanently removed by setting **DeleteFile**, it is moved to the "/scripts/deleted/" folder.



A script can only be deleted if it has no child objects.

EXAMPLE

```
IPS_DeleteScript($ScriptID, true); // Delete script including file
```

IPS_GetScript

```
array IPS_GetScript (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the script

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
IsBroken (until 3.4)	boolean	TRUE if an error occurred during the last execution of the script, otherwise FALSE (since 4.0 replaced by ScriptIsBroken)
LastExecute (until 3.4)	integer	Unix timestamp of the last call (since 4.0 replaced by ScriptExecuted)
ScriptCRC (since 5.1)	string	CRC32 of the file
ScriptIsBroken (since 4.0)	boolean	TRUE if an error occurred during the last execution of the script, otherwise FALSE
ScriptExecuted (since 4.0)	integer	Unix timestamp of the last call
ScriptFile	string	Filename of the script
ScriptID	integer	ID of the script
ScriptSize (since 5.1)	integer	Size in bytes
ScriptType	integer	Type of the script 0 `SCRIPTTYPE_PHP` PHP script 1 `SCRIPTTYPE_FLOW` Flow script 2 `SCRIPTTYPE_IPSWORKFLOW` IPSWorkflow
ScriptUpdated (since 5.1)	integer	Unix timestamp of the last update

DESCRIPTION

This function returns an array containing information about the script with the ID **ScriptID**.

EXAMPLE

```
// Since version 5.1
print_r(IPS_GetScript(46413));
/* Exemplary output:
Array
(
[ScriptID] => 46413
[ScriptType] => 0
[ScriptFile] => 46413.ips.php
[ScriptUpdated] => 1542622765
[ScriptExecuted] => 1542295981
[ScriptCRC] => 8132DEB6
[ScriptSize] => 42
[ScriptIsBroken] =>
)
*/

// Since version 4.0
print_r(IPS_GetScript(46413));
/* Exemplary output:
Array
(
[ScriptIsBroken] =>
[ScriptExecuted] => 1204933792
[ScriptFile] => 46413.ips.php
[ScriptID] => 46413
[ScriptType] => 0
)
*/

// Until version 3.4
print_r(IPS_GetScript(46413));
/* Exemplary output:
Array
(
[IsBroken] =>
[LastExecute] => 1204933792
[ScriptFile] => 46413.ips.php
[ScriptID] => 46413
[ScriptType] => 0
)
*/
```

IPS_GetScriptContent

Require: Symcon >= 3.1

```
string IPS_GetScriptContent (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the script

RETURNS

The content of the script

DESCRIPTION

This function returns the content of the script with the ID **ScriptID**. The content is read from the script file that is defined within the script object.



A practical example for this function is reading a script into an external editor that has no access to the file system, e.g., when using the JSON-RPC API.

EXAMPLE

```
$Content = IPS_GetScriptContent($ScriptID);
```

IPS_GetScriptEventList

```
array IPS_GetScriptEventList (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the script

RETURNS

An array of integer values containing the IDs of all events that are assigned to the script **ScriptID**. The order of the entries is not guaranteed.

DESCRIPTION

This function returns an array whose elements are the respective IDs of the events of the script with the ID **ScriptID**, i.e., the events that are direct children of the script. The events can be further processed by the function IPS_GetEvent.

EXAMPLE

```
$events = IPS_GetScriptEventList($_IPS['SELF']);
print_r($events); // determines all events of the current script
```

IPS_GetScriptFile

```
string IPS_GetScriptFile (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the script

RETURNS

File name of the script (relative to the "/scripts" folder)

DESCRIPTION

This function returns the file name of the script with the ID **ScriptID**. An exemplary use for this function is to include that file directly within other scripts.

EXAMPLE

```
include(IPS_GetScriptFile($ScriptID));
```

IPS_GetScriptIDByFile

```
int IPS_GetScriptIDByFile (string $FilePath)
```

PARAMETERS

FilePath	Relative path to the file from the scripts folder

RETURNS

ID of the found script, otherwise FALSE

DESCRIPTION

This function tries to determine the ID of the script whose script file matches **FilePath**. If a script with that file exists, its ID is returned, otherwise FALSE.



This function generates a **warning** if the file was not found. Use the @ operator, if you want to suppress this message. The exact function of this operator is described in the PHP manual.

EXAMPLE

```
$ScriptID = @IPS_GetScriptIDByFile("12345.ips.php");
if ($ScriptID === false)
echo "Script file not found!";
else
echo "The ID of the script is: ". $ScriptID;
```

IPS_GetScriptIDByName

```
int IPS_GetScriptIDByName (string $ScriptName, int $ParentID)
```

PARAMETERS

ScriptName	Name of the script
ParentID	Object whose direct child objects are searched for the script

RETURNS

ID of the found script, otherwise FALSE

DESCRIPTION

This function tries to determine the ID of a script with the name **ScriptName** and the parent with the ID **ParentID**. It returns the ID of the first found script whose name matches with **ScriptName**. If such a script is found, its ID is returned, otherwise FALSE.
In addition to the name, this function uses an additional criterion (**ParentID**) for the search. Thus, the probability to find the intended ID is considerably larger. Remember:



Caution: Names in IP Symcon can be chosen freely, so they are **not unique**. For this reason, the function can return a wrong ID.



This function generates a **warning** if the name was not found. Use the @ operator, if you want to suppress this message. The exact function of this operator is described in the PHP manual.

EXAMPLE

```
$ScriptID = @IPS_GetScriptIDByName("Rain sensing", $ParentID);
if ($ScriptID === false)
echo "Script not found!";
else
echo "The Script ID is: ". $ScriptID;
```

IPS_GetScriptList

```
array IPS_GetScriptList ()
```

RETURNS

An array of integer values containing the IDs of all scripts. The order of the entries is not guaranteed.

DESCRIPTION

This function determines the IDs of all scripts that are registered within IP-Symcon. The IDs are listed in an array. If no scripts exist, the array is empty.

EXAMPLE

```
$allScripts = IPS_GetScriptList();
print_r($allScripts);

/* returns, e.g.:
Array
(
[0] => 37659
[1] => 18326
etc. ...
*/
```

IPS_GetScriptTimer

```
int IPS_GetScriptTimer (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the script

RETURNS

Time in seconds that the script is called cyclically (not the remaining duration)

DESCRIPTION

The function returns the start value of the script timer of the script with the ID **ScriptID**. The value describes the duration in seconds. The function can be used to check if the value is set correctly.
If the start value is 0, the timer is deactivated and the script is not executed by the timer. As IPS_SetScriptTimer only accepts values less than 86400 (one day), the value is between 0 and 86399.

EXAMPLE

```
$TimerValue = IPS_GetScriptTimer($ScriptID); // determines the value of
the script timer
```

IPS_ScriptExists

```
bool IPS_ScriptExists (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the script

RETURNS

If the script exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

This function checks if the script with the ID **ScriptID** exists.

EXAMPLE

```
if (IPS_ScriptExists(34881))
echo "A script with this ID exists!";
```

IPS_SetScriptContent

Require: Symcon >= 3.1

```
bool IPS_SetScriptContent (int $ScriptID, string $Content)
```

PARAMETERS

ScriptID	ID of the script
Content	Content, including PHP tags (<? and ?>)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the content of the script with the ID **ScriptID**. The function replaces the content of the script file.



This function should be used with care as the previous content is overwritten without a warning.

EXAMPLE

```
$ScriptID = IPS_CreateScript(SCRIPTTYPE_PHP);
IPS_SetName($ScriptID, "My Script");
IPS_SetScriptContent($ScriptID, "<? echo 'Test!'; ?>");
```

IPS_SetScriptFile

```
bool IPS_SetScriptFile (int $ScriptID, string $FileName)
```

PARAMETERS

ScriptID	ID of the script
FileName	File name of the PHP file (relative to the "/scripts" folder)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function binds the file with the name **FileName** to the script with the ID **ScriptID**.
Usually, the file name of a script consists of the ID and the extension ".ips.php". The use of this function enables different file names. The file must already exist, otherwise the function fails with "File #... does not exist".
The file name is not connected to the script name, which is shown in the object tree. The script and file name can be completely different. The ID of the script is also unaffected by this function. Neither the name nor the script file name can be used to directly identify the script. The numerical ID of the script is used for direct identification.

EXAMPLE

```
$ScriptName = "Example.ips.php"; //script file
$ScriptID = IPS_CreateScript(SCRIPTTYPE_PHP);
IPS_SetName($ScriptID, "Example");

$Script = IPS_GetScript($ScriptID);
//Rename the file to $ScriptName
rename(IPS_GetKernelDir().'/scripts/'.$Script['ScriptFile'],
IPS_GetKernelDir().'/scripts/'.$ScriptName);
//Assign the new file name ("Example.ips.php") to the script
IPS_SetScriptFile($ScriptID, $ScriptName);
```

IPS_SetScriptTimer

```
bool IPS_SetScriptTimer (int $ScriptID, int $TimerValue)
```

PARAMETERS

ScriptID	ID of the script
TimerValue	Time in seconds in which the script is called cyclically

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the start value of the script timer of the script with the ID **ScriptID** to **TimerValue**. The value defines the time in seconds. The timer from the script counts down from the start value in steps of seconds. When the timer reaches 0 the script is executed and the script timer is reset to the start value. This process is repeated until the timer is explicitly deactivated. A timer is deactivated by setting the start value to 0. Negative values are treated like 0.



The duration of the timer can only be set on an exact second value if the timer value is 3600 or less. For values above 3600, irregularities can occur, depending on start time and/or divisor as the script timer shares the same limitations as normal cyclic events and their time templates. E.g., if a timer is set to 5 hours at 6:00, it triggers at 6:00, 11:00, 16:00, 21:00, and 1:00, even though 2:00 could have been expected.

TimerValue must be less than 86400 (one day). For larger values the function fails with "Interval must be less than 86400 seconds (1 day). Use a cyclic event for longer intervals" and the existing timer stays unchanged. Use a cyclic event for longer intervals, see IPS_SetEventCyclic.

EXAMPLE

```
// The script is run every 10 seconds
IPS_SetScriptTimer($ScriptID, 10);
```

Threads

IPS_GetScriptThread	returns information about a thread of the script engine
IPS_GetScriptThreadList	returns the IDs of all threads of the script engine
IPS_ScriptThreadExists	checks whether a specific thread of the script engine exists

IPS_GetScriptThread

```
array IPS_GetScriptThread (int $ThreadID)
```

PARAMETERS

ThreadID	ID of the thread

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
ThreadID	integer	ID of the thread
ExecuteCount	integer	Number of executions in this thread so far
ExecutionMin	integer	Shortest execution time in milliseconds
ExecutionAvg	integer	Average execution time in milliseconds
ExecutionMax	integer	Longest execution time in milliseconds
StartTime	integer	Unix timestamp of the start of the current execution, 0 if the thread is idle
Sender	string	Trigger of the current execution (value of $_IPS['SENDER'], e.g. "RunScript" or "TimerEvent"), empty if the thread is idle
SenderID	integer	ID of the triggering object (the event, the target instance or the script), 0 if the thread is idle (since version 9.0)
FilePath	string	What is currently executed: the script file relative to the program directory (e.g. "scripts/12345.ips.php"), the first 100 characters for script text, the executed function or timer for PHP modules, RunID and step for flow scripts; empty if the thread is idle
ScriptID	integer	ID of the executed script, 0 for script text or if the thread is idle
PeakMemoryUsage	integer	Highest memory usage of an execution in bytes since the last memory cleanup
RetainedMemoryUsage	integer	Memory in bytes that the thread keeps between executions, 0 after an unused thread has released its memory
MemoryCleanups	integer	Number of memory cleanups. If the memory usage of an execution exceeds the limit ThreadMemoryCleanup (default 10 MB), the thread releases its memory afterwards

DESCRIPTION

The function returns information about the thread with the ID **ThreadID**: the statistics of the previous executions, the memory usage and, if the thread is currently busy, the current execution. If the thread does not exist, the function fails with "ScriptThread #... does not exist".
PHP scripts are executed in a fixed pool of threads. The number of threads is set by the special switch ThreadCount (default 50, at least 25), and the threads have the IDs 1 to ThreadCount. Within a script, the system variable `$_IPS['THREAD']` contains the ID of the own thread.
IPS_GetScriptThreadList returns the IDs of all threads, IPS_GetScriptThreads returns the information of several threads at once.

EXAMPLE

```
// Information about the own thread
print_r(IPS_GetScriptThread($_IPS['THREAD']));

/* e.g. returns:
Array
(
[ThreadID] => 3
[ExecuteCount] => 152
[ExecutionMin] => 0
[ExecutionAvg] => 12
[ExecutionMax] => 1530
[StartTime] => 1758880000
[Sender] => RunScript
[SenderID] => 12345
[FilePath] => scripts/12345.ips.php
[ScriptID] => 12345
[PeakMemoryUsage] => 2097152
[RetainedMemoryUsage] => 2097152
[MemoryCleanups] => 0
)
*/
```

IPS_GetScriptThreadList

```
array IPS_GetScriptThreadList ()
```

RETURNS

An array of integer values with the IDs of all threads of the script engine. The order of the entries is not guaranteed.

DESCRIPTION

The function returns the IDs of all threads in which PHP scripts are executed. Without PHP support, the array is empty.
PHP scripts are executed in a fixed pool of threads. The number of threads is set by the special switch ThreadCount (default 50, at least 25), and the threads have the IDs 1 to ThreadCount. Within a script, the system variable `$_IPS['THREAD']` contains the ID of the own thread.
IPS_GetScriptThread returns information about a thread.

EXAMPLE

```
print_r(IPS_GetScriptThreadList());

/* e.g. returns:
Array
(
[0] => 1
[1] => 2
[2] => 3
etc. ...
)
*/
```

IPS_ScriptThreadExists

```
bool IPS_ScriptThreadExists (int $ThreadID)
```

PARAMETERS

ThreadID	ID of the thread to be checked

RETURNS

If the thread exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the thread with the ID **ThreadID** exists.
PHP scripts are executed in a fixed pool of threads. The number of threads is set by the special switch ThreadCount (default 50, at least 25), and the threads have the IDs 1 to ThreadCount. Within a script, the system variable `$_IPS['THREAD']` contains the ID of the own thread. Without PHP support, no threads exist.

EXAMPLE

```
if (IPS_ScriptThreadExists(1)) {
print_r(IPS_GetScriptThread(1));
}
```

Management of Traces

IPS_GetTraceIDList	returns the IDs of the trace entries of an object
IPS_GetTraceList	returns a trace entry with its cause chain

IPS_GetTraceIDList

```
array IPS_GetTraceIDList (int $ObjectID)
```

PARAMETERS

ObjectID	ID of the object whose trace entries are to be returned

RETURNS

An array with the IDs (TraceID) of the kept trace entries of the object, oldest first.

DESCRIPTION

In traces the kernel records what caused what: every entry represents one operation – e.g. writing a variable, the event triggered by it, its action or script, a RequestAction, a timer or a request to the WebServer – and refers to the entry that caused it via **ParentID** and to the entries it caused itself via **ChildIDs**. For every cause this results in a tree that shows, for example, why a variable has changed.
The function returns the IDs of all kept entries that belong to the object **ObjectID** – e.g. the write operations of a variable or the executions of an event. These are at most the last 25 entries. For an object without entries or a non-existing ID an empty array is returned. If 1 is passed as **ObjectID**, the IDs of all kept entries are returned in arbitrary order.
The entries themselves and their cause chain are returned by IPS_GetTraceList.
Traces are only kept in memory and are lost on a restart. The last 25 entries are kept per object, and the last 1000 entries in total for entries without an object (e.g. WebServer). An older entry is kept as long as a kept entry still descends from it, so chains stay complete. When an object is deleted, its entries are removed. Strings in **Data** are shortened to 100 bytes and marked with "...".

EXAMPLE

```
print_r(IPS_GetTraceIDList(10018));

/* e.g. returns:
Array
(
[0] => 1
[1] => 2
)
*/
```

IPS_GetTraceList

```
array IPS_GetTraceList (int $TraceID)
```

PARAMETERS

TraceID	ID of the trace entry

RETURNS

An array of trace entries, starting with the entry **TraceID**, followed by its parent entry and so on up to the start of the chain. Each entry contains the following information as key => value pairs:

INDEX	TYPE	DESCRIPTION
TraceID	integer	Unique, ascending ID of the entry
ParentID	integer	ID of the entry that caused this entry (0 = start of a chain)
ChildIDs	array	IDs of the entries caused by this entry
Depth	integer	Position in the chain, starting at 1 (0 = placeholder for an entry that is no longer available)
ObjectID	integer	ID of the object the entry belongs to (1 = no object, e.g. for WebServer)
Sender	string	Kind of entry (see sender table)
ExecutionUser	string	User on whose behalf it was executed (e.g. @admin; @unknown if no user was logged in)
TimeStamp	integer	Unix timestamp of the creation
Data	array	Additional information as key => value pairs, depending on the sender (see sender table)

Sender table

SENDER	DESCRIPTION	OBJECTID	DATA
Variable	A value was written to a variable (even without a change)	Variable	VariableID, Value, OldValue
RequestAction	The action of a variable was requested	Variable	VariableID, ActionID, Value, Sender
RequestAction	The RequestAction function of an instance was called for a status variable	Instance	Ident, Value

SENDER	DESCRIPTION	OBJECTID	DATA
TriggerEvent	A triggered event runs its action	Event	EventID, ActionID, Environment, Trigger, VariableID, Value
CyclicEvent	A cyclic event runs its action	Event	EventID, ActionID, Environment
Script	A script is executed	Script	Sender, ScriptID, FilePath, SenderID
Event	PHP code is executed on behalf of an event	Event	Sender, ScriptID, FilePath, SenderID
Instance	PHP code is executed for a target object	Target object	Sender, ScriptID, FilePath, SenderID
Timer	A timer of an instance is executed	Instance	TimerName, Instance, RunOnce
DataFlow	An instance receives data through the data flow	Receiving instance	InstanceID, ParentID
WebServer	A request to the WebServer (file/hook) or a JSON-RPC call	1	Path or Method, RemoteIP, ProxyIPs (only behind proxies)
System	Placeholder for an entry that is no longer available	1	Reason

DESCRIPTION

In traces the kernel records what caused what: every entry represents one operation – e.g. writing a variable, the event triggered by it, its action or script, a RequestAction, a timer or a request to the WebServer – and refers to the entry that caused it via **ParentID** and to the entries it caused itself via **ChildIDs**. For every cause this results in a tree that shows, for example, why a variable has changed.
The function returns the entry **TraceID** and all entries that caused it: the first element is the entry itself, the last element is the start of the chain (**ParentID** 0). The entries caused by an entry are listed in **ChildIDs** and can also be retrieved with this function. The IDs of the entries of an object are returned by IPS_GetTraceIDList.
If an entry of the chain is no longer available, the list ends with a placeholder: **Depth** 0, **Sender** "System", **ObjectID** 1 and the reason in **Data → Reason**. If there is no entry with the ID **TraceID**, the function fails with "No trace with ID … found".
Traces are only kept in memory and are lost on a restart. The last 25 entries are kept per object, and the last 1000 entries in total for entries without an object (e.g. WebServer). An older entry is kept as long as a kept entry still descends from it, so chains stay complete. When an object is deleted, its entries are removed. Strings in **Data** are shortened to 100 bytes and marked with "...".

EXAMPLE

```
print_r(IPS_GetTraceList(1));

/* e.g. returns:
```

```
Array
(
[0] => Array
(
[ChildIDs] => Array
(
)
[Data] => Array
(
[OldValue] => 0
[Value] => 5
[VariableID] => 10018
)
[Depth] => 1
[ExecutionUser] => @admin
[ObjectID] => 10018
[ParentID] => 0
[Sender] => Variable
[TimeStamp] => 1700000000
[TraceID] => 1
)
)
*/

// Output the cause chain of the newest entry of a variable
$ids = IPS_GetTraceIDList(10018);
if (count($ids) > 0) {
foreach (IPS_GetTraceList(end($ids)) as $entry) {
echo $entry['Depth'] . ': ' . $entry['Sender'] . ' #' .
$entry['ObjectID'] . PHP_EOL;
}
}
```

Management of Variables

.

IPS_CREATEVARIABLE

creates a new variable

IPS_DELETEVARIABLE

deletes a variable

IPS_GETVARIABLE

returns extensive information about a variable

IPS_GETVARIABLEEVENTLIST

returns a list of all events that are associated with a variable

IPS_GETVARIABLEIDBYNAME

returns the ID of a variable by its name

IPS_GETVARIABLELIST

returns a list of all existing variables

IPS_GETVARIABLEPRESENTATION

Returns all parameters of the currently used presentation

IPS_SETVARIABLECUSTOMACTION

sets a user-defined action script for a variable

IPS_SETVARIABLECUSTOMPRESENTATION

assigns a user-defined presentation to the variable

IPS_SETVARIABLECUSTOMPROFILE

sets a user-defined profile for a variable

IPS_VARIABLEEXISTS

checks if a variable exists

VARIABLE PRESENTATION

Functions in this group are used to manage variable presentations

IPS_GetPresentation	provides comprehensive information about a specific presentation
IPS_GetPresentationForm	returns the configuration form of a presentation for a variable type
IPS_GetPresentationFormat	returns the format template of a presentation for a variable type
IPS_GetPresentations	provides all presentations of the system
IPS_GetPresentationValue	returns the preview value of a presentation
IPS_PresentationExists	Checks whether a specific presentation exists
Management of Templates	Functions in this group are used to manage templates

IPS_CreateTemplate	creates a new template for a specific presentation
IPS_DeleteTemplate	deletes a template
IPS_GetTemplate	provides comprehensive information about a template
IPS_GetTemplateList	provides all available templates
IPS_GetTemplateListByPresentation	provides a list of the templates of a presentation
IPS_SetTemplateName	sets the name of a template
IPS_SetTemplateValues	sets the values of a template
IPS_TemplateExists	checks whether a template exists

VARIABLE PROFILES

IPS_CreateVariableProfile	creates a variable profile
IPS_DeleteVariableProfile	deletes a variable profile
IPS_GetVariableProfile	returns extensive information about a variable profile
IPS_GetVariableProfileList	returns a list of all existing variable profiles
IPS_GetVariableProfileListByType	returns a list of all existing variable profiles of a specific type
IPS_SetVariableProfileAssociation	modifies the association table of a variable profile
IPS_SetVariableProfileDigits	sets the number of decimal places of a variable profile

IPS_SetVariableProfileIcon	sets the icon for a variable profile
IPS_SetVariableProfileText	sets the prefix and suffix of a variable profile
IPS_SetVariableProfileValues	sets the minimum value, maximum value, and step size of a variable profile
IPS_VariableProfileExists	checks if a variable profile exists

IPS_CreateVariable

```
int IPS_CreateVariable (int $VariableType)
```

PARAMETERS

	VALUE	CONSTANT	DESCRIPTION
VariableType	0	VARIABLETYPE_BOOLEAN	Creates a variable of type **Boolean**
	1	VARIABLETYPE_INTEGER	Creates a variable of type **Integer**
	2	VARIABLETYPE_FLOAT	Creates a variable of type **Float**
	3	VARIABLETYPE_STRING	Creates a variable of type **String**

RETURNS

ID of the newly created variable

DESCRIPTION

This function creates a new IPS variable of the type **VariableType**.
It returns an ID that can be used to uniquely identify the new variable.

EXAMPLE

```
// Create a float variable with the name "Room temperature living room"
$VarID_RoomTemperature = IPS_CreateVariable(VARIABLETYPE_FLOAT);
IPS_SetName($VarID_RoomTemperature, "Room temperature living room"); //
Name the variable
IPS_SetParent($VarID_RoomTemperature, 12345); // Move the variable
below the object with the ID "12345"
```

IPS_DeleteVariable

```
bool IPS_DeleteVariable (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function deletes an existing variable with the ID **VariableID**.



A variable can only be deleted if it has no child objects.

EXAMPLE

```
// Delete the variable 47788
IPS_DeleteVariable(47788);
```

IPS_GetVariable

```
array IPS_GetVariable (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
VariableID	integer	ID of the variable
VariableProfile	string	Name of the system profile. Empty if no profile is specified
VariableAction	integer	ID of the instance that is notified about desired changes to the variable. 0 for none or deactivated connection
VariableCustomProfile	string	Name of the user defined profile. Empty if no profile is specified
VariableCustomAction	integer	ID of the script that starts on desired changes to the variable from the visualization. If an existing default action is disabled, the non-existent ScriptID 1 (`VARIABLE_ACTION_DISABLED`) is given. 0 (`VARIABLE_ACTION_NONE`) if no custom action is set.
VariableUpdated	integer	Unix timestamp of the most recent variable update
VariableChanged	integer	Unix timestamp of the most recent variable change
VariablePresentation	array	System presentation. Empty array if no presentation is specified.
VariableCustomPresentation	array	User-defined presentation. Empty array if no presentation is specified.
VariableType	integer	Type of the variable 0 `VARIABLETYPE_BOOLEAN` Boolean 1 `VARIABLETYPE_INTEGER` Integer 2 `VARIABLETYPE_FLOAT` Float 3 `VARIABLETYPE_STRING` String

INDEX	TYPE	DESCRIPTION
VariableValue	mixed	Current value of the variable. The data type matches **VariableType**. See also GetValue
VariableIsLocked	boolean	Indicates if the variable is above the variable limit and thus cannot be written

DESCRIPTION

This function returns an array that contains information about the variable with the ID **VariableID**.

EXAMPLE

```
print_r(IPS_GetVariable(40770));
/* Exemplary output:
Array
(
[VariableID] => 40770
[VariableProfile] => ~Switch
[VariableAction] => 0
[VariableCustomProfile] =>
[VariableCustomAction] => 0
[VariableUpdated] => 1246039629
[VariableChanged] => 1246039629
[VariablePresentation] => Array
(
[PRESENTATION] => {4153A8D4-5C33-C65F-C1F3-7B61AAF99B1C}
[PROFILE] => ~Switch
)

[VariableCustomPresentation] => Array
(
)

[VariableType] => 0
[VariableValue] => 1
[VariableIsLocked] =>
)
*/
```

IPS_GetVariableEventList

```
array IPS_GetVariableEventList (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

List of IDs of all events associated with the variable **VariableID**. The order of the entries is not guaranteed.

DESCRIPTION

This function returns an array whose elements are the respective IDs of the events in which the variable with the ID **VariableID** is used. The events can be processed further by the function IPS_GetEvent.

EXAMPLE

```
$events = IPS_GetVariableEventList(12345);
print_r($events); // determines all events associated with the variable
12345
```

IPS_GetVariableIDByName

```
int IPS_GetVariableIDByName (string $VariableName, int $ParentID)
```

PARAMETERS

VariableName	Name of the variable
ParentID	Object whose children are searched

RETURNS

ID of the found variable, otherwise **FALSE**

DESCRIPTION

This function tries to determine the ID of the variable called **VariableName** whose parent has the ID **ParentID**. It will use the ID of the first found variable whose name matches **VariableName**. If such a variable was found, its ID is returned, otherwise **FALSE**.
In addition to the name, the parent ID (**ParentID**) is used as a search criterion. Thus, the probability of finding the intended ID increases significantly. Remember:



Caution: Names in IP-Symcon can be chosen freely, so they are **not unique**. For this reason, the function can return a wrong ID.



This function generates a **warning** if the name was not found. Use the @ operator if you want to suppress this message. The exact functionality of this operator can be found in the PHP manual.

EXAMPLE

```
$VarID = @IPS_GetVariableIDByName("Rain fall", $ParentID);
if ($VarID === false)
echo "Variable not found!";
else
echo "The Variable ID is: ". $VarID;
```

IPS_GetVariableList

```
array IPS_GetVariableList ()
```

RETURNS

An array of integer values containing the IDs of all variables in IP-Symcon. The order of the entries is not guaranteed.

DESCRIPTION

This function determines the IDs of all registered IPS variables in IP-Symcon. The IDs are listed in an array. If no variable exists, the array is empty.

EXAMPLE

```
$allVariables = IPS_GetVariableList();
print_r($allVariables);

/* returns, e.g.:
Array
(
[0] => 37659
[1] => 18326
etc. ...
*/
```

IPS_GetVariablePresentation

Require: Symcon >= 8.1

```
array IPS_GetVariablePresentation (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable whose presentation is to be output

RETURNS

An array with all parameters of the current variable presentation

DESCRIPTION

The function returns the current presentation of the variable with the ID **VariableID**. The presentation is available as an array and contains all presentation parameters.

KEY	DESCRIPTION
PRESENTATION	The ID of the presentation used, formatted as a GUID.
Presentation parameters	The presentation parameters differ depending on the presentation. The parameters are at the same level as the PRESENTATION parameter

POSSIBLE PRESENTATIONS

PRESENTATION
Enumeration
Date/Time
Duration
Color
Legacy Profile
Shutter
Switch
Slider
Web content
Value display
Value input

EXAMPLE

```
print_r(IPS_GetVariablePresentation(12345));
/* Example output
Array
(
[DIGITS] => 0
[CUSTOM_GRADIENT] => []
[ICON] => temperature-half
[DECIMAL_SEPARATOR] => Client
[GRADIENT_TYPE] => 1
[MAX] => 25
[PRESENTATION] => {6B9CAEEC-5958-C223-30F7-BD36569FC57A}
[INTERVALS] => []
[INTERVALS_ACTIVE] =>
[MIN] => 15
[PERCENTAGE] =>
[PREFIX] =>
[STEP_SIZE] => 1
[SUFFIX] => °C
[THOUSANDS_SEPARATOR] =>
[USAGE_TYPE] => 0
)
*/
```

IPS_SetVariableCustomAction

```
bool IPS_SetVariableCustomAction (int $VariableID, int $ScriptID)
```

PARAMETERS

VariableID	ID of the variable
ScriptID	ID of the script that should be used as variable action. The nonexistent ScriptID 1 (VARIABLE_ACTION_DISABLED) can be given to disable the default action. If a valid ScriptID is specified, it is used instead of the default action. To enable the default action (if available), ScriptID 0 (VARIABLE_ACTION_NONE) must be specified.

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function assigns a custom action to the variable with the ID **VariableID**. The custom action is used in the visualization to forward a change in the variable to the proper function of the module. Examples of action scripts can be found here: Action Scripts
Besides a ScriptID, the following values can be specified:

VALUE	CONSTANT	DESCRIPTION
0	VARIABLE_ACTION_NONE	No custom action. The default action is used, if available.
1	VARIABLE_ACTION_DISABLED	The default action is disabled.

EXAMPLE

```
// Set script 44431 as action of variable 12345
IPS_SetVariableCustomAction(12345, 44431);

// Disable the default action of variable 12345
IPS_SetVariableCustomAction(12345, VARIABLE_ACTION_DISABLED);
```

```
// Enable the default action of variable 12345 again
IPS_SetVariableCustomAction(12345, VARIABLE_ACTION_NONE);
```

IPS_SetVariableCustomPresentation

Require: Symcon >= 8.0

```
bool IPS_SetVariableCustomPresentation (int $VariableID, array
$Presentation)
```

PARAMETERS

VariableID	ID of the variable to which the presentation is to be assigned
Presentation	The configuration of the presentation. Details can be found in the section Presentations.

RETURNS

If the function could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

The function assigns the presentation configured in **Presentation** to the variable **VariableID**. If an empty array is passed, the custom presentation is removed.

EXAMPLE

```
// Sets the presentation of a variable to Slider with a minimum,
maximum and suffix
IPS_SetVariableCustomPresentation(12345, ['PRESENTATION' =>
VARIABLE_PRESENTATION_SLIDER, 'MIN' => 0, 'MAX' => 100, 'SUFFIX' => '
%']);

// Sets the presentation of a variable to Enumeration with icons and
colors.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_ENUMERATION,
'DISPLAY' => 1 /* Icon */,
'OPTIONS' => json_encode([
['Value' => 1, 'Caption' => 'Automatic', 'IconActive' => true,
'IconValue' => 'circle-a'],
['Value' => 2, 'Caption' => 'Eco', 'IconActive' => true, 'IconValue' =>
'circle-e', 'Color' => 40448],
['Value' => 3, 'Caption' => 'Manual', 'IconActive' => true, 'IconValue'
=> 'circle-m'],
```

```
  ])
  ]);
```

IPS_SetVariableCustomProfile

Require: Symcon >= 3.0

```
bool IPS_SetVariableCustomProfile (int $VariableID, string
$ProfileName)
```

PARAMETERS

VariableID	ID of the variable
ProfileName	Name of the profile. Available profiles can be queried via IPS_GetVariableProfileList. If an empty string is used, the custom profile is disabled.

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the profile **ProfileName** as custom profile for the variable with the ID **VariableID**.



Before setting the custom profile, the function checks whether the profile exists and whether the profile type matches the variable type.



Standard profiles, which cannot be changed, start with a tilde (~).

EXAMPLE

```
IPS_SetVariableCustomProfile(12345, "~Switch");
```

IPS_VariableExists

```
bool IPS_VariableExists (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

If the variable exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

This function checks if the variable with the ID **VariableID** exists.

EXAMPLE

```
if (IPS_VariableExists(44788))
echo "Variable already exists!";
```

Variable Presentation

IPS_GETPRESENTATION

provides comprehensive information about a specific presentation

IPS_GETPRESENTATIONFORM

returns the configuration form of a presentation for a variable type

IPS_GETPRESENTATIONFORMAT

returns the format template of a presentation for a variable type

IPS_GETPRESENTATIONS

provides all presentations of the system

IPS_GETPRESENTATIONVALUE

returns the preview value of a presentation

IPS_PRESENTATIONEXISTS

Checks whether a specific presentation exists

MANAGEMENT OF TEMPLATES

Functions in this group are used to manage templates

IPS_CreateTemplate	creates a new template for a specific presentation
IPS_DeleteTemplate	deletes a template
IPS_GetTemplate	provides comprehensive information about a template
IPS_GetTemplateList	provides all available templates
IPS_GetTemplateListByPresentation	provides a list of the templates of a presentation
IPS_SetTemplateName	sets the name of a template
IPS_SetTemplateValues	sets the values of a template
IPS_TemplateExists	checks whether a template exists

IPS_GetPresentation

Require: Symcon >= 8.0

```
array IPS_GetPresentation (string $PresentationID)
```

PARAMETERS

PresentationID	The ID of the presentation, formatted as GUID

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
caption	string	The display name of the presentation.
chart	array	An array with parameters for the display of a chart (see chart table)
condition	array	A list of conditions that the parameters of the presentation must fulfill.
conversions	array	A list of statements that are used to adjust parameters for different types of variables
customIcon	boolean	Optional. TRUE if the presentation provides its own parameters for the icon (e.g., ICON)
form	array/string	The configuration form determines how the parameters can be adjusted in the variables dialog or the template manager.
format	array/string	The formatting of values within this presentation
groups	array	Optional. A list of groups with adjustments for specific variable types (see groups table)
id	string	The ID of the presentation, formatted as GUID
locale	array	The localization of the presentation
presentationParameters	array	A list describing the default values of the parameters of this presentation
presentationValue	mixed	The default value which is displayed in the presentation preview. Alternatively, it can contain PHP code as an array of strings that returns the value
restrictions	array	A list of restrictions that defines for which variables this presentation is available

chart table
Each value can either be a number or the name of the parameter whose value is to be used.

INDEX	TYPE	DESCRIPTION
min	integer/float/string	Parameter for the minimum value
max	integer/float/string	Parameter for the maximum value
stepSize	integer/float/string	Parameter for the step size

groups table
The first group whose **restrictions** contain the variable type of the variable is used. Its values extend or override the values of the presentation with the same name.

INDEX	TYPE	DESCRIPTION
id	string	The ID of the group, formatted as GUID
name	string	The name of the group
restrictions	array	Restrictions defining for which variables this group applies. The entry **variableType** is a list of variable types
presentationParameters	array	The default values of the parameters for this group
format	array/string	Optional. The formatting of values for this group
chart	array	Optional. The parameters for the display of a chart for this group
condition	array	Optional. Conditions that the parameters for this group must fulfill

DESCRIPTION

The function returns an array with information about the variable presentation with the ID **PresentationID**.

EXAMPLE

```
IPS_GetPresentation('{6B9CAEEC-5958-C223-30F7-BD36569FC57A}' /* Slider
*/);
```

IPS_GetPresentationForm

Require: Symcon >= 8.0

```
string IPS_GetPresentationForm (string $PresentationID, int
$VariableType, array $Parameters)
```

PARAMETERS

PresentationID	The ID of the presentation, formatted as GUID
VariableType	The type of variable whose form is to be returned
Parameters	The parameter list to be used for the form. It must not be empty and must contain the parameter PRESENTATION.

RETURNS

The form of the described presentation as a JSON-encoded string.

DESCRIPTION

The function returns the form for the presentation **PresentationID**, taking into account the **VariableType** and the **Parameters**.

EXAMPLE

```
IPS_GetPresentationForm('{6B9CAEEC-5958-C223-30F7-BD36569FC57A}' /*
Slider */, 1 /* Integer */, ['PRESENTATION' => '{6B9CAEEC-5958-C223-
30F7-BD36569FC57A}', 'MIN' => 0, 'MAX' => 100]);
```

IPS_GetPresentationFormat

```
string IPS_GetPresentationFormat (string $PresentationID, int
$VariableType)
```

PARAMETERS

PresentationID			The ID of the presentation, formatted as GUID
VariableType			The variable type for which the format is to be returned
	0	VARIABLETYPE_BOOLEAN	Boolean
	1	VARIABLETYPE_INTEGER	Integer
	2	VARIABLETYPE_FLOAT	Float
	3	VARIABLETYPE_STRING	String

RETURNS

The format template of the presentation as string.

DESCRIPTION

The function returns the format template the presentation **PresentationID** uses to display values of a variable of type **VariableType** as text, e.g. for GetValueFormatted. The template contains placeholders in curly braces: `{VALUE}` for the value, parameters of the presentation like `{PREFIX}` or `{SUFFIX}` as well as formatting instructions like `select`, `list` or `ipsnumber`, e.g. `{VALUE, select, true{On} false{Off}}`. The placeholders are not replaced – the parameter values of a variable are returned by IPS_GetVariablePresentation.
If the presentation defines its own group with a format for the variable type (e.g. the value presentation for Boolean), this format is returned, otherwise the general format of the presentation. If neither the group nor the presentation has a format, `{VALUE}` is returned. If the presentation contains translations, the template is translated if a translation exists. The restrictions of the presentation to specific variable types are not checked.
If the presentation does not exist, the function fails with "Presentation with ID … not found!", for an invalid GUID with "PresentationID … is not a valid GUID". All presentations are returned by IPS_GetPresentations, a single one by IPS_GetPresentation.

EXAMPLE

```
echo IPS_GetPresentationFormat('{60AE6B26-B3E2-BDB1-A3A1-BE232940664B}'
/* Switch */, VARIABLETYPE_BOOLEAN);
// e.g. returns: {VALUE, select, true{On} false{Off}}
```

```
echo IPS_GetPresentationFormat('{3319437D-7CDE-699D-750A-3C6A3841FA75}'
/* Value Presentation */, VARIABLETYPE_BOOLEAN);
// e.g. returns: {VALUE, list, list OPTIONS equals Value
found{{Caption}} other{{VALUE}}}
```

IPS_GetPresentations

Require: Symcon >= 8.0

```
string IPS_GetPresentations ()
```

RETURNS

A JSON-encoded list of all available presentations. The order of the entries is not guaranteed.
The fields of the individual presentations are described in more detail here.

DESCRIPTION

Returns a JSON-encoded list of all available presentations.

EXAMPLE

```
// All presentations as array
$presentations = json_decode(IPS_GetPresentations(), true);
```

IPS_GetPresentationValue

```
mixed IPS_GetPresentationValue (string $PresentationID, int
$VariableType, array $Parameters)
```

PARAMETERS

PresentationID		The ID of the presentation, formatted as GUID
VariableType		The variable type for which the value is to be returned
	0 VARIABLETYPE_BOOLEAN	Boolean
	1 VARIABLETYPE_INTEGER	Integer
	2 VARIABLETYPE_FLOAT	Float
	3 VARIABLETYPE_STRING	String
Parameters		The parameters of the presentation as key => value pairs, e.g. ['MIN' => 0, 'MAX' => 100]

RETURNS

The preview value of the presentation. The type depends on the presentation and the variable type.

DESCRIPTION

The function returns the value the presentation **PresentationID** shows in its preview, e.g. in the variable dialog or in the template manager. The presentation defines this value in **presentationValue** (see IPS_GetPresentation):

If a fixed value is defined there (e.g. **TRUE** for the switch), it is returned unchanged. **VariableType** and **Parameters** have no effect then.

If PHP code is defined there, it is executed and its return value is returned. In the code, the **Parameters** as well as the **VariableType** as VARIABLE_TYPE are available in **$_IPS**. The parameters are not completed with default values – if the code needs a parameter (e.g. MIN, MAX and STEP_SIZE for the slider), it must be passed.

If the code does not return a valid JSON-encodable result, the function fails with "Invalid JSON for presentation value: …". If the presentation does not exist, the function fails with "Presentation with ID … not found!", for an invalid GUID with "PresentationID … is not a valid GUID".

Together with IPS_GetPresentationFormat this allows creating a preview of a presentation.

EXAMPLE

```
// Fixed preview value
var_dump(IPS_GetPresentationValue('{60AE6B26-B3E2-BDB1-A3A1-
BE232940664B}' /* Switch */, VARIABLETYPE_BOOLEAN, []));
// returns: bool(true)

// Preview value from PHP code
var_dump(IPS_GetPresentationValue('{3319437D-7CDE-699D-750A-
3C6A3841FA75}' /* Value Presentation */, VARIABLETYPE_FLOAT, []));
// returns: float(123.456)

var_dump(IPS_GetPresentationValue('{6B9CAEEC-5958-C223-30F7-
BD36569FC57A}' /* Slider */, VARIABLETYPE_FLOAT, ['MIN' => 0, 'MAX' =>
100, 'STEP_SIZE' => 1]));
// returns: float(42)
```

IPS_PresentationExists

Require: Symcon >= 8.0

```
bool IPS_PresentationExists (string $PresentationID)
```

PARAMETERS

PresentationID	The ID of the presentation to be checked, formatted as GUID

RETURNS

If the presentation exists in the system, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the presentation with the ID **PresentationID** exists.

EXAMPLE

```
if (IPS_PresentationExists('{60AE6B26-B3E2-BDB1-A3A1-BE232940664B}' /*
Switch */)) {
echo 'Presentation already exists!';
}
```

Management of Templates

IPS_CreateTemplate	creates a new template for a specific presentation
IPS_DeleteTemplate	deletes a template
IPS_GetTemplate	provides comprehensive information about a template
IPS_GetTemplateList	provides all available templates
IPS_GetTemplateListByPresentation	provides a list of the templates of a presentation
IPS_SetTemplateName	sets the name of a template
IPS_SetTemplateValues	sets the values of a template
IPS_TemplateExists	checks whether a template exists

IPS_CreateTemplate

Require: Symcon >= 8.0

```
string IPS_CreateTemplate (string $PresentationID)
```

PARAMETERS

PresentationID	The ID of the presentation, formatted as GUID, for which a new template is to be created

RETURNS

The ID of the newly created template formatted as GUID

DESCRIPTION

Creates a new template for the presentation with the ID **PresentationID**. The ID of the template is returned.

EXAMPLE

```
$template = IPS_CreateTemplate('{6B9CAEEC-5958-C223-30F7-BD36569FC57A}'
/* Slider */);
```

IPS_DeleteTemplate

Require: Symcon >= 8.0

```
bool IPS_DeleteTemplate (string $TemplateID)
```

PARAMETERS

TemplateID	The ID of the template to be deleted, formatted as GUID

RETURNS

If the function could be executed successfully, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function deletes the template with the ID TemplateID.

EXAMPLE

```
// Deletes the template with the ID {8422E41F-00AA-33F7-E8A4-
BB516EC0056B}
IPS_DeleteTemplate('{8422E41F-00AA-33F7-E8A4-BB516EC0056B}');
```

IPS_GetTemplate

Require: Symcon >= 8.0

```
array IPS_GetTemplate (string $TemplateID)
```

PARAMETERS

TemplateID	The ID of the template, formatted as GUID

RETURNS

The following information is available as **key => value** pairs:

INDEX	TYPE	DESCRIPTION
TemplateID	string	The ID of the template
PresentationID	string	The ID of the presentation to which this template belongs
DisplayName	string	The display name of the template
Values	array	An array with the values of the template
IsReadOnly	bool	Whether the template is read-only, i.e. cannot be changed

DESCRIPTION

The function returns an array with information about the template with the ID **TemplateID**.

EXAMPLE

```
// Returns information about the room temperature template
print_r(IPS_GetTemplate('{868B087E-A38D-2155-EBE0-157AFBBF9E8C}' /*
Room temperature */));
/* Example output:
Array
(
[TemplateID] => {868B087E-A38D-2155-EBE0-157AFBBF9E8C}
[PresentationID] => {6B9CAEEC-5958-C223-30F7-BD36569FC57A}
[DisplayName] => Room Temperature
[Values] => Array
(
[MIN] => 15
[DIGITS] => 1
[CUSTOM_GRADIENT] => []
```

```
[DECIMAL_SEPARATOR] => Client
[GRADIENT_TYPE] => 1
[MAX] => 25
[INTERVALS] => []
[INTERVALS_ACTIVE] =>
[PERCENTAGE] =>
[PREFIX] =>
[STEP_SIZE] => 0.5
[SUFFIX] => °C
[THOUSANDS_SEPARATOR] =>
[USAGE_TYPE] => 0
)

[IsReadOnly] => 1
)
*/
```

IPS_GetTemplateList

Require: Symcon >= 8.0

```
array IPS_GetTemplateList ()
```

RETURNS

A list of the IDs of all available templates. The order of the entries is not guaranteed.

DESCRIPTION

The function returns the IDs of all available templates in an array. The function IPS_GetTemplate can be used for more information about the individual templates.

EXAMPLE

```
print_r(IPS_GetTemplateList());
```

IPS_GetTemplateListByPresentation

Require: Symcon >= 8.0

```
array IPS_GetTemplateListByPresentation (string $PresentationID)
```

PARAMETERS

PresentationID	The ID of the presentation whose templates are to be listed

RETURNS

An array with the IDs of all templates of the presentation. The order of the entries is not guaranteed.

DESCRIPTION

The function returns the IDs of all templates of the presentation with the ID **PresentationID**. The function IPS_GetTemplate can be used for more information about the individual templates.

EXAMPLE

```
print_r(IPS_GetTemplateListByPresentation('{6B9CAEEC-5958-C223-30F7-
BD36569FC57A}' /* Slider */));
```

IPS_SetTemplateName

Require: Symcon >= 8.0

```
bool IPS_SetTemplateName (string $TemplateID, string $Name)
```

PARAMETERS

TemplateID	The ID of the template whose name is to be changed
Name	The value to which the name of the template should be set

RETURNS

If the function could be executed successfully, it returns TRUE as the result, otherwise FALSE.

DESCRIPTION

The function sets the name of the template with the ID TemplateID to the value Name.

EXAMPLE

```
IPS_SetTemplateName('{3C257416-99FE-0386-D8D3-20437FF62CEB}' , 'New
Template');
```

IPS_SetTemplateValues

Require: Symcon >= 8.0

```
bool IPS_SetTemplateValues (string $TemplateID, array $Values)
```

PARAMETERS

TemplateID	The ID of the template whose values are to be adjusted
Values	An array to which the values of the template are to be set

RETURNS

If the function could be executed successfully, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the values of the template with the ID **TemplateID** to the value **Values**.

EXAMPLE

```
IPS_SetTemplateValues('{E35A875A-E9BC-4C6D-A550-F4DA423BCF18}',
['PROFILE' => '~Intensity.100']);
```

IPS_TemplateExists

Require: Symcon >= 8.0

```
bool IPS_TemplateExists (string $TemplateID)
```

PARAMETERS

TemplateID	ID of the template to be checked, formatted as GUID

RETURNS

If a template with the ID **TemplateID** exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

The function checks whether the template with the ID **TemplateID** exists.

EXAMPLE

```
if (IPS_TemplateExists('{868B087E-A38D-2155-EBE0-157AFBBF9E8C}' /* room
temperature */)) {
echo 'Template already exists!';
}
```

Variable Profiles

IPS_CreateVariableProfile	creates a variable profile
IPS_DeleteVariableProfile	deletes a variable profile
IPS_GetVariableProfile	returns extensive information about a variable profile
IPS_GetVariableProfileList	returns a list of all existing variable profiles
IPS_GetVariableProfileListByType	returns a list of all existing variable profiles of a specific type
IPS_SetVariableProfileAssociation	modifies the association table of a variable profile
IPS_SetVariableProfileDigits	sets the number of decimal places of a variable profile
IPS_SetVariableProfileIcon	sets the icon for a variable profile
IPS_SetVariableProfileText	sets the prefix and suffix of a variable profile
IPS_SetVariableProfileValues	sets the minimum value, maximum value, and step size of a variable profile
IPS_VariableProfileExists	checks if a variable profile exists

IPS_CreateVariableProfile

```
bool IPS_CreateVariableProfile (string $ProfileName, int $VariableType)
```

PARAMETERS

ProfileName	Name of the profile. Legal characters are A-Z, a-z, 0-9, period (.), hyphen (-), and underscore (_)

VariableType

VALUE	CONSTANT	DESCRIPTION
0	VARIABLETYPE_BOOLEAN	creates a variable profile of the type **Boolean**
1	VARIABLETYPE_INTEGER	creates a variable profile of the type **Integer**
2	VARIABLETYPE_FLOAT	creates a variable profile of the type **Float**
3	VARIABLETYPE_STRING	creates a variable profile of the type **String**

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates a new variable profile of the type **VariableType** and with the name **ProfileName**.

EXAMPLE

```
//Create a profile for Boolean variables
IPS_CreateVariableProfile("Switch", VARIABLETYPE_BOOLEAN);
//... further configuration of the profile is done here
```

IPS_DeleteVariableProfile

```
bool IPS_DeleteVariableProfile (string $ProfileName)
```

PARAMETERS

ProfileName	Name of the profile to be deleted

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function deletes an existing variable profile with the name **ProfileName**.

EXAMPLE

```
//Delete the profile "Switch"
IPS_DeleteVariableProfile("Switch");
```

IPS_GetVariableProfile

```
array IPS_GetVariableProfile (string $ProfileName)
```

PARAMETERS

ProfileName	Name of the profile

RETURNS

The following information is available as key => value pairs:

INDEX	TYPE	DESCRIPTION
Associations	array	Array with value, name, icon, color pairs (see Associations Table)
Icon	string	Icon of the variable profile
IsReadOnly	boolean	TRUE if the profile is a system created profile, which can not be changed
MaxValue	float	The maximum value used for the visualization
MinValue	float	The minimum value used for the visualization
StepSize	float	Step size for the visualization, 0 when the association table is used
Digits	integer	Number of decimal places
Prefix	string	Prefix for the visualization
Suffix	string	Suffix for the visualization
ProfileName	string	Name of the profile. (~ = System profile)
ProfileType	integer	Type of profile (see ProfileType table)

Associations Table

INDEX	TYPE	DESCRIPTION
Value	float	Value that is associated with the given name, icon, and color
Name	string	Name of the specified value
Icon	string	Icon of the specified value
Color	integer	Color value in HTML format, e.g., 0x0000FF for blue. Special case: -1 (COLOR_TRANSPARENT) for transparent

ProfileType
The profile type and the variable type (IPS_CreateVariable) must be identical so that a display in the WebFront can be shown correctly.

VALUE	CONSTANT	DESCRIPTION
0	VARIABLETYPE_BOOLEAN	A profile of the type **Boolean**
1	VARIABLETYPE_INTEGER	A profile of the type **Integer**
2	VARIABLETYPE_FLOAT	A profile of the type **Float**
3	VARIABLETYPE_STRING	A profile of the type **String**

DESCRIPTION

This function returns an array that contains information about the variable profile with the name **ProfileName**.

EXAMPLE

```
print_r( IPS_GetVariableProfile("~WindDirection") );

/* returns e.g.:
Array
(
[Associations] => Array
(
)

[Digits] => 1
[Icon] => WindDirection
[IsReadOnly] => 1
[MaxValue] => 360
[MinValue] => 0
[Prefix] =>
[ProfileName] => ~WindDirection
[ProfileType] => 2
[StepSize] => 60
[Suffix] => °
)
*/
```

IPS_GetVariableProfileList

```
array IPS_GetVariableProfileList ()
```

RETURNS

List of names of all variable profiles. The order of the entries is not guaranteed.

DESCRIPTION

This function lists the names of all existing variable profiles and returns them as an array.

EXAMPLE

```
$Profile = IPS_GetVariableProfileList();
print_r($Profile);

/* returns e.g.:
Array
(
[0] => ~Temperature
[1] => ~Humidity
[2] => ~AirPressure
etc. ...
*/
```

IPS_GetVariableProfileListByType

```
array IPS_GetVariableProfileListByType (int $VariableType)
```

PARAMETERS

	VALUE	CONSTANT	DESCRIPTION
VariableType	0	VARIABLETYPE_BOOLEAN	Search for the type **Boolean**
	1	VARIABLETYPE_INTEGER	Search for the type **Integer**
	2	VARIABLETYPE_FLOAT	Search for the type **Float**
	3	VARIABLETYPE_STRING	Search for the type **String**

RETURNS

List of names of all variable profiles of the given type. The order of the entries is not guaranteed.

DESCRIPTION

This function lists the names of all existing variable profiles of the type **VariableType** and returns them as an array.

EXAMPLE

```
$Profile = IPS_GetVariableProfileListByType(VARIABLETYPE_BOOLEAN);
print_r($Profile);

/* returns e.g.:
Array
(
[0] => ~Switch
[1] => ~Alert
[2] => ~Alert.Reversed
```

```
etc. ...
*/
```

IPS_SetVariableProfileAssociation

```
bool IPS_SetVariableProfileAssociation (string $ProfileName, mixed
$Value, string $Name, string $Icon, int $Color)
```

PARAMETERS

ProfileName	Name of the profile. Available profiles can be queried via IPS_GetVariableProfileList
Value	The value that should be associated with the name, icon, and color
Name	Name of the specified value
Icon	Icon of the specified value
Color	Color value in HTML format, e.g., 0x0000FF for blue. Special case: -1 (COLOR_TRANSPARENT) for transparent

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the **Name**, the **Icon**, and the **Color** for the value **Value** of the variable profile with the name **ProfileName**. The color is not displayed for profiles of type value display/slider (see Object presentation).



If **Name** and **Icon** are empty, the association for the specified value **Value** is deleted. Associations of Boolean profiles cannot be deleted.



Standard profiles, which cannot be changed, start with a tilde (~).

EXAMPLE

```
//Create an association for value 1 in white
IPS_SetVariableProfileAssociation("Temperature", 1, "Value 1",
"Speaker", 0xFFFFFF);

//Create an association for value "sum" in yellow
IPS_SetVariableProfileAssociation("Season", "sum", "Summer", "Sun",
0xFFFF00);

//Delete the association for value 1
IPS_SetVariableProfileAssociation("Temperature", 1, "", "",
COLOR_TRANSPARENT);
```

IPS_SetVariableProfileDigits

```
bool IPS_SetVariableProfileDigits (string $ProfileName, int $Digits)
```

PARAMETERS

ProfileName	Name of the profile. Available profiles can be queried via IPS_GetVariableProfileList
Digits	Number of decimal places displayed in the visualization

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the number of decimal places displayed in the visualization to **Digits** for the variable profile with the name **ProfileName**.



Standard profiles, which cannot be changed, start with a tilde (~).

EXAMPLE

```
IPS_SetVariableProfileDigits("Temperature", 1);
```

IPS_SetVariableProfileIcon

```
bool IPS_SetVariableProfileIcon (string $ProfileName, string $Icon)
```

PARAMETERS

ProfileName	Name of the profile. Available profiles can be queried via IPS_GetVariableProfileList
Icon	Name of the icon

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the default icon of the variable profile with the name **ProfileName** to **Icon**.



The precedence of icons is described in detail at Icons.



Standard profiles, which cannot be changed, start with a tilde (~).

EXAMPLE

```
IPS_SetVariableProfileIcon("Temperature", "Temperature");
```

IPS_SetVariableProfileText

```
bool IPS_SetVariableProfileText (string $ProfileName, string $Prefix,
string $Suffix)
```

PARAMETERS

ProfileName	Name of the profile. Available profiles can be queried via IPS_GetVariableProfileList
Prefix	Prefix of the value
Suffix	Suffix of the value

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the prefix and suffix of the variable profile with the name **ProfileName** to **Prefix** and **Suffix**, respectively.



Standard profiles, which cannot be changed, start with a tilde (~).

EXAMPLE

```
IPS_SetVariableProfileText("Switch", "", "%");
```

IPS_SetVariableProfileValues

```
bool IPS_SetVariableProfileValues (string $ProfileName, float
$MinValue, float $MaxValue, float $StepSize)
```

PARAMETERS

ProfileName	Name of the profile. Available profiles can be queried via IPS_GetVariableProfileList.
MinValue	Minimum value used for the visualization. This soft limitation does not affect the variable value.
MaxValue	Maximum value used for the visualization. This soft limitation does not affect the variable value.
StepSize	Step size used for the visualization to create the setpoint change bar. A step size of **0** activates the association list.

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the minimum value, maximum value, and step size of the variable profile with the name **ProfileName** to **MinValue**, **MaxValue** and **StepSize**, respectively.



Standard profiles, which cannot be changed, start with a tilde (~).

EXAMPLE

```
IPS_SetVariableProfileValues("Temperature", -10, 40, 0.5);
```

IPS_VariableProfileExists

```
bool IPS_VariableProfileExists (string $ProfileName)
```

PARAMETERS

ProfileName	Name of the variable profile

RETURNS

If the variable profile with the name **ProfileName** exists, **TRUE** is returned, otherwise **FALSE**.

DESCRIPTION

This function checks if the variable profile with the name **ProfileName** exists.

EXAMPLE

```
if (IPS_VariableProfileExists("Temperature"))
echo "Profile already exists!";
```

Accessing Variables

GetValue	returns the value of an IP-Symcon variable, independent of its type
GetValueBoolean	returns the value of an IP-Symcon variable of the type Boolean
GetValueColor	returns the value of a color variable as integer in the format 0xRRGGBB
GetValueFloat	returns the value of an IP-Symcon variable of the type Float
GetValueFormatted	returns the value of an IP-Symcon variable and formats it according to its profile
GetValueFormattedEx	returns a value formatted based on the profile of a variable
GetValueInteger	returns the value of an IP-Symcon variable of the type Integer
GetValueString	returns the value of an IP-Symcon variable of the type String
HasAction	checks whether an action is defined
RequestAction	executes the variable action of an IP-Symcon variable
RequestActionColor	executes the action of a color variable with a color value in the format 0xRRGGBB
RequestActionEx	executes the action of an IP-Symcon variable with individual sender
SetValue	sets the value of an IP-Symcon variable independent of its type
SetValueBoolean	sets the value of an IP-Symcon variable of the type Boolean
SetValueColor	sets the value of a color variable from a color value in the format 0xRRGGBB
SetValueFloat	sets the value of an IP-Symcon variable of the type Float
SetValueInteger	sets the value of an IP-Symcon variable of the type Integer
SetValueString	sets the value of an IP-Symcon variable of the type String

GetValue

```
mixed GetValue (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

Value of the specified variable in the type of the specified variable

DESCRIPTION

The function returns the content of the IP-Symcon variable with the ID **VariableID**. The returned variable type is the one that was selected when creating the variable (IPS_CreateVariable). If the variable was created as a string, the function returns a string.

EXAMPLE

```
$Window_open = GetValue(47788); // Boolean variable: true or false
$current_project = GetValue(46250); // String variable: e.g. "Rain
sensing"
```

GetValueBoolean

```
bool GetValueBoolean (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

Value of the specified variable

DESCRIPTION

The function returns the content of the IP-Symcon variable with the ID **VariableID**. The selected variable must have the type Boolean. If the variable's type is not Boolean, the function stops with an error. No conversion is done by this function. For a type-unsafe query you can use the function GetValue.

EXAMPLE

```
$Window_open = GetValueBoolean(47788); // Boolean variable: true or
false
```

GetValueColor

```
int GetValueColor (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

Color value in the format 0xRRGGBB (red, green and blue with 8 bits each)

DESCRIPTION

The function returns the content of the variable with the ID **VariableID** as color value in the format 0xRRGGBB. This allows reading a color regardless of how it is stored in the variable.

Integer variables already contain the color as 0xRRGGBB. Their value is returned unchanged; the presentation is not checked.

String variables must use the presentation Color. The value is a JSON object which is converted to RGB according to the parameter ENCODING of the presentation:

ENCODING	CONSTANT	ENCODING	JSON VALUE OF THE STRING VARIABLE	RANGES
0	COLOR_ENCODING_RGB	RGB	{"r": ..., "g": ..., "b": ...}	r, g, b: 0 - 255
1	COLOR_ENCODING_CMYK	CMYK	{"c": ..., "m": ..., "y": ..., "k": ...}	c, m, y, k: 0 - 100
2	COLOR_ENCODING_HSV	HSV	{"h": ..., "s": ..., "v": ...}	h: 0 - 360, s, v: 0 - 100
3	COLOR_ENCODING_HSL	HSL	{"h": ..., "s": ..., "l": ...}	h: 0 - 360, s, l: 0 - 100
4	COLOR_ENCODING_XY	xy	{"x": ..., "y": ...}	x, y: position on the CIE chromaticity diagram, 0 - 1

The xy encoding only stores the chromaticity, no brightness. The conversion uses the color space of the presentation (COLOR_SPACE); except for color space 5 (none), the coordinates

are limited to the triangle of the color space.
If a string variable does not use the presentation Color, the function fails with "Variable does not use presentation Color". For an unknown encoding it fails with "Invalid Color Encoding", for variables of type Boolean or Float with "Invalid Variable Type". The content of a string variable must be valid JSON with the fields listed above; an empty string causes an error.
The presentation can be read with IPS_GetVariablePresentation and set with IPS_SetVariableCustomPresentation. See also SetValueColor, RequestActionColor and GetValue.

EXAMPLE

```
$color = GetValueColor(12345);

// Split the color into its components
$red = ($color >> 16) & 0xFF;
$green = ($color >> 8) & 0xFF;
$blue = $color & 0xFF;

printf("#%06X (R: %d, G: %d, B: %d)", $color, $red, $green, $blue);
// e.g. returns: #FF8000 (R: 255, G: 128, B: 0)
```

GetValueFloat

```
float GetValueFloat (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

Value of the specified variable

DESCRIPTION

The function returns the content of the IP-Symcon variable with the ID **VariableID**. The selected variable must have the type Float. If the variable's type is not Float, the function stops with an error. No conversion is done by this function. For a type-unsafe query you can use the function GetValue.

EXAMPLE

```
$current_Temperature = GetValueFloat(47788);
```

GetValueFormatted

```
string GetValueFormatted (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

Value of the specified variable, formatted according to its presentation or profile

DESCRIPTION

The function returns the content of the IP-Symcon variable with the ID **VariableID** after being formatted according to the variable's presentation or profile. If a presentation is assigned to the variable, it is used for formatting, otherwise the variable profile. If the variable has neither a presentation nor a profile, the value is returned unformatted as string. The formatted result is always returned as string.

EXAMPLE

```
echo GetValueFormatted(47788); //Profile = ~Switch
//Returns: On or Off

echo GetValueFormatted(46250); //Profile = ~Temperature
//Returns: 17.5 °C
```

GetValueFormattedEx

Require: Symcon >= 5.5

```
string GetValueFormattedEx (int $VariableID, mixed $Value)
```

PARAMETERS

VariableID	ID of the variable
Value	Value to be formatted

RETURNS

Value formatted to match the presentation or profile of the variable

DESCRIPTION

The function returns the value **Value** after it has been formatted to match the presentation or variable profile of the variable with the ID **VariableID**. The actual value of the variable with the ID **VariableID** is irrelevant. Due to the formatting the variable type returned is always a string.

EXAMPLE

```
echo GetValueFormattedEx(12345, true); //Profile = ~Switch
//Results in: On

echo GetValueFormattedEx(23456, 17.5); //Profile = ~Temperature
//Results in: 17.5 °C
```

GetValueInteger

```
int GetValueInteger (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

Value of the specified variable

DESCRIPTION

The function returns the content of the IP-Symcon variable with the ID **VariableID**. The selected variable must have the type Integer. If the variable's type is not Integer, the function stops with an error. No conversion is done by this function. For a type-unsafe query you can use the function GetValue.

EXAMPLE

```
$Number_persons = GetValueInteger(47788);
```

GetValueString

```
string GetValueString (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

Value of the specified variable

DESCRIPTION

The function returns the content of the IP-Symcon variable with the ID **VariableID**. The selected variable must have the type String. If the variable's type is not String, the function stops with an error. No conversion is done by this function. For a type-unsafe query you can use the function GetValue.

EXAMPLE

```
$current_project = GetValueString(47788);
```

HasAction

Require: Symcon >= 5.3

```
bool HasAction (int $VariableID)
```

PARAMETERS

VariableID	ID of the variable

RETURNS

TRUE, if the variable has an action, otherwise FALSE

DESCRIPTION

The function checks the variable with the ID **VariableID** for an existing action.



The function only checks for the presence of an action, not its validity or correctness.

EXAMPLE

```
// Checks whether the variable with the ID 12345 has an action.
HasAction(12345);
```

RequestAction

Require: Symcon >= 5.0

```
bool RequestAction (int $VariableID, mixed $Value)
```

PARAMETERS

VariableID	ID of the variable
Value	Value for the action

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function executes the action of the Variable with the ID **VariableID** with the value **Value**. It executes the same action as the WebFront or other Visualizations. If a custom Action Script is defined, that PHP script is executed. Otherwise, the default action of an instance variable is executed, if available. If neither a custom nor a default action is defined, the function RequestAction causes an error. Usually, variable actions switch the corresponding parameter of a device and update the variable value accordingly. It is important to ensure that the type of the set value matches the type of the variable. If the variable has the type String, the value should have the type String as well. Furthermore, the System Variable with $_IPS["SENDER"] = Action is available in the script.



If the variable and value types do not match, the value is converted to the type of the variable. The result of this conversion can deviate from the expected result. It is recommended to pass the correct type.



It needs to be ensured, that Action Scripts do not execute RequestAction on themselves. This would cause an infinite loop.

EXAMPLE

```
// Switch Boolean variable 29117 to "true"
RequestAction(29117, true);

// Switch Integer variable 46250 to 18
RequestAction(46250, 18);
```

RequestActionColor

```
bool RequestActionColor (int $VariableID, int $ColorValue)
```

PARAMETERS

VariableID	ID of the variable
ColorValue	Color in the format 0xRRGGBB

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function executes the action of the variable with the ID **VariableID** with the color **ColorValue** (format 0xRRGGBB). It behaves like RequestAction, but first converts **ColorValue** into the format of the variable - exactly like SetValueColor:

Integer variables: **ColorValue** is passed to the action unchanged.

String variables with the presentation Color: the JSON object according to the parameter ENCODING of the presentation is passed to the action (RGB, CMYK, HSV, HSL or xy, see constants `COLOR_ENCODING_*`; the fields are described at SetValueColor).

The custom action of the variable is executed (e.g. an action script set via IPS_SetVariableCustomAction), otherwise the default action, usually the one of the parent instance. In the action script, $_IPS['SENDER'] = "Action" is available and $_IPS['VALUE'] contains the converted value. Unlike SetValueColor, the device is switched; whether and when the value of the variable is updated is up to the action.

The conversion happens before the action is executed: if a string variable does not use the presentation Color, the function fails with "Variable does not use presentation Color". For an unknown encoding it fails with "Invalid Color Encoding" without executing the action; for variables of type Boolean or Float with "Invalid Variable Type". If the variable has neither a custom nor a default action, it fails with "No valid action available".

EXAMPLE

```
// Switch the lamp to green
RequestActionColor(12345, 0x00FF00);
```

RequestActionEx

Require: Symcon >= 5.0

```
bool RequestActionEx (int $VariableID, mixed $Value, string $Sender)
```

PARAMETERS

VariableID	ID of the variable
Value	Value for the action
Sender	Name of the sender

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function executes the action of the Variable with the ID **VariableID** with the value **Value** and the sender **Sender**. The same switching action as in the WebFront and other Visualizations is used here. If a separate Action Script is defined for the variable, this PHP script is executed. Otherwise, if available, the standard action of an Instance Variable is executed. If there is neither a custom nor a standard action, the RequestActionEx function leads to an error. Variable actions usually switch the associated property of a device according to the new value and then set the value of the variable accordingly. If an error occurs while switching, an error message is output. It is important to ensure that the type of value matches the type of variable. If the target variable is of type string, the value should also be type string. Furthermore, the System Variable with $_IPS["SENDER"] = **Sender** is available in the script.



If the variable and value types do not match, the value is converted to the type of the variable. The result of this conversion can deviate from the expected result. It is recommended to pass the correct type.

EXAMPLE

```
// Switch on Boolean variable with Sender AlarmSystem
RequestActionEx(12345, true, "AlarmSystem");
```

```
// Set integer variable with Sender TemperatureAirConditioning
RequestActionEx(23456, 18, "TemperatureAirConditioning");
```

SetValue

```
bool SetValue (int $VariableID, mixed $Value)
```

PARAMETERS

VariableID	ID of the variable
Value	New value of the variable

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

The function writes the value **Value** into the IP-Symcon variable with the ID **VariableID**. It is important to ensure that the type of the set value matches the type of the variable. If the variable has the type String, the value should have the type String as well.



If the variable and value types do not match, the value is converted to the type of the variable. The result of this conversion can deviate from the expected result. It is recommended to pass the correct type.

EXAMPLE

```
SetValue(29117, true); // Boolean variable: Set flag for school
holidays
SetValue(46250, $current_project); // String variable: e.g. "Rain
sensing"
```

SetValueBoolean

```
bool SetValueBoolean (int $VariableID, bool $Value)
```

PARAMETERS

VariableID	ID of the variable
Value	TRUE/FALSE

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the value of the IP-Symcon variable with the ID **VariableID** to **Value**. The selected variable must have the type Boolean. If the variable's type is not Boolean, the function stops with an error. A conversion from Integer or Float (0 = FALSE, otherwise TRUE) is done. For setting the variable type-unsafely you can use the function SetValue.

EXAMPLE

```
SetValueBoolean(47788, true); // Switch on alarm system
```

SetValueColor

```
bool SetValueColor (int $VariableID, int $ColorValue)
```

PARAMETERS

VariableID	ID of the variable
ColorValue	Color in the format 0xRRGGBB

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the content of the variable with the ID **VariableID** to the color **ColorValue** in the format 0xRRGGBB. The color is stored according to the variable:
Integer variables receive **ColorValue** unchanged. The presentation is not checked.
String variables must use the presentation Color. **ColorValue** is converted according to the parameter ENCODING of the presentation and stored as JSON object. Only the lower 24 bits of **ColorValue** are evaluated.

ENCODING	CONSTANT	ENCODING	JSON VALUE OF THE STRING VARIABLE	RANGES
0	COLOR_ENCODING_RGB	RGB	{"r": ..., "g": ..., "b": ...}	r, g, b: 0 - 255
1	COLOR_ENCODING_CMYK	CMYK	{"c": ..., "m": ..., "y": ..., "k": ...}	c, m, y, k: 0 - 100
2	COLOR_ENCODING_HSV	HSV	{"h": ..., "s": ..., "v": ...}	h: 0 - 360, s, v: 0 - 100
3	COLOR_ENCODING_HSL	HSL	{"h": ..., "s": ..., "l": ...}	h: 0 - 360, s, l: 0 - 100
4	COLOR_ENCODING_XY	xy	{"x": ..., "y": ...}	x, y: position on the CIE chromaticity diagram, 0 - 1

The xy encoding only stores the chromaticity, no brightness. The conversion uses the color space of the presentation (COLOR_SPACE); except for color space 5 (none), the coordinates are limited to the triangle of the color space. For CMYK, HSV, HSL and xy the conversion results in floating point numbers.
Like SetValue, only the value of the variable is written; no action is executed, so the device is not switched. Use RequestActionColor for that.
If a string variable does not use the presentation Color, the function fails with "Variable does not use presentation Color". For an unknown encoding it fails with "Invalid Color Encoding" and leaves the variable unchanged; for variables of type Boolean or Float with "Invalid Variable Type". If the variable is read-only (e.g. a status variable of a module), the function fails with "Variable is marked as read-only and cannot be changed".
See also GetValueColor.

EXAMPLE

```
// Set orange
SetValueColor(12345, 0xFF8000);

// For a string variable with the presentation Color and the encoding
RGB
echo GetValue(12345);
// returns: {"b":0,"g":128,"r":255}
```

SetValueFloat

```
bool SetValueFloat (int $VariableID, float $Value)
```

PARAMETERS

VariableID	ID of the variable
Value	64-bit floating point number

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the value of the IP-Symcon variable with the ID **VariableID** to **Value**. The selected variable must have the type Float. If the variable's type is not Float, the function stops with an error. A conversion from Integer or Boolean (0/1) is done. For setting the variable type-unsafely you can use the function SetValue.

EXAMPLE

```
SetValueFloat(47788, $current_Temperature);
```

SetValueInteger

```
bool SetValueInteger (int $VariableID, int $Value)
```

PARAMETERS

VariableID	ID of the variable
Value	New value (64-bit integer, 32-bit on armhf systems)

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the value of the IP-Symcon variable with the ID **VariableID** to **Value**. The selected variable must have the type Integer. If the variable's type is not Integer, the function stops with an error. A conversion from Float (rounded) or Boolean (0/1) is done. For setting the variable type-unsafely you can use the function SetValue.



By default, integer variables are 64-bit integers with a range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Only on armhf systems they are 32-bit integers with a range of -2,147,483,648 to 2,147,483,647. armhf is supported until Symcon 10.0.

EXAMPLE

```
SetValueInteger(47788, $Number_persons);
```

SetValueString

```
bool SetValueString (int $VariableID, string $Value)
```

PARAMETERS

VariableID	ID of the variable
Value	Text

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function sets the value of the IP-Symcon variable with the ID **VariableID** to **Value**. The selected variable must have the type String. If the variable's type is not String, the function stops with an error. A conversion from Float, Integer, or Boolean (False/True) is done. For setting the variable type-unsafely you can use the function SetValue.

EXAMPLE

```
SetValueString(47788, $current_project);
```

Developer Area

ADVANCED FUNCTIONS

FUNCTION	DESCRIPTION
System Commands	All commands that are usually only available to module developers are usable to configure IP-Symcon. PHP can access the system completely - The Command Reference lists and explains all possibilities.
Module Commands	Every module and every option of modules that can be changed in the management console, can be read and modified by scripts as well. Thus, the complete control over the connected hardware modules is possible. The Module Reference contains a complete list of all action commands.
PHP	IP-Symcon offers the full potential of the script language PHP and offers all existing PHP commands. In addition, the individual php.ini can be used to make adjustments or load extensions. This enables extended functions, e.g., extended graphic functions (GD2) or communication with MySQL, MSSQL, or other databases.
Client/Server Model	The IP-Symcon service is embedded smoothly into the server operating system. Interactive sessions are not required and thus avoid problems due to unplanned or planned reboots of the system. After the reboot, the system is available again. The management console can always comfortably access the system. Extensions of the system can be done conviniently from the work station. Accessing the server directly or logging in via TeamViewer/AnyDesk/RDP is not needed. Changes in the management console are instantly handled by the server and are immediately available.
JSON-RPC Interface	The JSON-RPC interface is available since IP-Symcon 3.0 and allows the access to all functions of the Command Reference and Module Reference. Thus, the integration into any application is as easy as writing a PHP script within IP-Symcon. The functions can be used as described in the documentation. Some examples about the integration are found under Data Exchange.
WebServer	The integrated WebServer can be operated on multiple ports. One possibility is an SSL description with an individual certificate. A log file that is compatible to Webalizer can be configured for every instance.
WebFront	The web interface "WebFront" offers a quick possibility of visualization. As soon as devices are configured within IP-Symcon, they can be accessed from anywhere via LAN/WLAN/Web. It is easy to call the WebFront and operate. Documentation about the possibilities of extending the WebFront are found in the developer area. A modern browser is required for optimal use.

SDK/TOOLS

The IP-Symcon SDKs enable the creation of modules for IP-Symcon.

Data Exchange

REACHABILITY

The JSON-RPC API, which has been available since IP-Symcon 3.0, can be accessed via port 3777 by default. Port 3777 is only bound to the local system and does not require any authentication.
The JSON-RPC API is also active on every web server set up in IP-Symcon. This means that the WebFront and the mobile apps can have access to IP-Symcon on every host/port/SSL configuration.

PASSWORD AND FEATURES

The JSON-RPC API is authenticated via the license username and the Remote Access password. If no password is given, no access will be granted.
A special case are some WFC_* functions that are authenticated using the password of the respective WebFront configurator. In this case "webfront" is used as the username. These functions enable isolated access to the sub-areas of the object tree defined by the configurator. Only the WFC_GetConfigurators function is made available without any authentication. This is necessary to enable the listing of the WebFront configurators for the WebFront and the mobile apps.
In addition, the IP address Autostart Function is thereby implemented. In this function, only the ID/title/icon/position/editability of the respective configurator, whether a password is required and whether this configurator has been activated for the mobile apps is transmitted. By default, the use of mobile apps is deactivated in the configurator in order to avoid a security problem on the basic tree (ID = 0).

AUTHENTICATION

The "user" folder, which is shared for user-defined content. The basic authentication offered in the web server applies exclusively to it, provided the web server is used for the WebFront. If the web server is used for another directory, the authentication applies to all content that is located in this directory - but not to the JSON-RPC API, as this is subject to the aforementioned authentication rules.



The normal commands of the Command Reference and Module Reference can be used for the function calls.



Your license username must be used as username. The password can be set using Remote Access.

JSON-RPC INTERFACE (VIA PHP/IP-SYMCON)

READ KERNEL VERSION

```
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
$result = $rpc->IPS_GetKernelVersion();
echo "KernelVersion: ".$result;
```

START SCRIPTS

```
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
$rpc->IPS_RunScript(34956);
```

READ VARIABLES

```
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
echo $rpc->GetValueFormatted(58383);
```

CHANGE VARIABLES

```
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
$rpc->SetValue(38809, 18.5);
```

SWITCH HOMEMATIC DEVICE

```
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
$rpc->HM_WriteValueBoolean(20558, "STATE", false);
```

RETURN VALUE IF NOT AVAILABLE

```
try {
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
$rpc->IPS_GetKernelDir();
} catch (JSONRPCException $ e) {
echo 'RPC Problem: ', $e->getMessage(), "\n";
} catch (Exception $e) {
```

```
echo 'Server Problem: ', $e->getMessage(), "\n";
}
```

WEBSOCKET INTERFACE (VIA JAVASCRIPT)

Changes in the system can be received as quickly as possible via the Websocket interface.

FULL API

```
//IP-Symcon 5.1/IP-Symcon 5.2
//Normal
var connection = new WebSocket('ws://127.0.0.1:3777/api/',
[btoa('email:remoteaccesspw').replace(/=/g, '')]);
//SSL
var connection = new WebSocket('wss://127.0.0.1:3777/api/',
[btoa('email:remoteaccesspw').replace(/=/g, '')]);
//since IP-Symcon 5.3
//Normal
var connection = new WebSocket('ws://127.0.0.1:3777/api/',
[encodeURIComponent(btoa('email:remoteaccesspw'))]);
//SSL
var connection = new WebSocket('wss://127.0.0.1:3777/api/',
[encodeURIComponent(btoa('email:remoteaccesspw'))]);
```

WEBFRONT API

```
//IP-Symcon 5.1
//Normal
var connection = new WebSocket('ws://127.0.0.1:3777/api/wfc/12345',
[btoa('webfront:webfrontpw').replace(/=/g, '')]);
//SSL
var connection = new WebSocket('wss://127.0.0.1:3777/api/wfc/12345',
[btoa('webfront:webfrontpw').replace(/=/g, '')]);
//IP-Symcon 5.2
//Normal
var connection = new WebSocket('ws://127.0.0.1:3777/wfc/12345/api/',
[btoa('webfront:webfrontpw').replace(/=/g, '')]);
//SSL
var connection = new WebSocket('wss://127.0.0.1:3777/wfc/12345/api/',
[btoa('webfront:webfrontpw').replace(/=/g, '')]);
//since IP-Symcon 5.3
//Normal
var connection = new WebSocket('ws://127.0.0.1:3777/wfc/12345/api/',
[encodeURIComponent(btoa('webfront:webfrontpw'))]);
//SSL
var connection = new WebSocket('wss://127.0.0.1:3777/wfc/12345/api/',
[encodeURIComponent(btoa('webfront:webfrontpw'))]);
```

JSON-RPC INTERFACE (VIA CURL)

Invoke via Command Line

CHANGE VARIABLES

For the username mail%40provider.de or password @ must not be written. => Instead: %40
Enter username (mail), password and IP without <,>!



To encode and to take special characters into account, the page urlencoder.org can be used for username and password.

```
//SetValue Integer with the parameters ID = 12345 and value = 42
curl -i -X POST -H "Content-Type: application/json" -d "{\"jsonrpc\":
\"2.0\", \"id\": \"0\", \"method\": \"SetValue\", \"params\": [12345,
42]}" http://&lt;mail%40provider.de>:<Passwort>@<IP>:3777/api/
//SetValue Float with the parameters ID = 12346 and value = 1.23
curl -i -X POST -H "Content-Type: application/json" -d "{\"jsonrpc\":
\"2.0\", \"id\": \"0\", \"method\": \"SetValue\", \"params\": [12346,
1.23]}" http://&lt;mail%40provider.de>:<Passwort>@<IP>:3777/api/
//SetValue Boolean with the parameters ID = 12347 and value = false
curl -i -X POST -H "Content-Type: application/json" -d "{\"jsonrpc\":
\"2.0\", \"id\": \"0\", \"method\": \"SetValue\", \"params\": [12347,
false]}" http://&lt;mail%40provider.de>:<Passwort>@<IP>:3777/api/
//SetValue string with the parameters ID = 12348 and value = "my
string"
curl -i -X POST -H "Content-Type: application/json" -d "{\"jsonrpc\":
\"2.0\", \"id\": \"0\", \"method\": \"SetValue\", \"params\": [12348,
\"my string\"]}" http://&lt;mail%40provider.de>:
<Passwort>@<IP>:3777/api/
//Return-value if successful
{"jsonrpc":"2.0","id":"0","result":true}
```

READ VARIABLES

```
//GetValue with the parameters ID = 12345
curl -i -X POST -H "Content-Type: application/json" -d "{\"jsonrpc\":
\"2.0\", \"id\": \"0\", \"method\": \"GetValue\", \"params\": [12345]}"
http://&lt;mail%40provider.de>:<Passwort>@<IP>:3777/api/
//Return-value
{"jsonrpc":"2.0","id":"0","result":42}
```

Download (Archive)



It is always recommended to use the latest version of IP-Symcon. For testing purposes, we offer old versions for download for the respective platforms.

WINDOWS (32-BIT)

4.0, 4.1, 4.2, 4.3, 4.4

WINDOWS (64-BIT)

4.2, 4.3, 4.4, 5.0, 5.1, 5.2, 5.3, 5.4, 5.5, 6.0, 6.1, 6.2, 6.3, 6.4, 7.0, 7.1, 7.2, 8.0, 8.1, 9.0

RASPBERRY PI (ARMHF)

4.0, 4.1, 4.2, 4.3, 4.4, 5.0, 5.1, 5.2, 5.3, 5.4, 5.5, 6.0, 6.1, 6.2, 6.3, 6.4, 7.0, 7.1, 7.2, 8.0, 8.1, 9.0

RASPBERRY PI (ARM64)

6.0, 6.1, 6.2, 6.3, 6.4, 7.0, 7.1, 7.2, 8.0, 8.1, 9.0

UBUNTU (AMD64)

4.0, 4.1, 4.2, 4.3, 4.4, 5.0, 5.1, 5.2, 5.3, 5.4, 5.5, 6.0, 6.1, 6.2, 6.3, 6.4, 7.0, 7.1, 7.2, 8.0, 8.1, 9.0

MAC (32-BIT)

4.0, 4.1, 4.2, 4.3, 4.4, 5.0, 5.1, 5.2, 5.3, 5.4, 5.5, 6.0, 6.1, 6.2, 6.3, 6.4, 7.0, 7.1, 7.2, 8.0, 8.1, 9.0

CATAN (64-BIT)

9.0

LEGACY IP-SYMCON

3.4

LEGACY MANAGEMENT CONSOLE

3.4, 4.0, 4.1, 4.2, 4.3, 4.4, 5.0

Internal Command Reference



The functions in this section are intended for internal use, e.g. for the console. They may change or be removed at any time without notice and should not be used in your own scripts or modules.

HELPERS

Functions in this group are only available via the JSON-RPC interface.

IPS_GetConfigurationsFromChildren	returns the configuration which the child instances specify for an instance
IPS_GetFunctions	returns comprehensive information about several functions
IPS_GetIcons	returns all icons including the icons of the skins
IPS_GetLibraries	returns information about several libraries
IPS_GetModules	returns information about several modules
IPS_GetScriptThreads	returns information about several threads of the script engine
IPS_GetTimers	returns information about several timers
IPS_GetTraces	returns all traces of an object including their causes

LICENSE

Functions in this group return information about the license of IP-Symcon.

IPS_GetDemoExpiration	returns the expiration time of the demo mode
IPS_GetLicensee	returns the licensee
IPS_GetLimitDemo	returns whether a demo license is used
IPS_GetLimitFeatures	returns the enabled features of the license
IPS_GetLimitServer	returns the server limitation of the license
IPS_GetLimitVariables	returns the maximum number of variables of the license
IPS_GetLimitWebFront	returns the maximum number of visualizations of the license
IPS_GetSubscriptionExpiration	returns the end of the subscription

REPLICATION

Functions in this group return the status of the replication (mirroring), control the failover and map the object IDs of Sync Remote.

IPS_DemoteToReplicationSlave	demotes a slave promoted to master again

IPS_FindObjectID	returns the local ID for an ID of the remote system
IPS_FindRemoteID	returns the ID of the remote system for a local ID
IPS_FindServerID	returns the Sync Remote instance of a mirrored object
IPS_GetRemoteObject	returns the ID of a mirrored object on the remote system
IPS_GetReplicationFailoverTime	returns the point in time of the last failover
IPS_GetReplicationMode	returns the mode of the replication
IPS_GetReplicationSyncTime	returns the point in time of the last successful synchronization
IPS_IsReplicationActive	checks whether the replication is active
IPS_IsReplicationFailoverActive	checks whether the slave has taken over the tasks of the master
IPS_PromoteToReplicationMaster	promotes the slave to master

SNAPSHOTS

Functions in this group return the complete state of IP-Symcon and its changes for clients.

IPS_GetSnapshot	returns the complete state of IP-Symcon
IPS_GetSnapshotChanges	returns the changes since a specific message
IPS_GetSnapshotDiff	returns the differences to an earlier snapshot

SPECIAL SWITCHES

Functions in this group read and set the special switches (options) of IP-Symcon.

IPS_GetOption	returns the value of a special switch
IPS_GetOptionEx	returns the current, stored and default value of a special switch
IPS_GetOptionList	returns the names of all special switches
IPS_SetOption	sets the value of a special switch

STATISTICS

Functions in this group return runtime statistics of IP-Symcon.

IPS_GetActiveProxyConnections	returns the number of active video stream proxy connections
IPS_GetActiveWebRTCConnections	returns the number of open WebRTC sessions
IPS_GetActiveWebServerConnections	returns the number of open HTTP connections of the web server
IPS_GetActiveWebSocketConnections	returns the number of open WebSocket connections of the web server
IPS_GetAllocatorStats	returns the statistics of the memory allocator
IPS_GetConnectionList	returns the send queues of the connections of the web server

IPS_GetFileCacheStats	returns the size of the file cache of the WebServer
IPS_GetFlowScriptStatistic	returns the execution statistics of a flow plan
IPS_GetInstanceDataFlowStatistics	returns the statistics of the data flow per instance
IPS_GetInstanceMessageQueueSize	returns the number of pending messages to instances
IPS_GetInstanceMessageStatistics	returns the statistics of the message processing per instance
IPS_GetInstanceStats	returns the memory usage of the instances
IPS_GetMediaStats	returns the memory usage of the media objects
IPS_GetSettingsStats	returns the size of the settings
IPS_GetSnapshotBufferStats	returns the size of the snapshot buffer
IPS_GetSnapshotRingBufferStats	returns the size of the message ring buffer
IPS_GetStatsHistory	returns the history of the memory statistics
IPS_GetThreadStats	returns the memory usage of the script threads
IPS_GetTraceStats	returns the size of the trace memory
IPS_GetVariableStats	returns the memory usage of the variables

Helpers

The functions in this group are only available via the JSON-RPC interface and return the information of several objects in one call, e.g. for the console. They are not available in PHP scripts, where the respective functions for single objects are used instead.

IPS_GetConfigurationsFromChildren	returns the configuration which the child instances specify for an instance
IPS_GetFunctions	returns comprehensive information about several functions
IPS_GetIcons	returns all icons including the icons of the skins
IPS_GetLibraries	returns information about several libraries
IPS_GetModules	returns information about several modules
IPS_GetScriptThreads	returns information about several threads of the script engine
IPS_GetTimers	returns information about several timers
IPS_GetTraces	returns all traces of an object including their causes

IPS_GetConfigurationsFromChildren

```
array IPS_GetConfigurationsFromChildren (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

Object or associative array with the properties which the child instances specify for the instance. The key is the name of the property, the value is the required value. If no child instance specifies anything, the result is empty.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function determines the configuration of all child instances of the instance with the ID **InstanceID**, i.e. all instances which are connected to it as parent instance, via IPS_GetConfigurationForParent and merges them into one object. The management console uses the result to lock the properties specified this way in the configuration form of the instance.



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts – there IPS_GetConfigurationForParent can be called for each child instance.

If several child instances specify the same property with different values, the function fails with "Child nodes provide different configurations". If a child instance does not return valid JSON or no JSON object, the function fails with "Invalid JSON for InstanceID ..." or "JSON is missing object for InstanceID ...". If the instance does not exist, the function fails with "Flow for Instance #... does not exist".

EXAMPLE

```
// Determine the configuration which the child instances specify for
instance 23456 (e.g. a Client Socket)
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
print_r($rpc->IPS_GetConfigurationsFromChildren(23456));

/* e.g. returns:
Array
(
[Port] => 21000
)
*/
```

IPS_GetFunctions

```
array IPS_GetFunctions (array $FunctionNames)
```

PARAMETERS

FunctionNames	Array with the names of the functions (string). If the array is empty, all functions are returned

RETURNS

An array with one entry per function in the order of **FunctionNames**. If **FunctionNames** is empty, the order of the entries is not guaranteed. Each entry has the same structure as in IPS_GetFunction.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the information of the functions with the names **FunctionNames** in one call. If **FunctionNames** is empty, all functions registered in IP-Symcon are returned (see IPS_GetFunctionList).



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts ("This function is not available for PHP") – there the result can be determined with IPS_GetFunctionList and IPS_GetFunction.

If an entry of **FunctionNames** is not a string, the function fails with "Parameter type of FunctionNames does not match". If one of the functions does not exist, the function fails with "Function with name ... not found!".

EXAMPLE

```
// Query information about a function via the JSON-RPC interface
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
```

```
print_r($rpc->IPS_GetFunctions(["IPS_SemaphoreEnter"]));

/* e.g. returns:
Array
(
[0] => Array
(
[FunctionName] => IPS_SemaphoreEnter
[Parameters] => Array
(
[0] => Array
(
[Description] => Name
[Enumeration] => Array
(
)

[Type_] => 3
)

[1] => Array
(
[Description] => Milliseconds
[Enumeration] => Array
(
)

[Type_] => 1
)

)

[Result] => Array
(
[Description] => Result
[Enumeration] => Array
(
)

[Type_] => 0
)

)

)
*/
```

IPS_GetIcons

```
array IPS_GetIcons ()
```

RETURNS

An array with the icons grouped by skin. The structure is listed in the description.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns all available icons in one call. The keys of the array are the names of the skins, the values are arrays which map the name of each icon to the Base64 encoded file content.

KEY	CONTENT
"" (empty)	The icons of Symcon as Base64 encoded SVG files. The sub icons of adaptive icons (name with "-") are not included
Name of the skin	The icons of the installed skin as Base64 encoded PNG files



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts.

If the core instance "Skin Control" or "Util Control" is missing, the function fails with "Cannot find skin control" or "Cannot find util control". If the icons of a skin cannot be read, it fails with "Cannot load Icons for Skin ...: ...".

EXAMPLE

```
// Show the number of icons per skin via the JSON-RPC interface
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
foreach ($rpc->IPS_GetIcons() as $skin => $icons) {
printf("%s: %d icons\n", $skin == '' ? 'Symcon' : $skin,
count($icons));
```

```
}

/* e.g. returns:
Symcon: 165 icons
*/
```

IPS_GetLibraries

```
array IPS_GetLibraries (array $LibraryIDs)
```

PARAMETERS

LibraryIDs	Array with the LibraryIDs (string). If the array is empty, all libraries are returned

RETURNS

An array with one entry per library in the order of **LibraryIDs**. If **LibraryIDs** is empty, the order of the entries is not guaranteed. Each entry has the same structure as in IPS_GetLibrary.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the information of the libraries with the IDs **LibraryIDs** in one call. If **LibraryIDs** is empty, all libraries are returned.



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts ("This function is not available for PHP") – there the result can be determined with IPS_GetLibraryList and IPS_GetLibrary.

If an entry of **LibraryIDs** is not a string, the function fails with "Parameter type of LibraryIDs does not match". If one of the libraries does not exist, the function fails with "Cannot find library with GUID ...".

EXAMPLE

```
// Show name and version of all libraries via the JSON-RPC interface
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
foreach ($rpc->IPS_GetLibraries([]) as $library) {
```

```
printf("%s: %s (Build %d)\n", $library['Name'], $library['Version'],
$library['Build']);
}

/* e.g. returns:
SzenenSteuerung: 1.1 (Build 0)
*/
```

IPS_GetModules

```
array IPS_GetModules (array $ModuleIDs)
```

PARAMETERS

ModuleIDs	Array with the ModuleIDs (string). If the array is empty, all modules are returned

RETURNS

An array with one entry per module in the order of **ModuleIDs**. If **ModuleIDs** is empty, the order of the entries is not guaranteed. Each entry has the same structure as in IPS_GetModule.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the information of the modules with the IDs **ModuleIDs** in one call. If **ModuleIDs** is empty, all modules are returned.



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts ("This function is not available for PHP") – there the result can be determined with IPS_GetModuleList and IPS_GetModule.

If an entry of **ModuleIDs** is not a string, the function fails with "Parameter type of ModuleIDs does not match". If one of the modules does not exist, the function fails with "Cannot find module with GUID ...", on an invalid GUID with "ModuleID ... is not a valid GUID".

EXAMPLE

```
// Query information about a module via the JSON-RPC interface
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
print_r($rpc->IPS_GetModules(["{C3380B6E-2351-4A19-9668-
```

```
A17960F97E51}"]));

/* e.g. returns:
Array
(
[0] => Array
(
[Aliases] => Array
(
)

[ChildRequirements] => Array
(
)

[Implemented] => Array
(
[0] => {018EF6B5-AB94-40C6-AA53-46943E824ACF}
)

[LibraryID] => {0945206A-47AA-4FDD-9093-99051E410E82}
[Limit] => 0
[ModuleID] => {C3380B6E-2351-4A19-9668-A17960F97E51}
[ModuleName] => ThermoJet
[ModuleType] => 3
[ParentRequirements] => Array
(
[0] => {79827379-F36E-4ADA-8A95-5F8D1DC92FA9}
)

[Prefix] => TJ
[Translation] => Array
(
[de] => Array
(
[ThermoJet] => ThermoJet
)

)

[URL] =>
[Vendor] => ProJet
)

)
*/
```

IPS_GetScriptThreads

```
array IPS_GetScriptThreads (array $ThreadIDs)
```

PARAMETERS

ThreadIDs	Array with the IDs of the threads. If the array is empty, all threads are returned

RETURNS

An array with one entry per thread in the order of **ThreadIDs**. If **ThreadIDs** is empty, the order of the entries is not guaranteed. Each entry has the same structure as in IPS_GetScriptThread.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the information of the threads with the IDs **ThreadIDs** in one call. If **ThreadIDs** is empty, all threads are returned. If one of the threads does not exist, the function fails with "ScriptThread #... does not exist".



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts ("This function is not available for PHP") – there the result can be determined with IPS_GetScriptThreadList and IPS_GetScriptThread.

PHP scripts are executed in a fixed pool of threads. The number of threads is set by the special switch ThreadCount (default 50, at least 25), and the threads have the IDs 1 to ThreadCount. Within a script, the system variable `$_IPS['THREAD']` contains the ID of the own thread.

EXAMPLE

```
// Show all threads that are currently busy via the JSON-RPC interface
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
foreach ($rpc->IPS_GetScriptThreads([]) as $thread) {
if ($thread['StartTime'] > 0) {
printf("Thread %d: %s (%s) for %d s\n", $thread['ThreadID'],
$thread['FilePath'], $thread['Sender'], time() - $thread['StartTime']);
}
}

/* e.g. returns:
Thread 3: scripts/12345.ips.php (RunScript) for 0 s
*/
```

IPS_GetTimers

```
array IPS_GetTimers (array $TimerIDs)
```

PARAMETERS

TimerIDs	Array with the IDs of the timers. If the array is empty, all timers are returned

RETURNS

An array with one entry per timer in the order of **TimerIDs**. If **TimerIDs** is empty, the order of the entries is not guaranteed. Each entry has the same structure as in IPS_GetTimer.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the information of the timers with the IDs **TimerIDs** in one call. If **TimerIDs** is empty, all timers are returned. If one of the timers does not exist, the function fails with "Timer #... does not exist".



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts ("This function is not available for PHP") – there the result can be determined with IPS_GetTimerList and IPS_GetTimer.

Timers are registered by instances, e.g. by PHP modules with RegisterTimer and SetTimerInterval or as one-time timers with RegisterOnceTimer. The built-in modules use timers as well, e.g. "SearchExpires" for the end of the search mode. The TimerIDs are assigned at runtime and are not persistent, one-time timers disappear after their execution. Script timers (IPS_SetScriptTimer) are cyclic events and do not belong to these timers.

EXAMPLE

```
// Show all timers of an instance via the JSON-RPC interface
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
```

```
foreach ($rpc->IPS_GetTimers([]) as $timer) {
if ($timer['InstanceID'] == 12345) {
printf("%s: %d ms, next run: %s\n", $timer['Name'], $timer['Interval'],
$timer['NextRun'] ? date('H:i:s', $timer['NextRun']) : '-');
}
}

/* e.g. returns:
Update: 60000 ms, next run: 12:01:00
*/
```

IPS_GetTraces

```
array IPS_GetTraces (int $ObjectID)
```

PARAMETERS

ObjectID	ID of the object whose traces are to be returned

RETURNS

An array of trace entries. The order of the entries is not guaranteed. Each entry contains the following information as key => value pairs:

INDEX	TYPE	DESCRIPTION
TraceID	integer	Unique, ascending ID of the entry
ParentID	integer	ID of the entry that caused this entry (0 = start of a chain)
ChildIDs	array	IDs of the entries caused by this entry
Depth	integer	Position in the chain, starting at 1 (0 = placeholder for an entry that is no longer available)
ObjectID	integer	ID of the object the entry belongs to (1 = no object, e.g. for WebServer)
Sender	string	Kind of entry (see sender table)
ExecutionUser	string	User on whose behalf it was executed (e.g. @admin; @unknown if no user was logged in)
TimeStamp	integer	Unix timestamp of the creation
Data	array	Additional information as key => value pairs, depending on the sender (see sender table)

Sender table

SENDER	DESCRIPTION	OBJECTID	DATA
Variable	A value was written to a variable (even without a change)	Variable	VariableID, Value, OldValue
RequestAction	The action of a variable was requested	Variable	VariableID, ActionID, Value, Sender
RequestAction	The RequestAction function of an instance was called for a status variable	Instance	Ident, Value

SENDER	DESCRIPTION	OBJECTID	DATA
TriggerEvent	A triggered event runs its action	Event	EventID, ActionID, Environment, Trigger, VariableID, Value
CyclicEvent	A cyclic event runs its action	Event	EventID, ActionID, Environment
Script	A script is executed	Script	Sender, ScriptID, FilePath, SenderID
Event	PHP code is executed on behalf of an event	Event	Sender, ScriptID, FilePath, SenderID
Instance	PHP code is executed for a target object	Target object	Sender, ScriptID, FilePath, SenderID
Timer	A timer of an instance is executed	Instance	TimerName, Instance, RunOnce
DataFlow	An instance receives data through the data flow	Receiving instance	InstanceID, ParentID
WebServer	A request to the WebServer (file/hook) or a JSON-RPC call	1	Path or Method, RemoteIP, ProxyIPs (only behind proxies)
System	Placeholder for an entry that is no longer available	1	Reason

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

In traces the kernel records what caused what: every entry represents one operation – e.g. writing a variable, the event triggered by it, its action or script, a RequestAction, a timer or a request to the WebServer – and refers to the entry that caused it via **ParentID** and to the entries it caused itself via **ChildIDs**. For every cause this results in a tree that shows, for example, why a variable has changed.
The function returns all kept entries of the object **ObjectID** together with all entries of their cause chains (parents, grandparents, … up to the start of the chain). Entries that occur in several chains are only contained once. The result is a flat list; the tree structure can be rebuilt using **TraceID**, **ParentID** and **ChildIDs**. The result therefore corresponds to IPS_GetTraceList for every ID from IPS_GetTraceIDList. For an object without traces an empty array is returned. If 1 is passed as **ObjectID**, the chains of all kept entries are returned.



The function is intended to be called through the JSON-RPC interface, e.g. for the display in the console. It is not available in PHP scripts ("This function is not available for PHP") – there the result can be determined with IPS_GetTraceIDList and IPS_GetTraceList.

Traces are only kept in memory and are lost on a restart. The last 25 entries are kept per object, and the last 1000 entries in total for entries without an object (e.g. WebServer). An older entry is kept as long as a kept entry still descends from it, so chains stay complete. When an object is deleted, its entries are removed. Strings in **Data** are shortened to 100 bytes and marked with "...".



If a chain is deeper than 100 entries, the kernel assumes an endless loop (e.g. an event that writes its own trigger variable) and aborts the operation with "Insight trace depth exceeded 100. Assuming an endless loop. Aborting. Reverse Chain: …".

EXAMPLE

```
$rpc = new JSONRPC("http://user:password@127.0.0.1:3777/api/");
print_r($rpc->IPS_GetTraces(10018));

/* e.g. returns:
Array
(
[0] => Array
(
[ChildIDs] => Array
(
)
[Data] => Array
(
[OldValue] => 0
[Value] => 5
[VariableID] => 10018
)
[Depth] => 1
[ExecutionUser] => @admin
[ObjectID] => 10018
[ParentID] => 0
[Sender] => Variable
[TimeStamp] => 1700000000
[TraceID] => 1
)
[1] => Array
(
[ChildIDs] => Array
(
)
[Data] => Array
(
[OldValue] => 5
[Value] => 7
[VariableID] => 10018
)
[Depth] => 1
[ExecutionUser] => @admin
[ObjectID] => 10018
[ParentID] => 0
```

```
[Sender] => Variable
[TimeStamp] => 1700000000
[TraceID] => 2
)
)
*/
```

License

IPS_GetDemoExpiration	returns the expiration time of the demo mode
IPS_GetLicensee	returns the licensee
IPS_GetLimitDemo	returns whether a demo license is used
IPS_GetLimitFeatures	returns the enabled features of the license
IPS_GetLimitServer	returns the server limitation of the license
IPS_GetLimitVariables	returns the maximum number of variables of the license
IPS_GetLimitWebFront	returns the maximum number of visualizations of the license
IPS_GetSubscriptionExpiration	returns the end of the subscription

IPS_GetDemoExpiration

```
int IPS_GetDemoExpiration ()
```

RETURNS

Unix timestamp at which the demo mode expires. 0 if there is no time limit.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the time at which the demo mode of IP-Symcon expires as Unix timestamp. For a valid full version with a sufficient subscription and for the Community Edition, 0 is returned.
The time is determined at the start of IP-Symcon:

SITUATION	EXPIRATION TIME
No valid license	1 hour after the start
Demo license with valid subscription	End of the subscription
Demo license whose subscription has expired or could not be retrieved	6 hours after the start
Full version whose subscription expired before the build date of the installed version	48 hours after the start

After expiration, various actions fail with "Demo expired! ...". The build date of the installed version is returned by IPS_GetKernelDate, the end of the subscription by IPS_GetSubscriptionExpiration.

EXAMPLE

```
$expiration = IPS_GetDemoExpiration();
if ($expiration == 0) {
echo "No time limit";
} else {
echo "Demo mode ends on " . date("Y-m-d H:i", $expiration);
}
```

IPS_GetLicensee

```
string IPS_GetLicensee ()
```

RETURNS

Licensee of the active license. Without a valid license, the string is empty.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the licensee of the active license. This is the username the license was entered with in the license management, usually an e-mail address.
If no valid license can be read, IP-Symcon runs in demo mode and the function returns an empty string.
If the license is changed, the function immediately returns the new licensee, even if a restart of IP-Symcon is required to fully apply the license.

EXAMPLE

```
echo IPS_GetLicensee(); // e.g. returns "user@example.com"
```

IPS_GetLimitDemo

```
int IPS_GetLimitDemo ()
```

RETURNS

0 if a full version is licensed, otherwise a value other than 0 (demo license or demo mode)

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns whether IP-Symcon runs with a demo license. For a full version the value is 0. If no valid license is available, IP-Symcon runs in demo mode and the function returns 1.
When a demo expires is returned by IPS_GetDemoExpiration. Note that a full version also falls back to the time-limited demo mode if its subscription has expired for the installed version; the function still returns 0 in this case. Whether a time limit applies should therefore be checked with IPS_GetDemoExpiration.

EXAMPLE

```
if (IPS_GetLimitDemo() == 0) {
echo "Full version";
} else {
echo "Demo until " . date("Y-m-d H:i", IPS_GetDemoExpiration());
}
```

IPS_GetLimitFeatures

```
array IPS_GetLimitFeatures ()
```

RETURNS

An array of strings with the enabled features. The order of the entries is not guaranteed. The entry "*" stands for all features.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the features that are enabled with the active license. Some functions and modules are only available if the corresponding feature is enabled.
If the array contains the entry "*", all features are enabled. An entry of the form "NAME-suffix" (e.g. "BACNET-50") enables the feature "NAME" as well. Without a valid license, the array is empty.

EXAMPLE

```
print_r(IPS_GetLimitFeatures());

/* e.g. returns:
Array
(
[0] => *
)
*/
```

IPS_GetLimitServer

```
string IPS_GetLimitServer ()
```

RETURNS

Server limitation of the license. Without a limitation, the string is empty.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the server limitation stored in the active license. Most licenses do not contain such a limitation; in this case and without a valid license, an empty string is returned. The other limitations of the license are returned by IPS_GetLimitVariables, IPS_GetLimitWebFront and IPS_GetLimitFeatures.

EXAMPLE

```
$server = IPS_GetLimitServer();
if ($server == "") {
echo "No server limitation";
} else {
echo "Server limitation: " . $server;
}
```

IPS_GetLimitVariables

```
int IPS_GetLimitVariables ()
```

RETURNS

Maximum number of variables. 0 means unlimited.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the maximum number of variables that can be used with the active license. If it is 0, the number is not limited. In demo mode without a valid license, the limit is 1000.
If there are more variables than the limit allows, the excess variables are locked. If the license is upgraded, locked variables are unlocked accordingly.

EXAMPLE

```
$limit = IPS_GetLimitVariables();
if ($limit == 0) {
echo "Unlimited number of variables";
} else {
printf("%d of %d variables used", count(IPS_GetVariableList()),
$limit);
}
```

IPS_GetLimitWebFront

```
int IPS_GetLimitWebFront ()
```

RETURNS

Maximum number of visualizations per type. 0 means unlimited.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the maximum number of visualization instances that can be configured with the active license. The limit applies separately to WebFront and tile visualizations. If it is 0, the number is not limited. In demo mode without a valid license, the limit is 5.
With a limit of 1, some functions are only available with Symcon Professional/Symcon Unlimited ("This function is only available with Symcon Professional/Symcon Unlimited"). If there are more visualizations than the limit allows, their configuration fails with a corresponding error.
Older licenses do not contain an explicit limit for visualizations. If the number of variables is limited there, 5 is returned.

EXAMPLE

```
$limit = IPS_GetLimitWebFront();
if ($limit == 0) {
echo "Unlimited number of visualizations";
} else {
echo "At most " . $limit . " visualizations per type";
}
```

IPS_GetSubscriptionExpiration

```
int IPS_GetSubscriptionExpiration ()
```

RETURNS

Unix timestamp of the end of the subscription. 0 if no subscription is known.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the end of the subscription of the active license as Unix timestamp. If no subscription is known, e.g. without a valid license, 0 is returned.
A version of IP-Symcon can only be used with a full version if its build date (IPS_GetKernelDate) is not after the end of the subscription. Otherwise IP-Symcon runs in the time-limited demo mode (IPS_GetDemoExpiration). The Community Edition does not require a subscription.
The end of the subscription is retrieved from the license server at startup and when required. If the subscription is extended, the function returns the new date after the next update.

EXAMPLE

```
$subscription = IPS_GetSubscriptionExpiration();
if ($subscription == 0) {
echo "No subscription available";
} else {
echo "Subscription valid until " . date("Y-m-d", $subscription);
if ($subscription < IPS_GetKernelDate()) {
echo " (not sufficient for the installed version)";
}
}
```

Replication

With the paid extension Mirroring, a system can be operated as mirror (slave) of another system (master). The replication is configured in the core instance "Replication" or in the file "replication.json" and is read when IP-Symcon starts. Changes to the configuration therefore only take effect after a restart of the service.
If the replication is active, the slave loads the configuration from the master on startup. During operation, it synchronizes the changes of the master every 5 seconds. Changes made directly on the slave are overwritten in the process.

MODE	CONSTANT	NAME	DESCRIPTION
0	`REPLICATION_MODE_PARALLEL`	Parallel	The slave is fully operational and all its connections are active.
1	`REPLICATION_MODE_STANDBY`	Standby	The I/O instances of the slave do not establish connections and remain in the status Standby (106, `IS_STANDBY`). Events and timers of instances are not executed. The slave only becomes active on failover.

FAILOVER

In the mode Standby, the slave can take over the tasks of the master by being promoted to master with IPS_PromoteToReplicationMaster. In doing so, the synchronization is paused, all instances in the status Standby are activated and events as well as timers are executed again. If the automatic failover is active (parameter "FailoverActive"), this happens automatically as soon as the synchronization with the master has failed as many times in a row as specified in the parameter "FailoverCount" (default: 20). If the configuration cannot be loaded from the master on startup even after several attempts, the slave starts as master as well. With IPS_DemoteToReplicationSlave, the slave is demoted again and the synchronization is resumed.

SYNC REMOTE

With the module Sync Remote, the object tree of a remote Symcon system is mirrored beneath the Sync Remote instance. Each mirrored object gets its own local ID, including the root object (ID 0) of the remote system. For each Sync Remote instance, Symcon stores the mapping between the ID on the remote system (RemoteID) and the local ID. The ID of the local Sync Remote instance serves as ServerID. The mapping is stored in the settings and removed when the local object is deleted. In a replication, a slave takes over the mappings of the master.
Switching mirrored variables (SetValue, RequestAction), module functions as well as modifying IPS_ *functions (e.g. IPS_Set*, IPS_Delete*, IPS_RunScript, IPS_ApplyChanges) whose first parameter is a mirrored object are forwarded using the mapping to the remote system on which the object resides. Reading functions (IPS_Get*, IPSIs*, IPS*\Exists, IPS_HasChildren, IPS_HasPermission) and the functions for converting the IDs act locally on the mirrored object. Exceptions are IPS_GetConfigurationForm, IPS_GetMediaContent, IPS_GetScriptContent and IPS_GetFlowScriptStatistic, which are forwarded. The debug functions are not available for mirrored instances and report an error.

With IPS_GetRemoteObject, IPS_FindObjectID, IPS_FindRemoteID and IPS_FindServerID, object IDs can be mapped between the local system and the remote systems. If no mapping exists, the Find functions return 1.

IPS_DemoteToReplicationSlave	demotes a slave promoted to master again
IPS_FindObjectID	returns the local ID for an ID of the remote system
IPS_FindRemoteID	returns the ID of the remote system for a local ID
IPS_FindServerID	returns the Sync Remote instance of a mirrored object
IPS_GetRemoteObject	returns the ID of a mirrored object on the remote system
IPS_GetReplicationFailoverTime	returns the point in time of the last failover
IPS_GetReplicationMode	returns the mode of the replication
IPS_GetReplicationSyncTime	returns the point in time of the last successful synchronization
IPS_IsReplicationActive	checks whether the replication is active
IPS_IsReplicationFailoverActive	checks whether the slave has taken over the tasks of the master
IPS_PromoteToReplicationMaster	promotes the slave to master

IPS_DemoteToReplicationSlave

```
bool IPS_DemoteToReplicationSlave ()
```

RETURNS

If the function succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function demotes a slave which has been promoted to master with IPS_PromoteToReplicationMaster or by an automatic failover back to slave. The instances which were activated on the promotion are put back into the status Standby and the synchronization with the master is resumed.



Changes made on the slave during the failover are overwritten by the configuration of the master on the next synchronization.

If the replication is not active, the function fails with "Replication is not active!". If the slave has not been promoted, the function fails with "Replication must have been promoted before a demote is allowed!". If the replication is not in the mode Standby, the function fails with "Replication must be in Standby-Mode!".

EXAMPLE

```
// Demote back to slave after a failover as soon as the master is
running again
if (IPS_IsReplicationFailoverActive()) {
IPS_DemoteToReplicationSlave();
}
```

IPS_FindObjectID

```
int IPS_FindObjectID (int $RemoteID, int $ServerID)
```

PARAMETERS

RemoteID	ID of the object on the remote system
ServerID	ID of the Sync Remote instance

RETURNS

Local ID of the mirrored object. If no mapping exists, 1 is returned.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function reads the mapping of object IDs between the local system and remote systems which are mirrored by Sync Remote instances (see Sync Remote). **ServerID** is the ID of the local Sync Remote instance.
The function returns the local ID of the object which mirrors the object **RemoteID** of the remote system of the Sync Remote instance **ServerID**. The root object of the remote system has the **RemoteID** 0. If no mapping exists, e.g. because the object has not been synchronized yet, the function returns 1.
The reverse direction is returned by IPS_GetRemoteObject.

EXAMPLE

```
$serverID = 54321; // ID of the Sync Remote instance
$objectID = IPS_FindObjectID(23456, $serverID);
if ($objectID == 1) {
echo "The object is not mirrored";
} else {
echo "Local ID: " . $objectID;
}

/* e.g. returns:
```

```
Local ID: 12345
*/
```

IPS_FindRemoteID

```
int IPS_FindRemoteID (int $ObjectID, int $ServerID)
```

PARAMETERS

ObjectID	ID of the object on the local system
ServerID	ID of the Sync Remote instance

RETURNS

ID of the object on the remote system. If no mapping exists, 1 is returned.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function reads the mapping of object IDs between the local system and remote systems which are mirrored by Sync Remote instances (see Sync Remote). **ServerID** is the ID of the local Sync Remote instance.
The function determines the ID which the local object **ObjectID** has on the remote system of the Sync Remote instance **ServerID**. If no mapping exists, the function returns 1.
The reverse direction is returned by IPS_FindObjectID.

EXAMPLE

```
$remoteID = IPS_FindRemoteID(12345, 54321);
if ($remoteID == 1) {
echo "The object is not mirrored by this Sync Remote instance";
} else {
echo "ID on the remote system: " . $remoteID;
}
```

IPS_FindServerID

```
int IPS_FindServerID (int $ObjectID)
```

PARAMETERS

ObjectID	ID of the object on the local system

RETURNS

ID of the Sync Remote instance which mirrors the object. If no mapping exists, 1 is returned.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function reads the mapping of object IDs between the local system and remote systems which are mirrored by Sync Remote instances (see Sync Remote). It determines the ID of the Sync Remote instance which mirrors the object **ObjectID**. If **ObjectID** is not a mirrored object, the function returns 1.

EXAMPLE

```
// Determine the Sync Remote instance of a mirrored object
$serverID = IPS_FindServerID(12345);
if ($serverID == 1) {
echo "The object is not mirrored";
} else {
echo "Sync Remote instance: " . IPS_GetName($serverID);
}
```

IPS_GetRemoteObject

```
int IPS_GetRemoteObject (int $ObjectID)
```

PARAMETERS

ObjectID	ID of the mirrored object on the local system

RETURNS

ID of the object on the remote system

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the ID which the mirrored object **ObjectID** has on the remote system. The responsible Sync Remote instance is determined automatically. The function reads the mapping of object IDs between the local system and the mirrored remote systems (see Sync Remote). For the local representation of the root object of the remote system, it returns 0. If **ObjectID** is not a mirrored object, the function fails with "Cannot find remote server for #...".
The reverse direction is returned by IPS_FindObjectID.

EXAMPLE

```
$remoteID = IPS_GetRemoteObject(12345);
echo "ID on the remote system: " . $remoteID;

/* e.g. returns:
ID on the remote system: 23456
*/
```

IPS_GetReplicationFailoverTime

```
int IPS_GetReplicationFailoverTime ()
```

RETURNS

Unix timestamp of the last promotion to master. If the slave has never been promoted, 0 is returned.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the point in time at which the slave was last promoted to master (failover), either via IPS_PromoteToReplicationMaster or automatically.
The value is kept even after IPS_DemoteToReplicationSlave and is only reset on a restart of the service. Whether the failover is currently active is returned by IPS_IsReplicationFailoverActive.

EXAMPLE

```
$failoverTime = IPS_GetReplicationFailoverTime();
if ($failoverTime > 0) {
echo "Last failover: " . date("Y-m-d H:i:s", $failoverTime);
}

/* e.g. returns:
Last failover: 2026-09-14 03:12:45
*/
```

IPS_GetReplicationMode

```
int IPS_GetReplicationMode ()
```

RETURNS

Mode of the replication

0	REPLICATION_MODE_PARALLEL	Parallel
1	REPLICATION_MODE_STANDBY	Standby

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the configured mode of the replication. In the mode Parallel, the slave is fully operational, in the mode Standby, its I/O instances remain in the status Standby and events are not executed until a failover.
If the replication is not active (IPS_IsReplicationActive), the value has no meaning.

EXAMPLE

```
$modes = [REPLICATION_MODE_PARALLEL => "Parallel",
REPLICATION_MODE_STANDBY => "Standby"];
echo "Mode: " . $modes[IPS_GetReplicationMode()];

/* e.g. returns:
Mode: Standby
*/
```

IPS_GetReplicationSyncTime

```
int IPS_GetReplicationSyncTime ()
```

RETURNS

Unix timestamp of the last successful synchronization. If no synchronization has taken place yet, 0 is returned.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the point in time at which the slave was last successfully synchronized with the master (see replication). This includes loading the configuration on startup. During operation, the synchronization takes place every 5 seconds, so a considerably older point in time indicates problems with the connection to the master.
If the replication is not active, 0 is returned.

EXAMPLE

```
$syncTime = IPS_GetReplicationSyncTime();
if ($syncTime > 0) {
echo "Last synchronization " . (time() - $syncTime) . " seconds ago";
}

/* e.g. returns:
Last synchronization 3 seconds ago
*/
```

IPS_IsReplicationActive

```
bool IPS_IsReplicationActive ()
```

RETURNS

TRUE if the replication is active, otherwise FALSE

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function checks whether the replication is active, i.e. whether the system is a slave. For this, the extension Mirroring must be licensed and the replication must have been activated when the service was started.
The result remains TRUE even after a promotion to master (IPS_PromoteToReplicationMaster). Whether the slave has currently taken over the tasks of the master is returned by IPS_IsReplicationFailoverActive.

EXAMPLE

```
if (IPS_IsReplicationActive() && !IPS_IsReplicationFailoverActive()) {
echo "This system is a slave and is synchronized by the master";
}
```

IPS_IsReplicationFailoverActive

```
bool IPS_IsReplicationFailoverActive ()
```

RETURNS

TRUE if the slave has been promoted to master, otherwise FALSE

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function checks whether the slave has been promoted to master and has thus taken over the tasks of the master (failover). This is the case after calling IPS_PromoteToReplicationMaster, after an automatic failover or if the configuration could not be loaded from the master on startup. As long as the result is TRUE, no synchronization with the master takes place. After IPS_DemoteToReplicationSlave, the result is FALSE again.



The function does not return whether the automatic failover is configured.

EXAMPLE

```
if (IPS_IsReplicationFailoverActive()) {
echo "Failover since " . date("Y-m-d H:i:s",
IPS_GetReplicationFailoverTime());
}

/* e.g. returns:
Failover since 2026-09-14 03:12:45
*/
```

IPS_PromoteToReplicationMaster

```
bool IPS_PromoteToReplicationMaster ()
```

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function promotes the slave to master, so that it takes over the tasks of the master (failover). The synchronization with the master is paused, the point in time for IPS_GetReplicationFailoverTime is set and all instances in the status Standby (106) are activated. The result of the activation is logged per instance in the message log.
The same happens automatically if the automatic failover is configured and the synchronization with the master fails repeatedly. The slave can be demoted again with IPS_DemoteToReplicationSlave.
If the replication is not active, the function fails with "Replication is not active!". If the replication is not in the mode Standby (IPS_GetReplicationMode), the function fails with "Replication must be in Standby-Mode!".

EXAMPLE

```
// Promote the slave to master manually, e.g. for a planned maintenance
of the master
if (IPS_IsReplicationActive() && !IPS_IsReplicationFailoverActive()) {
IPS_PromoteToReplicationMaster();
}
```

Snapshots

A snapshot is the complete state of IP-Symcon at a specific point in time. It contains all objects with their properties (e.g. variables with value and presentation, instances with configuration and status, scripts, events, media and links), all variable profiles and templates as well as information about the server, license, special switches, language and time zone. Clients such as the console, the visualizations, the apps or IPSView load the snapshot once to initialize themselves. Afterwards they follow all changes via the messages of IP-Symcon and apply them to their local state. They receive the messages via the WebSocket interface (see Data exchange) or by polling with IPS_GetSnapshotChanges. For this purpose every message carries a consecutive number, and the snapshot contains the number of the last message already included in it under **timestamp**. Visualizations use a snapshot that is filtered to their contents.
With IPS_GetSnapshotDiff a client that already has a snapshot can retrieve only the differences instead of the complete snapshot.

IPS_GetSnapshot	returns the complete state of IP-Symcon
IPS_GetSnapshotChanges	returns the changes since a specific message
IPS_GetSnapshotDiff	returns the differences to an earlier snapshot

IPS_GetSnapshot

```
string IPS_GetSnapshot ()
```

RETURNS

The snapshot as a JSON-encoded string. Via the JSON-RPC interface it is returned directly as an object.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.



The structure of the snapshot may change between versions of IP-Symcon.

The function returns the complete current state of IP-Symcon, the so-called snapshot. It is based on the saved settings (settings.json) and is enriched with runtime information. Clients initialize themselves with it and afterwards follow the changes via messages.
Among others, the snapshot contains the following entries:

KEY	DESCRIPTION
timestamp	Number of the last message included in the snapshot. Starting from this number, the following changes can be retrieved with IPS_GetSnapshotChanges
timezone	Time zone of the server in IANA format (e.g. "Europe/Berlin"). This allows clients e.g. to render graphs in the time zone of the server
language	System language of IP-Symcon (e.g. "de-DE")
server	Information about the server: version, revision, platform, architecture, date and starttime
license	Information about the license: licensee, edition, limits (demo, variables, webfront, server, features) and expiration (demo, subscription)
options	All special switches with their current, pending and default value
compatibility	Version and date of the server (version, date). Only present for compatibility reasons
objects	All objects. The key is "ID" followed by the ObjectID (e.g. "ID12345"). The type-specific data is located under **data**

KEY	DESCRIPTION
profiles	All variable profiles. **readOnly** is **TRUE** for system profiles (name starts with "~")
templates	All templates for presentations

Further entries of the saved settings, e.g. **sync** or **permissions**, are included unchanged.
Every object under **objects** contains its general properties (e.g. name, ident, parentID, position, type) as well as the summary. Depending on the object type, runtime information is added under **data**: status and visualizationType for instances, isLocked, profile and customProfile for variables, crc and size for scripts, nextRun for events and crc, size and isAvailable for media.
The snapshot is returned as uncompressed JSON. Via HTTP the response is compressed with gzip if the client specifies this in the "Accept-Encoding" header.
If the special switch "DebugDisableSnapshot" is active, the function fails with "This function is disabled for debugging purposes!".

EXAMPLE

```
// Retrieve the snapshot and output some information
$snapshot = json_decode(IPS_GetSnapshot(), true);
echo "Last message: " . $snapshot['timestamp'] . "\n";
echo "Time zone: " . $snapshot['timezone'] . "\n";
echo "Objects: " . count($snapshot['objects']) . "\n";
echo "Profiles: " . count($snapshot['profiles']) . "\n";

// Objects are stored under "ID" followed by the ObjectID
print_r($snapshot['objects']['ID12345']);

/* e.g. returns:
Last message: 990
Time zone: Europe/Berlin
Objects: 166
Profiles: 111
Array
(
[data] => Array
(
[action] => 54227
[customAction] => 0
[customPresentation] => Array
(
)

[customProfile] =>
[isLocked] =>
[lastChange] => 1789241479
[lastUpdate] => 1790080547
[presentation] => Array
(
[MAX] => 254
[MIN] => 1
[PERCENTAGE] => 1
[PRESENTATION] => {6B9CAEEC-5958-C223-30F7-BD36569FC57A}
```

```
[SUFFIX] => %
[USAGE_TYPE] => 2
)

[profile] =>
[type] => 1
[value] => 254
)

[disabled] =>
[hidden] =>
[hiddenMaximize] =>
[hiddenTitle] =>
[icon] =>
[ident] => Brightness
[info] =>
[name] => Brightness
[parentID] => 23456
[position] => 0
[readOnly] =>
[summary] =>
[type] => 2
)
*/
```

IPS_GetSnapshotChanges

```
string IPS_GetSnapshotChanges (int $LastTimestamp)
```

PARAMETERS

LastTimestamp	Number of the last known message (**timestamp** from the snapshot or **TimeStamp** of the last received message)

RETURNS

A JSON-encoded array with all messages after **LastTimestamp** in ascending order. Via the JSON-RPC interface it is returned directly as an array.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.



This function is planned to be removed in the future, once the old iOS/Android apps are no longer in use and IPSView has switched to WebSockets. Changes should be received via the WebSocket interface instead, see Data exchange.

The function returns all messages that occurred after the message with the number **LastTimestamp**. Clients that initialized themselves with IPS_GetSnapshot can thereby keep their state up to date by polling regularly, see Snapshots.
For this purpose IP-Symcon stores the last messages in a ring buffer. Every message receives a consecutive number (**TimeStamp**). Despite its name, this is not a point in time but a counter that restarts on every start of IP-Symcon. The size of the ring buffer is defined by the special switch "MessageRingBufferSize" (default: 8192 messages).
Every entry of the array contains the following fields:

FIELD	TYPE	DESCRIPTION
TimeStamp	integer	Consecutive number of the message
SenderID	integer	ID of the object that sent the message (0 for the kernel)
Message	integer	ID of the message, see messages
Data	array	Data of the message. Structure and number depend on the respective message

If **LastTimestamp** equals the number of the most recent message, the function returns an empty array. If **LastTimestamp** is 0, it returns only the most recent message.
The following errors are possible:
If **LastTimestamp** is greater than the number of the most recent message, the function fails with "Requests into the future are not possible".
If **LastTimestamp** lies back more messages than the ring buffer can hold, it fails with "Request range is bigger than message buffer".
If requested messages have already been overwritten in the ring buffer, it fails with "Invalid request. Check timestamp" or "Invalid timestamp requested ...".
If the special switch "DebugDisableSnapshotChanges" is active, it fails with "This function is disabled for debugging purposes!".
In these cases, and after a restart of IP-Symcon, the client has to reload the snapshot.

EXAMPLE

```
// Load the snapshot and remember the number of the last included
message
$snapshot = json_decode(IPS_GetSnapshot(), true);
$lastTimestamp = $snapshot['timestamp'];

// ... later: retrieve all messages that occurred since then
$changes = json_decode(IPS_GetSnapshotChanges($lastTimestamp), true);
foreach ($changes as $message) {
printf("#%d: Sender %d, Message %d, Data %s\n", $message['TimeStamp'],
$message['SenderID'], $message['Message'],
json_encode($message['Data']));
$lastTimestamp = $message['TimeStamp'];
}

/* e.g. returns:
#991: Sender 0, Message 10202, Data ["Kernel","*** IPS
READY",1790081169]
*/
```

IPS_GetSnapshotDiff

```
string IPS_GetSnapshotDiff (string $Hash)
```

PARAMETERS

Hash	Hash of the snapshot the client already has. For an unknown hash (e.g. an empty string) the complete snapshot is returned

RETURNS

A JSON-encoded string. If **Hash** is known, it contains the differences as JSON Patch (array), otherwise the complete snapshot (object) with the additional entry **hash**. Via the JSON-RPC interface the result is returned directly as an array or object.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function determines the current snapshot like IPS_GetSnapshot and calculates its hash (CRC32 of the JSON-encoded snapshot, as a hexadecimal string in upper case, e.g. "2ADC9D3C"). The current snapshot is cached under this hash so that it can serve as the base for the next call.

If a cached snapshot exists for **Hash**, the function returns the differences between it and the current snapshot as JSON Patch according to RFC 6902. This is an array of operations with the fields **op** (e.g. "add", "remove", "replace"), **path** and, if applicable, **value**. The new hash is contained in the operation for the path "/hash".

Otherwise, e.g. on the first call with an empty string or after a restart of IP-Symcon, it returns the complete snapshot with the additional entry **hash**.

A client can distinguish the two cases by whether an array or an object is returned.

Every hash can only be used once, as the associated snapshot is discarded after the differences have been calculated. If the snapshot is unchanged, the JSON Patch is empty and the hash stays the same. As it has already been used, however, the next call with this hash returns the complete snapshot again.

At most 10 snapshots are cached, snapshots older than 24 hours are discarded. If several clients use the function, snapshots can therefore be displaced. The client then receives the complete snapshot again.

If the special switch "DebugDisableSnapshot" is active, the function fails with "This function is disabled for debugging purposes!".

EXAMPLE

```
// First call with an unknown hash: returns the complete snapshot
including the hash
$snapshot = json_decode(IPS_GetSnapshotDiff(''), true);
$hash = $snapshot['hash'];

// ... later: retrieve only the differences as JSON Patch
$patch = json_decode(IPS_GetSnapshotDiff($hash), true);
foreach ($patch as $operation) {
echo $operation['op'] . ' ' . $operation['path'] . "\n";
// Remember the new hash for the next call
if ($operation['path'] == '/hash') {
$hash = $operation['value'];
}
}

/* e.g. returns:
replace /hash
replace /objects/ID12345/data/lastChange
replace /objects/ID12345/data/lastUpdate
replace /objects/ID12345/data/value
replace /timestamp
*/
```

Special Switches

IPS_GetOption	returns the value of a special switch
IPS_GetOptionEx	returns the current, stored and default value of a special switch
IPS_GetOptionList	returns the names of all special switches
IPS_SetOption	sets the value of a special switch

IPS_GetOption

```
mixed IPS_GetOption (string $Option)
```

PARAMETERS

Option	Name of the option, e.g. "ThreadCount"

RETURNS

Stored value of the option. The data type (bool, int, float or string) matches the type of the option.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the value of the special switch **Option**. An overview of all available options is returned by IPS_GetOptionList.
The value stored in the settings is returned. If the option was changed with IPS_SetOption, this already is the new value, even if it only takes effect after a restart of IP-Symcon. The value currently in effect and the default value are returned by IPS_GetOptionEx.
If the option does not exist, the function fails with "Cannot find option ... node in settings".

EXAMPLE

```
echo IPS_GetOption("ThreadCount"); // e.g. returns 50

var_dump(IPS_GetOption("ScriptWatch"));

/* e.g. returns:
bool(true)
*/
```

IPS_GetOptionEx

```
array IPS_GetOptionEx (string $Option)
```

PARAMETERS

Option	Name of the option, e.g. "ThreadCount"

RETURNS

The following key => value pairs are included. The data type of the values (bool, int, float or string) matches the type of the option:

INDEX	TYPE	DESCRIPTION
Current	mixed	Value currently in effect, i.e. the value at the last start of IP-Symcon
Default	mixed	Default value of the option
Pending	mixed	Stored value that applies from the next start (matches IPS_GetOption)

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the value currently in effect, the default value and the stored value of the special switch **Option**.
If an option is changed with IPS_SetOption, the new value is stored immediately and returned as **Pending**. **Current** keeps the value IP-Symcon was started with until the next restart. If **Current** and **Pending** differ, a restart is therefore required to apply the change. Without a change since the start, both values are equal.
If the option does not exist, the function fails with "Cannot find option ... node in settings". For options that are still present in the settings but are no longer used by the installed version, it fails with "Cannot find default value for this option".

EXAMPLE

```
$option = IPS_GetOptionEx("ThreadCount");
print_r($option);
```

```
if ($option['Current'] !== $option['Pending']) {
echo "The change only takes effect after a restart";
}

/* e.g. returns:
Array
(
[Current] => 50
[Default] => 50
[Pending] => 100
)
The change only takes effect after a restart
*/
```

IPS_GetOptionList

```
array IPS_GetOptionList ()
```

RETURNS

An array of strings with the names of all special switches. The order of the entries is not guaranteed.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the names of all special switches that are used by the installed version of IP-Symcon. Options that are only left in the settings (e.g. from older versions) are not included.
The values of the individual options can be queried with IPS_GetOption or IPS_GetOptionEx and changed with IPS_SetOption.

EXAMPLE

```
foreach (IPS_GetOptionList() as $option) {
echo $option . " = " . var_export(IPS_GetOption($option), true) . "\n";
}

/* e.g. returns:
ArchiveCommitInterval = 60
ArchiveRecordLimit = 10000
ArchiveWatch = false
BackupCount = 25
etc. ...
*/
```

IPS_SetOption

```
bool IPS_SetOption (string $Option, mixed $Value)
```

PARAMETERS

Option	Name of the option, e.g. "ThreadCount"
Value	New value of the option

RETURNS

If the function succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function sets the special switch **Option** to the value **Value**. The names of all options are returned by IPS_GetOptionList.



Most options are only applied by IP-Symcon at startup. A change therefore usually only takes effect after a restart of IP-Symcon.

The new value is stored in the settings immediately and returned by IPS_GetOption. IPS_GetOptionEx keeps returning the value IP-Symcon was started with as **Current** until a restart has taken place.
Value is converted to the data type of the option. For a boolean option, an integer or float value greater than 0 results in **TRUE**, a string only results in **TRUE** for "True". For an integer or float option, a string is interpreted as a number, for a string option numbers are stored as text.
If the option does not exist, the function fails with "Cannot find option ... node in settings". New options cannot be created with this function.

EXAMPLE

```
// Disable logging of script executions
IPS_SetOption("ScriptWatch", false);

// Increase the number of PHP threads
IPS_SetOption("ThreadCount", 100);

// The changes only take effect after a restart of IP-Symcon
```

Statistics

The functions in this group return runtime statistics of the kernel of IP-Symcon, e.g. about the memory usage of individual areas, the processing of messages and data or connections. They are mainly used for the diagnostics in the console. Which values are returned and how they are structured may change at any time.

IPS_GetActiveProxyConnections	returns the number of active video stream proxy connections
IPS_GetActiveWebRTCConnections	returns the number of open WebRTC sessions
IPS_GetActiveWebServerConnections	returns the number of open HTTP connections of the web server
IPS_GetActiveWebSocketConnections	returns the number of open WebSocket connections of the web server
IPS_GetAllocatorStats	returns the statistics of the memory allocator
IPS_GetConnectionList	returns the send queues of the connections of the web server
IPS_GetFileCacheStats	returns the size of the file cache of the WebServer
IPS_GetFlowScriptStatistic	returns the execution statistics of a flow plan
IPS_GetInstanceDataFlowStatistics	returns the statistics of the data flow per instance
IPS_GetInstanceMessageQueueSize	returns the number of pending messages to instances
IPS_GetInstanceMessageStatistics	returns the statistics of the message processing per instance
IPS_GetInstanceStats	returns the memory usage of the instances
IPS_GetMediaStats	returns the memory usage of the media objects
IPS_GetSettingsStats	returns the size of the settings
IPS_GetSnapshotBufferStats	returns the size of the snapshot buffer
IPS_GetSnapshotRingBufferStats	returns the size of the message ring buffer
IPS_GetStatsHistory	returns the history of the memory statistics
IPS_GetThreadStats	returns the memory usage of the script threads
IPS_GetTraceStats	returns the size of the trace memory
IPS_GetVariableStats	returns the memory usage of the variables

IPS_GetActiveProxyConnections

```
int IPS_GetActiveProxyConnections ()
```

RETURNS

Number of clients currently receiving a proxied video stream

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the number of clients currently receiving a video stream proxied by the web server. For each MJPEG or RTSP source, e.g. a camera in the visualization, Symcon establishes only one connection and distributes the stream to all requesting clients. The active clients across all of these streams are counted.
On systems without support for proxying video streams, the function returns 0.
WebRTC connections of the VideoServer are returned by IPS_GetActiveWebRTCConnections.

EXAMPLE

```
echo "Active proxy connections: " . IPS_GetActiveProxyConnections();

/* e.g. returns:
Active proxy connections: 2
*/
```

IPS_GetActiveWebRTCConnections

```
int IPS_GetActiveWebRTCConnections ()
```

RETURNS

Number of open WebRTC sessions of the VideoServer

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the number of WebRTC sessions of the VideoServer which have not been closed yet. WebRTC is used, for example, to transfer media streams of cameras or door intercoms to the visualization.
If the VideoServer is not running, the function fails with "VideoServer is not available!". On systems without VideoServer, it fails with "VideoServer is not available for this build!".
Proxied MJPEG and RTSP streams are returned by IPS_GetActiveProxyConnections.

EXAMPLE

```
echo "Open WebRTC sessions: " . IPS_GetActiveWebRTCConnections();

/* e.g. returns:
Open WebRTC sessions: 1
*/
```

IPS_GetActiveWebServerConnections

```
int IPS_GetActiveWebServerConnections ()
```

RETURNS

Number of currently open HTTP and HTTPS connections

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the number of currently open HTTP and HTTPS connections across all servers operated by the web server of Symcon. This includes the default port 3777 as well as the ports of the WebServer instances.
WebSocket connections are not included, they are returned by IPS_GetActiveWebSocketConnections.

EXAMPLE

```
echo "Open HTTP connections: " . IPS_GetActiveWebServerConnections();

/* e.g. returns:
Open HTTP connections: 4
*/
```

IPS_GetActiveWebSocketConnections

```
int IPS_GetActiveWebSocketConnections ()
```

RETURNS

Number of currently open WebSocket connections

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the number of currently open WebSocket connections (unencrypted and encrypted) across all servers operated by the web server of Symcon – the default port 3777 as well as the ports of the WebServer instances. Visualizations and the console, for example, are connected via WebSocket to receive messages.
The regular HTTP connections are returned by IPS_GetActiveWebServerConnections. Details about the individual connections and their send queues are returned by IPS_GetConnectionList.

EXAMPLE

```
echo "Open WebSocket connections: " .
IPS_GetActiveWebSocketConnections();

/* e.g. returns:
Open WebSocket connections: 3
*/
```

IPS_GetAllocatorStats

```
string IPS_GetAllocatorStats ()
```

RETURNS

Multi-line text with the statistics of the memory allocator

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the statistics of the memory allocator IP-Symcon uses to reserve memory as text. Content and format depend on the allocator in use and are taken over unchanged from it:

PLATFORM	ALLOCATOR	RETURN
Linux, macOS	mimalloc	Output of mimalloc (table with heap, page and segment statistics)
Linux (diagnostics builds)	jemalloc	Output of jemalloc (malloc_stats_print)
Windows	System allocator	"The system allocator is in use. No statistics are available."

The notice of the system allocator is returned in the configured language. The output is intended for troubleshooting and not for automated evaluation. The memory occupied by the allocator is additionally recorded in the series "Allocator" of IPS_GetStatsHistory.

EXAMPLE

```
// Output the statistics of the memory allocator
echo IPS_GetAllocatorStats();
```

IPS_GetConnectionList

```
array IPS_GetConnectionList ()
```

RETURNS

An array with one entry per connection. The order of the entries is not guaranteed. The structure is listed in the description.

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the connections through which the web server sends data to clients. These are WebSocket connections (e.g. messages to visualizations) and proxy connections (proxied video streams). For each of these connections, Symcon maintains a send queue.

KEY	TYPE	DESCRIPTION
Remote	string	IP address and port of the client, e.g. "192.168.1.20:52814"
ConnectionType	string	"WebSocket" or "Proxy"
ConnectedSince	int	Unix timestamp at which the first message was queued for the connection
Error	bool	TRUE if the connection has been marked as faulty because of an overflowing queue
QueueSize	int	Number of messages waiting in the queue for transmission
QueueBytes	int	Size of the waiting messages in bytes
TransferredCount	int	Number of all queued messages
TransferredBytes	int	Size of all queued messages in bytes
LastSuccessfulQueueing	int	Unix timestamp of the most recently queued message
LastSuccessfulTransfer	int	Unix timestamp of the most recently completed transmission

A connection is marked as faulty and does not receive any further messages if its queue contains more than 65,536 messages or 32 MB, or if no transmission has been completed for 5 minutes. If no transmission has been completed for 10 seconds, more than 32,768 messages or 8 MB are already sufficient. The specified limits are the default values.

Entries are only removed when no transmission has taken place for one hour. The list can therefore also contain connections which have already been closed. The number of currently open connections is returned by IPS_GetActiveWebSocketConnections and IPS_GetActiveProxyConnections.

EXAMPLE

```
foreach (IPS_GetConnectionList() as $connection) {
printf("%s (%s): %d messages queued, %d messages total%s\n",
$connection['Remote'], $connection['ConnectionType'],
$connection['QueueSize'], $connection['TransferredCount'],
$connection['Error'] ? ', faulty' : '');
}

/* e.g. returns:
192.168.1.20:52814 (WebSocket): 0 messages queued, 1532 messages total
*/
```

IPS_GetFileCacheStats

```
array IPS_GetFileCacheStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of files in the cache
TotalBytes	integer	Estimated memory usage of the cache in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the size of the file cache of the WebServer. So that large, frequently requested files of the console and the tile visualization (e.g. the JavaScript of the console and the code editor) do not have to be read and compressed on every request, they are stored once gzip compressed including their ETag hash in memory after the start of IP-Symcon. As long as the cache has not been built yet, the function returns 0 for both values. The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement.

EXAMPLE

```
// Output the size of the file cache
print_r(IPS_GetFileCacheStats());

/* e.g. returns:
Array
(
[Count] => 5
[TotalBytes] => 4718592
)
*/
```

IPS_GetFlowScriptStatistic

```
array IPS_GetFlowScriptStatistic (int $ScriptID)
```

PARAMETERS

ScriptID	ID of the flow plan

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
RunCount	integer	Number of started runs since the start of IP-Symcon
Queued	array	Queued and running runs (see run table)
Completed	array	The last (at most 25) finished runs (see run table)

Run table

INDEX	TYPE	DESCRIPTION
RunID	integer	Consecutive number of the run
QueueTime	integer	Unix timestamp at which the run was queued
StartTime	integer	Unix timestamp of the start (0 = not started yet)
CurrentStep	array	Path to the current step (see step table)
CurrentRetry	integer	Number of retries of the current step so far
FlowScript	string	Content of the flow plan (JSON) at the time of the run
Parameters	array	Parameters of the run as key => value pairs
Status	integer	Status of the run (see status table)
Executed	array	Executed steps (see execution table)

Step table

INDEX	TYPE	DESCRIPTION
ActionList	string	Name of the parameter with the nested action list (empty = top level)
ActionStep	integer	Index of the action in this action list, starting at 0

Execution table

INDEX	TYPE	DESCRIPTION
Step	array	Path to the step (see step table)
Executed	integer	Unix timestamp of the execution
Duration	integer	Duration of the execution in milliseconds
Success	boolean	**TRUE** if the step was successful
Result	string	Output or error message of the step
Retries	integer	Number of retries

Status table

VALUE	DESCRIPTION
0	Queued
1	Running
2	Finished successfully
3	Skipped
4	Aborted (details in the log)
5	Aborted by the user
6	Aborted by a rerun
7	Aborted by a warning
8	Aborted by an error
9	Aborted by an action

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the execution statistics of the flow plan with the ID **ScriptID**, e.g. for the display of the runs in the console. It contains the queued and running as well as the last 25 finished runs with their executed steps. The statistics are only kept in memory and are empty after a restart of IP-Symcon. If the flow plan has not been executed since then, **RunCount** is 0 and the lists are empty.
If **ScriptID** is not a flow plan, the function fails with "Script ... is not a flow script!", if the script does not exist, with "Script #... does not exist".
A path (**CurrentStep**, **Step**) starts at the top level of the flow plan. Every further entry descends into a nested action list of the previous action, e.g. into the actions of a condition.
There are no PHP constants for the status values.

EXAMPLE

```
// Output the finished runs of a flow plan
$statistic = IPS_GetFlowScriptStatistic(12345);
foreach ($statistic['Completed'] as $run) {
printf("Run %d: status %d, %d steps\n", $run['RunID'], $run['Status'],
count($run['Executed']));
}

/* e.g. returns:
Run 1: status 2, 3 steps
Run 2: status 8, 2 steps
*/
```

IPS_GetInstanceDataFlowStatistics

```
array IPS_GetInstanceDataFlowStatistics ()
```

RETURNS

An array with one entry per instance. The order of the entries is not guaranteed. Each entry contains the following key => value pairs:

INDEX	TYPE	DESCRIPTION
InstanceID	integer	ID of the sending instance
TxCount	integer	Number of data sent to the parent instance
TxDuration	integer	Total processing duration in the parent instance in milliseconds
TxMaxDuration	integer	Longest processing in the parent instance in milliseconds
RxCount	integer	Number of deliveries to child instances
RxDuration	integer	Total processing duration in the child instances in milliseconds
RxMaxDuration	integer	Longest processing in a child instance in milliseconds

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the statistics of the data flow per instance. Tx records the data an instance sends to its parent instance and the duration until it has processed the data. Rx records the data an instance sends to its child instances and the duration until they have processed the data. Each delivery to a child instance is counted separately. The values are summed up since the start of IP-Symcon, durations are rounded down to whole milliseconds.



The statistics are only recorded if the special switch "DataFlowWatch" is enabled (e.g. with IPS_SetOption). Otherwise the function fails with "This function is only available if the special option 'DataFlowWatch' is enabled!".

If "DataFlowWatch" is enabled, a message is additionally written to the log if a processing takes longer than "FlowWatchTxLimit" or "FlowWatchRxLimit" milliseconds (0 = disabled).

EXAMPLE

```
// Output the instances with the longest processing in the data flow
$statistics = IPS_GetInstanceDataFlowStatistics();
usort($statistics, function ($a, $b) {
return ($b['TxDuration'] + $b['RxDuration']) <=> ($a['TxDuration'] +
$a['RxDuration']);
});
foreach (array_slice($statistics, 0, 3) as $entry) {
printf("#%d: Tx %d (%d ms), Rx %d (%d ms)\n", $entry['InstanceID'],
$entry['TxCount'], $entry['TxDuration'], $entry['RxCount'],
$entry['RxDuration']);
}

/* e.g. returns:
#23456: Tx 120 (845 ms), Rx 5310 (2210 ms)
#34567: Tx 0 (0 ms), Rx 2655 (512 ms)
*/
```

IPS_GetInstanceMessageQueueSize

```
int IPS_GetInstanceMessageQueueSize ()
```

RETURNS

Number of not yet completely processed messages to instances

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the number of messages that are to be delivered to instances and have not yet been completely processed by their MessageSink. A message to several instances counts once per instance. In normal operation the value is 0 or very small. A permanently rising value indicates that instances process messages more slowly than they arrive. Which instances are affected can be determined with IPS_GetInstanceMessageStatistics.
The function is available independently of the special switch "MessageQueueWatch".

EXAMPLE

```
echo IPS_GetInstanceMessageQueueSize(); // e.g. returns 0
```

IPS_GetInstanceMessageStatistics

```
array IPS_GetInstanceMessageStatistics ()
```

RETURNS

An array with one entry per instance. The order of the entries is not guaranteed. Each entry contains the following key => value pairs:

INDEX	TYPE	DESCRIPTION
InstanceID	integer	ID of the receiving instance
Count	integer	Number of processed messages
Duration	integer	Total processing duration in milliseconds
MaxDuration	integer	Longest processing of a message in milliseconds

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the statistics of the message processing per instance. It records how many messages an instance has processed in its MessageSink and how long this took. The values are summed up since the start of IP-Symcon, durations are rounded down to whole milliseconds. The current number of not yet processed messages is returned by IPS_GetInstanceMessageQueueSize.



The statistics are only recorded if the special switch "MessageQueueWatch" is enabled (e.g. with IPS_SetOption). Otherwise the function fails with "This function is only available if the special option 'MessageQueueWatch' is enabled!".

If "MessageQueueWatch" is enabled, a message is additionally written to the log if the processing of a message takes longer than "MessageQueueWatchLimit" milliseconds (0 = disabled).

EXAMPLE

```
// Output the average processing duration of the messages per instance
foreach (IPS_GetInstanceMessageStatistics() as $entry) {
printf("#%d: %d messages, %.1f ms on average, max. %d ms\n",
$entry['InstanceID'], $entry['Count'], $entry['Duration'] /
$entry['Count'], $entry['MaxDuration']);
}

/* e.g. returns:
#23456: 1520 messages, 0.4 ms on average, max. 37 ms
*/
```

IPS_GetInstanceStats

```
array IPS_GetInstanceStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of loaded instances
TotalBytes	integer	Estimated memory usage of the instances in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the memory usage of the instances. The instances whose module is loaded are counted. For each instance the configuration, not yet applied changes of the configuration and the attributes are taken into account, for PHP modules additionally the buffers (SetBuffer).
The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement. The value is also recorded in the series "Instances" of IPS_GetStatsHistory.

EXAMPLE

```
// Output the memory usage of the instances
print_r(IPS_GetInstanceStats());

/* e.g. returns:
Array
(
[Count] => 187
[TotalBytes] => 1572864
)
*/
```

IPS_GetMediaStats

```
array IPS_GetMediaStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of media objects whose content is cached in memory
TotalBytes	integer	Estimated memory usage of all media objects including the cached contents in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the memory usage of the media objects. **Count** only counts the media objects whose content (Base64 encoded) is cached in memory, e.g. media objects with an active cache. **TotalBytes** additionally contains the management data of all media objects such as file name and checksum.
The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement. The value is also recorded in the series "Medias" of IPS_GetStatsHistory.

EXAMPLE

```
// Output the memory usage of the media objects
print_r(IPS_GetMediaStats());

/* e.g. returns:
Array
(
[Count] => 3
[TotalBytes] => 262144
)
*/
```

IPS_GetSettingsStats

```
array IPS_GetSettingsStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of objects in the settings
TotalBytes	integer	Estimated memory usage of the settings in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the size of the settings (settings.json) IP-Symcon holds in memory.
Count is the number of objects of the object tree in the settings.
The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement. The value is also recorded in the series "Settings" of IPS_GetStatsHistory. As the determination walks the whole settings tree, the series is only sampled once per hour.

EXAMPLE

```
// Output the size of the settings
print_r(IPS_GetSettingsStats());

/* e.g. returns:
Array
(
[Count] => 1843
[TotalBytes] => 9437184
)
*/
```

IPS_GetSnapshotBufferStats

```
array IPS_GetSnapshotBufferStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of buffered snapshots
TotalBytes	integer	Estimated memory usage of the snapshots in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the size of the snapshot buffer. For querying the differences between two snapshots (e.g. by the console), IP-Symcon buffers every returned snapshot, so that only the differences have to be transferred on the next request. A snapshot is discarded after its use or after 24 hours, at most 10 snapshots are kept.
The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement. The value is also recorded in the series "SnapshotBuffer" of IPS_GetStatsHistory. The series is only sampled once per hour.

EXAMPLE

```
// Output the size of the snapshot buffer
print_r(IPS_GetSnapshotBufferStats());

/* e.g. returns:
Array
(
[Count] => 2
[TotalBytes] => 14680064
)
*/
```

IPS_GetSnapshotRingBufferStats

```
array IPS_GetSnapshotRingBufferStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of messages in the ring buffer
TotalBytes	integer	Estimated memory usage of the ring buffer including the messages in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the size of the message ring buffer. In it, IP-Symcon keeps the most recently sent messages, so that clients like the console only need to fetch the changes that occurred since a snapshot. The size of the ring buffer is set with the special switch "MessageRingBufferSize" (default: 8192 messages). If the ring buffer is full, the oldest messages are overwritten, so **Count** reaches at most this value.
The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement. The value is also recorded in the series "SnapshotRingBuffer" of IPS_GetStatsHistory.

EXAMPLE

```
// Output the size of the message ring buffer
print_r(IPS_GetSnapshotRingBufferStats());

/* e.g. returns:
Array
(
[Count] => 8192
[TotalBytes] => 3145728
)
*/
```

IPS_GetStatsHistory

```
string IPS_GetStatsHistory ()
```

RETURNS

JSON encoded array with one entry per hour (directly as array via the JSON-RPC interface). Each entry contains the following key => value pairs:

INDEX	TYPE	DESCRIPTION
TimeStamp	integer	Unix timestamp of the beginning of the hour
<Series>	array	For each recorded series an array with the keys **Count** and **Bytes**, each containing **Min**, **Max** and **Avg** (minimum, maximum and average of the hour)

Series

SERIES	COUNT	BYTES
WorkingSet	Number of handles of the process	Occupied memory (working set) of the process
Shared	Number of threads of the process	Shareable memory (executable, libraries, OPcache)
Allocator	mimalloc: abandoned pages, otherwise 0	Memory occupied by the allocator (mimalloc/jemalloc only, otherwise 0)
Traces	see IPS_GetTraceStats	see IPS_GetTraceStats
SnapshotRingBuffer	see IPS_GetSnapshotRingBufferStats	see IPS_GetSnapshotRingBufferStats
SnapshotBuffer	see IPS_GetSnapshotBufferStats (hourly)	see IPS_GetSnapshotBufferStats (hourly)
Settings	see IPS_GetSettingsStats (hourly)	see IPS_GetSettingsStats (hourly)
ScriptEngine	see IPS_GetThreadStats	see IPS_GetThreadStats
Medias	see IPS_GetMediaStats	see IPS_GetMediaStats
Variables	see IPS_GetVariableStats	see IPS_GetVariableStats
Instances	see IPS_GetInstanceStats	see IPS_GetInstanceStats
VideoServer	Reserved, currently 0	Reserved, currently 0

DESCRIPTION

This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the hourly aggregated history of the memory statistics of the last 7 days, e.g. for the display in the console. As long as IP-Symcon is fully started, the statistics are sampled once per minute and aggregated to minimum, maximum and average per hour. Expensive series ("SnapshotBuffer" and "Settings") are only sampled once per hour, there all three values are equal. Series without a sample in an hour are missing in the respective entry. The history is only kept in memory and is empty after a restart of IP-Symcon. At most 168 entries (7 days) are kept, the current entry is the last one in the array.

In PHP the function returns the history as JSON encoded string which can be evaluated with json_decode. Via the JSON-RPC interface the history is returned directly as array.

All values in **Bytes** are specified in bytes. The number of series and their meaning may change with new versions.

EXAMPLE

```
// Output the memory of the last hours
$history = json_decode(IPS_GetStatsHistory(), true);
foreach (array_slice($history, -3) as $hour) {
printf("%s: %.1f MB (max. %.1f MB)\n", date('Y-m-d H:i',
$hour['TimeStamp']), $hour['WorkingSet']['Bytes']['Avg'] / 1048576,
$hour['WorkingSet']['Bytes']['Max'] / 1048576);
}

/* e.g. returns:
2026-09-26 10:00: 182.4 MB (max. 190.1 MB)
2026-09-26 11:00: 184.0 MB (max. 188.7 MB)
2026-09-26 12:00: 183.2 MB (max. 185.9 MB)
*/
```

IPS_GetThreadStats

```
array IPS_GetThreadStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of script threads currently holding a PHP environment
TotalBytes	integer	Memory reserved by the PHP memory manager across all threads in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the memory usage of the script threads of the PHP script environment. So that scripts can start quickly, threads keep their PHP environment after the execution. **Count** is the number of these threads, **TotalBytes** the sum of the memory the PHP memory manager (Zend Memory Manager) keeps reserved in these threads. Information about the individual threads is returned by IPS_GetScriptThreads.
TotalBytes is the value reported by the PHP memory manager at the end of the last execution in the respective thread. The value is also recorded in the series "ScriptEngine" of IPS_GetStatsHistory.

EXAMPLE

```
// Output the memory usage of the script threads
print_r(IPS_GetThreadStats());

/* e.g. returns:
Array
(
[Count] => 4
[TotalBytes] => 8388608
)
*/
```

IPS_GetTraceStats

```
array IPS_GetTraceStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of stored trace entries
TotalBytes	integer	Estimated memory usage of the trace entries in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the size of the memory in which IP-Symcon keeps the chains of causes (traces) of actions, events, scripts and data flows. The individual entries of an object are returned by IPS_GetTraces.
The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement. The value is also recorded in the series "Traces" of IPS_GetStatsHistory.

EXAMPLE

```
// Output the size of the trace memory
print_r(IPS_GetTraceStats());

/* e.g. returns:
Array
(
[Count] => 12873
[TotalBytes] => 6291456
)
*/
```

IPS_GetVariableStats

```
array IPS_GetVariableStats ()
```

RETURNS

An array with the following key => value pairs:

INDEX	TYPE	DESCRIPTION
Count	integer	Number of variables
TotalBytes	integer	Estimated memory usage of the variables in bytes

DESCRIPTION



This function is intended for internal use only and may change or be removed at any time without notice.

The function returns the memory usage of the variables. For each variable the value, the (custom) variable profile and the (custom) presentation are taken into account.
The bytes are an estimate of the occupied memory based on the stored data structures and not an exact measurement. The value is also recorded in the series "Variables" of IPS_GetStatsHistory.

EXAMPLE

```
// Output the memory usage of the variables
print_r(IPS_GetVariableStats());

/* e.g. returns:
Array
(
[Count] => 1204
[TotalBytes] => 983040
)
*/
```

Compatibilityfunctions

Compatibilityfunctions are the deprecated features that should no longer be used and are disabled by default. These were abolished due to performance or replaced by improved replacement functions. For reasons of compatibility with old IP-Symcon installations, these can be activated via the special switch "compatibility functions" if required. However, this requires a lot of resources and is strongly discouraged.

IPS_GetInstanceChildrenIDs

```
array IPS_GetInstanceChildrenIDs (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

An array of InstanceIDs from the children instances. The order of the entries is not guaranteed.

DESCRIPTION



This function should not be used since IP-Symcon 2.6 and was removed with IP-Symcon 4.0.

The command returns all IDs of the children instances of the specific instance **InstanceID**. (Seen by the hardware (physical) view)



This command is only useful for I / O and splitter instances.

EXAMPLE

```
print_r(IPS_GetInstanceChildrenIDs(12345));


// Example to replace the deprecated command
// "IPS_GetInstanceChildrenIDs".
// IPS_GetInstance is used

$InstanceIDs = IPS_GetInstanceList();
foreach($InstanceIDs as $IID)
if(IPS_GetInstance($IID)['ConnectionID'] == 12345)
echo $IID . PHP_EOL;
```

IPS_GetInstanceParentID

```
int IPS_GetInstanceParentID (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

ID of the parent instance

DESCRIPTION



This function should not be used since IP-Symcon 2.6 and was removed with IP-Symcon 4.0.

The command returns the ID of the parent instance for the instance **InstanceID**. (Seen by the hardware (physical) view)



This command is only useful for devices and splitter instances.

F

EXAMPLE

```
// Up to Version 2.5
echo IPS_GetInstanceParentID(12345);

// Since Version 2.6
echo IPS_GetInstance(12345)['ConnectionID'];
```

IPS_GetStatusVariable

```
array IPS_GetStatusVariable (int $InstanceID, string $VariableIdent)
```

PARAMETERS

InstanceID	ID of the instance
VariableIdent	Status variable identifier. A listing can be found on IPS_GetStatusVariableIdents

RETURNS

The following information are available as key => value pairs:

INDEX	TYPE	DESCRIPTION
VariableID	integer	Current ID of the associated IP Symcon variable
VariableIdent	string	Status variable identifier
VariableName	string	Default name of the created variable
VariablePosition	integer	Default position of the created variable
VariableProfile	string	Default profile name of the created variable
VariableType	integer	Compatible variable type (See: IPS_GetVariable)
VariableHasAction	boolean	Indicates whether an action is internally defined, which was linked to this status variable
VariableUseAction	boolean	Specifies the existing action, in which the visualization should be used

DESCRIPTION



The documentation of this function only for compatibility reasons and should not be used in IPSymcon 2.5 and newer versions. Since version 2.5, the functionality is only emulated and should be replaced by other functions. Since version 4.0, these functions can only be accessed if the Special Switch "Compatibility Functions" is activated.

The command returns an array containing information about the status variable of the instance **InstanceID** and the status variable with the status variable identifier **VariableIdent**.

EXAMPLE

```
if(IPS_StatusVariableExists(12345, "StatusVariable"))
{
print_r(IPS_GetStatusVariable(12345, "StatusVariable"));
}
```

IPS_GetStatusVariableID

```
int IPS_GetStatusVariableID (int $InstanceID, string $VariableIdent)
```

PARAMETERS

InstanceID	Instance ID
VariableIdent	Status variable identifier. A listing can be retrieved via IPS_GetStatusVariableIdents.

RETURNS

Variable ID of the variable associated with the status variable

DESCRIPTION



The documentation for this function is only available for reasons of compatibility and, since IP-Symcon 2.5, should no longer be used. Since version 2.5, this function is only emulated and should be replaced by the corresponding substitute functions. A replacement function is in the example (see below).

The function returns the VariableID of the variable associated with the instance's status variable **InstanceID** and the status variable identifier **VariableIdent**.

EXAMPLE

```
//Display the status of, for example, an FS20 device without knowing
the ID of the "Status" variable
echo GetValue(IPS_GetStatusVariableID(12345, "StatusVariable"));

// since version 2.6 replacement function
$InstanceID = "StatusVariable";
$VariableIdent = 12345;

$migrateIdents = Array(
"F05_StatusVar" => "Status",
"F10_TemperatureVar" => "Temperature",
"F12_Var0" => "Status0",
"F12_Var1" => "Status1",
"F1D_CounterVar1" => "Counter1",
```

```
"F1D_CounterVar2" => "Counter2",
"F20_Var0" => "Port0",
"F20_Var1" => "Port1",
"F20_Var2" => "Port2",
"F20_Var3" => "Port3",
"F26_TemperatureVar" => "Temperature",
"F26_VDDVar" => "VDD",
"F26_VADVar" => "VAD",
"F26_XSENSVar" => "XSENS",
"F28_TemperatureVar" => "Temperature",
"F29_Var0" => "Status0",
"F29_LatchVar0" => "Latch0",
"F29_Var1" => "Status1",
"F29_LatchVar1" => "Latch1",
"F29_Var2" => "Status2",
"F29_LatchVar2" => "Latch2",
"F29_Var3" => "Status3",
"F29_LatchVar3" => "Latch3",
"F29_Var4" => "Status4",
"F29_LatchVar4" => "Latch4",
"F29_Var5" => "Status5",
"F29_LatchVar5" => "Latch5",
"F29_Var6" => "Status6",
"F29_LatchVar6" => "Latch6",
"F29_Var7" => "Status7",
"F29_LatchVar7" => "Latch7",
"F2C_PositionVar" => "Position",
"F3A_Var0" => "Status0",
"F3A_Var1" => "Status1"
);

if(isset($migrateIdents[$VariableIdent]))
{
$VariableIdent = $migrateIdents[$VariableIdent];
}

$result = IPS_GetObjectIDByIdent($VariableIdent, $InstanceID);
echo $result;
```

IPS_GetStatusVariableIdents

```
array IPS_GetStatusVariableIdents (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

An array of strings that indicate the status variable identifiers of the instance. The order of the entries is not guaranteed.

DESCRIPTION



The documentation of this function only for compatibility reasons and should not be used in IPSymcon 2.5 and newer versions. Since version 2.5, the functionality is only emulated and should be replaced by other functions. Since version 4.0, these functions can only be accessed if the Special Switch "Compatibility Functions" is activated.

The command returns all known state variables of the instance **InstanceID** and returns the status variable identifiers in an array of values.

EXAMPLE

```
print_r(IPS_GetStatusVariableIdents(12345));
```

IPS_GetUptime

```
int IPS_GetUptime ()
```

RETURNS

Point of time of the most recent start of IP Symcon

DESCRIPTION

The function returns the date and time of the start time of IP Symcon as Unix Timestamp.



This function is implemented until version 3.4. Since version 4.0 it is replaced by IPS_GetKernelStartTime.

EXAMPLE

```
// returns, e.g., 1204675012
echo IPS_GetUptime();
// returns, e.g., "2008.03.04 23:56:52"
echo gmdate("Y.m.d H:i:s.", IPS_GetUptime());
```

IPS_HasInstanceChildren

```
bool IPS_HasInstanceChildren (bool $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

The return value is **TRUE** if the instance has a child instance, otherwise, **FALSE**.

DESCRIPTION



This function should not be used since IP-Symcon 2.6 and was removed with IP-Symcon 4.0.

The command indicates whether a particular instance of **InstanceID**, seen by the hardware (physical) view, has a child instance e.g. a splitter or device module.



This command is only useful for I / O and splitter instances.

EXAMPLE

```
if(IPS_HasInstanceParent(12345))
{
echo "Has a child instance";
}
```

IPS_HasInstanceParent

```
bool IPS_HasInstanceParent (int $InstanceID)
```

PARAMETERS

InstanceID	ID of the instance

RETURNS

The return value is TRUE if the instance has a parent instance, otherwise, FALSE.

DESCRIPTION



This function should not be used since IP-Symcon 2.6 and was removed with IP-Symcon 4.0.

The command indicates whether a particular instance InstanceID, seen by the hardware (physical) view, has a parent instance e.g. a splitter or I/ O module.



This command is only useful for devices and splitter instances.

EXAMPLE

```
// Up to Version 2.5
if(IPS_HasInstanceParent(12345))
{
echo "Has a parent instance";
}

// Since Version 2.6
if(IPS_GetInstance(12345)['ConnectionID'] > 0)
{
echo "Has a parent instance";
}
```

IPS_IsPersistent

```
bool IPS_IsPersistent (int $ObjectID)
```

PARAMETERS

ObjectID	The ID of the object

RETURNS

TRUE, if the object is saved, FALSE otherwise

DESCRIPTION



This function should not be used any more since version 2.6. As all objects within IP Symcon are persistent, this function is not needed any more.

This function checks if the object with the ID **ObjectID** is saved in the Settings. This is currently done for all IDs that are bigger than 10000. Otherwise, the object is discarded when IP Symcon shuts down.

EXAMPLE

```
if (IPS_IsPersistent(34881))
echo "The object is saved on shut down!";
```

IPS_SetLinkChildID

Require: Symcon >= 2.1

```
bool IPS_SetLinkChildID (int $LinkID, int $LinkedObject)
```

PARAMETERS

LinkID	ID of the link
LinkedObject	ID of the target object for the link

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



The documentation of this function only for compatibility reasons and should not be used in IPSymcon 2.6 and newer versions. Since version 4.0, this command can only be accessed if the Special Switch "Compatibility Functions" is activated.

The command binds a specific object with the ID **LinkedObject** to the link with the ID **LinkID** and thus the link refers to the object.

EXAMPLE

```
IPS_SetLinkChildID($LinkID, 12345); //Refer to object 12345
```

IPS_SetStatusVariableUseAction

```
bool IPS_SetStatusVariableUseAction (int $InstanceID, string
$VariableIdent, bool $UseAction)
```

PARAMETERS

InstanceID	ID of the instance
VariableIdent	Status variable identifier. A listing can be found on IPS_GetStatusVariableIdents
UseAction	Indicates whether the action should be used in the visualization

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



The documentation of this function only for compatibility reasons and should not be used in IPSymcon 2.5 and newer versions. Since version 2.5, the functionality is only emulated and should be replaced by other functions. Since version 4.0, these functions can only be accessed if the Special Switch "Compatibility Functions" is activated.

The command sets whether the internally linked action of a specific instance **InstanceID** and the status variable with the status variable identifier **VariableIdent** should be used.

EXAMPLE

```
if(IPS_StatusVariableExists(12345, "StatusVariable"))
{
IPS_SetStatusVariableUseAction(12345, "StatusVariable", false); // Do
not make it switchable via WebFront
}

// Replacement function since version 2.6
$VariableIdent = "Statusvariable";
$InstanceID = 12345;
```

```
$UseAction = false;

$id = IPS_GetObjectIDByIdent($VariableIdent, $InstanceID);

if ($UseAction) {
IPS_SetVariableCustomAction($id, VARIABLE_ACTION_NONE); // Enable
default action
} else {
IPS_SetVariableCustomAction($id, VARIABLE_ACTION_DISABLED); // Disable
default action
}
```

IPS_StatusVariableExists

```
bool IPS_StatusVariableExists (int $InstanceID, string $VariableIdent)
```

PARAMETERS

InstanceID	ID of the Instance
VariableIdent	Status variable identifier. A listing can be found on IPS_GetStatusVariableIdents

RETURNS

The return value is **TRUE** if the status variable exists, otherwise **FALSE**

DESCRIPTION



The documentation of this function only for compatibility reasons and should not be used in IPSymcon 2.5 and newer versions. Since version 2.5, the functionality is only emulated and should be replaced by other functions. Since version 4.0, these functions can only be accessed if the Special Switch "Compatibility Functions" is activated.

The command checks whether a specific instance **InstanceID** has the state variable with the status variable identifier **VariableIdent**.

EXAMPLE

```
if(IPS_StatusVariableExists(12345, "StatusVariable"))
{
echo IPS_GetStatusVariableID(12345, "StatusVariable");
}
```

Limitations

The restrictions described below are necessary for reasons of performance and security of the system. It is not planned to change these.

LIMITATION	DESCRIPTION
Maximum number of objects	50000. The address space of 10000-59999 is used. Depending on the license/IP-Symcon version, this value may be lower.
Maximum size of a string variable	Over 1024kb an error is generated and the content is not written. For reasons of performance and stability, data over 8kb should be transferred elsewhere. The 1024kb limit simply serves as a hard limit in order to contain catastrophic effects during runtime due to faulty scripts.
Maximum size of the RegisterVariable Buffer	For size 8kb and above a warning is written in the log. Only the last 64kb of data is kept in the buffer. Older data will be discarded without further warning.
Maximum size of the Cutter Buffer	For size 8kb and above a warning is written in the log. Only the last 64kb of data is kept in the buffer. Older data will be discarded without further warning.
Maximum number of PHP-Threads	The default setting of 50 PHP threads is completely sufficient for almost all systems. The minimum is 25 PHP threads. If a value lower than 25 is selected, 25 PHP threads will still be created. If more PHP threads are required, the value can be increased to 100, for example. Much higher values make little sense in most cases and consume an unnecessary amount of RAM.
Maximum memory per PHP thread	The default setting is 32MB per PHP thread. This value can be changed to a maximum of 64M per thread in the php.ini file. Consult the PHP documentation for this.
Maximum memory per buffer	The buffer has a soft limit of 256kB and a hard limit of 1024kB. This means that IP-Symcon throws a "warning" in the messages if the buffer is larger than 256kB and the maximum size of the buffer is 1024kB. Data larger than 1024kB will be truncated.
Query of data via AC_*functions	A maximum of 10000 lines are returned, regardless of whether the limit was selected higher. No warning is given. Queries should be optimized accordingly so that the aggregated data can be accessed. This minimizes the load on the database and thus also minimizes access times for the WebFront visualization of the individual accesses.
Maximum associations per variable profile	A maximum of 128 (since 5.0; previously 32) associations are allowed per variable profile. The reasoning to this is that every change to the profile associations must be transferred to all WebFronts, which results in a high overhead if there are a large number of associations. In addition, this feature is not designed for e.g. a playlist selection, as this would negatively influence the design and layout of the WebFront.
Maximum number of simultaneous RTSP streams	The hardware of the Symcon server and the resolution of the streams determine how many simultaneous streams are possible.

Mirroring

Require: Symcon >= 7.0



Mirroring is a paid extension that can be purchased for any existing IP-Symcon license. For a suitable demo version, please contact our Support. The extension can be purchased directly in the Shop.

A system can be configured as a mirror of another server system. In this case, the mirror is regularly updated by the server and adapts its configuration to that of the server. If the server fails, the mirror can be used and take over the tasks of the server, thereby optimizing the availability of the overall system.



Mirroring contains various parameters which should be configured to suit the use case. In order to optimize the commissioning of this extension, we recommend contacting our Support so that the optimal configuration for the use case can be worked out and implemented together.

To use mirroring, remote access must be enabled on the server. The initial settings for mirroring can be configured in the core instance “Replication”. These are then created in the file "replication.json" in the same directory as "settings.json" – see Installation of the corresponding operating system. Changes after the initial activation of replication can only be made in this file, as the rest of the system is a 1:1 mirror of the main system.

PARAMETER OF REPLICATION.JSON

PARAMETERS	TYPE	DEFAULT	DESCRIPTION
Active	Boolean	false	Mirroring is active if the value is set to true, otherwise it inactive and the system acts normally.

PARAMETERS	TYPE	DEFAULT	DESCRIPTION
Mode	Integer	0	Operating mode of the mirrored system 0 REPLICATION_MODE_PARALLEL: Parallel: The system is fully operational and all its connections are active. This means that the system can be used directly if the server fails. However, either a gateway must support the connection of multiple devices or a separate gateway must be available. If the mirrored system is used although the server is still running, these changes will be overwritten during the next mirroring and will therefore expire 1 REPLICATION_MODE_STANDBY: Standby: The system is in standby and all its connections are inactive. If the server fails, the mirrored system is promoted to master by Failover - automatically (see FailoverActive) or manually with IPS_PromoteToReplicationMaster. The required connections are then established and the mirrored system can be used.
Host	String	""	The URL under which the server system can be reached
Port	Integer	3777	The port under which the server system provides IP-Symcon
UseSSL	Boolean	false	If true, the connection to the server is encrypted via SSL
VerifyPeer	Boolean	true	If true, it is verified that the SSL certificate of the host is correct (only if UseSSL is also true)
VerifyHost	Boolean	true	If true, it is verified that the host URL from the SSL certificate matches the host from the property (only if UseSSL is also true)
Username	String	""	The user name, i.e. the license address, of the server system
Password	String	""	The password for remote access to the server system
Timeout	Integer	10000	The timeout for requests to the server in milliseconds. If request takes longer, the connection is marked as faulty and no mirroring is performed
RetryCount	Integer	10	Number of attempts to load the configuration from the server on startup. If all attempts fail, the mirrored syster starts as master in the mode Standby (failover)
RetryBackoff	Integer	20	Waiting time in seconds after a failed attempt to load th configuration from the server on startup

PARAMETERS	TYPE	DEFAULT	DESCRIPTION
FailoverActive	Boolean	true	If true, the mirrored system in the mode Standby is automatically promoted to master as soon as the mirroring during operation has failed as many times in a row as specified in FailoverCount. If false, only a manual promotion is possible
FailoverCount	Integer	20	Number of consecutive failed mirrorings during operatic after which the automatic failover is triggered (only if FailoverActive is true)
Synchronization	Object	not set	Detailed settings for not mirroring aspects of the server while retaining your own settings

PARAMETER FOR SYNCHRONIZATION

PARAMETER	TYPE	DESCRIPTION
ExcludeObjects	Array	Objects whose IDs are in this list are not updated during a mirroring and retain the configuration of the mirrored system
ExcludeProfiles	Array	Variable profiles whose names are in this list are not updated during a mirroring and retain the configuration of the mirrored system

SDK/Tools

The SDK is used to develop own modules and to be able to integrate them via the "Module Control". The developed modules can be made available to other users.

SDK

Version 3.0 and above
SDK for Excel
SDK for Skins
Version 4.0 and above
SDK for PHP

TOOLS

Guid Generator
Module Generator
Module Validator
Network-Configuration-Tool
Recoverytool
Visual Studio Code
AI Agents (llms.txt)

AI Agents

The complete documentation is additionally available as Markdown optimized for AI agents and Large Language Models (LLMs), e.g. Claude, ChatGPT, Cursor or GitHub Copilot. The files are automatically regenerated on every update of the website and are therefore always up to date with this documentation.
In contrast to the HTML page or the PDF, the files contain only the content without any layout. Therefore, they require significantly fewer tokens and can be processed faster by AI agents.

STRUCTURE

The files follow the llms.txt standard:

FILE	DESCRIPTION
llms.txt	Index of all files with a short description and approximate size in tokens. The entry point for AI agents.
llms/function-index.md	All functions of the command reference and the SDK with signature and short description, each with a link to the file containing parameters and examples.
llms/function-index-modules.md	All module specific functions (e.g. ENO*, KNX*) with signature and short description, each with a link to the module file.
llms/modules/*.md	One file per module, e.g. enocean.md or knx.md, including the module functions.
llms/functions/*.md	The command reference, one file per group, e.g. management-variables.md.
llms-full.txt	The complete documentation in a single file, e.g. for uploading into a knowledge base.

No file is larger than about 80,000 characters, so AI agents can fetch every file completely. Larger sections are split into multiple files, e.g. one file per form element of the configuration forms. Every file containing multiple pages starts with a table of contents, which also links to the subpages in separate files.
The file names are identical in all languages. The German version is available at /de/llms.txt.

USAGE

Most AI agents can fetch the files directly. It is sufficient to tell the agent the address of the index, e.g. in the CLAUDE.md or AGENTS.md file of your module project:

```
The Symcon documentation for AI agents is available at
https://www.symcon.de/en/llms.txt.
For PHP functions, load https://www.symcon.de/en/llms/function-index.md
(command reference and SDK) or https://www.symcon.de/en/llms/function-
index-modules.md (module functions) first.
```

Alternatively, the file llms-full.txt can be uploaded as a knowledge base into a project (e.g. Claude Projects or NotebookLM).

FILES

llms.txt – Index
llms-full.txt – Complete documentation in a single file
Start here
- Function Index `function-index.md`
- Function Index Modules `function-index-modules.md`
- Getting Started `getting-started.md`
- Procedures `how-to.md`
- Basics `concepts.md`
- Automations `concepts/automations.md`
- Logic Scripts `concepts/automations/logic-scripts.md`
- Repository `concepts/automations/logic-scripts/repository.md`

Components
- Components – Overview `components/index.md`
- Remote Access `components/remote-access.md`
- Tray `components/tray.md`
- Service `components/service.md`
- Management Console `components/management-console.md`
- Tile Visualization `components/tile-visualization.md`
- WebFront Visualization `components/webfront-visualization.md`
- Object Presentation `components/object-presentation.md`
- Icons `components/icons.md`

Command Reference
- Command Reference – Overview `functions/index.md`
- Process Control `functions/process-control.md`
- Management of Permissions `functions/management-permissions.md`
- Management of Events `functions/management-events.md`
- Management of Instances `functions/management-instances.md`
- Configuration `functions/management-instances/configuration.md`
- Management of Categories `functions/management-categories.md`
- Management of Links `functions/management-links.md`
- Management of Media `functions/management-media.md`
- Management of Modules `functions/management-modules.md`
- Management of Objects `functions/management-objects.md`
- Program Information `functions/program-information.md`
- Management of Scripts `functions/management-scripts.md`
- Management of Traces `functions/management-traces.md`
- Management of Variables `functions/management-variables.md`
- Accessing Variables `functions/access-variables.md`

Developer Area
- Developer Area – Overview `developer/index.md`
- Data Exchange `developer/data-exchange.md`
- Download (Archive) `developer/download-archive.md`

Internal Command Reference developer/internal-command-reference.md
Compatibilityfunctions developer/compatibility-functions.md
Limitations developer/limitations.md
Mirroring developer/mirroring.md
SDK/Tools developer/sdk-tools.md
SDK (PHP) developer/sdk-tools/sdk-php.md
Configuration Forms developer/sdk-tools/sdk-php/configuration-forms.md
Module developer/sdk-tools/sdk-php/module.md
Special Switches developer/special-switches.md
Module Reference
Module Reference – Overview modules/index.md
Module Reference: Devices
1-Wire modules/1-wire.md
ABL modules/abl.md
Alfen modules/alfen.md
ALLNET modules/allnet.md
BACnet modules/bacnet.md
Catan modules/catan.md
digitalSTROM modules/digitalstrom.md
DMX / ArtNet modules/dmx-artnet.md
EgiGeoZone modules/egigeozone.md
ekey modules/ekey.md
ekey bionyx modules/ekeybionyx.md
EnOcean modules/enocean.md
FHZ1X00PC modules/fhz1x00pc.md
FS10 Weather modules/fs10-weather.md
GARDENA smart system modules/gardena-smart-system.md
Geofency modules/geofency.md
Heating Control modules/heating-control.md
Home Connect modules/home-connect.md
HomeMatic modules/homematic.md
Image Grabber modules/image-grabber.md
IPS-868 modules/ips-868.md
IR Trans modules/irtrans.md
KEBA modules/keba.md
KNX modules/knx.md
LCN modules/lcn.md
LJQuick modules/ljquick.md
Matter modules/matter.md
M-Bus modules/mbus.md
Mennekes modules/mennekes.md
Modbus RTU/TCP modules/modbus-rtu-tcp.md
Möhlenhoff Alpha 2 modules/moehlenhoff-alpha-2.md
MQTT modules/mqtt.md
NEA Smart modules/nea-smart.md

- OCPP modules/ocpp.md
- OPC UA modules/opc-ua.md
- SageGlass (BACnet) modules/sageglass-bacnet.md
- Shutter Control modules/shutter-control.md
- Siemens OZW modules/siemens-ozw.md
- SNMP modules/snmp.md
- Snom modules/snom.md
- PLC: Siemens, Vipa, Logo modules/sps-siemens-vipa-logo.md
- PLC: Wago, Beckhoff, ABB modules/sps-wago-beckhoff-abb.md
- Sync Remote modules/sync-remote.md
- Technische Alternative modules/technische-alternative.md
- Door Intercom modules/door-intercom.md
- Voice over IP modules/voip.md
- Weishaupt modules/weishaupt.md
- WinLIRC modules/winlirc.md
- Wireless M-Bus modules/wireless-m-bus.md
- WMRS200 modules/wmrs200.md
- W&T modules/wut.md
- XBee modules/xbee.md
- Eaton xComfort modules/xcomfort.md
- Z-Wave modules/z-wave.md
- Zevvy modules/zevvy.md

Module Reference: Logic

- Active List modules/active-list.md
- Presence Simulation modules/presence-simulation.md
- Image Archive modules/image-archive.md
- Countdown modules/countdown.md
- CSV ZIP Export modules/csv-zip-export.md
- Dummy Module modules/dummy-module.md
- Egg Timer modules/egg-timer.md
- Group Control modules/group-control.md
- JSON Decoder modules/json-decoder.md
- JSON Exporter modules/json-exporter.md
- Logic Gate modules/logic-gate.md
- Computation Module modules/computation-module.md
- RGBMultiplexer modules/rgbmultiplexer.md
- DragPointer modules/dragpointer.md
- Game Collection modules/game-collection.md
- Scene Control modules/scene-control.md
- Dew Point Temperature Calculation modules/dew-point-temperature-calculation.md
- Staircase Light Controls modules/staircase-light-controls.md
- Renamer modules/renamer.md
- Rain Central modules/rain-central.md
- Variable Comparison modules/variable-comparison.md
- Virtuelle Devices modules/virtuelle-devices.md

Random Lighting modules/random-lighting.md
Module Reference: Energy
Work Efficiency modules/work-efficiency.md
Operating Hours Counter modules/operating-hours-counter.md
Energy Dashboard modules/energy-dashboard.md
Energy Manager modules/energy-manager.md
Energy Distribution modules/energy-distribution.md
Energy Counter modules/energy-counter.md
Power Billing Module modules/power-billing-module.md
Power price modules/power-price.md
Consumption per Category modules/consumption-per-category.md
Consumption within Timespan modules/consumption-within-timespan.md
Consumption Behaviour modules/consumption-behaviour.md
Calculated Counter modules/calculated-counter.md
Virtual Counter modules/virtual-counter.md
Reading (Day) modules/reading-day.md
Meter Overflow modules/meter-overflow.md
Module Reference: Visualizations
Tile Visualization modules/tile-visualization.md
WebFront Visualization modules/webfront-visualization.md
Module Reference: Voice assistents
Amazon Alexa modules/amazon-alexa.md
Google Assistant modules/google-assistant.md
Module Reference: Notifications
Alerting modules/alerting.md
Notification modules/notification.md
Announcement modules/announcement.md
Dynamic Mail modules/dynamic-mail.md
Done Notifier modules/done-notifier.md
IMAP modules/imap.md
MediaPlayer modules/mediaplayer.md
POP3 modules/pop3.md
Popup Module modules/popup-module.md
SMS modules/sms.md
SMTP modules/smtp.md
Spotify modules/spotify.md
Fault Manager modules/fault-manager.md
SymconReport modules/symconreport.md
Phone Announcement modules/phone-announcement.md
Phone Chain modules/phone-chain.md
TelegramBot modules/telegrambot.md
Text to Speech modules/text-to-speech.md
TTSAWSPolly modules/ttsawspolly.md
Consumption Alert modules/consumption-alert.md
Water Alert modules/water-alert.md
Watchdog modules/watchdog.md
Module Reference: Core Instances

Archive Control modules/archive-control.md
Permission Control modules/permission-control.md
Calendar Control modules/calendar-control.md
Connect Control modules/connect-control.md
Cutter modules/cutter.md
DNS-SD Control modules/dns-sd-control.md
Event Control modules/event-control.md
Location Control modules/location-control.md
Module Control modules/module-control.md
Notification Control modules/notification-control.md
Presence Control modules/presence-control.md
RegisterVariable modules/registervariable.md
Skin Control modules/skin-control.md
SSDP Control modules/ssdp-control.md
System Information modules/system-information.md
Tailscale VPN modules/tailscale-vpn.md
TextParser modules/textparser.md
Translation Control modules/translation-control.md
Util Control modules/util-control.md
WebHook Control modules/webhook-control.md
WebServer modules/webserver.md
Module Reference: I/O Instances
Client Socket modules/clientsocket.md
HID modules/hid.md
HTTP Client modules/httpclient.md
Multicast Socket modules/multicastsocket.md
Serial Port modules/serialport.md
Server Sent Event Client modules/serversenteventclient.md
Server Socket modules/serversocket.md
UDP Socket modules/udpsocket.md
Virtual I/O modules/virtualio.md
WebSocket Client modules/websocketclient.md
Module Reference: Backups
Backup modules/backup.md
Module Reference: Legacy
RRDTool modules/rrdtool.md
Shutter Control (legacy) modules/shutter-control-legacy.md
USBMapper modules/usbmapper.md
Web Graph modules/web-graph.md
Wunderground Weather modules/wunderground-weather.md

SDK (Excel)

Require: Symcon >= 3.0

DESCRIPTION

Since IP-Symcon 3.0, IP-Symcon supports the JSON-RPC interface and thus the database can easily be read into Microsoft Excel.

REQUIREMENTS

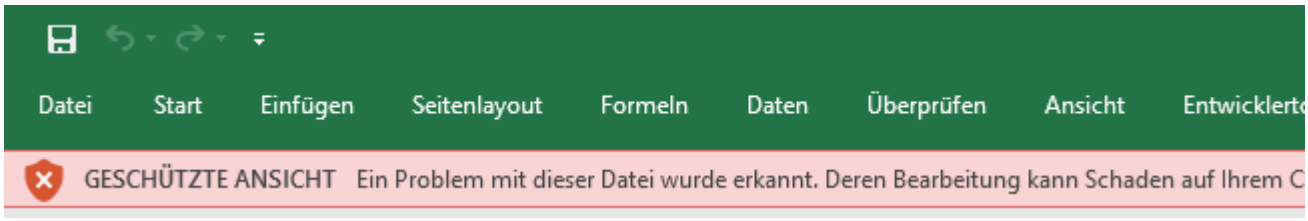
IP-Symcon 3.0 or above
Microsoft Excel or similar with Visual Basic for Applications support
"Visual Basic for Applications" experience (optional)
Download the prepared Excel file

SETUP IN MICROSOFT EXCEL

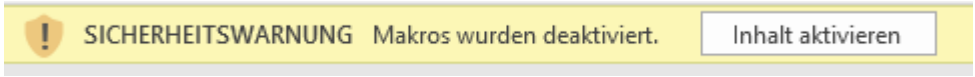
The downloaded file must be unzipped and opened in Excel.

OPENING THE FILE FOR THE FIRST TIME

When the file is opened for the first time, two warnings appear, which must be clicked through and allowed.
First warning:

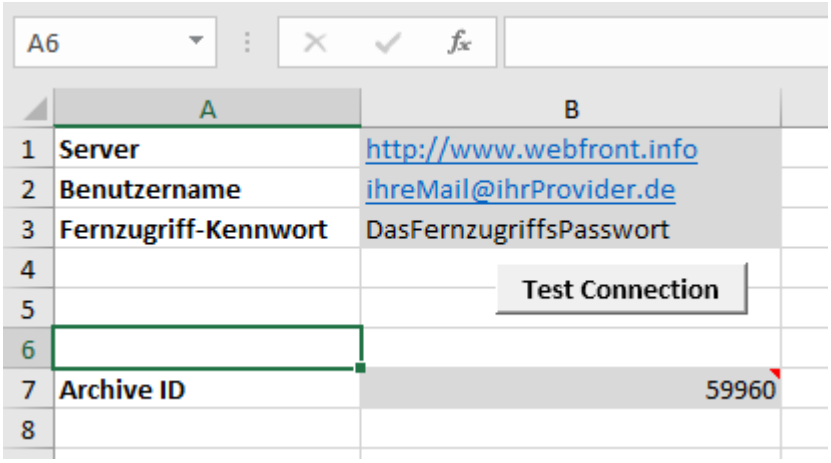


If you click on the text of the message, "Enable Editing" must be selected. This is a standard security measure because the file contains scripts.
Then either a popup or a yellow security warning appears, both of which warn of the macros contained. These must be activated to ensure functionality. Either click "Activate macros" in the popup or "Enable Content" in the yellow message.



CONFIGURE ACCESS

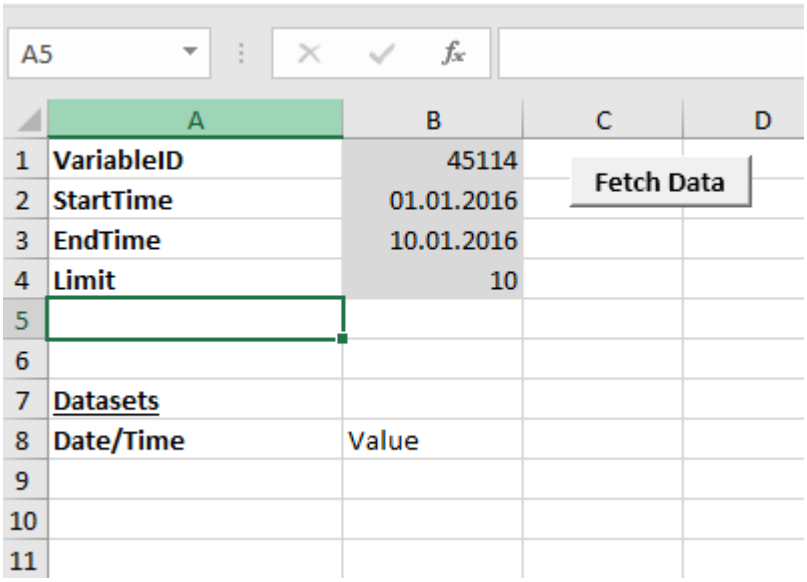
The connection to the IPS database can be tested in the "Configuration" tab.
Make sure that Remote Access is activated.



Server, user name and remote access password can be specified in the configuration table. Complete URLs or IP addresses can be entered for servers.
When using IP addresses, make sure that this is specified in the format "http://192.168.1.1:3777".
The remote access password and user name are only required if the data is not being accessed locally (127.0.0.1).
The connection is established via "Test Connection" and the InstanceID of the archive is read out.

READ RAW DATA

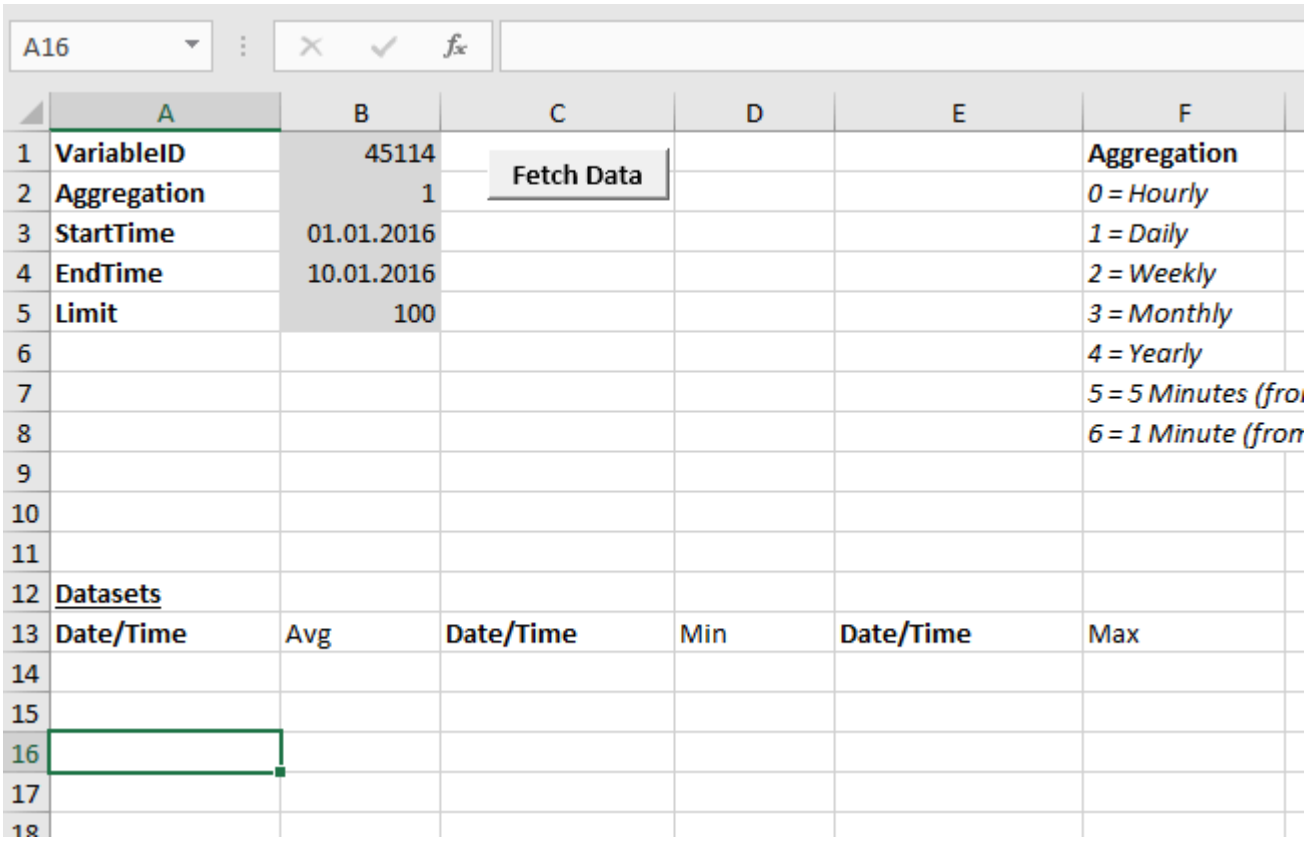
The raw data of a variable can be read out in the "Raw Data" tab. These are then put out line by line accompanied by date and value.



The VariableID must be entered. The start and end time can be 0 and therefore do not set any limits. Limit sets the maximum number of data records that should be read. Due to the system, however, there is a limit of 10,000 datasets.
The data is read in via "Fetch Data" and entered in the table. The most recent dataset appears first.

READ AGGREGATION

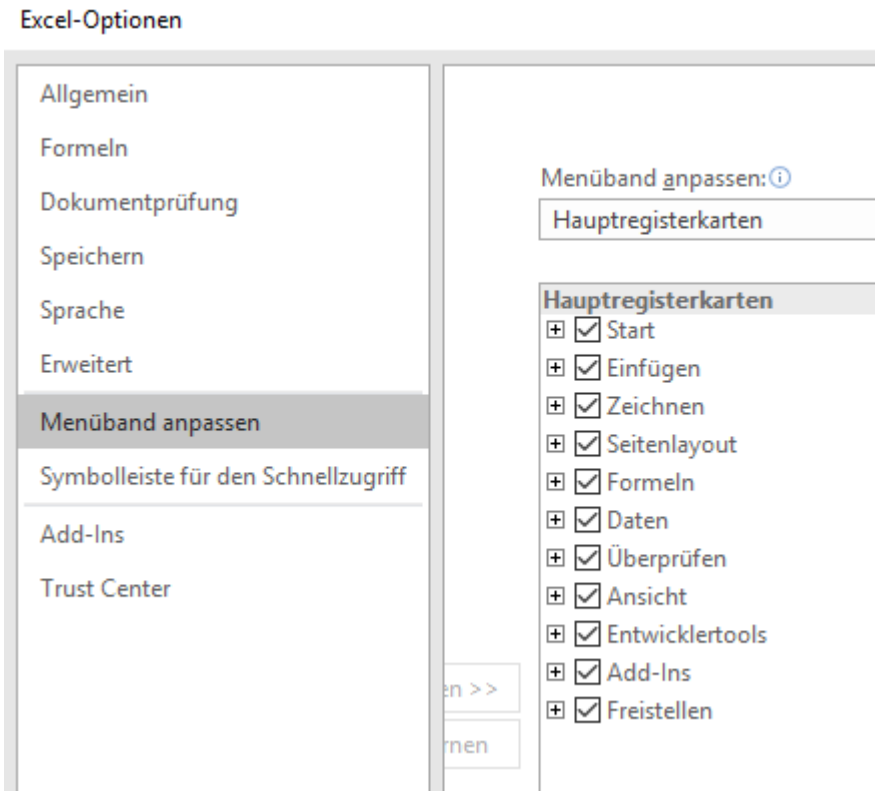
The third tab contains the "Aggregation". The VariableID, start/end time and the limit are required again.
In addition, the aggregation can be selected. With the values 0-6 you can choose from minutely up to annually accurate aggregation.



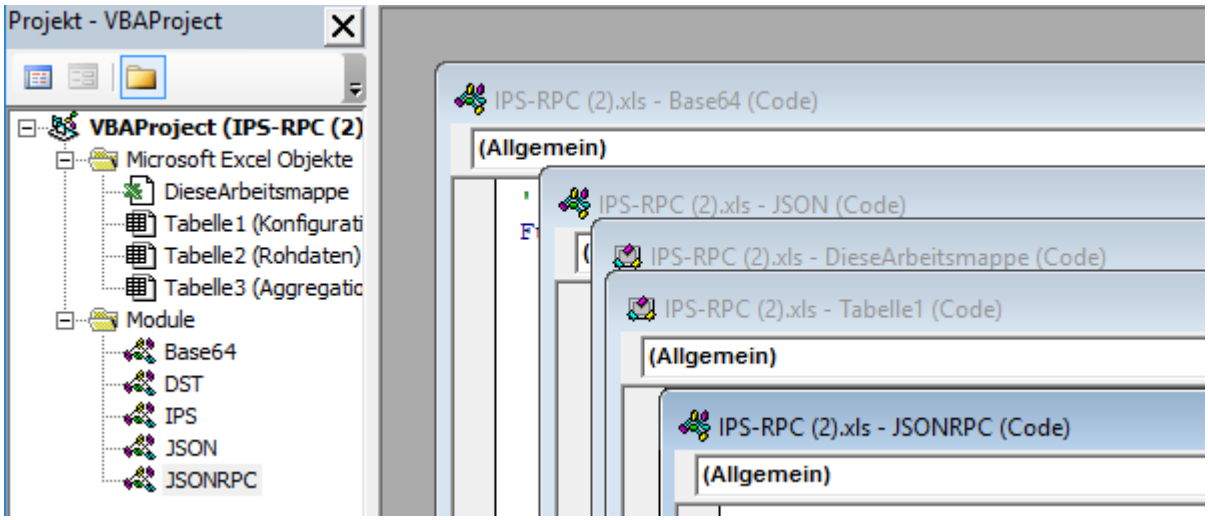
The data is read in via "Fetch Data" and entered in the table. An average as well as minima and maxima within an aggregation are put out.

VISUAL BASIC FOR APPLICATIONS AND THE CONFIGURATION

In order to manage or change the scripts and macros that enable reading, the developer tools need to be activated in the Quick Bar.
This can easily be selected via "Right-click->File->Customize ribbon". Here, simply the activation of the "Developer" menu is required.



Once activated, you will find the "Visual Basic" button under this menu item, which will take you to the Developer Area.
It looks like this:



Here the required scripts can be adapted and used to further process IP-Symcon's data. These scripts also contain the logic for establishing a connection to IP-Symcon, as well as the JSON-RPC interface required for this.

SDK (PHP)

Require: Symcon >= 4.0

DESCRIPTION

As of version 4.0, modules can be created using PHP.

REQUIREMENTS

PHP knowledge
IP-Symcon 4.0 or above

VIDEO-TUTORIALS

For many topics related to the SDK, video tutorials (German) are available on our YouTube channel.

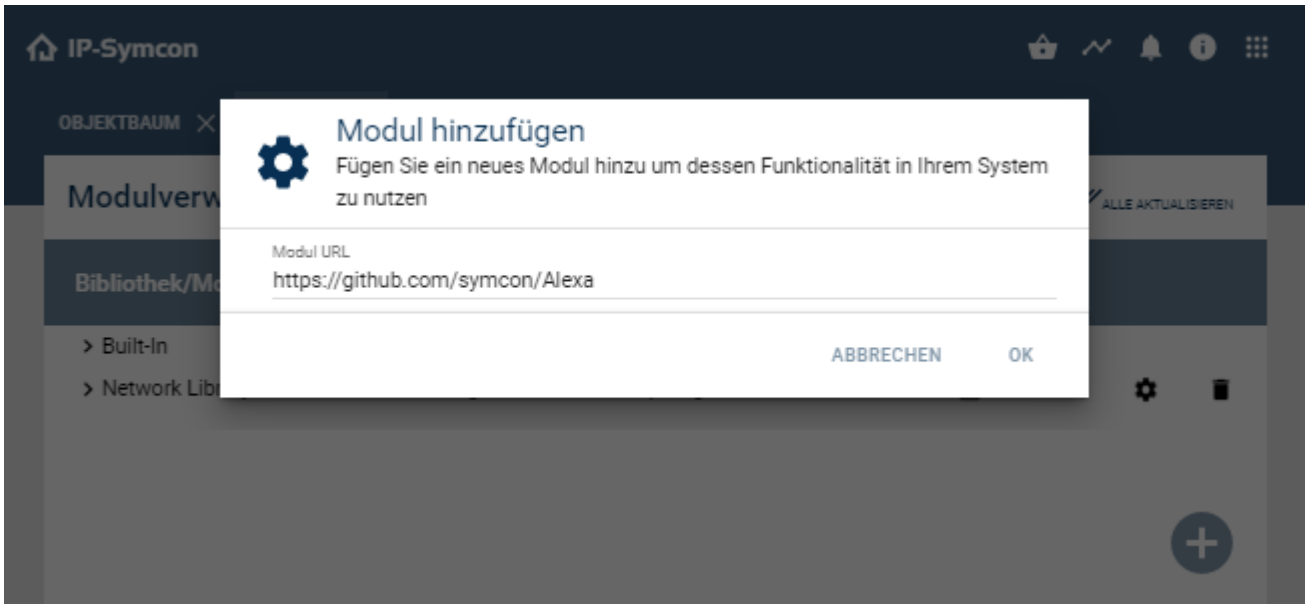
INTEGRATION IN IP-SYMCON

The data structure can be seen here .
The first step would be creating the library , as well as setting up all necessary files.
These can also be found in the structure overview.
Then the Modules can be developed.



In order to reload a module in the local development, a short restart of the service is necessary. For an update via the repository this step is not necessary.

After development has been completed, it is recommended that the library and its module(s) are combined into a repository.
This could, for example, look like this: Computation Module
The modules can then be added directly to the Module Control via the repository URL. It is also possible to submit this module for the Module store.


IP-Symcon
OBJEKTBAUM
Modulverw
ALLE AKTUALISIEREN
Bibliothek/Mo
> Built-In
> Network Libr
Modul hinzufügen
Fügen Sie ein neues Modul hinzu um dessen Funktionalität in Ihrem System zu nutzen
Modul URL
https://github.com/symcon/Alexa
ABBRECHEN
OK

Actions

Require: Symcon >= 6.0

DESCRIPTION

It is possible to define your own actions as part of the Library. After installing the library, these can be used as described under Actions .
If actions are saved or configured, this results in three values: The action target, the ActionID and the action parameters. The action target represents the target object, for example the underlying object in the case of an Event. The ActionID uniquely identifies the selected action. The parameters are a set of values which individually belong to the action. This allows a user to further parameterize an action and, for example, specify the value to which a variable should be switched.

STRUCTURE

For each action that should be part of the library, a JSON file must be created in the "actions" subfolder of the library. This JSON file must contain a JSON object that follows the following structure.

PARAMETER	DATA TYPE	DESCRIPTION
id	string	Unique GUID for the identification of the action. GUID Generator
caption	string	Visible name of the action
form	array / string	This field can contain an array of configuration elements as in the "elements" or "actions" areas of the Configuration Forms. Alternatively, the form can be generated dynamically by a PHP script. In this case, this field contains the corresponding code, which the configuration form returns as an object. The PHP code can be split into multiple strings in an array of strings. If an action that has already been defined is to be processed, for example in an Event, the defined parameters are available as system variables, see SelectAction. The settings of the form represent the parameters of the action, with each named configuration element making its value available under its "name".
action	array / string	PHP code that defines the execution of the action. If the action is carried out, the PHP code from this field is carried out. In doing so the parameters are available as system variables.
priority (optional)	int	Priority of the action, in a selection actions are sorted according to priority
category (optional)	string	(**default:** "other") The short form of the category of the action. Possible categories are listed under category .
restrictions (optional)	array	(**default:** []) A list of restrictions which limit the conditions under which this action is offered. The possible restrictions are described in the tables restrictions .

PARAMETER	DATA TYPE	DESCRIPTION
locale (optional)	object	(**default:** []) Here, as in the Localization of a Module, the action can be localized. This localizes both the form and the caption.
format (optional)	string	(**default:** content of caption) This string represents a short formatting of the action with its parameters, which can be displayed in different places. The formatting is noted in ICU syntax . There the parameters (name according to the parameter) and the target (name "TARGET") can be used as variables. Some additional Symcon-specific types are available for formatting. The content of format is translated by the locale.
description (optional)	string	(**default:** "") This text is displayed as a description for the action in the action selection. The content of description is translated by the locale.
readable (optional)	array / string	(**default:** content of action, whereby all status variables of parameters are replaced by the set values) PHP code, which returns the action as readable code. This is used, for example, for "insert commands" in the script editor. In doing so the parameters are available as system variables.
deprecated (optional)	array / string	(**default:** Not deprecated) If this parameter is set, this action is considered as deprecated and the action is not shown in any selections. Whenever a dialog to edit the action is opened, the PHP code in this parameter is executed with the currently set parameters. This action is immediately transformed to the action definition which is returned from the code.

CATEGORY

SHORT FORM	NAME	DESCRIPTION
target	Target specific	Actions that belong to a specific target, usually a module
math	Mathematical operations	Mathematical operations to modify objects
set	Set value	Actions that set the value of a variable
switch	Switch value	Actions that switch the value of a variable
expert	Expert	Actions that require a deeper understanding of IP-Symcon or PHP
other	Other actions	Actions that don't fit into any other category

RESTRICTIONS

Restrictions for all object types

PARAMETER	DATA TYPE	DESCRIPTION
objectType	array	The action is only offered for targets of the specified types (0 = category, 1 = instance, 2 = variable, 3 = script, 4 = event, 5 = media, 6 = link)

PARAMETER	DATA TYPE	DESCRIPTION
includeEnvironments	array	A list of environments in which the action is offered, see Environments
excludeEnvironments	array	A list of environments in which the action is not offered. If it matches "includeEnvironments", the action is not displayed. See Environments
writable	bool	If this restriction is set to true, the action is only offered if the target object is not write-protected. If the target is a variable, it must not have a Variable Action.

Restrictions for Target Objects of the Type Instance

PARAMETER	DATA TYPE	DESCRIPTION
hasActionStatusVariables	bool	If this restriction is set to true, the action is only offered if the instance has at least one switchable status variable
hasInstanceFunctions	bool	If this restriction is set to true, the action is only offered if the instance has at least one public function that was not inherited from the IPSModule
moduleID	array	The action is only offered if the ModuleID of the target instance is included in this list
moduleType	array	The action is only offered if the module type of the target instance is included in this list (since Symcon 9.0) 0 `MODULETYPE_CORE` Core 1 `MODULETYPE_IO` I/O 2 `MODULETYPE_SPLITTER` Splitter 3 `MODULETYPE_DEVICE` Device 4 `MODULETYPE_CONFIGURATOR` Configurator 5 `MODULETYPE_DISCOVERY` Discovery 6 `MODULETYPE_VISUALIZATION` Visualization
hasIdent	array	The action is only offered if the target instance has a child for each entry of the list, that has the corresponding ident (since Symcon 6.1)
hasNoIdent	array	The action is only offered if the target instance has NO child that has an ident corresponding to an entry of the list (since Symcon 8.0)

Restrictions for Target Objects of the Variable Type

PARAMETER	DATA TYPE	DESCRIPTION
requestAction	bool	If this restriction is set to true, the action is only offered if the variable has a variable action
variableType	array	The action is only offered if the variable type of the target variable is contained in this list (0 = Boolean, 1 = Integer, 2 = Float, 3 = String)

PARAMETER	DATA TYPE	DESCRIPTION
profilesInclude	array	The action is only offered if the name of the profile the target variable is currently using is included in this list
profilesExclude	array	The action is only offered if the name of the profile the target variable is currently using is NOT in this list
profileIsEnum	bool	If this restriction is set to true, the action is only offered if the profile of the variable represents an enumeration (step size = 0, at least one association)
profileIsPercentage	bool	If this restriction is set to true, the action is only offered if the profile of the variable represents a percentage value (suffix = "%", max value > min value)
presentation	array	The action is only offered if the Id of the used presentation is included in this list

Restrictions for Target Objects of the Type Automation

PARAMETER	DATA TYPE	DESCRIPTION
scriptType	array	The action is only offered if the automation type of the automation is included in this list (0 = PHP script, 1 = Flow script, 2 = IPSWorkflow) (since version 6.1)

Restrictions for Target Objects of the Type Event

PARAMETER	DATA TYPE	DESCRIPTION
eventType	array	The action is only offered if the event type of the event is included in this list (0 = Triggered, 1 = Cyclic, 2 = Schedule) (since version 6.3)

Environments

ENVIRONMENT	DESCRIPTION
Default	The default environment in which most actions are defined, many choices are limited to these actions or offer them in addition
TestCommands	Action selection in the "Test commands" dialog of an instance, "Default" actions are not offered here
EventTrigger	Action selection in a triggered event
EventCyclic	Action selection in a cyclical event
EventSchedule	Action selection in a weekly plan
FlowScript	Action selection in the schedule
ScriptEditor	Action selection in the "Add command" dialog of a script editor
Other environments	You can define your own environments, for example to offer special actions as part of an instance configuration

SYSTEM VARIABLES

Both in a possible PHP script for the form and in the script for the execution, the target of the action and the set parameters can be accessed via system variables. The following system variables are available in these scripts:

SYSTEM VARIABLE	DESCRIPTION
$_IPS['TARGET']	ObjectID of the target of the action
$_IPS [(name of the parameter)] (e.g. $ _IPS ['VALUE'])	The set or selected value of the parameter. These fields are not available in a script for the form if the action is initially selected and therefore no parameters have yet been selected. Therefore, the PHP function isset should be used to check whether the system variables are available.
Further system variables	Additional system variables are available based on the execution of the action. If the parameter names overlap, the parameter values are used and the regular system variables cannot be accessed.

FORMATTING

All types of the ICU format can be used in the format field. In addition, Symcon-specific types were introduced. As usual with ICU, these consist of three parts: The parameter designation, the type designation and optional additional parameters.

profile

The value is presented based on the profile of the target variable. If the target does not have a profile, the value is displayed directly.
Examples: {VALUE, profile}

valueFormatted

The formatted value of the variable with the ObjectID of the value is displayed.
Examples: {VARIABLE, valueFormatted}

object

The name of the object with the ObjectID of the value is displayed.

PARAMETER	DESCRIPTION
ident (optional)	If "ident" is specified as the parameter followed by a parameter name, the name of the child with the ident that corresponds to the parameter is displayed instead. If the value is behind ident in single quotation marks ('), the value in between them is directly used as Ident.

Examples: {VARIABLE, object}, {TARGET, object, ident IDENT}

action

The formatting of the action, which is coded as a JSON string in the value, is shown.

PARAMETER	DESCRIPTION
targetID (optional)	If "targetID" is specified as the parameter followed by a parameter name, the TargetID of the formatted action is set to the value of the corresponding parameter. If "ident" is also given immediately afterwards, followed by a parameter name, the target ID is instead displayed on the child's object ID with the ident that corresponds to the parameter. If the value is behind ident in single quotation marks ('), the value in between them is directly used as Ident.

Beispiele: {ACTION, action}, {ACTION, action, targetID VARIABLE}, {ACTION, action, targetID TARGET ident IDENT}

scheduleAction

The names of weekly plan actions of weekly plans are displayed under the specified parent object with the ID of the value.

PARAMETER	DESCRIPTION
parent	If "parent" is specified as the parameter followed by a parameter name, the object is used as the parent object with the value as its ID
action	If "action" is specified as the parameter, followed by an ActionID in single quotation marks ('), only the names of weekly plan actions that carry out the corresponding action are displayed

Examples: {SCHEDULE_ACTION, scheduleAction, parent PARENT}, {SCHEDULE_ACTION, scheduleAction, parent PARENT action '{7938A5A2-0981-5FE0-BE6C-8AA610D654EB}'}

selectPresentation

Different formattings depending on the presentation used by a variable.

PARAMETER	DESCRIPTION
ID of a presentation	If the ID of a presentation is used as a parameter followed by a formatting, the formatting is used if the variable uses the presentation.
other	If "other" is used as a parameter followed by a formatting, the formatting is used if the ID of the used presentation was not otherwise found as a parameter.

Examples: {TARGET, selectPresentation, {6B9CAEEC-5958-C223-30F7-BD36569FC57A}{Slider} {05CC3CC2-A0B2-5837-A4A7-A07EA0B9DDFB}{Color} other{Other presentation}}

EXAMPLES

switchValueString.json

```
{
"id": "{A4D52B67-BE4B-4AD0-964F-B9BA2556AAB0}",
"caption": "Switch to Value",
"form": [
{
"type": "ValidationTextBox",
"name": "VALUE",
"caption": "Value"
}
],
"action": "RequestAction($_IPS['TARGET'], $_IPS['VALUE']);",
"restrictions": {
"objectType": [ 2 ],
"variableType": [ 3 ],
"profileIsEnum": false,
"requestAction": true
},
"priority": 10,
"locale": {
"de": {
"Switch to Value": "Schalte auf Wert",
"Value": "Wert",
```

```
"Set to {VALUE, profile}": "Setze auf {VALUE, profile}"
}
},
"format": "Set to {VALUE, profile}"
}
```

setValueIntegerPreviousReminder.json

```
{
"id": "{92971E6F-4AC1-BEF8-B325-9E044D27B3EB}",
"caption": "Set to Value",
"form": [
"$form = [",
" {",
" 'type' => 'SelectColor',",
" 'name' => 'VALUE',",
" 'allowTransparent' => false,",
" 'caption' => 'Value',",
" 'writable' => true",
" }",
"];",
"if (isset($_IPS['VALUE'])) {",
" $form[] = {",
" 'type' => 'Label'",
" 'caption' => 'Previously selected value: ' . $_IPS['VALUE']",
" };",
"}",
"return $form;"
],
"action": "SetValue($_IPS['TARGET'], $_IPS['VALUE']);",
"restrictions": {
"objectType": [ 2 ],
"variableType": [ 1 ],
"profilesExclude": [ "~HexColor" ]
},
"priority": 5,
"locale": {
"de": {
"Set to Value": "Setze auf Wert",
"Value": "Wert",
"Set to {VALUE, profile}": "Setze auf {VALUE, profile}"
}
},
"format": "Set to {VALUE, profile}"
}
```

switchStatusVariable.json

```
{
"id": "{E616C2B2-A827-1712-4AE0-841C57B3DD74}",
"caption": "Switch Status Variable",
"form": [
"$options = [];",
"$firstVariable = false;",
```

```
"foreach (IPS_GetChildrenIDs($_IPS['TARGET']) as $childID) {",
" $object = IPS_GetObject($childID);",
" if (($object['ObjectType'] === OBJECTTYPE_VARIABLE) &&
($object['ObjectIdent'] !== '')) {",
" if (!HasAction($childID)) {",
" continue;",
" }",
" if (!$firstVariable) {",
" $firstVariable = $object;",
" }",
" $options[] = [",
" 'caption' => $object['ObjectName'],",
" 'value' => $object['ObjectIdent']",
" ];",
" }",
"}",
"return [",
" [",
" 'type' => 'Select',",
" 'name' => 'IDENT',",
" 'caption' => 'Status Variable',",
" 'options' => $options,",
" 'value' => $firstVariable['ObjectIdent'],",
" 'onChange' => 'IPS_UpdateFormField(\"ACTION\", \"targetID\",
IPS_GetObjectIDByIdent($IDENT, ' . $_IPS['TARGET'] . '), $id);'",
" ],",
" [",
" 'type' => 'SelectAction',",
" 'name' => 'ACTION',",
" 'caption' => 'Action',",
" 'targetID' => isset($_IPS['IDENT']) ?
IPS_GetObjectIDByIdent($_IPS['IDENT'], $_IPS['TARGET']) :
$firstVariable['ObjectID']",
" ]",
"];"
],
"action": [
"$targetVariableID = IPS_GetObjectIDByIdent($_IPS['IDENT'],
$_IPS['TARGET']);",
"$action = json_decode($_IPS['ACTION'], true);",
"echo IPS_RunActionWait($action['actionID'], $targetVariableID,
$action['parameters']);"
],
"restrictions": {
"objectType": [ 1 ],
"hasActionStatusVariables": true
},
"locale": {
"de": {
"Switch Status Variable": "Schalte Statusvariable",
"Status Variable": "Statusvariable",
"{TARGET, object, ident IDENT}: Switch Variable": "{TARGET, object,
ident IDENT}: Schalte Variable"
}
},
"format": "{TARGET, object, ident IDENT}: {ACTION, action, targetID
TARGET ident IDENT}",
"readable": [
"$targetVariableID = IPS_GetObjectIDByIdent($_IPS['IDENT'],
```

```
$_IPS['TARGET']);",
"$action = json_decode($_IPS['ACTION'], true);",
"$action['parameters']['TARGET'] = $targetVariableID;",
"$action['parameters']['ENVIRONMENT'] = $_IPS['ENVIRONMENT'];",
"$action['parameters']['PARENT'] = $_IPS['PARENT'];",
"echo IPS_GetActionReadableCode($action['actionID'],
$action['parameters']) . \"\\n\";"
]
}
```

multiplySetVariable.json (deprecated action)

```
{
"id": "{085176EC-BEE5-3732-15E5-8C82875CABD4}",
"caption": "Multiply with Value of other Variable",
"form": [
"$variable = IPS_GetVariable($_IPS['TARGET']);",
"$types = [];",
"if ($variable['VariableType'] === VARIABLETYPE_FLOAT) {",
" $types = [VARIABLETYPE_INTEGER, VARIABLETYPE_FLOAT];",
"}",
"else {",
" $types = [ $variable['VariableType'] ];",
"}",
"return [",
" [",
" 'type' => 'SelectVariable',",
" 'name' => 'VARIABLE',",
" 'caption' => 'Variable',",
" 'validVariableTypes' => $types",
" ]",
"];"
],
"action": "SetValue($_IPS['TARGET'], GetValue($_IPS['TARGET']) *
GetValue($_IPS['VARIABLE']));",
"restrictions": {
"objectType": [ 2 ],
"variableType": [ 1, 2 ],
"profileIsEnum": false,
"writable": true
},
"locale": {
"de": {
"Multiply with Value of other Variable": "Multipliziere mit Wert einer
anderen Variablen",
"Variable": "Variable",
"Multiply by value of {VARIABLE, object}": "Multipliziere mit Wert von
{VARIABLE, object}",
"Set the target variable to its current value multiplied with the value
of another variable": "Setze die Zielvariable auf ihren aktuellen Wert
multipliziert mit dem Wert einer anderen Variablen"
}
},
"format": "Multiply by value of {VARIABLE, object}",
"category": "math",
"description": "Set the target variable to its current value multiplied
```

```
with the value of another variable",
"deprecated": [
"return [",
" 'actionID' => '{A3153696-013A-41B1-A001-5E8085D95465}',",
" 'parameters' => [",
" 'DYNAMIC' => true,",
" 'FACTOR' => 0,",
" 'VARIABLE' => $_IPS['VARIABLE']",
" ]",
"];"
]
}
```

Libraries

Require: Symcon >= 4.0

DESCRIPTION

The library is the basis for every module development. Several modules can also be combined to form a library. The required directory structure can be viewed under Structure . This can be made available via a repository (we recommend Github or Bitbucket).

INTEGRATION IN IP-SYMCON

The library.json file must be available, which is located in the main directory.
Based on the directory structure, IP-Symcon can read in the entire library via the "Module Control".

PARAMETER	DATA TYPE	DESCRIPTION
id	string	Each module has its own GUID for unique identification (see info box below). GUID generator
author	string	Under what name was the library developed?
compatibility (since 4.3)	array	Checks whether the required version is installed. The kernel version and/or date can be checked. (For further description see table)
name	string	The name of the entire library. (A-Z, a-z, 0-9, spaces, underscores are allowed characters. However, spaces and underscores may not be at the beginning or the end. An empty name is also not valid.)
url	string	URL to the homepage (Must begin with http:// or https://. May alternatively be left empty)
version	string	Version number. This is represented as any desired string. We remmomend the form "number.number". E.g: "4.2"
build	integer	build number
date	integer	Unix timestamp



The GUID is a UUID and has the format 8-4-4-4-12. The numbers indicate the number of digits. The digits consist of characters between 0-9 and A-F. There must always be hyphens and curly brackets. Only capital letters may be used. (Example: {12345678-90AB-CDEF-1234-567890ABCDEF})

COMPATIBILITY

PARAMETER	DATA TYPE	DESCRIPTION
version (optional)	string	Minimum version as a string. E.g: "4.2"
date (optional)	integer	Date as UnixTimestamp. E.g: 1491343200

EXAMPLES

library.json

```
{
"id": "{F96B257F-85E7-47CF-8340-8FE850AACD10}",
"author": "Symcon GmbH",
- name: "Misc Modules",
"url": "https://www.symcon.de",
"compatibility": {
"version": "4.2",
"date": 1491343200
},
"version": "1.0",
"build": 0,
"date": 0
}
```

Presentations

Require: Symcon >= 8.0

DESCRIPTION

A visual overview of all presentations can be found in the Object Presentation.
The presentation of a variable can be set using the functions RegisterVariable and MaintainVariable. The presentation of a variable is determined by an array with the following "Key"->"Value" pairs.

KEY	DESCRIPTION
PRESENTATION	The ID of the presentation to be used, formatted as GUID (constants of the Presentations)
TEMPLATE	(optional) The ID of the template to be used, formatted as a GUID (constants of existing Templates)
Presentation Parameters	(optional) Instead of a template, any number of parameters for a presentation can be set. The parameters are set at the same level as the PRESENTATION parameter



If a parameter is set directly, the values of the template are ignored

Example:
The presentation of a variable is set to a slider with its own suffix when it is created.

```
$this->RegisterVariableFloat('Value', 'Value', [
'PRESENTATION' => VARIABLE_PRESENTATION_SLIDER,
'SUFFIX' => ' %'
]);
```

PRESENTATIONS

PRESENTATION
Enumeration
Date/Time
Duration
Color
Shutter
Switch

PRESENTATION
Slider
Web Content
Value Presentation
Value Input

Enumeration

Require: Symcon >= 8.0

PARAMETER

Information on the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_ENUMERATION {52D9E126-D7D2-2CBB-5E62-4CF7BA7C5D82}
ICON	String	Default Icon	Icons
OPTIONS	String	Options	JSON-encoded list of objects, the exact parameters are described under Options
LAYOUT	Integer	Layout	0: Column, 1: Row, 2: Grid
DISPLAY	Integer	Display...	0: Caption, 1: Icon, 2: Caption and Icon

OPTIONEN

NAME	TYPE	PARAMETER	DESCRIPTION
Value	Boolean/String/Integer/Float	Value	Depends on the type of the variable
Caption	String	Caption	
IconActive	Boolean	Overwrite Icon	
IconValue	String	Icon	Icons
Color	Integer	Color	

BEISPIELE

```
// When registering a variable for PHP modules
// Only icons are displayed. One option is highlighted in color.
$this->RegisterVariableInteger('Modus', 'Modus', [
'PRESENTATION' => VARIABLE_PRESENTATION_ENUMERATION,
'DISPLAY' => 1 /* Icon */,
'OPTIONS' => json_encode([
['Value' => 1, 'Caption' => 'Automatic', 'IconActive' => true,
'IconValue' => 'circle-a'],
['Value' => 2, 'Caption' => 'Eco', 'IconActive' => true, 'IconValue' =>
'circle-e', 'Color' => 40448],
['Value' => 3, 'Caption' => 'Manual', 'IconActive' => true, 'IconValue'
```

```
=> 'circle-m'],
])
]);

// For existing variables via script
// Only icons are displayed. One option is highlighted in color.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_ENUMERATION,
'DISPLAY' => 1 /* Icon */,
'OPTIONS' => json_encode([
['Value' => 1, 'Caption' => 'Automatic', 'IconActive' => true,
'IconValue' => 'circle-a'],
['Value' => 2, 'Caption' => 'Eco', 'IconActive' => true, 'IconValue' =>
'circle-e', 'Color' => 40448],
['Value' => 3, 'Caption' => 'Manual', 'IconActive' => true, 'IconValue'
=> 'circle-m'],
])
]);
```

Date/Time

Require: Symcon >= 8.0

PARAMETER

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	GUID of the Presentation	VARIABLE_PRESENTATION_DATE_TIME {497C4845-27FA-6E4F-AE37-5D951D3BDBF9}
DATE	Integer	Date Display	0: None, 1: Year, Month, and Day, 2: Month and Day, 3: Year and Day
MONTH_TEXT	Boolean	Format Month as	false: Number, true: Text
DAY_OF_THE_WEEK	Boolean	Show Day of the Week	
TIME	Integer	Time Display	0: None, 1: Hours and Minutes, 2: Hours, Minutes, and Seconds

TEMPLATES

```
// Registers a variable with a date template.
$this->RegisterVariableFloat('Date', 'Date', [
'PRESENTATION' => VARIABLE_PRESENTATION_DATE_TIME,
'TEMPLATE' => VARIABLE_TEMPLATE_DATE,
]);
```

NAME	CONSTANT	GUID
Date	VARIABLE_TEMPLATE_DATE	{B4C70F3E-6613-DA1A-7279-5DEE8DEB1B24}
Time	VARIABLE_TEMPLATE_TIME	{362DA268-56A2-E771-5E53-17E38B5D82E6}
Date/Time	VARIABLE_TEMPLATE_DATE_TIME	{BB0E9933-0403-BD3A-D1C9-255646934B00}

EXAMPLES

```
// When registering a variable for PHP modules
// Only icons are displayed. One option is highlighted in color.
$this->RegisterVariableInteger('Alarm', 'Alarm Time', [
'PRESENTATION' => VARIABLE_PRESENTATION_DATE_TIME,
'DATE' => 0 /* None */,
'TIME' => 2 /* Hours, Minutes, and Seconds */,
]);
// Shows only the time: 05:09:37
$this->RegisterVariableInteger('Alarm', 'Alarm Time', [
'PRESENTATION' => VARIABLE_PRESENTATION_DATE_TIME,
'DATE' => 0 /* None */,
'TIME' => 2 /* Hours, Minutes, and Seconds */,
]);

// For existing variables via script
// Displays a date as follows: Sun, May 3, 2026 05:09
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_DATE_TIME,
'DATE' => 1 /* Year, Month, and Day */,
'DAY_OF_THE_WEEK' => true,
'TIME' => 1 /* Hours and Minutes */,
]);
// Shows only the time: 05:09:37
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_DATE_TIME,
'DATE' => 0 /* None */,
'TIME' => 2 /* Hours, Minutes, and Seconds */,
]);
```

Duration

Require: Symcon >= 8.0

PARAMETERS

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	Presentation GUID	VARIABLE_PRESENTATION_DURATION {08A6AF76-394E-D354-48D5-BFC690488E4E}
COUNTDOWN_TYPE	Integer	Display Type	0: Value in Variable, 1: Duration until Value in Variable, 2: Duration since Value in Variable
FORMAT	Integer	Format	0: Seconds only, 1: Minutes and Seconds, 2: Hours, Minutes, and Seconds, 3: Hours and Minutes
MILLISECONDS	Boolean	Show Milliseconds	

EXAMPLES

```
// When registering a variable for PHP modules
// Formats the value of the variable as follows: 03:54
$this->RegisterVariableInteger('Countdown', 'Countdown', [
'PRESENTATION' => VARIABLE_PRESENTATION_DURATION,
'COUNTDOWN_TYPE' => 0 /* Value in Variable*/,
'FORMAT' => 1 /* Minutes and Seconds */,
]);
// Formats the duration until the timestamp in the variable as follows:
03:10:08
$this->RegisterVariableInteger('Countdown', 'Countdown', [
'PRESENTATION' => VARIABLE_PRESENTATION_DURATION,
'COUNTDOWN_TYPE' => 1 /* Duration until Value in Variable*/,
'FORMAT' => 2 /* Minutes and Seconds */,
]);

// For existing variables via script
// Formats the value of the variable as follows: 03:54
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_DURATION,
'COUNTDOWN_TYPE' => 0 /* Value in Variable*/,
'FORMAT' => 1 /* Minutes and Seconds */,
]);
// Formats the duration until the timestamp in the variable as follows:
03:10:08
```

```
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_DURATION,
'COUNTDOWN_TYPE' => 1 /* Duration until Value in Variable*/,
'FORMAT' => 2 /* Minutes and Seconds */,
]);
```

Color

PARAMETER

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_COLOR {05CC3CC2-A0B2-5837-A4A7-A07EA0B9DDFB}
ENCODING	Integer	Encoding	0 `COLOR_ENCODING_RGB` RGB 1 `COLOR_ENCODING_CMYK` CMYK 2 `COLOR_ENCODING_HSV` HSV 3 `COLOR_ENCODING_HSL` HSL 4 `COLOR_ENCODING_XY` xy
PRESET_VALUES	String	Default Values	JSON-encoded list of objects, the exact parameters are described under Colors Not available for encoding xy
COLOR_SPACE	Integer	Color Space	0: Custom, 1: sRGB, 2: AdobeRGB, 3: DCI-P3, 4: Rec2020
CUSTOM_COLOR_SPACE	String	Custom color space	JSON-encoded list of objects, the exact parameters are described under Custom Color Space
COLOR_CURVE	Integer	Color Curve	0: None, 1: Custom, 2: Daylight, 3: Daylight (Spring), 4: Daylight (Summer), 5. Daylight (winter)
CUSTOM_COLOR_CURVE	String	Custom Color Curve	JSON-encoded list of objects, the exact parameters are described under Custom Color Curve

COLORS

PARAMETER	TYPE	DESCRIPTION
Color	Integer	Color as Integer

CUSTOM COLOR SPACE

A color space is described by 4 objects with the following parameters. The 1st entry is for red, the 2nd for green, the 3rd for blue and the 4th for the white point.

PARAMETER	TYPE	DESCRIPTION
x	Float	X position of the color on the CIE standard color chart in the range 0-1
y	Float	Y position of the color on the CIE standard color chart in the range 0-1

Example:

```
// sRGB
[
{"x":0. 64, "y":0.33},
{"x":0.3, "y":0.6},
{"x":0.15, "y":0.06},
{"x":0.3127, "y":0.329}
]
```

CUSTOM COLOR CURVE

A color curve consists of any number of colors that represent the color temperature over the course of a day.

PARAMETER	TYPE	DESCRIPTION
x	Float	X position of the color on the CIE standard color chart in the range 0-1
y	Float	Y position of the color on the CIE standard color chart in the range 0-1

Example:

```
// Daylight (Winter)
[
{"x": 0. 477, "y" 0.4137},
{"x": 0.4599, "y" 0.4106},
{"x": 0.433, "y" 0.4027},
{"x": 0.4103, "y" 0.3932},
{"x": 0.3918, "y" 0.3835},
{"x": 0.3761, "y" 0. 374},
{"x": 0.3631, "y" 0.3652},
{"x": 0.352, "y" 0.357},
{"x": 0.3439, "y" 0.3507},
{"x": 0.3367, "y" 0.3447},
{"x": 0.3348, "y" 0.343},
]
```

TEMPLATES

```
// Registers a variable with shades of green as its default values.
$this->RegisterVariableString('Color', 'Color', [
'PRESENTATION' => VARIABLE_PRESENTATION_COLOR,
'TEMPLATE' => VARIABLE_TEMPLATE_COLOR_FOREST,
]);
```

NAME	KONSTANTE	GUID
Rainbow	VARIABLE_TEMPLATE_COLOR_RAINBOW	{0C711895-2F8E-DBFE-1700-84173491D229}
Forest	VARIABLE_TEMPLATE_COLOR_FOREST	{A7467E68-5C39-5BD9-C0C8-BCE6004FEEAA}

BEISPIELE

```
// When registering a variable for PHP modules
// Allows color selection in RGB format, with red, green, and blue as
the default values.
$this->RegisterVariableString('ColorRGB', 'Color', [
'PRESENTATION' => VARIABLE_PRESENTATION_COLOR,
'ENCODING' => COLOR_ENCODING_RGB,
'PRESET_VALUES' => json_encode([['Color' => 16711680], ['Color' =>
65280], ['Color' => 255]]),
]);
// Allows color selection in the xy format within a custom color space
$this->RegisterVariableString('ColorXY', 'Color', [
'PRESENTATION' => VARIABLE_PRESENTATION_COLOR,
'ENCODING' => COLOR_ENCODING_XY,
'COLOR_SPACE' => 0 /* Custom */,
'SELECTION' => 1 /* CIE Diagram */,
'CUSTOM_COLOR_SPACE' => json_encode([
['x' => 0.692, 'y' => 0.308], // Red
['x' => 0.170, 'y' => 0.700], // Green
['x' => 0.153, 'y' => 0.048], // Blue
['x' => 0.3127,'y' => 0.329] // White point
])
]);

// For existing variables via script
// Allows color selection in RGB format, with red, green, and blue as
the default values.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_COLOR,
'ENCODING' => COLOR_ENCODING_RGB,
'PRESET_VALUES' => json_encode([['Color' => 16711680], ['Color' =>
65280], ['Color' => 255]]),
]);
// Erlaubt die Farbauswahl im Format xy in einem eigenen Farbraum
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_COLOR,
'ENCODING' => COLOR_ENCODING_XY,
'COLOR_SPACE' => 0 /* Benutzerdefiniert */,
'SELECTION' => 1 /* CIE Diagramm */,
```

```
'CUSTOM_COLOR_SPACE' => json_encode([
['x' => 0.692, 'y' => 0.308], // Red
['x' => 0.170, 'y' => 0.700], // Green
['x' => 0.153, 'y' => 0.048], // Blue
['x' => 0.3127,'y' => 0.329] // White point
])
]);
```

Shutter

Require: Symcon >= 8.0

PARAMETER

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_SHUTT {6075FC22-69AF-B110-3749-C24138883082}
USAGE_TYPE	Integer	Usage Type	0: Open, 1: Rotation
OPEN_OUTSIDE_VALUE	Integer/Float	Open At / Outside At	
CLOSE_INSIDE_VALUE	Integer/Float	Closed At / Inside At	
MAX_ROTATION_INSIDE	Integer/Float	Maximum Rotation Inside	
MAX_ROTATION_OUTSIDE	Integer/Float	Maximum Rotation Outside	
SUN_POSITION	Integer	Sun Position	0: Left, 1: Right, 2: (None)

TEMPLATES

```
// Registers a variable that displays slats rotated to the right.
$this->RegisterVariableFloat('Lamelle', 'Lamelle', [
'PRESENTATION' => VARIABLE_PRESENTATION_SHUTTER,
'TEMPLATE' => VARIABLE_TEMPLATE_SHUTTER_LAMELLA_RIGHT,
]);
```

NAME	CONSTANT	GUID
Slat right	VARIABLE_TEMPLATE_SHUTTER_LAMELLA_RIGHT	{3BE75DE9-7D84-C082-2E77-9ED3AEE04D63}
Slat left	VARIABLE_TEMPLATE_SHUTTER_LAMELLA_LEFT	{22A0DF9C-C200-154A-641B-3A3CB096DB6D}

EXAMPLES

```
// When registering a variable for PHP modules
// A variable that represents the opening of the shutter. With a
variable value of 150, the presentation is closed - at 0 open.
$this->RegisterVariableInteger('Shutter', 'Shutter', [
'PRESENTATION' => VARIABLE_PRESENTATION_SHUTTER,
'CLOSE_INSIDE_VALUE' => 150,
'USAGE_TYPE' => 0 /* Open */,
'SUN_POSITION' => 2 /* None */,
'OPEN_OUTSIDE_VALUE' => 0,
]);
// A variable that represents the rotation of slats. With a variable
value of 100, the presentation is rotated outwards - at 0 inside.
$this->RegisterVariableInteger('Shutter', 'Shutter', [
'PRESENTATION' => VARIABLE_PRESENTATION_SHUTTER,
'CLOSE_INSIDE_VALUE' => 100,
'OPEN_OUTSIDE_VALUE' => 0,
'USAGE_TYPE' => 1 /* Rotation */,
'SUN_POSITION' => 1 /* Right */,
'MAX_ROTATION_INSIDE' => -30,
'MAX_ROTATION_OUTSIDE' => 30,
]);

// For existing variables via script
// A variable that represents the opening of the shutter. With a
variable value of 150, the presentation is closed - at 0 open.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_SHUTTER,
'CLOSE_INSIDE_VALUE' => 150,
'USAGE_TYPE' => 0 /* Open */,
'SUN_POSITION' => 2 /* None */,
'OPEN_OUTSIDE_VALUE' => 0,
]);
// A variable that represents the rotation of slats. With a variable
value of 100, the presentation is rotated outwards - at 0 inside.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_SHUTTER,
'CLOSE_INSIDE_VALUE' => 100,
'OPEN_OUTSIDE_VALUE' => 0,
'USAGE_TYPE' => 1 /* Rotation */,
'SUN_POSITION' => 1 /* Right */,
'MAX_ROTATION_INSIDE' => -30,
'MAX_ROTATION_OUTSIDE' => 30,
]);
```

Legacy Profile

Require: Symcon >= 8.0

PARAMETER

Information on the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_LEGACY {4153A8D4-5C33-C65F-C1F3-7B61AAF99B1C}
PROFILE	String	Profile	Variable Profiles

EXAMPLES

```
// When registering a variable for PHP modules
$this->RegisterVariableFloat('Temperature', 'Temperature', [
'PRESENTATION' => VARIABLE_PRESENTATION_LEGACY,
'PROFILE' => '~Temperature',
]);

// For existing variables via script
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_LEGACY,
'PROFILE' => '~Temperature',
]);
```

Switch

Require: Symcon >= 8.0

PARAMETERS

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_SWITCH {60AE6B26-B3E2-BDB1-A3A1-BE232940664B}
USE_ICON_FALSE	Boolean	Individual icons based on value	
ICON_TRUE	String	Icon for true, also false if USE_ICON_FALSE false	Icons
ICON_FALSE	String	Icon for false if USE_ICON_FALSE true	Icons
GLOW_COLOR	Integer	Color of glow while active	
GLOW_INTENSITY	Integer	Intensity of glow while active	
USAGE_TYPE	Integer	Variable Usage	0: On/Off, 1: Mute Switch, 2: None of these

EXAMPLES

```
// When registering a variable for PHP modules
// A switch that glows green.
$this->RegisterVariableInteger('Status', 'Status', [
'PRESENTATION' => VARIABLE_PRESENTATION_SWITCH,
'GLOW_COLOR' => 3210585,
'GLOW_INTENSITY' => 20,
'USAGE_TYPE' => 0 /* On/Off */
]);
// A switch with custom icons to mute
$this->RegisterVariableInteger('Mute', 'Mute', [
'PRESENTATION' => VARIABLE_PRESENTATION_SWITCH,
'USE_ICON_FALSE' => true,
'ICON_FALSE' => 'volume',
'USAGE_TYPE' => 1 /* Mute */,
'ICON_TRUE' => 'volume-xmark',
]);
```

```
// For existing variables via script
// A switch that glows green.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_SWITCH,
'GLOW_COLOR' => 3210585,
'GLOW_INTENSITY' => 20,
'USAGE_TYPE' => 0 /* On/Off */
]);
// A switch with custom icons to mute
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_SWITCH,
'USE_ICON_FALSE' => true,
'ICON_FALSE' => 'volume',
'USAGE_TYPE' => 1 /* Mute */,
'ICON_TRUE' => 'volume-xmark',
]);
```

Slider

Require: Symcon >= 8.0

PARAMETER

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_SLIDER {6B9CAEEC-5958-C223-30F7-BD36569FC57A}
MIN	Integer/Float	Minimum Value	
MAX	Integer/Float	Maximum Value	
STEP_SIZE	Integer/Float	Step Size	
GRADIENT_TYPE	Integer	Gradient	0: Default, 1: Temperature, 2: Tuneable White, 3: Custom
CUSTOM_GRADIENT	String	Custom Gradient	JSON-encoded list of objects with the parameters 'Value' and 'Color', where Color is encoded analogously to SelectColor
USAGE_TYPE	Integer	Variable Usage	0: Temperature, 1: Tuneable White, 2: Intensity, 3: Volume, 4: Progress, 5: None of these
PREFIX	String	Prefix	
SUFFIX	String	Suffix	
PERCENTAGE	Boolean	Display Type	false: Absolute, true: Percentage
THOUSANDS_SEPARATOR	String	Thousands Separator	Client: Default from Client, otherwise directly the value
DIGITS	Integer	Digits	
DECIMAL_SEPARATOR	String	Decimal Separator	Client: Default from Client, otherwise directly the value
ICON	String	Icon	
INTERVALS_ACTIVE	Boolean	Use updated parameters for specific intervals	
INTERVALS	String	Intervals	JSON-encoded list of objects, the exact parameters are described under Intervals

INTERVALS

NAME	TYPE	PARAMETER	DESCRIPTION
IntervalMinValue	Integer/Float	Interval Start	
IntervalMaxValue	Integer/Float	Interval End	
ConstantActive	Boolean	Display	false: Formatted Value, true: Constant
ConstantValue	String	Constant	
ConversionFactor	Integer/Float	Conversion Factor	
PrefixActive	Boolean	Overwrite Prefix	
PrefixValue	String	Prefix	
SuffixActive	Boolean	Overwrite Suffix	
SuffixValue	String	Suffix	
DigitsActive	Boolean	Overwrite Digits	
DigitsValue	Integer	Digits	
IconActive	Boolean	Overwrite Icon	
IconValue	String	Icon	

TEMPLATES

```
// Registers a variable with a room temperature template.
$this->RegisterVariableFloat('TargetValue', 'Temperature', [
'PRESENTATION' => VARIABLE_PRESENTATION_SLIDER,
'TEMPLATE' => VARIABLE_TEMPLATE_SLIDER_ROOM_TEMPERATURE,
]);
```

NAME	KONSTANTE	GUID
Room Temperature	VARIABLE_TEMPLATE_SLIDER_ROOM_TEMPERATURE	{868B087E-A38D-2155-EBE0-157AFBBF9E8C}
Color Temperature	VARIABLE_TEMPLATE_SLIDER_COLOR_TEMPERATURE	{66062309-21A9-26C0-213F-775C52E1473B}
Energy	VARIABLE_TEMPLATE_SLIDER_ENERGY	{BC799412-0C66-551F-CAEC-7566F5D52BD9}

NAME	KONSTANTE	GUID
Power	VARIABLE_TEMPLATE_VALUE_PRESENTATION_BATTERY	{8EC19DF0-89FB-A77E-ED7D-047A949CF292}

EXAMPLES

```
// When registering a variable for PHP modules
// A slider with a range of 0–2000 W. Values of 1000 W or higher are
displayed in kW.
$this->RegisterVariableInteger('Power', 'Power', [
'PRESENTATION' => VARIABLE_PRESENTATION_SLIDER,
'ICON' => 'bolt',
'SUFFIX' => ' W',
'MIN' => 0,
'MAX' => 2000,
'INTERVALS_ACTIVE' => true,
'INTERVALS' => json_encode([
[
'IntervalMinValue' => 1000,
'IntervalMaxValue' => 999999,
'ConversionFactor' => 1000,
'SuffixActive' => true,
'SuffixValue' => ' kW',
'DigitsActive' => true,
'DigitsValue' => 2,
'ConstantActive' => false,
'PrefixActive' => false,
]
])
]);

// For existing variables via script
// A slider with a range of 0–2000 W. Values of 1000 W or higher are
displayed in kW.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_SLIDER,
'ICON' => 'bolt',
'SUFFIX' => ' W',
'MIN' => 0,
'MAX' => 2000,
'INTERVALS_ACTIVE' => true,
'INTERVALS' => json_encode([
[
'IntervalMinValue' => 1000,
'IntervalMaxValue' => 999999,
'ConversionFactor' => 1000,
'SuffixActive' => true,
'SuffixValue' => ' kW',
'DigitsActive' => true,
'DigitsValue' => 2,
'ConstantActive' => false,
'PrefixActive' => false,
```

```
  ]
  ])
  ]);
```

Web Content

Require: Symcon >= 8.0

PARAMETER

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_WEB_CONTENT {9DE1D610-5106-97FB-714D-1AADEDF8377A}
HTML_TYPE	Integer	Display Type	0: HTML Inhalt, 1: Webseite
PADDING	Boolean	Remove Padding	

EXAMPLES

```
// When registering a variable for PHP modules
// The variable value is treated as a link. The website is displayed
without padding in the tiles.
$this->RegisterVariableString('Website', 'Website', [
'PRESENTATION' => VARIABLE_PRESENTATION_WEB_CONTENT,
'HTML_TYPE' => 1 /* Website*/,
'PADDING' => true,
]);

// For existing variables via script
// The variable value is treated as a link. The website is displayed
without padding in the tiles.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_WEB_CONTENT,
'HTML_TYPE' => 1,
'PADDING' => true,
]);
```

Value Presentation

Require: Symcon >= 8.0

PARAMETER

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_VALUE_PRESENTATION {3319437D-7CDE-699D-750A-3C6A3841FA75}
ICON	String	Default Icon	Icons
COLOR	Integer	Default Color	
PREFIX	String	Prefix	
SUFFIX	String	Suffix	

FLOAT AND INTEGER

NAME	TYPE	PARAMETER	DESCRIPTION
USAGE_TYPE	Integer	Variable Usage	0: None of these, 1: Temperature
PERCENTAGE	Boolean	Display Type	true: Percentage, false: Absolute
MIN	Integer/Float	Minimum Value	
MAX	Integer/Float	Maximum Value	
THOUSANDS_SEPARATOR	String	Thousands Separator	Client: Default from Client, otherwise directly the value
DIGITS	Integer	Digits	
DECIMAL_SEPARATOR	String	Decimal Separator	Client: Default from Client, otherwise directly the value
INTERVALS_ACTIVE	Boolean	Use updated parameters for specific intervals	
INTERVALS	String	Intervals	JSON-encoded list of objects, the exact parameters are described under Intervals

INTERVALS

NAME	TYPE	PARAMETER	DESCRIPTION
IntervalMinValue	Integer/Float	Interval Start	
IntervalMaxValue	Integer/Float	Interval End	
ConstantActive	Boolean	Display	false: Formatted Value, true: Constant
ConstantValue	String	Constant	
ConversionFactor	Integer/Float	Conversion Factor	
PrefixActive	Boolean	Overwrite Prefix	
PrefixValue	String	Prefix	
SuffixActive	Boolean	Overwrite Suffix	
SuffixValue	String	Suffix	
DigitsActive	Boolean	Overwrite Digits	
DigitsValue	Integer	Digits	
IconActive	Boolean	Overwrite Icon	
IconValue	String	Icon	
ColorActive	Boolean	Overwrite Color	
Color	Integer	Color	

BOOLEAN AND STRING

NAME	TYPE	PARAMETER	DESCRIPTION
MULTILINE	Boolean	Multiline	
OPTIONS	String	Options	JSON-encoded list of objects, the exact parameters are described under Options described

OPTIONS

NAME	TYPE	PARAMETER	DESCRIPTION
Value	Boolean/String	Value	Depends on the type of the variable
Caption	String	Caption	
IconActive	Boolean	Overwrite Icon	
IconValue	String	Icon	Icons
ColorActive	Boolean	Overwrite Color	
ColorValue	Integer	Color	

TEMPLATES

```
// Registers a variable with a temperature template.
$this->RegisterVariableFloat('Temperature', 'Room temperature', [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_PRESENTATION,
'TEMPLATE' => VARIABLE_TEMPLATE_VALUE_PRESENTATION_ROOM_TEMPERATURE,
]);
```

NAME	KONSTANTE	GU
Room Temperature	VARIABLE_TEMPLATE_VALUE_PRESENTATION_ROOM_TEMPERATURE	{90AF8F BBFD-E0 35FAB6[
Power	VARIABLE_TEMPLATE_VALUE_PRESENTATION_POWER	{2FED3[073D-60 901B- 2586A1A
Energy	VARIABLE_TEMPLATE_VALUE_PRESENTATION_ENERGY	{C899FC 063E-89 9DA4- 28ADD2
Battery	VARIABLE_TEMPLATE_VALUE_PRESENTATION_BATTERY	{7BD38C 07F2-5B 8F7F- 15398B8
Battery (Color)	VARIABLE_TEMPLATE_VALUE_PRESENTATION_BATTERY_COLOR	{C90EF3 165E-D0 032C- F468F48

EXAMPLES

```
// When registering a variable for PHP modules
// An alert variable with an icon and background color
$this->RegisterVariableBoolean('Alarm', 'Alarm', [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_PRESENTATION,
'OPTIONS' => json_encode([
['Value' => false, 'Caption' => 'OK', 'IconActive' => true, 'IconValue'
=> 'siren'],
['Value' => true, 'Caption' => 'Alarm', 'IconActive' => true,
'IconValue' => 'siren-on', 'ColorValue' => 16711680, 'ColorActive' =>
true],
])
]);
// Power in watts, with values above 1000 displayed in kilowatts
$this->RegisterVariableInteger('Power', 'Power', [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_PRESENTATION,
'ICON' => 'bolt',
'SUFFIX' => ' W',
'INTERVALS_ACTIVE' => true,
'INTERVALS' => json_encode([
[
'IntervalMinValue' => 1000,
```

```
'IntervalMaxValue' => 999999,
'ConversionFactor' => 1000,
'SuffixActive' => true,
'SuffixValue' => ' kW',
'DigitsActive' => true,
'DigitsValue' => 2,
'ConstantActive' => false,
'PrefixActive' => false,
]
])
]);

// For existing variables via script
// Only icons are displayed. One option is highlighted in color.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_PRESENTATION,
'OPTIONS' => json_encode([
['Value' => false, 'Caption' => 'OK', 'IconActive' => true, 'IconValue'
=> 'siren'],
['Value' => true, 'Caption' => 'Alarm', 'IconActive' => true,
'IconValue' => 'siren-on', 'ColorValue' => 16711680, 'ColorActive' =>
true],
])
]);
// Power in watts, with values above 1000 displayed in kilowatts
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_PRESENTATION,
'ICON' => 'bolt',
'SUFFIX' => ' W',
'INTERVALS_ACTIVE' => true,
'INTERVALS' => json_encode([
[
'IntervalMinValue' => 1000,
'IntervalMaxValue' => 999999,
'ConversionFactor' => 1000,
'SuffixActive' => true,
'SuffixValue' => ' kW',
'DigitsActive' => true,
'DigitsValue' => 2,
'ConstantActive' => false,
'PrefixActive' => false,
]
])
]);
```

Value Input

Require: Symcon >= 8.0

PARAMETER

Information about the visual properties and meanings of the parameters can be found in the Object Presentation.

NAME	TYPE	PARAMETER	DESCRIPTION
PRESENTATION	String	The ID of the presentation	VARIABLE_PRESENTATION_VALUE_INPUT {6F477326-1683-A2FD-D2E7-477F366ECB62}
PREFIX	String	Prefix	
SUFFIX	String	Suffix	
MULTILINE	Boolean	Multiline Input	

EXAMPLES

```
// When registering a variable for PHP modules
// Allows numbers to be entered freely.
$this->RegisterVariableFloat('Strompreis', 'Kosten pro kWh', [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_INPUT,
'SUFFIX' => ' €',
]);
// Allows free text to be entered over multiple lines.
$this->RegisterVariableString('Einkaufsliste', 'Einkaufsliste', [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_INPUT,
'MULLTILINE' => true,
]);

// For existing variables via script
// Allows numbers to be entered freely.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_INPUT,
'SUFFIX' => ' €',
]);
// Allows free text to be entered over multiple lines.
IPS_SetVariableCustomPresentation(12345, [
'PRESENTATION' => VARIABLE_PRESENTATION_VALUE_INPUT,
'MULLTILINE' => true,
]);
```

Data flow

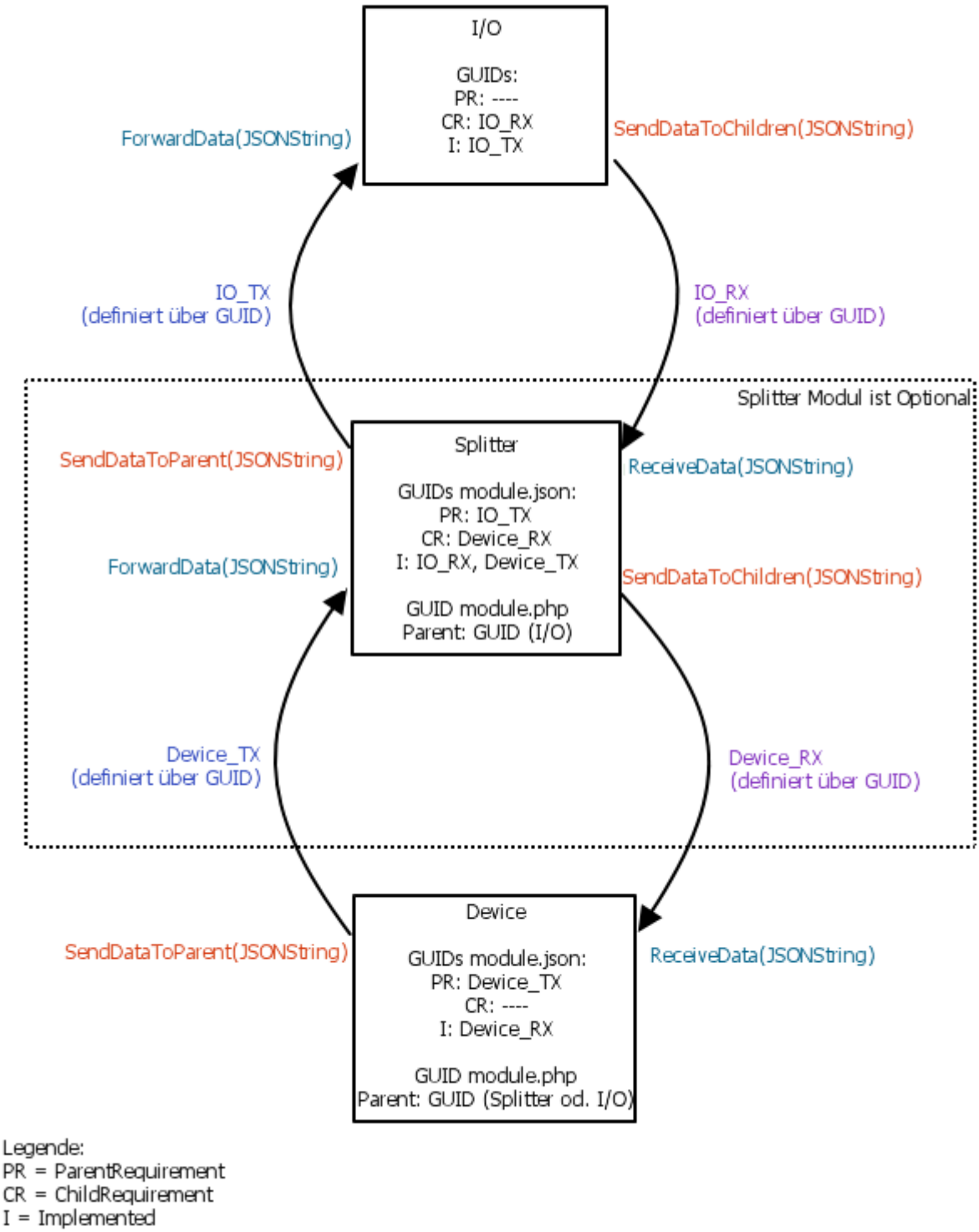
DESCRIPTION

The data flow should explain how data is sent between "Device <-> Splitter <-> I / O" in IP-Symcon and what has to be set up for it.



For a simple creation of a module including the data flow, the use of the module generator is recommended.

Typical structure



DIRECTION OF DATA FLOW

There are 2 directions of data flow.

I/O TO DEVICE

As can be seen in the figure, this flow direction is from the parent instance via SendDataToChildren to the children. The incoming data is processed within the children by

the overwritable function ReceiveData.
Optionally and if necessary, a splitter can be inserted between I/O and device.

DEVICE TO I/O

As can be seen in the figure, this flow direction is from the children instance via SendDataToParent to the parent. The incoming data is processed within the parent by the overwritable function ForwardData.
Optionally and if necessary, a splitter can be inserted between the device and I/O.

I/O MODULES

In the case of self-developed modules, the GUIDs for devices and splitters must be specified in the respective module.json file.
These can be created using the GUID Generator.
On the I/O side, the following data packages are available for I/O types:

I/O MODULE	MODULE GUID	SUPPORTED DATA PACKAGES
Client Socket	{3CFF0FD9-E306-41DB-9B5A-9D06D38576C3}	Simple
HID	{E6D7692A-7F4C-441D-827B-64062CFE1C02}	Extended (event)
HTTP Client	{4CB91589-CE01-4700-906F-26320EFCF6C4}	Extended (HTTP Request)
Multicast Socket	{BAB408E0-0A0F-48C3-B14E-9FB2FA81F66A}	Simple Extended (Socket)
Serial Port	{6DC3D946-0D31-450F-A8C6-C42DB8D7D4F1}	Simple
Server Sent Event Client	{2FADB4B7-FDAB-3C64-3E2C-068A4809849A}	Extended (SSE)
Server Socket	{8062CF2B-600E-41D6-AD4B-1BA66C32D6ED}	Simple Extended (Socket)
UDP Socket	{82347F20-F541-41E1-AC5B-A636FD3AE2D8}	Simple Extended (Socket) Erweitert (UDP)
Virtual I/O	{6179ED6A-FC31-413C-BB8E-1204150CF376}	Simple Extended (Socket)
WebSocket Client	{D68FD31F-0E90-7019-F16C-1949BD3079EF}	Simple

DATA PACKAGES

The data packets are encoded in JSONString.
These contain the GUID of the data packet type and the actual data.
Using the GUID as an identifier, the respective module knows what data is in the data packet and how it is formatted.

In modules based on IPSModuleStrict the field **Buffer** is transferred HEX-encoded: encode it with `bin2hex` when sending and decode it with `hex2bin` when receiving (instead of `utf8_encode`/`utf8_decode` with IPSModule). Symcon delivers upper-case hex and also accepts the lower-case output of `bin2hex` when sending. The other fields (e.g. **ClientIP** or the fields of the packets Extended (SSE) and Extended (HTTP Request)) stay unchanged. The examples show IPSModule; the Simple type also shows the variant for IPSModuleStrict.

SIMPLE

RX GUID: {018EF6B5-AB94-40C6-AA53-46943E824ACF}
TX GUID: {79827379-F36E-4ADA-8A95-5F8D1DC92FA9}

PARAMETER	DATA TYPE	DESCRIPTION
Buffer	String	Any data content

```
// Example for sending to the parent (TX packet) of the simple type
public function SendData() {
$this->SendDataToParent(json_encode([
'DataID' => "{79827379-F36E-4ADA-8A95-5F8D1DC92FA9}",
'Buffer' => utf8_encode("Hello World String"),
]));
}

// Received data from the parent (RX packet) of the simple type
public function ReceiveData($JSONString) {
$data = json_decode($JSONString);
$data['Buffer'] = utf8_decode($Data['Buffer']);

// Output in the message window for debug purposes
IPS_LogMessage("DATA", print_r($data, true));
}
```

```
// IPSModuleStrict: sending to the parent (TX packet) of the simple
type
public function SendData(): void {
$this->SendDataToParent(json_encode([
'DataID' => "{79827379-F36E-4ADA-8A95-5F8D1DC92FA9}",
'Buffer' => bin2hex("Hello World String"),
]));
}

// IPSModuleStrict: received data from the parent (RX packet) of the
simple type
public function ReceiveData(string $JSONString): string {
$data = json_decode($JSONString, true);
```

```
$data['Buffer'] = hex2bin($data['Buffer']);

// Output in the message window for debug purposes
IPS_LogMessage("DATA", print_r($data, true));
return '';
}
```

EXTENDED (EVENT)

RX GUID: {FD7FF32C-331E-4F6B-8BA8-F73982EF5AA7}
TX GUID: {4A550680-80C5-4465-971E-BBF83205A02B}

PARAMETER	DATA TYPE	DESCRIPTION
Buffer	String	Any data content
EventID	Integer	ID for the event

```
// Example for sending to the parent (TX packet) of the type Extended
(Event)
public function SendData() {
$this->SendDataToParent(json_encode([
'DataID' => "{4A550680-80C5-4465-971E-BBF83205A02B}",
'Buffer' => utf8_encode("Hello World String"),
'EventID' => 123
]));
}

// Received data from the parent (RX packet) of the type Extended
(Event)
public function ReceiveData($JSONString) {
$data = json_decode($JSONString);
$data['Buffer'] = utf8_decode($Data['Buffer']);

// Output in the message window for debug purposes
IPS_LogMessage("DATA", print_r($data, true));
}
```

EXTENDED (SOCKET)

RX GUID: {7A1272A4-CBDB-46EF-BFC6-DCF4A53D2FC7}
TX GUID: {C8792760-65CF-4C53-B5C7-A30FCC84FEFE}

PARAMETER	DATA TYPE	DESCRIPTION
Buffer	String	Any data content
Type	Integer	Type of connection (0 = Data, 1 = Connect, 2 = Disconnect)
ClientIP	String	Connection IP address
ClientPort	Integer	Connection port

```
// Example for sending to the parent (TX packet) of the extended type
(socket)
public function SendData() {
$this->SendDataToParent(json_encode([
'DataID' => "{C8792760-65CF-4C53-B5C7-A30FCC84FEFE}",
'Buffer' => utf8_encode("Hello World String"),
'Type' => 0,
'ClientIP' => "192.168.0.8",
'ClientPort' => 502
]));
}

// Received data from the parent (RX packet) of the type extended
(socket)
public function ReceiveData($JSONString) {
$data = json_decode($JSONString);
$data['Buffer'] = utf8_decode($Data['Buffer']);

// Output in the message window for debug purposes
IPS_LogMessage("DATA", print_r($data, true));
}
```

EXTENDED (UDP)

RX GUID: {9082C662-7864-D5CA-863F-53999200D897}
TX GUID: {8E4D9B23-E0F2-1E05-41D8-C21EA53B8706}

PARAMETER	DATA TYPE	DESCRIPTION
Buffer	String	Any data content
ClientIP	String	IP address of the connection, if empty then the host address set in the UDP socket is used. (Will be ignored if broadcast is active)
ClientPort	Integer	Connection port, if set to 0, the port set in the UDP socket is used
Broadcast	Boolean	Broadcast deactivated/activated (False = deactivated, True = activated)

```
// Example for sending to the parent (TX packet) of the extended type
(UDP)
public function SendData() {
$this->SendDataToParent(json_encode([
'DataID' => "{8E4D9B23-E0F2-1E05-41D8-C21EA53B8706}",
'Buffer' => utf8_encode("Hello World String"),
'ClientIP' => "192.168.0.8",
'ClientPort' => 502
'Broadcast' => false,
]));
}

// Received data from the parent (RX packet) of the extended type (UDP)
public function ReceiveData($JSONString) {
$data = json_decode($JSONString);
$data['Buffer'] = utf8_decode($Data['Buffer']);
```

```
// Output in the message window for debug purposes
IPS_LogMessage("DATA", print_r($data, true));
}
```

EXTENDED (SSE)

RX GUID: {5A709184-B602-D394-227F-207611A33BDF}

PARAMETER	DATA TYPE	DESCRIPTION
Event	String	Any type of event
Data	String	Any data content
Retry	String	In milliseconds before another attempt to send
ID	String	ID for the event

TX GUID: {79827379-F36E-4ADA-8A95-5F8D1DC92FA9}
TX is not evaluated but has to be set

```
// Received data from the parent (RX packet) of the extended type (SSE)
public function ReceiveData($JSONString) {
$data = json_decode($JSONString);

// Output in the message window for debug purposes
IPS_LogMessage("DATA", print_r($data, true));
}
```

Further information is available at w3.org/TR/eventsource

EXTENDED (HTTP REQUEST)

RX GUID: {018EF6B5-AB94-40C6-AA53-46943E824ACF}

PARAMETER RX	DATA TYPE	DESCRIPTION
Buffer	String	Any data content

TX GUID: {D4C1D08F-CD3B-494B-BE18-B36EF73B8F43}

PARAMETER TX	DATA TYPE	DESCRIPTION
RequestMethod	String	GET or POST
RequestURL	String	URL of the website to be requested
RequestData	String	Which data value should be requested
Timeout	Integer	Milliseconds until a timeout occurs

```
// Example for sending to the parent (TX packet) of the extended type
(HTTP request)
public function SendData() {
$this->SendDataToParent(json_encode([
'DataID' => "{D4C1D08F-CD3B-494B-BE18-B36EF73B8F43}",
'RequestMethod' => utf8_encode("POST"),
'RequestURL' => utf8_encode("https://reqbin.com/echo/post/form"),
'RequestData' => utf8_encode("dummy-post-data"),
'Timeout' => 10000
]));
}

// Received data from the parent (RX packet) of the simple type
public function ReceiveData($JSONString) {
$data = json_decode($JSONString);
$data['Buffer'] = utf8_decode($Data['Buffer']);

// Output in the message window for debug purposes
IPS_LogMessage("DATA", print_r($data, true));
}
```

REQUIREMENTS AND SETUP

The GUIDs for the data packet types used are defined in the module.json .
So that a data flow can be established between two entities, the sender must enter the type of communication used for the appropriate requirement.
The recipient enters the GUID of the communication type under Implemented.
If several GUIDs of different data packet types are entered in the module.json , a module can also interpret and process several data packet types.



The respective parent of each child must also be set up via ConnectParent, RequireParent or ForceParent. With the newer IPSModuleStrict, the necessary instances for the data flow are automatically created by the management console. If this automation is not suitable, the respective module can influence the automation of the management console via the function GetCompatibleParents.

EXAMPLE

Communication between I/O, splitter and device.
The device has the splitter as a parent and the splitter has the I/O as a parent.
Excerpt from the respective module.json
Example GUIDs of the data packet types:
IO_TX: {65465465-6546-6546-6546-65465465}
IO_RX: {AE3AE3A-AE3A-AE3A-AE3A-AE3AE3AE}
Device_TX: {78978978-7897-7897-7897-78978978}
Device_RX: {12312312-1231-1231-1231-12312312}

```
// Inside the I/O (parent of the splitter)
// I/Os usually have no other parents. So the list should normally be
empty.
"parentRequirements": [],
// GUID of the data packet type that is used via SendDataToChildren
"childRequirements": ["{AE3AE3A-AE3A-AE3A-AE3A-AE3AE3AE}"],
// List of GUIDs of the data packet types that are obeyed.
// This is processed by the ForwardData function (data come from the
child).
"implemented": ["{65465465-6546-6546-6546-65465465}"],

// Within the splitter (parent of the device, child of the I/O)
// GUID of the data packet type which is used via SendDataToParent
"parentRequirements": ["{65465465-6546-6546-6546-65465465}"],
// GUID of the data packet type that is used via SendDataToChildren
"childRequirements": ["{12312312-1231-1231-1231-12312312}"],
// List of GUIDs of the data packet types that are obeyed.
// This is processed either by the ForwardData function (data come from
the child) or ReceiveData (data come from the parent).
"implemented": ["{AE3AE3A-AE3A-AE3A-AE3A-AE3AE3AE}", "{78978978-7897-
7897-7897-78978978}"],

// Inside the device (child of the splitter)
// GUID of the data packet type which is used via SendDataToParent
"parentRequirements": ["{78978978-7897-7897-7897-78978978}"],
// Devices usually have no other children. So the list should normally
be empty.
"childRequirements": [],
// List of GUIDs of the data packet types that are obeyed.
// This is processed by the ReceiveData function (data come from the
parent).
"implemented": ["{12312312-1231-1231-1231-12312312}"],
```

Data management

Require: Symcon >= 4.0

DESCRIPTION

A module basically has four different forms of managing data. These are properties, attributes, buffers and status variables. These differ in terms of task, access options and persistence.
Properties
Attributes
Buffer
Status Variables

PROPERTIES

Properties are persistent data of a module which should/must be configured by the user. This data is only saved when "Apply" is clicked. Properties are, for example, required login data, device ID or interval for calling up sensor values. This is usually done via the configuration page, which is defined via the form.json.

FUNCTION NAME	BRIEF DESCRIPTION
ReadPropertyBoolean	Returns the value of a boolean property
ReadPropertyFloat	Returns the value of a float property
ReadPropertyInteger	Returns the value of an integer property
ReadPropertyString	Returns the value of a string property
RegisterPropertyBoolean	Creates a boolean property
RegisterPropertyFloat	Creates a float property
RegisterPropertyInteger	Creates an integer property
RegisterPropertyString	Creates a string property

```
public function Create() {
//Never delete this line!
parent::Create();
$this->RegisterPropertyBoolean("EmulateStatus", true);
$this->RegisterPropertyFloat("Faktor", 0.5);
$this->RegisterPropertyInteger("DeviceID", 0);
$this->RegisterPropertyString("Text", "");
}
```

ATTRIBUTES

Attributes are persistent data of a module, which are only set by the module itself and saved immediately. These are, for example, tokens for encrypted connections or saved values of a scene control.

FUNCTION NAME	BRIEF DESCRIPTION
ReadAttributeBoolean	Returns the value of a boolean attribute
ReadAttributeFloat	Returns the value of a float attribute
ReadAttributeInteger	Returns the value of an integer attribute
ReadAttributeString	Returns the value of a string attribute
RegisterAttributeBoolean	Creates a boolean attribute
RegisterAttributeFloat	Creates a float attribute
RegisterAttributeInteger	Creates an integer attribute
RegisterAttributeString	Creates a string attribute
WriteAttributeBoolean	Writes in a boolean attribute
WriteAttributeFloat	Writes to a float attribute
WriteAttributeInteger	Writes in an integer attribute
WriteAttributeString	Writes to a string attribute

Example

```
public function Create() {
//Never delete this line!
parent::Create();
$this->RegisterAttributeBoolean("BoolAttr", true);
$this->RegisterAttributeInteger("IntAttr", 5);
$this->RegisterAttributeFloat("FloatAttr", 3.7);
$this->RegisterAttributeString("StrAttr", "lalala");
}
public function BumpAndShow() {
var_dump($this->ReadAttributeBoolean("BoolAttr"));
var_dump($this->ReadAttributeInteger("IntAttr"));
var_dump($this->ReadAttributeFloat("FloatAttr"));
var_dump($this->ReadAttributeString("StrAttr"));
$this->WriteAttributeBoolean("BoolAttr", !$this-
>ReadAttributeBoolean("BoolAttr"));
$this->WriteAttributeInteger("IntAttr", $this-
>ReadAttributeInteger("IntAttr")*2);
$this->WriteAttributeFloat("FloatAttr", $this-
>ReadAttributeFloat("FloatAttr")+0.1);
$this->WriteAttributeString("StrAttr", $this-
>ReadAttributeString("StrAttr") . "öäü");
}
```

BUFFER

Buffers are non-persistent data of a module, which should only be managed by the module itself. These are, for example, incoming data records that are not transmitted in one, but

arrive gradually and therefore have to be put together. These are only taken from the buffer and processed by the module when the data record is complete.

FUNCTION NAME	BRIEF DESCRIPTION
GetBuffer	Returns the content of a buffer
GetBufferList	Returns an array of all buffers
SetBuffer	Creates a buffer

Example

```
public function ReceiveData($JSONString) {
//Decode JSONString
$data = json_decode($JSONString);
//Parse and write values to our buffer
$this->SetBuffer("Test", utf8_decode($data->Buffer));
//Print buffer
IPS_LogMessage("IOTest", $this->GetBuffer("Test"));
}
```

STATUS VARIABLES

Status variables are persistent data of a module, which can be changed by the module at any time. These are visible in the object tree and are available for further processing and display in the Visualizations. Status variables are, for example, sensor values, actuator values to be displayed on the WebFront and status values for further processing.

FUNCTION NAME	BRIEF DESCRIPTION
DisableAction	Disables the default action
EnableAction	Activates the default action
GetValue	Returns the value of a variable
SetValue	Sets the value of a variable
MaintainAction	Calls DisableAction or EnableAction
MaintainVariable	Configures a status variable
RegisterVariableBoolean	Creates a Boolean status variable
RegisterVariableFloat	Creates a float status variable
RegisterVariableInteger	Creates an integer status variable
RegisterVariableString	Creates a string status variable
RequestAction	Sets the value of a status variable
UnregisterVariable	Deletes a status variable

Example

```
public function Create() {
//Never delete this line!
parent::Create();
// Variables
$this->RegisterVariableString("TextData", "TextData", "");
IPS_SetHidden($this->GetIDForIdent("TextData"), true);
$this->RegisterVariableString("SimulationView", "SimulationView",
"~HTMLBox");
$this->RegisterVariableInteger("SimulationCounter", "SimulationCounter"
, "");
$this->RegisterVariableFloat("Factor", "Zoom Factor Wall Display",
"Factor.Display");
$this->RegisterVariableBoolean("Active", "Simulation active",
"~Switch");
$this->EnableAction("Active");
}
```

HTML-SDK

Require: Symcon >= 7.1

DESCRIPTION

The HTML SDK enables PHP modules to use an individual object presentation. This presentation allows full flexibility thanks to HTML. Messages between the HTML presentation and the module can be used to update the presentation at runtime or to inform the module about user interaction.

USE HTML AS PRESENTATION

If the HTML SDK is to be used, the visualization must be activated with the function SetVisualizationType. If the tile is then to be displayed in the visualization, the function GetVisualizationTile is used to return the HTML content.

TRANSLATION

Analogous to configuration forms, the texts of the HTML elements of the visualization are also localized based on the user's language.

ICONS

The Symcon icons can be used within the HTML presentation by loading the icons.js as script:

```
<script src="/icons.js"></script>
```

When the script is included, icons can be shown as explained in the documentation of Font Awesome. All icons that can be used normally in Symcon can be used as well. The Classic icons in the style Light can be used via CSS class fa-light while the Brands icons are used via class fa-brands. The custom icons can be used via class fa-kit.

MESSAGES

The function GetVisualizationTile is initially called once. If the display is to be adjusted at runtime, this must be done via messages. Alternatively, messages enable a channel from the visualization to the module in order to switch variables or perform other interactions.

It is also possible to realize a comparable communication via a HTML-Box. However, on the one hand, this is quite complex and, on the other hand, involves the risk of creating security gaps that could be exploited. In comparison, communication via HTML SDK is secured in both directions with the visualization password.

MODULE TO VISUALIZATION

If a message is to be sent from the module to the visualization, the function UpdateVisualizationValue must be used in the module. In order to receive and process the content of this message in the display, the function handleMessage must be implemented via JavaScript. This function receives the content that the content that was sent with UpdateVisualizationTile exactly as input. The format of this data is absolutely free and should be selected to suit the module.

VISUALIZATION FOR MODULE

The visualization can use the function requestAction via JavaScript. In return, this function executes the PHP function RequestAction on the module side, allowing the module to respond to the message.

SPECIAL FUNCTIONS IN JAVASCRIPT

The HTML SDK offers a range of functions that can be used within the display via JavaScript.

FUNCTION	DESCRIPTION
handleMessage	This function must be defined to receive messages from the module
requestAction	This function sends a message to the module
translate	Translates text based on localization
translateHTML	Translates the text of HTML elements based on the localization

WEBINAR AND EXAMPLES

At the time of publication, the functionality of the HTML SDK was presented in a webinar on YouTube.
Some sample implementations can also be viewed and tried out in our test repository:

EXAMPLE	DESCRIPTION
Duck counter	A small counter with detailed comments
Heat pump	A complex and extensive example for the representation of a heat pump

handleMessage

Require: Symcon >= 7.1

```
void handleMessage (mixed Data)
```

PARAMETERS

Data	Any data

RETURNS

No return

DESCRIPTION

This function must be implemented if the visualization is to receive and process messages from the module. If the function UpdateVisualizationValue is used in the module, this function is called in return with the same parameter value.

EXAMPLE

```
function handleMessage(data) {
// In this example, data simply contains text that may be translated
and displayed
document.getElementById('info').textContent = translate(data);
}
```

openObject

Require: Symcon >= 8.2

```
void openObject (int ObjectID)
```

PARAMETERS

ObjectID	The ID of the object to be opened.

RETURNS

No return

DESCRIPTION

This function opens an object in the visualization from which the function was executed. If the object is a category, the system jumps to this category. All other objects are displayed as a maximized tile.

EXAMPLE

```
// Opens the object with the ID 12345
openObject(12345);
```

requestAction

Require: Symcon >= 7.1

```
void requestAction (string Ident, mixed Value)
```

PARAMETERS

Ident	An ident
Value	A value

RETURNS

No return

DESCRIPTION

This function sends a message to the module and executes the function RequestAction with the given parameters.



Even if RequestAction is usually used to switch a variable with the given ident, the function is not limited to this. This means that any other behavior can be triggered when using an ident that does not match any variable.

EXAMPLE

```
// Set counter to 5
requestAction('Counter', 5);
```

translate

Require: Symcon >= 7.1

```
string translate (string Original)
```

PARAMETERS

Original	The original English text to be translated

RETURNS

The translated text

DESCRIPTION

Translates the text *original* based on the localization of the module and the language of the user. If no suitable translation exists, the original text is returned.

EXAMPLE

```
// Output the translated text to the console
console.log(translate('Hallo Welt'));
```

translateHTML

Require: Symcon >= 7.1

```
void translateHTML (object Node)
```

PARAMETERS

Node	An HTML element

RETURNS

No return

DESCRIPTION

Translates and overwrites the text content of the HTML element *node* and all its children.



When the visualization is initially loaded, this function is automatically applied once to the entire document. It is therefore only really necessary to use it if elements are dynamically adapted.

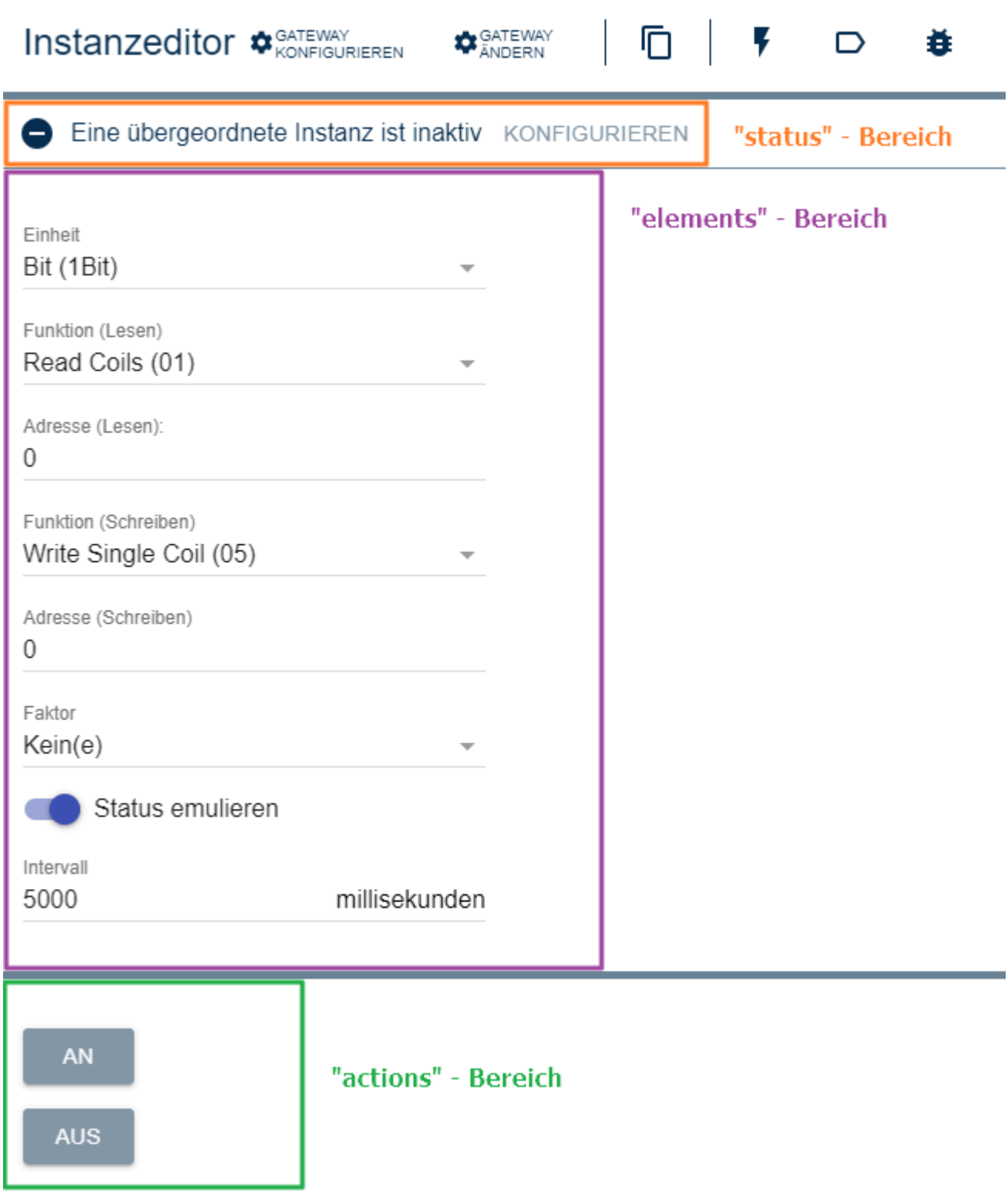
EXAMPLE

```
// Übersetze Inhalt des Textblockes 'info'
translateHTML(document.getElementById('info'));
```

Configuration Forms

Require: Symcon >= 4.0
The configuration forms are JSON files that are used to configure the properties of IP-Symcon instances.
These are usually loaded from the Management Console and displayed accordingly.
The JSON file describing the configuration form has up to three areas which can, independently of one another, be individually described.
A translation of the "caption" and "label" can be added via the Localizations via the locale.json.

FORM AREAS



elements
Form fields defined in the "elements" area can set the properties of an instance. The name of a form field, if available, corresponds to the name of the property which is set via IPS_SetProperty. If the "elements" area is not to be visible, it must not be defined.
actions
The "actions" area is a so-called test environment. The form fields defined here cannot change the properties of the respective instance. If the "actions" area is not to be visible, it must not be defined.
The general rule: The test environment can only be used if changes have been accepted in the elements area. This prevents settings that have not been accepted from being tested.
status
The "status" area defines possible status messages. No form fields are allowed in the "status"

area, only status messages which are explained under Status Message. If the module is not to display any status messages, this area does not have to be defined.

OVERVIEW OF THE INSERTABLE FORM FIELDS

FORM FIELD / TYPE	SINCE IP-SYMCON	DESCRIPTION
Button	4.0	Button
CheckBox	4.0	Checkbox
ColumnLayout	6.0	Column view of several form fields
Configurator	5.0	Describes a configurator
ExpansionPanel	5.0	Form fields that can be opened and closed
HorizontalSlider	4.0	A horizontal slider
Image	5.0	Creates an image on the module page
IntervalBox	4.0	Specification of numbers with text description / unit (outdated)
Label	4.0	A label
List	4.2	An editable list
NumberSpinner	4.0	Input field in which only numbers are allowed. (optionally with decimal places)
OpenObjectButton	5.3	Button that opens an object
PasswordTextBox	4.0	Input field for passwords
PopupAlert	5.0	Alarm, which opens a popup immediately
PopupButton	5.0	Button that opens a popup
ProgressBar	5.2	Progress bar
QrCode	7.0	Creates a QR code
RadioButtonGroup	9.1	Group of radio buttons
RowLayout	5.0	Row view of multiple form fields
ScriptEditor	6.0	Script editor
Select	4.0	Drop-down menu
SelectAction	6.0	Selection of an action
SelectCategory	4.0	Selection dialog for a category
SelectColor	4.2	Selection dialog for a color
SelectCondition	6.1	Input of conditions
SelectDate	5.0	Selection dialog for a date
SelectDateTime	5.0	Selection dialog for a date and time
SelectEvent	4.0	Selection dialog for an event
SelectFile	4.2	Selection dialog for a file
SelectIcon	6.1	Selection dialog for an icon
SelectInstance	4.0	Selection dialog for an instance
SelectLink	4.0	Selection dialog for a link

FORM FIELD / TYPE	SINCE IP-SYMCON	DESCRIPTION
SelectMedia	4.0	Selection dialog for a media file
SelectModule	5.5	Selection dialog for an instance of a module
SelectObject	4.0	Selection dialog for an object
SelectProfile	5.5	Selection dialog for a variable profile
SelectScript	4.0	Selection dialog for a script
SelectTime	5.0	Selection dialog for a time
SelectValue	6.3	Selection of a value matching a variable
SelectVariable	4.0	Selection dialog for a variable
TestCenter	5.0	Test area for the status variables of the instance
Tree	5.0	Creates a tree
ValidationTextBox	4.0	Input field for text

EXAMPLES

Basic structure of the JSON file

```
{
"elements":
[
... // Insert form fields here
],
"actions":
[
... // Insert form fields here
],
"status":
[
... // Insert status messages here
]
}
```

Complete example using an SMS module

```
{
"elements":
[
{ "type": "ValidationTextBox", "name": "Username", "caption":
"Username" },
{ "type": "ValidationTextBox", "name": "Password", "caption":
"Password" },
{ "type": "NumberSpinner", "name": "APIID", "caption": "APIID" },
{ "type": "ValidationTextBox", "name": "Sender", "caption": "Sender" },
{ "type": "Select", "name": "SMSType", "caption": "Type",
```

```
"options": [
{ "caption": "SMS", "value": 0 },
{ "caption": "Combi-SMS", "value": 1 },
{ "caption": "MMS", "value": 2 }
]}
],
"actions":
[
{ "type": "ValidationTextBox", "name": "Number", "caption": "Number" },
{ "type": "ValidationTextBox", "name": "Message", "caption": "Message"
},
{ "type": "Button", "caption": "Send Message", "onClick":
"SMS_Send($id, $Number, $Message);" },
{ "type": "Button", "caption": "Read Balance", "onClick": "echo
'Balance: '.sprintf('%.2f', SMS_RequestBalance($id)).' credits';" }
],
"status":
[
{ "code": 102, "icon": "active", "caption": "Login information valid"
},
{ "code": 201, "icon": "error", "caption": "Authentication failed" },
{ "code": 202, "icon": "error", "caption": "No credits left" }
]
}
```

VALIDATION

A JSON schema is available for configuration forms: formSchema.json
The schema describes all form fields and their parameters as they are supported by the Management Console. Unknown form fields, typos in parameter names, wrong data types (e.g. `"digits": "2"` instead of `"digits": 2`) and parameters that have no effect on the respective form field are reported as errors.
With the schema, editors like Visual Studio Code or PhpStorm offer autocompletion and check the form.json while editing. To do this, the schema is specified in the form.json:

```
{
"$schema":
"https://www.symcon.de/assets/files/validation/formSchema.json",
"elements": [ ... ]
}
```

Alternatively, the schema can be assigned to the file form.json in the settings of the editor, in Visual Studio Code e.g. via the setting `json.schemas`.
Forms that are generated dynamically via GetConfigurationForm must also match the schema. Parameters like "options" or "columns", which are often only added there, are therefore optional in the schema.
The schemas for the other files of a module are listed under Structure.

Button

A button that executes an action when pressed.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption of the button
confirm (optional)	(**default:** "") If this parameter does not contain an empty string, the text is displayed in a yes/no dialog when the button is pressed. The onClick action is only carried out if the dialog is confirmed with "Yes". If the parameter is empty, the action is carried out directly. (since IP-Symcon 5.0)
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onClick script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the button can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
~~label~~	~~Name and visible caption of the button~~ *(deprecated, cannot be changed)*
link (optional)	If this parameter is true, the output of the onClick script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. If the parameter is not set, a decision whether it is opened as a link or displayed in the dialog is made on the basis of the output (see info box). (since IP-Symcon 5.1)
name (optional)	Name of the button
onClick	Script that is executed when the button is clicked. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). No PHP-tags are required. There are a few other variables available, which are discussed in the information field below.
type	Button
visible (optional)	(**default:** true) If true, the button is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	Fixed width of the button in pixels or % as a string, e.g. "40%" or "250px". If the value is not set or "", the width is automatically selected based on the caption (since IP-Symcon 5.2)



The variable $id is available in all areas. In addition, variables are created for all form fields if a name has been assigned. The variable can then be used accordingly as $Name. The $Name variable is only available within the same scope (e.g. actions or elements).

If the parameter onClick contains an "echo" in combination with 'http'/'https', the browser is opened and the URL is called. If there is an "echo" in combination with a 'mailto:', the default mail program is opened with all the parameters set. If an "echo" is present in combination with a 'link:', 'link:' is filtered out and the remaining content is opened in the browser. This makes relative links possible. (available since IP-Symcon 4.1)

EXAMPLE

```
//The example returns the instance ID.
{ "type": "Button", "caption": "On", "onClick": "echo $id;" }

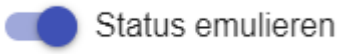
// Special example:
// The example opens the browser with the specified address when
"clicked": 'https://www.symcon.de'
{
"type": "Button",
"caption": "Symcon",
"onClick": "echo 'https://www.symcon.de';"
}

// The example asks whether the value of a variable should actually be
set to 0
{
"type": "Button",
"caption": "Reset",
"onClick": "SetValue(12345, 0);",
"confirm": "This sets the variable back to 0. Important data can be
lost in the process. Are you sure?"
}

// The example runs a multi-line script
{
"type": "Button",
"caption": "Reset",
"onClick": [
"SetValue(12345, 0);",
"echo 'value was reset';"
]
}
```

CheckBox

Creates a checkbox.
If created in the "elements" area, the checkbox sets a property to **True** or **False** when accepted.
The parameter **"name"** defines which property is set.



PARAMETERS

PARAMETER	DESCRIPTION
caption	visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.0)
enabled (optional)	(**default:** true) If true, the CheckBox can be used, otherwise it is displayed as deactivated (since Symcon 5.2)
items (optional)	(**default:** []) List of configuration elements which are only displayed while the CheckBox is activated. The values of the elements are retained if it is deactivated again (cannot be changed) (since Symcon 9.1)
itemsPosition (optional)	(**default:** 0) The position of the configuration elements defined by "items" (since Symcon 9.1) 0: The elements are displayed below the CheckBox 1: The elements are displayed to the right of the CheckBox
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 6.0)
name (optional)	Name of the checkbox/the property to be set
onChange (optional)	(**default:** "") Script, which is executed with a click when the checkbox is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since Symcon 5.2)
type	CheckBox
value (optional)	(**default:** false) The value of the CheckBox - If there is an associated property, this parameter is overwritten by the property in the elements area (since Symcon 5.2)
visible (optional)	(**default:** true) If true, the CheckBox is visible, otherwise it is invisible (since Symcon 5.2)
width (optional)	Fixed width of the checkbox in pixels or % as a string, e.g. "40%" or "250px". If the value is not set or "", the width is automatically selected based on the caption (since Symcon 5.3)

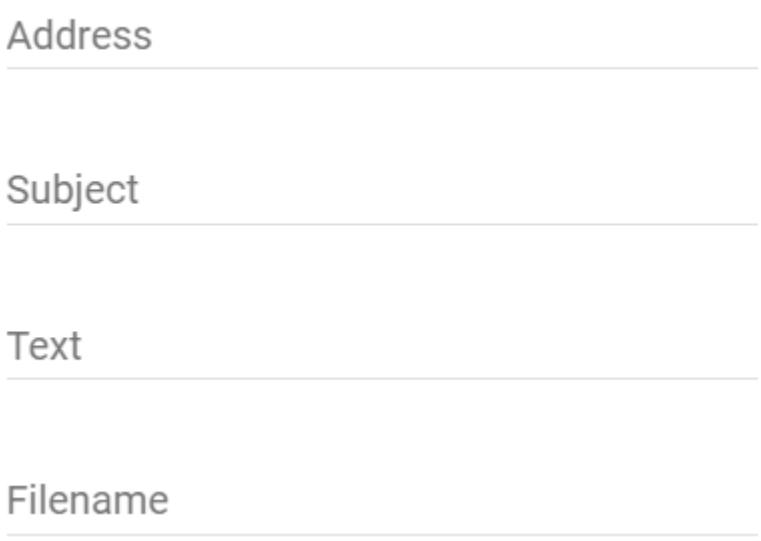
EXAMPLE

```
{ "type": "CheckBox", "name": "status", "caption": "emulate Status" }

//Configuration elements which are only displayed while the CheckBox is
activated
{ "type": "CheckBox", "name": "UseProxy", "caption": "Use proxy",
"itemsPosition": 1,
"items": [
{ "type": "ValidationTextBox", "name": "ProxyHost", "caption": "Host"
},
{ "type": "NumberSpinner", "name": "ProxyPort", "caption": "Port" }
]
}
```

ColumnLayout

Require: Symcon >= 6.0
Creates a column in which form fields are displayed one below one another. This is particularly useful for dynamically adapting the content.



This item is not supported by the Legacy Console. If a module with this element is opened in the legacy console, all the elements contained are displayed one below the other in the form.

PARAMETERS

PARAMETER	DESCRIPTION
halign (optional)	(**default:** "left") Horizontal alignment of the elements within the column (since Symcon 9.1) "left": The elements are aligned to the left "center": The elements are centered "right": The elements are aligned to the right

items	List of configuration items within the column
name (optional)	Name of the column
type	ColumnLayout
valign (optional)	(**default:** "") Vertical alignment of the elements within the available height, e.g., if the column is placed in a RowLayout next to a higher column. If the parameter is set, the column uses the full available height (since Symcon 9.1)
"": The column is as high as its content and is aligned according to the parameter valign of the surrounding RowLayout
"top": The elements are aligned to the top
"middle": The elements are aligned to the middle
"bottom": The elements are aligned to the bottom
"between": The elements are distributed over the height, the first element at the top and

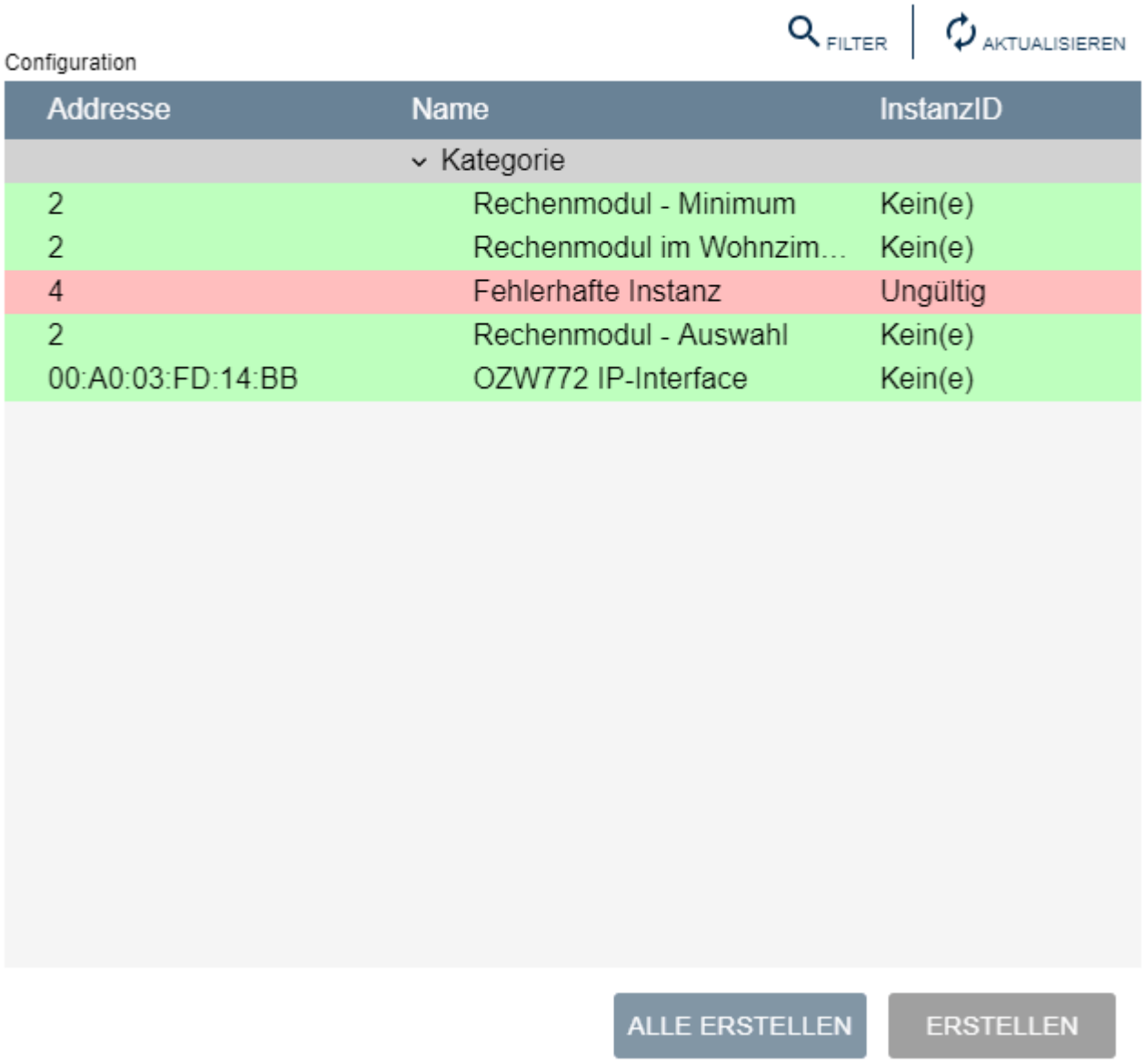
the last one at the bottom |
| visible (optional) | (**default:**true) If true, the column is visible, otherwise it is invisible |

EXAMPLE

```
// column layout
{
"type": "ColumnLayout",
"items": [
{
"type": "SelectLocation",
"name": "Location",
"caption": "My Location"
},{
"type": "Button",
"caption": "Test",
"onClick": "TM_Update($id);"
},{
"type": "Button",
"caption": "Test2",
"onClick": "TM_Update2($id);"
}
]
}
```

Configurator

Require: Symcon >= 5.0
Describes the appearance and functionality of a configurator.



This item is not supported by the Legacy Console. If a module with this element is opened in the legacy console, an error message appears.

GENERAL PARAMETERS

PARAMETER	DESCRIPTION
caption (optional)	visible caption of the configuration list

PARAMETER	DESCRIPTION
columns (optional)	(**default:** columns for address ("address") and name ("name")) Columns of the configuration list
delete (optional)	(**default:** false) If true, a button is displayed behind each entry with InstanceID, which deletes the corresponding instance and updates the line in the configurator
discoveryInterval (optional)	(**default:** 0) If the console requests the including configuration form for Device Search, this parameter defines how many seconds need to pass before another request is allowed to be executed; If multiple elements of the type Configurator are included within the configuration form, the maximum is used (since IP-Symcon 6.3)
onExpand (optional)	(**default:** "") Script which is executed after expanding an entry of the configurator. If the script consists of several lines, the individual lines can also be defined as an array. In this version, the tree does not offer the currently selected entry as the variable value, but the expanded entry, see Selected Line as a Variable in on-Actions. Otherwise it has the same properties as onClick of the Button (since IP-Symcon 6.4)
name (optional)	Name of the configuration list
rowCount (optional)	(**default:** 0) visible number of lines; If there are more lines in the list, a scroll bar is shown - if the value is 0, the configuration list fills the remaining space in the instance editor. Only one configuration item can get all of the remaining space.
sort(optional)	Sorting of the table. If this is not set, the list remains unsorted
type	Configurator
values (optional)	Entries of the form
visible (optional)	(**default:** true) If true, the configurator is visible, otherwise it is invisible (since IP-Symcon 5.2)



If the parameters columns, sort or values are to be changed via UpdateFormField, the new values must be coded as JSON.

PARAMETERS FOR COLUMNS

PARAMETER	DESCRIPTION
caption	visible caption of a column
~~label~~	~~visible caption of a column~~ *(deprecated)*
name	unique name of a column to refer to; the name cannot be "id", "parent", "create", "expanded" or "InstanceID"
quickFilter (optional)	(**default:** true) If true, the text of this column is considered during the search via quick filter. Should the value be false for all columns, the quick filter is hidden. (since IP-Symcon 7.2)
visible (optional)	(**default:** true) If false, the column is not displayed

PARAMETER	DESCRIPTION
width	Width of the column in pixels, as a CSS string (e.g. "100px"); a single column may have the value "auto", which means that the width of this column uses the remaining available space. In the case of a tree, the value "auto" must be set for a column. The expansion symbols are displayed in this column.

PARAMETERS FOR SORT

PARAMETER	DESCRIPTION
column	Name of the column by which the list is sorted
direction	The direction of the sort; "ascending" for an ascending order and "descending" for a descending order

PARAMETERS FOR VALUES

PARAMETER	DESCRIPTION
address (optional)	(**default** : "") If the default columns were used, this value is used for the address column.
create (optional)	(**default** : error) Specification of the configuration of the created instance (More information)
expanded (optional)	(**default** : false) If this value is true, the entry is expanded directly when the configuration form is loaded
instanceID (optional)	(**default** : 0) InstanceID of the device in the object tree, 0 if the device has not yet been created
id (optional)	An identifier with which the entry for a tree display can be addressed as a parent node, id must be greater than 0
name (optional)	(**default** : "") If the default columns were used, this value is used for the name column. This value additionally defines the name of the instance to be created. If the instance already exists, the name column shows the name of the instance and not the content of this parameter.
parent (optional)	(**default** : 0) The id of a value can be set in this field, which becomes the parent node of this value in the tree display. For this, the parent value must come before of the child in the array. If "parent" is 0, the element is inserted at the top level.

PARAMETER FOR CREATE

There are various options for defining the instances to be created and their initial configuration.

VARIANT 1: CONFIGURATION OF A SINGLE INSTANCE

In the first variant, a single instance and its configuration is defined. In this case create is an object that contains information about the ModuleID used and the initial configuration: *Parameters for configuration*

PARAMETER	DESCRIPTION
moduleID	The module ID of the instance required or to be created
configuration	The required configuration of the instance, see IPS_SetConfiguration, not coded as a JSON string
location (optional)	(**default:** []) The list of strings indicates the initial position in the object tree. If the list is empty, the instance is created in the main category. Otherwise, the categories with the respective name are used hierarchically and created if necessary.
info (optional)	(**default:** "") The initial description of the newly created instance; if an existing instance is used, the description is not overwritten.
position (optional)	(**default:** 0) The initial position of the newly created instance; if an existing instance is used, the position is not overwritten.
name (optional)	(**default:** module name or value of the "name" column) The initial name of the newly created instance; if an existing instance is used, the name is not overwritten. (since IP-Symcon 5.5)
noDiscovery (optional)	(**default:** false) If this parameter is set to true, this entry is not displayed as a device in the Device Search and cannot be marked as read. In the case of a configuration as a chain (variant 2), the value of the first entry in the chain is used. If there are several possible configurations (variant 3), each configuration must have this parameter set to true so that the selection does not appear in the Device Search. (since IP-Symcon 5.4)
statusVariables (optional)	(**default:** {}) This object specifies an initial configuration of the status variables. For each parameter of this status variable object, the variable with the ident of the parameter name is configured. The associated parameter is another object that defines the configuration of the variables. The variable configuration is used after the instance configuration has been adopted. Variables that are created later cannot be configured here. (since IP-Symcon 5.3)

Parameters for status variables

PARAMETER	DESCRIPTION
name (optional)	(**default:** specification by created instance) The status variable is renamed to this value. The name is translated through the localization of the configurator.

If an instance is created, it is created with the ModuleID and configuration specified by the configuration. As is usual with the creation of an instance, the new instance is linked to a valid physical parent instance, if possible. The physical parent instance of the configurator is preferably used here.

VARIANT 2: CONFIGURATION OF A CHAIN

Alternatively, a chain of configurations can be used. This defines the entire physical chain for the newly created instance, usually the device, splitter and I/O. In this variant, data.create represents a list, the elements of which define the instances of the physical chain as in variant 1. The list begins with the device to be created and then continues with the next parent instance.
When creating a chain, existing parent instances are preferably reused, provided that these and their parent instances meet the requirements of the chain. However, the device itself will always be newly created.

VARIANT 3: SEVERAL POSSIBLE CONFIGURATIONS

Finally, several configurations can be made available for selection. Here create is configured as an object with key/value pairs, whereby each value corresponds to a configuration as described in variant 1 or variant 2. If an instance is to be created in this representation, the user is given a selection of the designations. The instance is created with the settings of the selected item. The keys "moduleID" and "configuration" cannot be used here.

COLORS IN THE CONFIGURATION LIST

What the individual colors mean in messages and in the configurator can be read under the color codes for messages and configurators. The element presents this information as follows:

COLOR	CONDITION
green	InstanceID is 0, create is defined
red	InstanceID is not 0, create is not defined or the defined InstanceID does not exist
gray	The instance with the ID InstanceID is not configured as specified by create. If create represents a chain, the line is also displayed in gray if a physical parent is configured differently than defined in the chain
white	InstanceID is 0, create is not defined (category)
white	The instance with the ID instanceID is configured as specified by create. If create represents a chain, the line is only displayed in white if all physical parents are configured as defined in the chain

USE THE CONFIGURATOR FORM FIELD IN THE DEVICE SEARCH

The console offers Instances that can be created centrally via the Device Search. This is done by regularly calling GetConfigurationForm for Instances of the type Configurator and Discovery and evaluating the parameter values for configurator fields. Only configurator and discovery instances that contain at least one configurator field are displayed in the Device Search. In this, values can also be empty. If the own instance is to be seen in the Device Search, the following conditions must therefore be met:

The instance must be of the type Configurator or Discovery
The search for devices must be started automatically when executing GetConfigurationForm.
If the search takes longer, values should be left empty in an initial return, but after the search has been completed, the devices that can be created must be included in the return of GetConfigurationForm
The search should not be restarted when GetConfigurationForm is called repeatedly if it is currently running
An example implementation can be found here: SymconTest

SELECTED LINE AS A VARIABLE IN ON-ACTIONS

The use of the values of the configurator in scripts is analogous to Lists. All values are available there, including expanded, instanceID and create. However, the configurator does not save any values in a property.

EXAMPLE

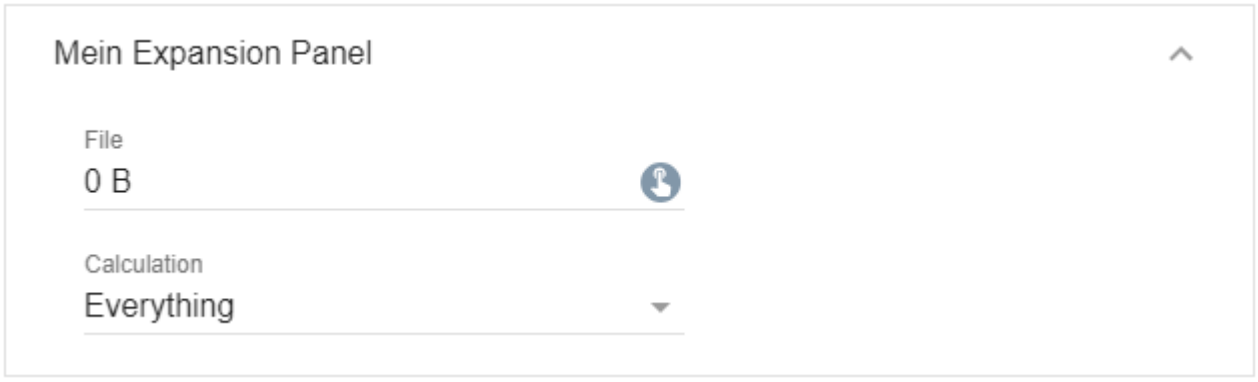
```
// Basic example
{
"type": "Configurator",
"name": "Configuration",
"caption": "Configuration",
"delete": true,
"values": [
{
"id": 1,
"name": "Category",
"address": ""
},{
"parent": 1,
"name": "Calculation module - minimum",
"address": "2",
"create": {
"moduleID": "{A7B0B43B-BEB0-4452-B55E-CD8A9A56B052}",
"configuration": {
"Calculation": 2,
"Variables": "[]"
}
}
},{
"parent": 1,
"name": "Computing module in the living room",
"address": "2",
"create": {
"moduleID": "{A7B0B43B-BEB0-4452-B55E-CD8A9A56B052}",
"configuration": {
"Calculation": 2,
"Variables": "[]"
},
"location": [
"Ground floor",
"Living room"
]
}
},{
"parent": 1,
"instanceID": 53398,
"name": "Incorrect instance",
"address": "4"
},{
"parent": 1,
"name": "Calculation module - selection",
"address": "2",
"create": {
"Maximum": {
"moduleID": "{A7B0B43B-BEB0-4452-B55E-CD8A9A56B052}",
"configuration": {
"Calculation": 3,
"Variables": "[]"
}
},
```

```
"Average": {
"moduleID": "{A7B0B43B-BEB0-4452-B55E-CD8A9A56B052}",
"configuration": {
"Calculation": 4,
"Variables": "[]"
}
}
}
}, {
"parent": 1,
"name": "OZW772 IP-Interface",
"address": "00:A0:03:FD:14:BB",
"create": [
{
"moduleID": "{33765ABB-CFA5-40AA-89C0-A7CEA89CFE7A}",
"configuration": {}
},
{
"moduleID": "{1C902193-B044-43B8-9433-419F09C641B8}",
"configuration": {
"GatewayMode":1
}
},
{
"moduleID": "{82347F20-F541-41E1-AC5B-A636FD3AE2D8}",
"configuration": {
"Host":"172.17.31.95",
"Port":3671,
"Open":true
}
}
]
}
]
}

// Example with renaming of status variables
{
"type": "Configurator",
"values": [{
"name": "Calculation module (renamed status variables)",
"address": "Address",
"create": {
"moduleID": "{A7B0B43B-BEB0-4452-B55E-CD8A9A56B052}",
"configuration": {},
"statusVariables": {
"Minimum": {
"name": "Alternative name for minimum"
},
"Maximum": {
"name": "Alternative name for maximum"
}
}
}
}]
}
```

ExpansionPanel

Require: Symcon >= 5.0
Creates an Expansion Panel, which contains further form fields and can be opened and closed in order to display the content in a space-saving manner.



This item is not supported by the Legacy Console. If a module with this element is opened in the legacy console, all elements contained are displayed directly in the form.

PARAMETERS

PARAMETER	DESCRIPTION
caption	Expansion panel caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onClick script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
expanded (optional)	(**default:** false) If true, the expansion panel is expanded, otherwise it is collapsed (since IP-Symcon 5.2)
items	List of form fields within the expansion panel (cannot be changed)
link (optional)	(**default:** false) If this parameter is true, the output of the onClick script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the button
onClick (optional)	(**default:** "") Script that is executed when you click on the drop-down icon. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.3)
type	ExpansionPanel
visible (optional)	(**default:** true) If true, the expansion panel is visible, otherwise it is invisible (since IP-Symcon 5.2)

PARAMETER	DESCRIPTION
width (optional)	Fixed width of the expansion panel in pixels or % as a string, e.g. "40%" or "250px". If the value is not set or "", the full width is used automatically (since IP-Symcon 5.3)

EXAMPLE

```
// Set up an Expansion Panel
{
"type": "ExpansionPanel",
"caption": "My Expansion Panel",
"items": [
{
"type": "SelectFile",
"name": "File",
"caption": "File"
},
{
"type": "Select",
"name": "Calculation",
"caption": "Calculation",
"options": [
{ "caption": "Everything", "value": 0 },
{ "caption": "Sum", "value": 1 },
{ "caption": "Minimum", "value": 2 },
{ "caption": "Maximum", "value": 3 },
{ "caption": "Average", "value": 4 },
{ "caption": "Count", "value": 5 }
]
}
]
}
```

HorizontalSlider

Creates a horizontal slider that can be moved between the values of "**minimum**" and "**maximum**".



PARAMETERS

PARAMETER	DESCRIPTION
caption	Caption of the slider
displayValue (optional)	(**default:** false) On true, the value of the slider is displayed right of the slider. (since IP-Symcon 8.1)
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the slider can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
maximum (optional, since IP-Symcon 5.3)	(**default:** 100) Maximum value that can be set in the slider
minimum (optional, since IP-Symcon 5.3)	(**default:** 0) Minimum value that can be set in the slider
name (optional)	Name of the slider
onChange (optional, since IP-Symcon 5.3)	(**default:** "") Script which is executed when the controller is moved. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button
prefix (optional)	(**default:** "") Visible caption / unit before the value display (since IP-Symcon 8.1)
stepSize (optional)	(**default:** 1) Step size in which the slider can be adjusted (since IP-Symcon 5.3)
suffix (optional)	(**default:** "") Visible caption / unit after the value display (since IP-Symcon 8.1)
type	Horizontal slider
value (optional)	(**default:** minimum) The value of the slider - If there is an associated property, the parameter in the elements area is overwritten by the property (since IP-Symcon 5.2)

PARAMETER	DESCRIPTION
visible (optional)	(**default:** true) If true, the slider is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the slider in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
// The example returns the InstanceID.
{ "type": "HorizontalSlider", "name": "Slider", "caption": "ID-slider",
"minimum": 0, "maximum": 16, "onChange": "echo $id;" }
```

Image

Require: Symcon >= 5.0
Creates an image that is displayed on the module page. The image can be specified directly or loaded from a media object.



This item is not supported by the Legacy Console. If a module with this element is opened in the legacy console, an error message appears.

PARAMETERS

PARAMETER	DESCRIPTION
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
image (optional)	Content of the image as a Data-URI (More info, cannot be combined with **mediaID**, either **image** or **mediaID** must be set)
link (optional)	If this parameter is true, the output of the onClick script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. If the parameter is not set, a decision is made on the basis of the output whether it is opened as a link or displayed in the dialog (see info box). (since IP-Symcon 6.0)
mediaID (optional)	MediaID of the displayed image or stream (since IP-Symcon 5.5) (cannot be combined with **image**, either **image** or **mediaID** must be set)
name (optional)	Name of the image
type	Image
visible (optional)	(**default:** true) If true, the image is visible, otherwise it is invisible (since IP-Symcon 5.2)
center (optional)	(**default:** false) If **center** is set, the image is centered horizontally within the displayed area, otherwise it is left-justified. This setting has no function within a RowLayout.
width (optional)	Fixed width of the image in pixels or % as a string, e.g. "40%" or "250px". If the value is not set or "", the width is automatically selected based on the image (since IP-Symcon 6.0)
onClick	Script that is executed when the image is clicked. If the script consists of several lines, the individual lines can also be defined as an array. It has the same properties as onClick of the Button. (since IP-Symcon 6.0)



If the parameter onClick contains an "echo" in combination with 'http'/'https', the browser is opened and the URL is called. If there is an "echo" in combination with a 'mailto:', the default mail program is opened with all the parameters set (since IP-Symcon 6.0)

EXAMPLE

```
// Display the image base64-coded
{
"type": "Image",
"image": "
iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDI
}

// Display image from media
{
"type": "Image",
"mediaID": 12345
}
```

IntervalBox (deprecated)



As of 5.0 the IntervalBox is deprecated and will be removed at a future date. The NumberSpinner supports the parameter **suffix** and thus offers all functions of the IntervalBox. Therefore the parameters of the IntervalBox cannot be changed.

Creates an IntervalBox which has a number input field, which has a label/unit behind the input field.
If created in the "elements" area, the IntervalBox sets a property to the entered numerical value when adopted.
The parameter **"name"** defines which property is set.



PARAMETERS

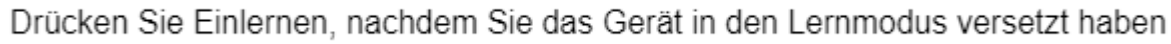
PARAMETER	DESCRIPTION
caption	Visible label/unit behind the input field
type	IntervalBox
name (optional)	Name of the interval box/the property to be set

EXAMPLE

```
{"type": "IntervalBox", "name": "CacheInterval", "caption": "Seconds" }
```

Label

Creates a label that provides an overview.



PARAMETERS

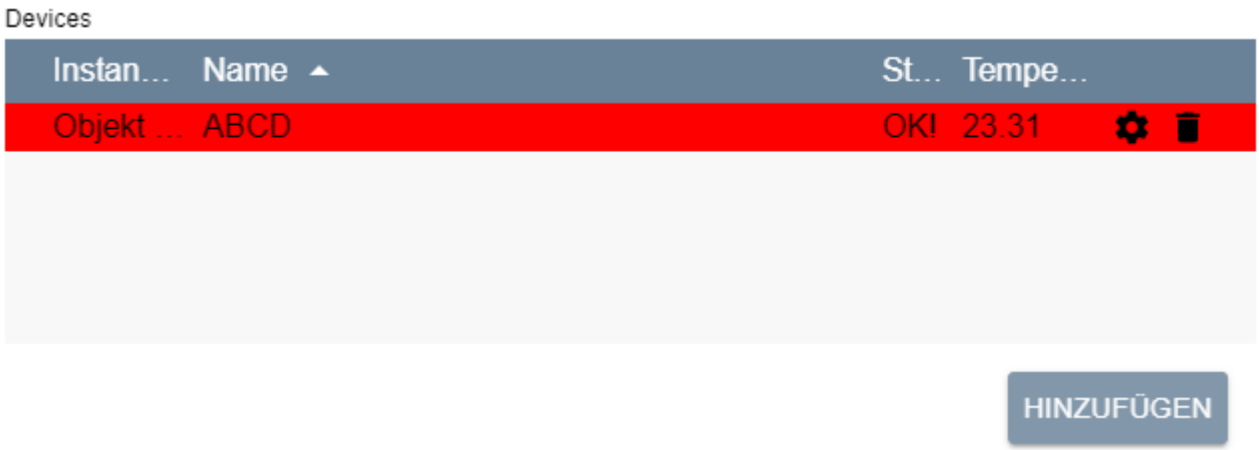
PARAMETER	DESCRIPTION
bold (optional)	(**default:** false) If true, the label is displayed in bold
caption	visible caption
color (optional)	(**default:** 0 (black)) The label is displayed in the specified color
italic (optional)	(**default:** false) If true, the label is displayed in italics
~~label~~	~~visible caption~~ *(outdated, cannot be changed)*
link (optional)	(**default:** true) If this parameter is set, URLs within the label that begin with "http://" or "https://" are displayed as links (since IP-Symcon 5.3)
name (optional)	Name of the label
type	Label
underline (optional)	(**default:** false) If true, the label is displayed underlined
visible (optional)	(**default:** true) If true, the label is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	Fixed width of the label in pixels or % as a string, e.g. "40%" or "250px". If the value is not set or "", the width is automatically selected based on the caption (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "Label", "caption": "Press learn after you have put the
device into learn mode" }
```

List

Require: Symcon >= 4.2
Creates a list with the heading **caption**.
The displayed columns are defined in **columns**.
Initial entries in the list can be specified via **values**.



GENERAL PARAMETERS

PARAMETER	DESCRIPTION
add (optional)	(**default:** false) If true, an "Add" button is displayed below the table, which adds a new element with standard values to the table
caption (optional)	Visible caption of the list
columns	Columns of the list
delete (optional)	(**default:** false) If true, a button is displayed behind each entry in the list, which deletes the corresponding element
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onAdd, onChangeOrder, onDelete, onEdit or onSelectionChanged script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.0)
form (optional)	(**default:** based on the "edit" parameters of columns) An individual form for the dialog for adding and editing rows. The definition is done analogously to the definition of the form for Actions either statically as array of configuration elements or as PHP code, which can generate an individual form for every row. Use cases, sequence and examples are described in the section Individual Forms for Editing (since Symcon 7.0)
link (optional)	(**default:** false) If this parameter is true, the output of the onAdd, onChangeOrder, onDelete, onEdit or onSelectionChanged script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 6.0)

PARAMETER	DESCRIPTION
multiAdd (optional)	(**default:** true, if neither onAdd nor form is set and only a single editable column with an object selection exists, otherwise false) The types SelectObject, SelectCategory, SelectEvent, SelectInstance, SelectLink, SelectMedia, SelectScript or SelectVariable count as object selection. If this parameter is true, adding an entry directly opens the object selection of the first visible column with an object selection, in which multiple objects can be selected at once. An entry is added for every selected object, whose further values receive their add value and can be adjusted via the dialog afterwards, see Adding Multiple Objects. When adding multiple objects, onAdd is still executed only once. If this parameter is false, adding an entry opens the dialog. However, if only a single editable column with an object selection exists and form is not set, the object selection for a single object is opened directly instead. (since Symcon 9.1)
name (optional)	Name of the list / the property to be set
onAdd (optional)	(**default:** "") Script which is executed after adding an entry to the list. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). In this version, the list does not offer the currently selected entry as the variable value but the added entry instead, see Selected line as variable in on-Actions. Otherwise it has the same properties as onClick of the Button (since Symcon 5.3)
onChangeOrder (optional)	(**default:** "") Script which is executed after moving an entry within the list. If the script consists of several lines, the individual lines can also be defined as an array. In this version, the list does not offer the currently selected entry as the variable value but the moved entry instead, see Selected line as variable in on-Actions. Otherwise it has the same properties as onClick of the Button (since Symcon 6.3)
onDelete (optional)	(**default:** "") Script which is executed after deleting an entry in the list. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). In this version, the list does not offer the currently selected entry as a variable value but the deleted entry instead, see Selected line as variable in on-Actions. Otherwise it has the same properties as onClick of the Button (since Symcon 5.3)
onEdit (optional)	(**default:** "") Script which is executed after editing an entry in the list. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). In this version, the list does not offer the currently selected entry as a variable value but the edited entry instead, see Selected line as variable in on-Actions. Otherwise it has the same properties as onClick of the Button (since Symcon 5.3)

PARAMETER	DESCRIPTION
onSelectionChanged (optional)	(**default:** "") Script which is executed after the user has selected another row via click or arrow key. If the script consists of several lines, the individual lines can also be defined as an array. The list offers the newly selected row as the variable value. The script is not executed when the selection is removed or when the values are changed via UpdateFormField, see Reacting to the selection. Otherwise it has the same properties as onClick of the Button (since Symcon 9.1)
rowCount (optional)	(**default:** 0) visible number of lines; If there are more lines in the list, a scroll bar is displayed - if the value is 0, the list fills the remaining space available in the instance editor. Only one configuration item can get all of the remaining space.
sort (optional)	Sort the table. If this is not set, the list remains unsorted
type	List
values (optional)	Initial entries in the list
visible (optional)	(**default:** true) If true, the list is visible, otherwise it is invisible (since Symcon 5.2)
enabled (optional)	(**default:** true) If true, the list can be used, otherwise it is displayed deactivated (since Symcon 6.0)
changeOrder (optional)	(**default:** false) If true, the order of the entries can be changed using drag & drop. The sorting of the list according to columns, however, is not possible in that case. (since Symcon 6.0)
loadValuesFromConfiguration (optional)	(**default:** true) This parameter specifies whether the list should be filled with the values of the property, see Saving and loading lists . If the value is set to false, it is necessary for a successful review in the Module Store to fill the list in the interests of the user, see Review guidelines Point 13 (since Symcon 6.0)

PARAMETERS FOR COLUMNS

PARAMETER	DESCRIPTION
add (optional)	Initial value in this column for a newly added entry, necessary if **add** is set to true in the list. Alternatively, an object can be provided which generates the value automatically, see Parameters for add (since Symcon 9.1)
align (optional)	(**default:** "left") Alignment of the column (since Symcon 5.2) "left": Content is displayed left-aligned "center": Content is displayed centered "right": Content is displayed right-aligned
caption	Visible caption of a column
confirm (optional)	(**default:** "") Query text before an onClick action is performed. The action is only carried out if the query is confirmed with "Yes". If no query text is defined, the action is carried out directly (since Symcon 5.2)

PARAMETER	DESCRIPTION
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onClick script of this column is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.1)
edit (optional)	Description of an editable element
~~label~~	~~Visible caption of a column~~ *(deprecated)*
link (optional)	(**default:** false) If this parameter is true, the output of the onClick script of this column is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 7.1)
name	Unique name of a column to refer to; the name must not be "rowColor", "editable" or "deletable", since these values are used for other functions, see Parameters for values
onClick (optional)	(**default:** "") Script which is executed when a field in this column is clicked. It has the same properties as onClick of the Button. The script is only executed if this field is not empty, i.e. 0, false or "" (since Symcon 5.2)
quickFilter (optional)	(**default:** false) If true, the text of this column is considered during the search via quick filter. Should the value be false for all columns, the quick filter is hidden. (since Symcon 7.2)
sortColumn (optional)	If sortColumn is set and sorted according to this column, the values of the column are compared with the name of this parameter instead of the values of this column
save (optional)	(**default:** true, if the element can be edited or **add** is an object for automatic generation, otherwise false) If true, the values of this column should be saved as module properties, see Save and load lists
visible (optional)	(**default:** true) If false, the column is not displayed
width	Width of the column in pixels, as a CSS string (e.g. "100px"); a single column may have the value "auto", which means that the width of this column uses the remaining available space

PARAMETERS FOR EDIT

PARAMETER	DESCRIPTION
type	Type of representation, according to the possible entries: CheckBox, IntervalBox, NumberSpinner, PasswordTextBox, Select, SelectCategory, SelectColor, SelectEvent, SelectFile, SelectInstance, SelectLink, SelectMedia, SelectObject, SelectScript, SelectVariable, ValidationTextBox, List, Tree
Further parameters	The other parameters depend on the type selected; the parameters "name" and "caption" of the type are ignored



If the "type" List or Tree is used, the value of this column is not JSON-coded again. It is only necessary to parse the value of the higher-level list once. Each sub-list or subtree is then directly available as an object. Similarly, the values in "values" do not have to be coded twice.

The "add" parameter of the column is translated according to the localization, provided the "type" of "edit" is ValidationTextBox or PasswordTextBox or "edit" is not set at all

It is possibly to update only specific parts of columns or values with the function UpdateFormField by using a dot notation. In this case, the target is provided as columns.x.y or values.x.y. x refers to the index of the affected column or value and y refers to the name of the updated field. It is also possible to update a single column or value completely by using columns.x or values.x as target. For example, updating the width of the second column can be done via $this->UpdateFormField('list', 'columns.1.width', '200px');.

PARAMETERS FOR ADD

Instead of a fixed value, the parameter **add** of a column can also be an object which describes the strategy by which the value of a newly added entry is generated automatically (since Symcon 9.1). The value is generated when the entry is added and is treated like a regular value afterwards. The generated value is unique within the column: If a random value matches the value of an existing entry, a new value is generated. Generated values are not translated.

PARAMETER	DESCRIPTION
strategy	Strategy of the generation: "increment": The highest numeric value of this column across all entries of the list is increased by **step**. Non-numeric values are ignored. If there is no numeric value yet, **start** is used or, without **start**, the value of **step**. If **start** is set, a value less than **start** is never generated (with a negative **step** never a greater one) "random-string": A random string of the characters a-z, A-Z and 0-9 with the length **length** "random-integer": A random integer between **min** and **max** "guid": A random GUID (UUID version 4), e.g. "{3F2504E0-4F89-4D3A-9A0C-0305E82C3301}"
step (optional)	(**default:** 1) Step by which the highest value is increased, only for the strategy "increment"; must be a number other than 0
start (optional)	Value of the first entry and lower bound of all following values (upper bound with a negative **step**), only for the strategy "increment"; must be a number
length (optional)	(**default:** 32) Number of characters of the generated string, only for the strategy "random-string"
min (optional)	(**default:** 0) Lower bound (inclusive) of the random integer, only for the strategy "random-integer"; must be less than **max**
max (optional)	(**default:** 2147483647) Upper bound (inclusive) of the random integer, only for the strategy "random-integer"; must be greater than **min**

To keep generated values, columns with such an add value are saved by default, i.e. the parameter "save" is true by default for these columns. If a loaded entry is missing the value of such a column, e.g. because the column was added later, the value is generated by the same strategy when loading. The default value that on-actions receive when no entry is selected is generated by the strategy as well. If no unique value can be generated because all possible values are already in use, e.g. with "random-integer" and a small range between min and max, adding the entry fails with an error message.

```
// Columns with automatically generated values
{
"caption": "ID",
"name": "ID",
"width": "50px",
"add": { "strategy": "increment", "step": 10 }
}, {
"caption": "Token",
"name": "Token",
"width": "auto",
"add": { "strategy": "random-string", "length": 16 },
"edit": { "type": "ValidationTextBox" }
}, {
"caption": "Random number",
"name": "Number",
"width": "100px",
"add": { "strategy": "random-integer", "min": 1, "max": 100 },
"edit": { "type": "NumberSpinner" }
}, {
"caption": "GUID",
"name": "GUID",
"width": "300px",
"add": { "strategy": "guid" }
}
```

The strategy "increment" determines the next value from the current entries. If the entry with the highest value is deleted, its value is assigned again when the next entry is added. If a value should never be reused, e.g. because it serves as a permanent ID towards an external system, the module can remember the highest value ever assigned and specify it via **start**.

```
// Deleted IDs are not assigned again
public function GetConfigurationForm(): string
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);
// The attribute HighestID is updated to the highest value of the list
in ApplyChanges, but never decreases
$form['elements'][0]['columns'][0]['add'] = ['strategy' => 'increment',
'start' => $this->ReadAttributeInteger('HighestID') + 1];
return json_encode($form);
}
```

PARAMETERS FOR SORT

PARAMETER	DESCRIPTION
column	Name of the column by which the list is sorted
direction	The direction of the sort; "ascending" for an ascending order and "descending" for a descending order

PARAMETERS FOR VALUES

PARAMETER	DESCRIPTION
deletable (optional)	(**default:** true) If false, the line cannot be deleted (since Symcon 5.4)
editable (optional)	(**default:** true) If false, the line cannot be edited, although the list itself has editable columns (since Symcon 5.4)
rowColor (optional)	(**default:** transparent) Background color of the line as hex code (default colors see below), empty string for transparency
icons (optional)	(**default:** {}) An object that contains the used icons for each column. Within the object, the key defines the name of the column and the value the name of the used icon. If there is no entry for a column, no icon is shown for that column. (since Symcon 9.0)
Further parameters (optional)	(**default:** "") For each initial entry in the list, the values for all columns are defined here. For this purpose, a parameter with the name of the column is entered for each column. The assignment of the parameter corresponds to the value of this column for the corresponding entry.

COLOR	HEX CODE
Light green	#C0FFC0
Purple	#C0C0FF
Yellow	#FFFFC0
Gray	#DFDFDF
Red	#FFC0C0

What the individual colors mean in messages and in the configurator can be read under the color codes for messages and configurators.
All icons supported by iron-icon are usable: Iron Icons Demo
In addition, the following Symcon specific icons can be used:



SAVING AND LOADING LISTS

When the list is saved, all list elements that can be edited or that have set the **save** parameter are saved as module properties.
If the list is reloaded and the parameter **loadValuesFromConfiguration** is not set to false, the values that were saved as module properties are entered first. If **loadValuesFromConfiguration** is set to false, values are not loaded from the property and only as described in the next steps. If values from the property should be used as basis anyway, they must be provided within the **values**. The other elements, including the color, are filled with the initial values from **values**.
If there are more lines than elements in **values**, elements of the superfluous lines that were not saved as module properties are filled with the initial value **add** of the corresponding column.

SELECTED LINE AS A VARIABLE IN ON-ACTIONS

In the definition of the configuration form, the values of the table can be used as the variable **$Name**. The object can be accessed like an array. If the corresponding object is accessed with a string index, the corresponding value is returned as defined in values of the selected line. If a numeric index is accessed or the object is iterated, the corresponding line is returned, the individual parameters of which can in turn be referenced by a string index.
Example:

```
// Output the name of the currently selected row in the devices table
{ "type": "Button", "caption": "Output", "onClick":
"print_r($devices['name']);" }

// Output the name of the first line of the devices table
```

```
{ "type": "Button", "caption": "Output", "onClick":
"print_r($devices[0]['name']);" }
```

REACTING TO THE SELECTION

Using the parameter **onSelectionChanged**, the module can react when the user selects another row of the list (since Symcon 9.1). Typical use cases are:

- Enabling buttons that refer to the selected row, e.g. "Test device"
- Showing details of the selected row in other form fields, e.g. in a Label
- Loading dependent data, e.g. the channels of the selected device in a second list

The values of the newly selected row are available as described in Selected line as variable in on-Actions via $(Name of the list)['Name of the column'], $id and all other form fields as usual. As the script is not executed within the module, it typically calls a function of the module, which adjusts the other form fields via UpdateFormField.

SITUATION	ONSELECTIONCHANGED
Click on another row	Is executed
Changing the row via arrow key	Is executed
Click on the row that is already selected	Is not executed
Removing the selection, e.g. by clicking on an empty area of the list or deleting the selected row	Is not executed
Adding, editing or moving a row	Is not executed, onAdd, onEdit or onChangeOrder are executed instead
Changing values via UpdateFormField	Is not executed. The selection stays at the same position if a row still exists there, otherwise it is removed

The onClick of a column, on the other hand, is executed on every click on a non-empty field of this column, even if the row is already selected. If a click selects another row, both scripts are executed. For actions that only depend on the selected row and not on the clicked column, onSelectionChanged is therefore the better choice.

Notes:

- As removing the selection does not trigger onSelectionChanged, scripts referring to the selection, e.g. the onClick of a button, should expect that no row is selected. $(Name of the list)['Name of the column'] then returns the value **add** of the column.
- The script is executed on every change, e.g. also when browsing through the list via arrow key. It should therefore be fast. Time-consuming actions belong in a button.
- An output of the script is shown as a message like for the Button.

Example: A list of devices whose details and channels are shown when a device is selected. The button "Test device" is only enabled once a device is selected.

```
// Example: form.json
{
"actions": [
{
"type": "List",
"name": "Devices",
"caption": "Devices",
```

```
"rowCount": 5,
"columns": [
{ "caption": "Name", "name": "Name", "width": "auto", "add": "" },
{ "caption": "Address", "name": "Address", "width": "150px", "add": ""
}
],
"values": [],
"onSelectionChanged": "DEV_UISelectDevice($id, $Devices['Address']);"
},
{ "type": "Label", "name": "DeviceDetails", "caption": "Please select a
device" },
{
"type": "List",
"name": "Channels",
"caption": "Channels",
"rowCount": 3,
"columns": [
{ "caption": "Channel", "name": "Channel", "width": "auto", "add": "" }
],
"values": []
},
{
"type": "Button",
"name": "TestButton",
"caption": "Test device",
"enabled": false,
"onClick": "DEV_UITestDevice($id, $Devices['Address']);"
}
]
}
```

```
// Example: module.php
<?php

declare(strict_types=1);

class DeviceOverview extends IPSModule
{
// In a real module, this list originates e.g. from a search in the
network
private const DEVICES = [
'192.168.1.20' => ['Name' => 'Living room', 'Channels' =>
['Temperature', 'Humidity']],
'192.168.1.21' => ['Name' => 'Basement', 'Channels' => ['Temperature',
'Water level', 'Pump']]
];

public function GetConfigurationForm()
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);
$values = [];
foreach (self::DEVICES as $address => $device) {
$values[] = ['Name' => $device['Name'], 'Address' => $address];
}
$form['actions'][0]['values'] = $values;
return json_encode($form);
```

```
}

// onSelectionChanged of the list Devices: Shows the details of the
selected row
public function UISelectDevice(string $Address)
{
$device = self::DEVICES[$Address] ?? ['Name' => '', 'Channels' => []];
$this->UpdateFormField('DeviceDetails', 'caption', sprintf('%s (%s)
with %d channels', $device['Name'], $Address,
count($device['Channels'])));

$channels = [];
foreach ($device['Channels'] as $channel) {
$channels[] = ['Channel' => $channel];
}
$this->UpdateFormField('Channels', 'values', json_encode($channels));
$this->UpdateFormField('TestButton', 'enabled', true);
}

// onClick of the button "Test device"
public function UITestDevice(string $Address)
{
if ($Address == '') {
echo 'Please select a device';
return;
}
echo Sys_Ping($Address, 1000) ? 'The device responds' : 'The device
does not respond';
}
}
```

ADDING MULTIPLE OBJECTS

If a list has only a single editable column of the type SelectObject, SelectCategory, SelectEvent, SelectInstance, SelectLink, SelectMedia, SelectScript or SelectVariable and **form** is not set, the "Add" button directly opens the object selection instead of the dialog. Additionally, objects can be dropped onto the list via drag & drop. Whether multiple objects can be added at once is defined by **multiAdd**.
If **multiAdd** is explicitly set to true, the object selection is opened directly even if the list has further editable columns or **form** is set (since Symcon 9.1). This is especially useful if the object is the essential part of a row and the further parameters are rather secondary. The user can add many objects in a single step and only needs to edit the rows that should differ from the default afterwards.

ACTION	BEHAVIOR
Add	The object selection of the first visible column whose **edit** is an object selection is opened. A separate row is added for every selected object. All further values of the row receive their add value. onAdd is executed once for all added rows
Drag & drop	The dropped objects are added just like via the object selection. Objects that are not accepted by the column are skipped
Edit	The dialog is opened as usual, so the further values can be adjusted afterwards

If **form** is set, the form must contain an element with the name of this column that offers an object selection, e.g. a SelectVariable. If the form is generated by PHP code, the code is executed before the object selection is opened. Just like when confirming the dialog, all named elements of the form are written into the row, see Hints for Individual Forms. If there is no matching column or the form contains no matching element, an error message is shown and the dialog is opened instead.

```
// Example: Monitor whether variables are updated regularly
// As multiAdd is explicitly true, "Add" directly opens the object
selection, even though the list has further editable columns.
// This is ideal especially if the further parameters are rather
secondary: The default values from add fit most
// variables, so any number of variables can be added in a single step.
// Only single rows that should differ from the default are adjusted
afterwards via the edit dialog.
{
"type": "List",
"name": "Variables",
"caption": "Monitored Variables",
"rowCount": 10,
"add": true,
"delete": true,
"multiAdd": true,
"columns": [{
"caption": "Variable",
"name": "VariableID",
"width": "auto",
"add": 0,
"edit": { "type": "SelectVariable" }
}, {
"caption": "Timeout",
"name": "Timeout",
"width": "100px",
"add": 60,
"edit": { "type": "NumberSpinner", "suffix": "min", "minimum": 1 }
}, {
"caption": "Active",
"name": "Active",
"width": "60px",
"add": true,
"edit": { "type": "CheckBox" }
}]
}
```

INDIVIDUAL FORMS FOR EDITING

By default, adding or editing a row shows a dialog that contains exactly one input field for every column with an **edit** parameter. The parameter **form** of the list defines an arbitrary form for this dialog instead (since Symcon 7.0). The form can contain all configuration elements that are possible within the regular configuration form, e.g. RowLayout, ExpansionPanel, PopupButton, a Select with elements per option or another list. Every named element of the form becomes a value of the row, regardless of whether a column with that name exists.
An individual form is the right choice if

a row has more settings than can reasonably be shown as columns. The list then only shows an overview and all details are edited within the dialog
the input fields depend on the type of the row, e.g. a shutter requires different settings than a dimmer
the input fields require hints, grouping, validation or dependencies between each other (onChange)
a row has nested data, e.g. a list of translations or roles per row
the possible values are only known at runtime, e.g. options that depend on other instances or on the value of the row
single entries of a list that is managed by the module should be edited, e.g. the parameters of a device or access tokens. Instead of a separate PopupButton with a Select to choose the entry, a list with **form** and **onEdit** is sufficient. The user directly clicks the edit icon of the desired row

STATIC FORM OR PHP CODE

The form is defined analogously to the form for Actions:

FORM	BEHAVIOR
Array of configuration elements	The form is the same for all rows. It is created when the dialog is opened for the first time and reused afterwards
PHP code as string or as array of strings	The code is executed every time the dialog is opened and returns the array of configuration elements via return. This way, every row can get its own form, e.g. depending on the type of the row

Just like the onClick script of a Button, the PHP code is executed as a standalone script and not within the module. Therefore, $this is not available and functions of the module are called via their prefix. The following variables are available:

VARIABLE	CONTENT
$id	ID of the instance
$(name of the list)	The list as described in Selected Line as a Variable in on-Actions. A string index returns the value of the edited row, e.g. $Devices['Type']. When adding, it returns the add value of the column instead. All rows are available via a numeric index or foreach
$(name of an element)	The current, possibly not yet applied value of every other named element of the configuration form

The return value can be a PHP array or a form decoded with json_decode. Any other output of the script, e.g. by echo or a PHP warning, invalidates the result. Errors are shown within the dialog instead of the form. As the code is executed every time the dialog is opened, it should be fast.
The following approach, which is also used by the modules integrated in Symcon, has proven itself: The form.json contains the form of the rows statically as a template. GetConfigurationForm replaces it with a short PHP code that calls a public function of the module with the required values of the row, e.g. "return json_decode(SHD_UIMotorForm($id, $Motors['WeatherStationID'] ?? 0), true);". This function loads the template, adjusts it for the row (e.g. options and visibility) and returns it as JSON string, see example 4.

SEQUENCE WHEN EDITING

ACTION	BEHAVIOR
Add	The form is loaded. Every form field for which a column with the same name exists gets the add value of that column. All other fields keep their current value, see hints
Edit	The form is loaded. Every form field gets the value with the same name from the row, both values of columns and values without a column
Confirm	The dialog can only be confirmed if all visible fields are valid, e.g. according to validate of a ValidationTextBox. Afterwards, the value of every named element is written into the row under its name, also from RowLayout, ColumnLayout, ExpansionPanel, PopupButton, the items of a CheckBox and the items of the options of a Select. When editing, all other values of the row are kept, when adding, columns without a matching form field get their add value. Afterwards, onAdd or onEdit is executed
Cancel	The row remains unchanged, onAdd or onEdit is not executed
onAdd, onEdit, onDelete	All values of the row are available via $(name of the list)['Name'], including values of form fields without a column
Apply	Values of columns are saved according to their save parameter. Values without a matching column are always saved
Load	Saved values without a column remain in the row and are available again when editing the next time

If **form** is set, an edit icon is always shown, except for rows with "editable": false. The **edit** parameters of the columns are not used for the dialog, but still define the display within the list, e.g. the caption of the option of a Select or the name of an object, and whether the column is saved by default. Visible columns with a CheckBox as **edit** can still be toggled directly within the list, which executes onEdit without a dialog. Adding directly via the object selection and adding via drag & drop are only available with **form** if **multiAdd** is explicitly set to true, see Adding Multiple Objects.

HINTS FOR INDIVIDUAL FORMS

Columns without **edit** are not saved by default. If such a column is filled via the form, "save": true must be set, as its value is lost when applying otherwise. Values without a column, however, are always saved.
A static form is reused. Fields for which the row, or the columns when adding, provide no value keep the value of the last opened dialog. Changes via UpdateFormField, e.g. of visible or options, are kept as well. Therefore, a column with an add value should exist for every form field, if required invisible with "visible": false and "save": true. Alternatively, the form is generated via PHP code, which sets visibility and options based on the row.
If the form contains a List or a Tree, a column with the same name and "edit": { "type": "List" } or "edit": { "type": "Tree" } should exist, if required invisible. Only then the value is available as array within the row and the property, otherwise as JSON encoded string.
When adding, $(name of the list)['Name'] contains the add value of the column. For values without a column or without an add value, the index is not set. The PHP code should therefore provide a default value via ??, e.g. $Devices['Type'] ?? 0.

Scripts of the elements within the dialog, e.g. onChange, receive $id and the values of the elements of the dialog as variables, but not the list. Via UpdateFormField, the module can adjust the elements of the opened dialog by their name, e.g. options or visibility. Therefore, the names of the elements within the dialog should differ from the names of the other elements of the configuration form.
Confirming the dialog only changes the list within the configuration form. It is saved when applying. If a change should take effect immediately, e.g. be sent to a device, this has to be done in onAdd or onEdit, see example 3.

EXAMPLES FOR INDIVIDUAL FORMS

Example 1: Static form for rows with many settings. The list only shows an overview. The columns "Name" and "Limit" have no **edit** and therefore require "save": true. The settings that are only visible within the dialog have invisible columns, so they are set to their add value when adding.

```
// Example 1: Static form for rows with many settings
{
"type": "List",
"name": "Sensors",
"caption": "Sensors",
"rowCount": 5,
"add": true,
"delete": true,
"columns": [{
"caption": "Active",
"name": "Active",
"width": "60px",
"add": true,
"edit": { "type": "CheckBox" }
}, {
"caption": "Name",
"name": "Name",
"width": "auto",
"add": "",
"save": true
}, {
"caption": "Variable",
"name": "VariableID",
"width": "250px",
"add": 0,
"edit": { "type": "SelectVariable" }
}, {
"caption": "Limit",
"name": "Limit",
"width": "100px",
"add": 25,
"save": true
}, {
"name": "Hysteresis",
"visible": false,
"add": 0.5,
"save": true
}, {
```

```
"name": "Delay",
"visible": false,
"add": 60,
"save": true
}, {
"name": "Message",
"visible": false,
"add": "",
"save": true
}],
"form": [{
"type": "RowLayout",
"items": [
{ "type": "CheckBox", "name": "Active", "caption": "Active" },
{ "type": "ValidationTextBox", "name": "Name", "caption": "Name",
"validate": "^.+$" }
]
}, {
"type": "SelectVariable",
"name": "VariableID",
"caption": "Variable",
"validVariableTypes": [1, 2],
"width": "400px"
}, {
"type": "RowLayout",
"items": [
{ "type": "NumberSpinner", "name": "Limit", "caption": "Limit",
"digits": 1 },
{ "type": "NumberSpinner", "name": "Hysteresis", "caption":
"Hysteresis", "digits": 1, "minimum": 0 }
]
}, {
"type": "ExpansionPanel",
"caption": "Advanced",
"items": [
{ "type": "NumberSpinner", "name": "Delay", "caption": "Delay",
"suffix": "s", "minimum": 0 },
{ "type": "ValidationTextBox", "name": "Message", "caption": "Message
text", "multiline": true, "width": "400px" }
]
}]
}
```

Example 2: Different fields depending on the type of the row, without PHP code (since Symcon 9.1). The fields are attached to the options of the Select and are only shown while the corresponding option is selected. The values of the fields of the other options are kept and are saved within the row as well.

```
// Example 2: Fields depending on the type of the row
{
"type": "List",
"name": "Devices",
"caption": "Devices",
"rowCount": 5,
"add": true,
"delete": true,
```

```
"columns": [{
"caption": "Name",
"name": "Name",
"width": "auto",
"add": "",
"save": true
}, {
"caption": "Type",
"name": "Type",
"width": "150px",
"add": 0,
"edit": {
"type": "Select",
"options": [
{ "caption": "Switch", "value": 0 },
{ "caption": "Dimmer", "value": 1 },
{ "caption": "Shutter", "value": 2 }
]
}
}, {
"caption": "Status variable",
"name": "StatusID",
"width": "250px",
"add": 0,
"edit": { "type": "SelectVariable" }
},
{ "name": "MinimumLevel", "visible": false, "add": 0, "save": true },
{ "name": "RampTime", "visible": false, "add": 1, "save": true },
{ "name": "RunTimeUp", "visible": false, "add": 30, "save": true },
{ "name": "RunTimeDown", "visible": false, "add": 30, "save": true },
{ "name": "Invert", "visible": false, "add": false, "save": true }],
"form": [{
"type": "ValidationTextBox",
"name": "Name",
"caption": "Name"
}, {
"type": "SelectVariable",
"name": "StatusID",
"caption": "Status variable"
}, {
"type": "Select",
"name": "Type",
"caption": "Type",
"options": [{
"caption": "Switch",
"value": 0
}, {
"caption": "Dimmer",
"value": 1,
"items": [
{ "type": "NumberSpinner", "name": "MinimumLevel", "caption": "Minimum
brightness", "suffix": "%", "minimum": 0, "maximum": 100 },
{ "type": "NumberSpinner", "name": "RampTime", "caption": "Dimming
speed", "suffix": "s", "digits": 1, "minimum": 0 }
]
}, {
"caption": "Shutter",
"value": 2,
"items": [{
```

```
"type": "RowLayout",
"items": [
{ "type": "NumberSpinner", "name": "RunTimeUp", "caption": "Run time
up", "suffix": "s", "minimum": 1 },
{ "type": "NumberSpinner", "name": "RunTimeDown", "caption": "Run time
down", "suffix": "s", "minimum": 1 }
]
}, {
"type": "CheckBox",
"name": "Invert",
"caption": "Invert position"
}]
}]
}]
}
```

Example 3: An individual form per row, generated by the module. The list shows the parameters of a device, every row requires a different input element: a CheckBox, a NumberSpinner with limits, a Select with options or a text field. The list is located within **actions** and is therefore not saved as property. Its entries are filled in GetConfigurationForm and every change is applied directly via onEdit.

```
// Example 3: form.json
{
"actions": [{
"type": "List",
"name": "Parameters",
"caption": "Device parameters",
"rowCount": 10,
"columns": [
{ "name": "Ident", "visible": false },
{ "caption": "Parameter", "name": "Caption", "width": "auto" },
{ "caption": "Value", "name": "DisplayValue", "width": "200px" },
{ "name": "Value", "visible": false }
],
"values": [],
"form": "return json_decode(DEV_UIParameterForm($id,
$Parameters['Ident']), true);",
"onEdit": "DEV_WriteParameter($id, $Parameters['Ident'],
$Parameters['Value']);"
}]
}
```

```
// Example 3: module.php
<?php

declare(strict_types=1);

class ParameterDevice extends IPSModule
{
// In a real module, the device provides this definition
private const PARAMETERS = [
'LedMode' => ['caption' => 'LED mode', 'type' => 'enum', 'options' =>
```

```
[0 => 'Off', 1 => 'On', 2 => 'Status']],
'Brightness' => ['caption' => 'Brightness', 'type' => 'number',
'minimum' => 0, 'maximum' => 100, 'suffix' => '%'],
'ChildLock' => ['caption' => 'Child lock', 'type' => 'bool'],
'Label' => ['caption' => 'Label', 'type' => 'string']
];

public function Create()
{
//Never delete this line!
parent::Create();

$this->RegisterAttributeString('ParameterValues', '{"LedMode": 2,
"Brightness": 80, "ChildLock": false, "Label": ""}');
}

public function GetConfigurationForm()
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);
$form['actions'][0]['values'] = $this->GetParameterRows();
return json_encode($form);
}

// Called by the PHP code in "form" and returns the form for the edited
row
public function UIParameterForm(string $Ident): string
{
$parameter = self::PARAMETERS[$Ident];

// The element is named like the column "Value" and therefore
automatically gets the value of the row
$element = ['name' => 'Value', 'caption' => $parameter['caption']];
switch ($parameter['type']) {
case 'bool':
$element['type'] = 'CheckBox';
break;

case 'number':
$element['type'] = 'NumberSpinner';
$element['minimum'] = $parameter['minimum'];
$element['maximum'] = $parameter['maximum'];
$element['suffix'] = $parameter['suffix'];
break;

case 'enum':
$element['type'] = 'Select';
$element['options'] = [];
foreach ($parameter['options'] as $value => $caption) {
$element['options'][] = ['caption' => $caption, 'value' => $value];
}
break;

default:
$element['type'] = 'ValidationTextBox';
break;
}
return json_encode([$element]);
}
```

```
// Called via onEdit after the dialog was confirmed
public function WriteParameter(string $Ident, $Value)
{
// In a real module, the value is sent to the device here
$values = json_decode($this->ReadAttributeString('ParameterValues'),
true);
$values[$Ident] = $Value;
$this->WriteAttributeString('ParameterValues', json_encode($values));

// Update the list, so the formatted value is shown
$this->UpdateFormField('Parameters', 'values', json_encode($this-
>GetParameterRows()));
}

private function GetParameterRows(): array
{
$values = json_decode($this->ReadAttributeString('ParameterValues'),
true);
$rows = [];
foreach (self::PARAMETERS as $ident => $parameter) {
$value = $values[$ident];
switch ($parameter['type']) {
case 'bool':
$display = $value ? 'Yes' : 'No';
break;

case 'number':
$display = $value . ' ' . $parameter['suffix'];
break;

case 'enum':
$display = $parameter['options'][$value];
break;

default:
$display = $value;
break;
}
$rows[] = [
'Ident' => $ident,
'Caption' => $parameter['caption'],
'DisplayValue' => $display,
'Value' => $value
];
}
return $rows;
}
}
```

Example 4: Template within the form.json, which is adjusted for every row via PHP code. The selection of the weather station is filled at runtime and the selection of the sensor depends on the weather station of the row. If another weather station is selected within the dialog, onChange adjusts the sensor selection via UpdateFormField. The column "WeatherStationID" has an **edit** of the type SelectInstance, so the list shows the name of the instance and the column is saved.

```
// Example 4: form.json
{
"elements": [{
"type": "List",
"name": "Motors",
"caption": "Motors",
"rowCount": 5,
"add": true,
"delete": true,
"columns": [{
"caption": "Name",
"name": "Name",
"width": "auto",
"add": "",
"save": true
}, {
"caption": "Position",
"name": "PositionID",
"width": "250px",
"add": 0,
"edit": { "type": "SelectVariable" }
}, {
"caption": "Weather station",
"name": "WeatherStationID",
"width": "250px",
"add": 0,
"edit": { "type": "SelectInstance" }
}, {
"name": "Sensor",
"visible": false,
"add": "",
"save": true
}],
"form": [{
"type": "ValidationTextBox",
"name": "Name",
"caption": "Name"
}, {
"type": "SelectVariable",
"name": "PositionID",
"caption": "Position"
}, {
"type": "Select",
"name": "WeatherStationID",
"caption": "Weather station",
"options": [],
"onChange": "SHD_UIWeatherStationChanged($id, $WeatherStationID);"
}, {
"type": "Select",
"name": "Sensor",
"caption": "Brightness sensor",
"options": [],
"visible": false
}]
}]
}
```

```
// Example 4: module.php
<?php

declare(strict_types=1);

class Shading extends IPSModule
{
// GUID of the module of the weather station
private const WEATHER_STATION_MODULE_ID = '{5C9A1E3B-7D24-4F68-A0B1-
2E3C4D5F6A7B}';

public function Create()
{
//Never delete this line!
parent::Create();

$this->RegisterPropertyString('Motors', '[]');
}

public function GetConfigurationForm()
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);

// The form of the rows depends on the weather station of the row and
is therefore generated every time it is opened
$form['elements'][0]['form'] = "return
json_decode(SHD_UIMotorForm(\$id, \$Motors['WeatherStationID'] ?? 0),
true);";
return json_encode($form);
}

// Called by the PHP code in "form" and returns the adjusted template
public function UIMotorForm(int $WeatherStationID): string
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);
$motorForm = $form['elements'][0]['form'];
foreach ($motorForm as &$element) {
switch ($element['name'] ?? '') {
case 'WeatherStationID':
$element['options'] = $this->GetWeatherStationOptions();
break;

case 'Sensor':
$element['options'] = $this->GetSensorOptions($WeatherStationID);
$element['visible'] = $WeatherStationID !== 0;
break;
}
}
return json_encode($motorForm);
}

// Called via onChange within the dialog and adjusts the elements of
the opened dialog
public function UIWeatherStationChanged(int $WeatherStationID)
{
$this->UpdateFormField('Sensor', 'options', json_encode($this-
>GetSensorOptions($WeatherStationID)));
$this->UpdateFormField('Sensor', 'visible', $WeatherStationID !== 0);
```

```
}

private function GetWeatherStationOptions(): array
{
$options = [['caption' => 'None', 'value' => 0]];
foreach (IPS_GetInstanceListByModuleID(self::WEATHER_STATION_MODULE_ID)
as $instanceID) {
$options[] = ['caption' => IPS_GetName($instanceID), 'value' =>
$instanceID];
}
return $options;
}

private function GetSensorOptions(int $WeatherStationID): array
{
$options = [];
if (($WeatherStationID !== 0) && IPS_InstanceExists($WeatherStationID))
{
foreach (IPS_GetChildrenIDs($WeatherStationID) as $childID) {
if (IPS_VariableExists($childID)) {
$options[] = ['caption' => IPS_GetName($childID), 'value' =>
IPS_GetObject($childID)['ObjectIdent']];
}
}
}
return $options;
}
}
```

Further examples for the use of the form parameter can be found within the Test Repository.

REGISTER AND READ OUT VALUES FROM THE TABLE IN PHP

If the table was defined in the configuration form in the **elements** area, the table can be registered with the RegisterPropertyString function. The table is then saved as a JSON-encoded string.
The ReadPropertyString function can be used to read out the stored string. In a next step the function **json_decode** should be used to convert the read string into an array.
Example:

```
$arrString = $this->ReadPropertyString("devices");
$arr = json_decode($arrString);
```

EXAMPLE

```
// Simple editable list of instances and associated values
// The list initially contains an entry
{
"type": "List",
```

```
"name": "Devices",
"caption": "Devices",
"rowCount": 5,
"add": true,
"delete": true,
"sort": {
"column": "Name",
"direction": "ascending"
},
"columns": [{
"caption": "InstanceID",
"name": "InstanceID",
"width": "75px",
"add": 0,
"edit": {
"type": "SelectInstance"
}
}, {
"caption": "Name",
"name": "Name",
"width": "auto",
"add": ""
}, {
"caption": "State",
"name": "State",
"width": "40px",
"add": "New!"
}, {
"caption": "Temperature",
"name": "Temperature",
"width": "75px",
"add": 20.0,
"edit": {
"type": "NumberSpinner",
"digits": 2
}
}],
"values": [{
"InstanceID": 12435,
"Name": "ABCD",
"State": "OK!",
"Temperature": 23.31,
"rowColor": "#ff0000" //rot
}]
}
```

If the value **InstanceID** of the first entry in the example table is changed to 54321 and **Temperature** to 37.0 and the table is saved, the new values for **InstanceID** and **Temperature** are persistently saved as module properties. Since the other elements, i.e. **name**, **state** and the color of the line **rowColor**, cannot be edited, they are not saved either.
If the table is now reloaded, the saved values for **InstanceID** and **Temperature** are loaded first. The non-editable fields **name** and **state** are then assigned the values from **values** i.e., "ABCD" and "OK!" filled. The color is also loaded from **values**, which gives the line a red background.
If one has also added another line to the table and set the values **InstanceID** and **Temperature** to 314 and -5.3, respectively, then these would also be loaded from the module properties. Since there is only one element in **values**, the non-editable fields are

filled according to the standard values from **add**. The **name** field of the second line would remain empty and **state** would be set to "OK!". Since no color is defined, the background of this line would be transparent.

```
// List with alternative sorting
{
"type": "List",
"name": "sortList",
"rowCount": 5,
"add": false,
"columns": [
{
"caption": "Text",
"name": "Text",
"width": "auto",
"sortColumn": "Number"
},
{
"caption": "Number",
"name": "Number",
"width": "200px",
"visible": false
}
],
"values": [
{
"Text": "One",
"Number": "1"
},
{
"Text": "Two",
"Number": "2"
},
{
"Text": "Three",
"Number": "3"
}
]
}
```

NumberSpinner

Creates an input field for numbers with the heading **caption**.
If created in the "Elements" area, the NumberSpinner sets a property to the entered numerical value when accepted.
The parameter **"name"** defines which property is set.



The digits parameter is used to determine whether the entered number is interpreted as an integer or a float (floating point number).



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption of the NumberSpinner
digits (optional)	(**default:** 0) Indicates the number of decimal places. (Cannot be combined with hex)
enabled (optional)	(**default:** true) If true, the NumberSpinner can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
hex (optional)	(**default:** false) If true, then a hexadecimal input {1..9, A..F} is allowed. (Cannot be combined with digits)
minimum (optional)	(**default:** no minimum value) Defines a minimum value - If the entered value is less than the minimum value, an error is displayed and changes cannot be accepted (since IP-Symcon 5.3)
maximum (optional)	(**default:** no maximum value) Defines a maximum value - If the entered value is greater than the maximum value, an error is displayed and changes cannot be accepted (since IP-Symcon 5.3)
moreDigits (optional)	(**default:** false) If false, the NumberSpinner is limited to the specified number of decimal places. If this is exceeded, an error is displayed and changes cannot be accepted. If true, any number of decimal places can be specified (cannot be combined with hex; since IP-Symcon 6.0)
name (optional)	Name of the NumberSpinner / the property to be set
onChange (optional)	(**default:** "") Script which is executed when the NumberSpinner value is changed. If the script consists of several lines, the individual lines can also be defined as an array. It has the same properties as onClick of the Button. (since IP-Symcon 8.1)
suffix (optional)	(**default:** "") Visible caption / unit behind the input field
type	NumberSpinner

PARAMETER	DESCRIPTION
value (optional)	(**default:** 0) The value of the NumberSpinner - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the NumberSpinner is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
// without digits/hex parameters
{ "type": "NumberSpinner", "name": "Port", "caption": "Port" }
// with digits parameter
{ "type": "NumberSpinner", "name": "Latitude", "caption": "Latitude
(N)", "digits": 2}
// with hex parameters
{ "type": "NumberSpinner", "name": "Value", "caption": "Hexvalue",
"hex": true}
// with suffix
{ "type": "NumberSpinner", "name": "Seconds", "caption": "Seconds",
"suffix": "seconds"}
```

OpenObjectButton

Require: Symcon >= 5.3
A button that opens an object when pressed. The instance configurator is opened for an instance and the script editor for a script. For all other object types, the edit dialog is opened in the object tree.



PARAMETERS

PARAMETERS	DESCRIPTION
caption	Visible caption of the button
enabled (optional)	(**default:** true) If true, the button can be used, otherwise it is displayed as deactivated
name (optional)	Name of the button
objectID	ID of the object to be opened
type	OpenObjectButton
visible (optional)	(**default:** true) If true, the button is visible, otherwise it is invisible
width (optional)	Fixed width of the button in pixels or % as a string, e.g. "40%" or "250px". If the value is not set or "", the width is automatically selected based on the caption

EXAMPLE

```
{ "type": "OpenObjectButton", "caption": "To the Connect Control",
"objectID": 42563 }
```

PasswordTextBox

Creates an input field for passwords with the heading **caption**.
If created in the "Elements" area, the PasswordTextBox sets a property to the entered String when accepted.
The parameter **"Name"** defines which property is set.



PARAMETER

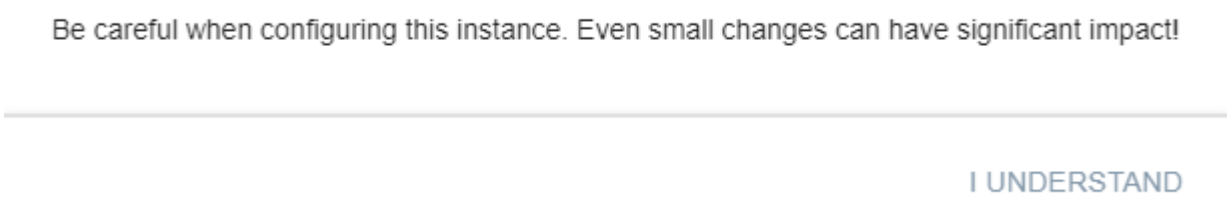
PARAMETERS	DESCRIPTION
caption	Visible caption of the password input field
enabled (optional)	(**default:** true) If true, the PasswordTextBox can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
name (optional)	Name of the password input field/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the PasswordTextBox value is changed. If the script consists of several lines, the individual lines can also be defined as an array. It has the same properties as onClick of the Button. (since IP-Symcon 8.1)
type	PasswordTextBox
validate (optional)	(**default:** "") If the value does not meet the regular expression specified here, an error is displayed and changes cannot be accepted (since IP-Symcon 5.3)
value (optional)	(**default:** "") The value of the PasswordTextBox - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the image is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "PasswordTextBox", "name": "Password", "caption": "Passwort"
}
```

PopupAlert

Require: Symcon >= 5.0
If the configuration form contains this element, it will be opened when the configuration page is loaded. After the popup has been closed, it is also possible to open it again using dynamics (visible to true).



GENERAL PARAMETERS

PARAMETERS	DESCRIPTION
name (optional)	Name of the PopupAlert
popup	Definition of the popup to be opened (parameters can be changed since Symcon 6.1 by linking the parameter names, e.g. popup.closeCaption)
type	PopupAlert
visible (optional)	(**default:** true) If visible is initially true, the popup is opened when the instance configuration is opened, otherwise nothing happens. By changing to true, the popup can be opened afterwards. A change to false closes a currently open popup. (since Symcon 5.2)

PARAMETERS FOR POPUP

PARAMETERS	DESCRIPTION
buttons (optional)	(**default:** []) A list of buttons which is shown in the popup at the bottom right. The close button is always displayed first, followed by the buttons described here in sequence (since Symcon 6.1)
caption (optional)	(**default:** "") Visible title of the popup (since Symcon 8.2)
closeCaption (optional)	(**default:** "OK", "Cancel" if there are several (link: ./#Parameters_for_pages text: pages)) The caption of the close button at the bottom right of the popup

PARAMETERS	DESCRIPTION
initialPage (optional)	(**default:** 0) A reference to the page from pages that is shown when the popup is opened. This can be a number corresponding to the index within the pages list, a string corresponding to the name of the page, or a script returning the page as a number or string. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the defined page does not exist, the first page is shown. A change to initialPage only takes effect the next time the popup is opened. (since Symcon 9.1)
items	List of form fields in the popup (cannot be changed). Alternatively, PHP code can be specified as a string or an array of strings which returns the list of form fields via return (since Symcon 9.1). When using pages, items is the default for pages without their own items and can be omitted
pages (optional)	(**default:** []) A list of pages of the popup that can be linked together to enable the functionality of a wizard. If the list is empty, the regularly defined content is displayed. See Wizard (since Symcon 9.1)
width (optional)	(**default:** "") Initial width of the popup in pixels or % as a string, e.g. "40%" or "250px". Percentages refer to the width of the browser window. If the value is not set or "", the default width is used (since Symcon 9.0)

PARAMETERS FOR PAGES

The pages field contains a list of objects, each describing a page. If the list contains at least two pages, the buttons on the far right additionally show "Previous" and "Next" or "OK", which can be used to switch between pages. The close button is then labeled "Cancel" by default. For the second button, "Next" is displayed if the page determined by nextPage exists, otherwise "OK", which closes the popup. "Previous" is active if the current page was reached via "Next". If a page was left on the way that defines a script for "onConfirm" but none for "onUndo", "Previous" is deactivated and none of the previous pages can be reached again. If the list contains only one page, it is displayed like a regular popup without "Previous" and "Next". Use cases, the exact flow and examples are described in the section Wizard and in detail for the PopupButton. Each of these objects can use the following parameters:

PARAMETER	DESCRIPTION
buttons (optional)	(**default:** buttons of popup) A list of buttons which is shown while this page is active
caption (optional)	(**default:** caption of popup) Visible title of the popup while this page is active
items (optional)	(**default:** items of popup) List of form fields of the page. Alternatively, PHP code can be specified as a string or an array of strings which returns the list of form fields via return. The code is executed when switching to the page and can thus generate content depending on previous inputs. See Values and scripts
name (optional)	A unique name for referencing, typically for nextPage or initialPage

PARAMETER	DESCRIPTION
nextPage (optional)	(**default:** next page) A reference to the page to be shown when "Next" is clicked. This can be a number corresponding to the index within the pages list, a string corresponding to the name of the page, or a script returning the next page as a number or string. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the defined page does not exist, e.g. for -1, "OK" is shown and clicking it closes the popup. A string that does not match the name of a page is executed as a script. An index must be returned by the script as a number, a returned string is always interpreted as a name. The script is executed again when the page is shown and after every change of a value, so the caption "Next" or "OK" always fits. If the script is started again before a previous execution has finished, only the result of the most recently started execution is used
onConfirm (optional)	(**default:** "") Script that is executed when "Next" or "OK" is pressed. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the page change is prevented by validate, onConfirm is not executed. An output of the script is shown as a message, the page change takes place anyway. If onConfirm is defined without onUndo, "Previous" is no longer possible after leaving the page
onUndo (optional)	(**default:** "") Script that is executed when returning to this page via "Previous". If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. It should revert the changes of the onConfirm of this page. The form fields of the page are available with the values they had when the page was left. Only a non-empty onUndo allows returning to a page with onConfirm. "onUndo": "" behaves like a missing onUndo. If there is nothing to revert but "Previous" should remain possible, a comment is sufficient, e.g. "onUndo": "// Nothing to revert"
validate (optional)	(**default:** "") Script that is executed when "Next" or "OK" is pressed, after the form fields of the page are valid, e.g. according to validate of a ValidationTextBox. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the script returns a non-empty string via return, it is displayed as an error, no page change takes place and onConfirm is not executed. The string is translated using the translations of the module. The script must not produce any output, e.g. via echo

The parameters closeCaption and width always apply to the whole popup and cannot be defined per page.

PARAMETERS FOR BUTTONS

PARAMETERS	DESCRIPTION
caption	Visible caption of the button

PARAMETERS	DESCRIPTION
onClick	Script that is executed when the button is clicked. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the script does not produce any output, the popup is closed after the script has been processed. Otherwise the output is shown as a red error message at the bottom of the popup. There is an exception here if the return begins with the prefix "MESSAGE:". In this case the popup is closed and the return is displayed after the prefix in a message dialog.
enabled (optional)	(**default:** true) If true, the button can be used, otherwise it is displayed as deactivated

WIZARD

Using the parameter **pages**, the popup becomes a wizard which guides the user step by step through a process (since Symcon 9.1). The pages of the PopupAlert behave exactly like those of the PopupButton. Use cases, the flow of Next, Previous and Cancel, the available variables, controlling page changes, notes and further examples are described for the PopupButton.

In contrast to the PopupButton, the PopupAlert is opened without a click of the user:

If visible is true when the configuration form is loaded, the wizard is opened immediately. If GetConfigurationForm sets visible depending on the configuration, the wizard appears exactly when the instance still has to be set up.

The module can open the wizard at any time via UpdateFormField using visible, e.g. via a button "Start setup", and close it with false. The wizard then starts on the page defined by initialPage or popup.initialPage. This also applies if the wizard is already open: If the module first sets popup.initialPage and then visible to true, the wizard switches to this page, e.g. via $this->UpdateFormField('SetupWizard', 'popup.initialPage', 'settings') and $this->UpdateFormField('SetupWizard', 'visible', true). The inputs are kept, "Previous" is deactivated afterwards.

SETUP WIZARD

A proven pattern for setting up a module:

The wizard is placed in actions and is made visible in GetConfigurationForm as long as a required property is not set.

The first page welcomes the user and asks via a RadioButtonGroup how the setup should be done. The values of the options correspond to the names of the following pages, so nextPage can return the selection directly. One option allows skipping the setup.

One page shows the result of a search. Its items are generated by the module via PHP code, so the search is executed again every time the page is shown.

The onConfirm of the last settings page writes the values into the fields of the configuration form via UpdateFormField. The user sees the result and saves it with "Apply changes". The onUndo resets the fields to the saved values.

Final pages are placed at the end of the list and are reached via their name: a summary and a note when the setup is skipped. The summary ends the wizard via "nextPage": -1 as the page for skipping still follows.

A button in actions opens the wizard again.

Example: Setup wizard for a device in the network. As long as the property Host is empty, the wizard opens when the instance configuration is opened.

```
// Example: form.json
{
"elements": [
{ "type": "ValidationTextBox", "name": "Host", "caption": "Host" },
{ "type": "NumberSpinner", "name": "Interval", "caption": "Update
interval", "suffix": "seconds", "minimum": 10 }
],
"actions": [
{
"type": "Button",
"caption": "Start setup",
"onClick": "SET_UIOpenWizard($id);"
},
{
"type": "PopupAlert",
"name": "SetupWizard",
"visible": false,
"popup": {
"caption": "Setup",
"width": "600px",
"pages": [
{
"caption": "Welcome",
"items": [
{ "type": "Label", "caption": "This wizard sets up the connection to
the device." },
{
"type": "RadioButtonGroup",
"name": "WizardStart",
"caption": "How should the device be found?",
"value": "search",
"options": [
{ "caption": "Search in the network", "value": "search" },
{ "caption": "Enter address manually", "value": "manual" },
{ "caption": "Skip setup", "value": "skip" }
]
}
],
"nextPage": "return $WizardStart;"
},
{
"name": "search",
"caption": "Devices found",
"items": "return json_decode(SET_UIGetSearchItems($id), true);",
"validate": "if (($WizardFoundHost ?? '') == '') { return 'Please
select a device'; }",
"nextPage": "settings"
},
{
"name": "manual",
"caption": "Address",
"items": [
{ "type": "ValidationTextBox", "name": "WizardManualHost", "caption":
"Host", "validate": "^[a-zA-Z0-9.:-]+$" }
]
},
{
"name": "settings",
```

```
"caption": "Settings",
"items": [
{ "type": "NumberSpinner", "name": "WizardInterval", "caption": "Update
interval", "suffix": "seconds", "value": 60 }
],
"validate": "if ($WizardInterval < 10) { return 'The interval must be
at least 10 seconds'; }",
"onConfirm": "SET_UIApply($id, ($WizardStart == 'search') ?
($WizardFoundHost ?? '') : $WizardManualHost, $WizardInterval);",
"onUndo": "SET_UIRevert($id);",
"nextPage": "summary"
},
{
"name": "summary",
"caption": "Done",
"items": [
{ "type": "Label", "caption": "The settings have been entered in the
configuration form. Please save them with \"Apply changes\"." }
],
"nextPage": -1
},
{
"name": "skip",
"caption": "Setup skipped",
"items": [
{ "type": "Label", "caption": "Without a host, the instance does not
work. The setup can be continued at any time via \"Start setup\"." }
]
}
]
}
}
]
}
```

```
// Example: module.php
<?php

declare(strict_types=1);

class SetupWizardDevice extends IPSModule
{
public function Create()
{
//Never delete this line!
parent::Create();

$this->RegisterPropertyString('Host', '');
$this->RegisterPropertyInteger('Interval', 60);
}

public function GetConfigurationForm()
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);
// The wizard opens automatically as long as the instance is not set up
$form['actions'][1]['visible'] = ($this->ReadPropertyString('Host') ==
```

```
'');
return json_encode($form);
}

// Button "Start setup"
public function UIOpenWizard()
{
$this->UpdateFormField('SetupWizard', 'visible', true);
}

// items of the page "search": Executed every time the page is shown
public function UIGetSearchItems(): string
{
// In a real module, the devices are searched here, e.g. via DNS-SD
$devices = ['192.168.1.20' => 'Device in the living room',
'192.168.1.21' => 'Device in the basement'];

if (count($devices) == 0) {
return json_encode([
['type' => 'Label', 'caption' => 'No device was found. Please go back
and enter the address manually.']
]);
}
$options = [];
foreach ($devices as $host => $name) {
$options[] = ['caption' => $name . ' (' . $host . ')', 'value' =>
$host];
}
return json_encode([
['type' => 'Select', 'name' => 'WizardFoundHost', 'caption' =>
'Device', 'options' => $options, 'value' => $options[0]['value']]
]);
}

// onConfirm of the page "settings": Enters the values in the
configuration form
public function UIApply(string $Host, int $Interval)
{
$this->UpdateFormField('Host', 'value', $Host);
$this->UpdateFormField('Interval', 'value', $Interval);
}

// onUndo of the page "settings": Restores the saved values
public function UIRevert()
{
$this->UpdateFormField('Host', 'value', $this-
>ReadPropertyString('Host'));
$this->UpdateFormField('Interval', 'value', $this-
>ReadPropertyInteger('Interval'));
}
}
```

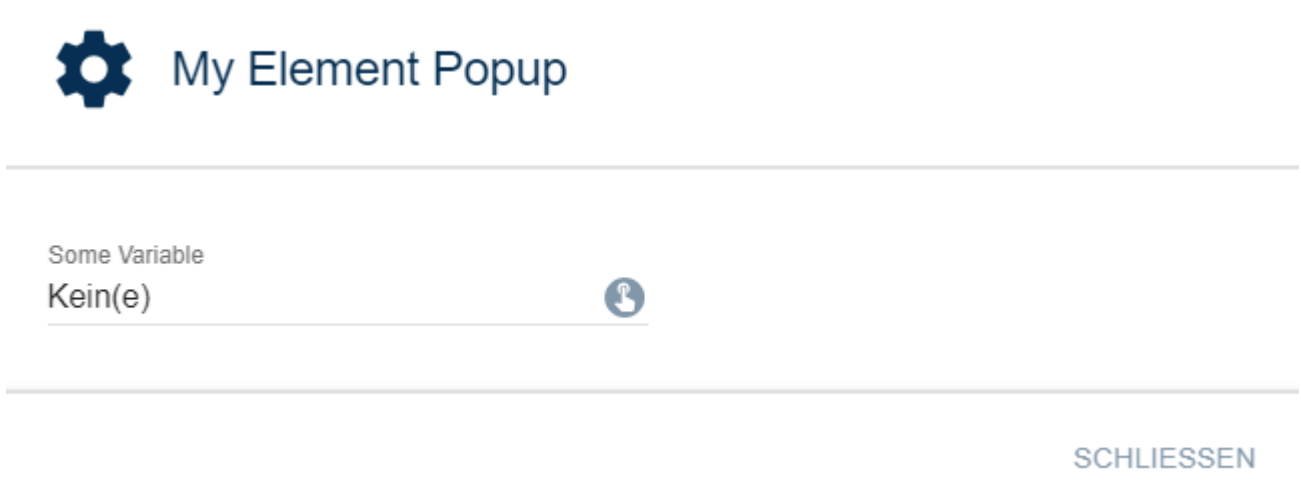
EXAMPLE

```
{
"type": "PopupAlert",
"popup": {
"closeCaption": "I understand",
"items": [
{
"type": "Label",
"caption": "Be careful when configuring this instance. Even small
changes can have significant impact!"
}
]
}
},
{
"type": "PopupAlert",
"popup": {
"items": [
{
"type": "SelectVariable",
"name": "VariableActionTest",
"caption": "Some Action Variable"
}
]
}
},
{
"type": "PopupAlert",
"popup": {
"caption": "A title that is never seen since every page overrides it",
"closeCaption": "Close dialog",
"items": [
{
"type": "Label",
"caption": "This element is never seen since every page has its own
items"
}
],
"pages": [
{
"caption": "Start",
"items": [
{
"type": "Label",
"caption": "Next page: Selection"
}
],
"nextPage": 2
},
{
"caption": "Never visible",
"items": [
{
"type": "Label",
"caption": "This page is never shown since the start page jumps
directly to the page at index 2, the selection"
}
]
},
```

```
{
"caption": "Selection",
"items": [
{
"type": "Select",
"name": "NextPage",
"options": [
{"value": "page1", "caption": "Go to page 1"},
{"value": "page2", "caption": "Go to page 2"},
{"value": "close", "caption": "Done, close"}
],
"caption": "Choose next page"
}
],
"nextPage": "return $NextPage;"
},
{
"name": "page1",
"caption": "Page 1",
"items": [
{
"type": "Label",
"caption": "This is page 1 with 'Undo'"
}
],
"onUndo": "echo 'Undo page 1';",
"onConfirm": "echo 'Confirm page 1';"
},
{
"name": "page2",
"caption": "Page 2",
"items": [
{
"type": "Label",
"caption": "This is page 2, there is no 'Undo'. This is also the last
page of the wizard."
}
],
"onConfirm": "echo 'Confirm page 2';"
}
]
}
}
```

PopupButton

Require: Symcon >= 5.0
Creates a button which, when clicked, opens a popup containing additional form fields.



GENERAL PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption of the button
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onClick script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.0)
enabled (optional)	(**default:** true) If true, the button can be used, otherwise it is displayed as deactivated This parameter has no effect on a possibly opened popup (since Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onClick script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 6.0)
name (optional)	Name of the PopupButton
onClick	Script that is executed when the button is clicked. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since Symcon 5.3)
popup	Definition of the popup to be opened (parameters can be changed since Symcon 6.1 by linking the parameter names, e.g. popup.closeCaption)
type	PopupButton
visible (optional)	(**default:** true) If true, the button is visible, otherwise it is invisible This parameter has no effect on a possibly opened popup (since Symcon 5.2)

PARAMETER	DESCRIPTION
width (optional)	Fixed width of the button in pixels or % as a string, e.g. "40%" or "250px". If the value is not set or "", the width is automatically selected based on the caption (since Symcon 5.2)

PARAMETERS FOR POPUP

PARAMETERS	DESCRIPTION
buttons (optional)	(**default:** []) A list of buttons which is shown in the popup at the bottom right. The close button is always displayed first, followed by the buttons described here in sequence (since Symcon 6.1)
caption (optional)	(**default:** "") Visible title of the popup
closeCaption (optional)	(**default:** "Close", "Cancel" if there are several (link: ./#Parameters_for_pages text: pages)) The caption of the close button at the bottom right of the popup (since version 6.1)
initialPage (optional)	(**default:** 0) A reference to the page from pages that is shown when the popup is opened. This can be a number corresponding to the index within the pages list, a string corresponding to the name of the page, or a script returning the page as a number or string. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the defined page does not exist, the first page is shown. A change to initialPage only takes effect the next time the popup is opened. (since Symcon 9.1)
items	List of form fields in the popup (cannot be changed). Alternatively, PHP code can be specified as a string or an array of strings which returns the list of form fields via return (since Symcon 9.1). When using pages, items is the default for pages without their own items and can be omitted
pages (optional)	(**default:** []) A list of pages of the popup that can be linked together to enable the functionality of a wizard. If the list is empty, the regularly defined content is displayed. See Wizard (since Symcon 9.1)
width (optional)	(**default:** "") Initial width of the popup in pixels or % as a string, e.g. "40%" or "250px". Percentages refer to the width of the browser window. If the value is not set or "", the default width is used (since Symcon 9.0)

PARAMETERS FOR PAGES

The pages field contains a list of objects, each describing a page. If the list contains at least two pages, the buttons on the far right additionally show "Previous" and "Next" or "OK", which can be used to switch between pages. The close button is then labeled "Cancel" by default. For the second button, "Next" is displayed if the page determined by nextPage exists, otherwise "OK", which closes the popup. "Previous" is active if the current page was reached via "Next". If a page was left on the way that defines a script for "onConfirm" but none for "onUndo", "Previous" is deactivated and none of the previous pages can be reached again. If the list contains only one page, it is displayed like a regular popup without "Previous" and "Next". Use cases, the exact flow and examples are described in the section Wizard. Each of these objects can use the following parameters:

PARAMETER	DESCRIPTION
buttons (optional)	(**default:** buttons of popup) A list of buttons which is shown while this page is active
caption (optional)	(**default:** caption of popup) Visible title of the popup while this page is active
items (optional)	(**default:** items of popup) List of form fields of the page. Alternatively, PHP code can be specified as a string or an array of strings which returns the list of form fields via return. The code is executed when switching to the page and can thus generate content depending on previous inputs. See Values and scripts
name (optional)	A unique name for referencing, typically for nextPage or initialPage
nextPage (optional)	(**default:** next page) A reference to the page to be shown when "Next" is clicked. This can be a number corresponding to the index within the pages list, a string corresponding to the name of the page, or a script returning the next page as a number or string. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the defined page does not exist, e.g. for -1, "OK" is shown and clicking it closes the popup. A string that does not match the name of a page is executed as a script. An index must be returned by the script as a number, a returned string is always interpreted as a name. The script is executed again when the page is shown and after every change of a value, so the caption "Next" or "OK" always fits. If the script is started again before a previous execution has finished, only the result of the most recently started execution is used
onConfirm (optional)	(**default:** "") Script that is executed when "Next" or "OK" is pressed. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the page change is prevented by validate, onConfirm is not executed. An output of the script is shown as a message, the page change takes place anyway. If onConfirm is defined without onUndo, "Previous" is no longer possible after leaving the page
onUndo (optional)	(**default:** "") Script that is executed when returning to this page via "Previous". If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. It should revert the changes of the onConfirm of this page. The form fields of the page are available with the values they had when the page was left. Only a non-empty onUndo allows returning to a page with onConfirm. "onUndo": "" behaves like a missing onUndo. If there is nothing to revert but "Previous" should remain possible, a comment is sufficient, e.g. "onUndo": "// Nothing to revert"
validate (optional)	(**default:** "") Script that is executed when "Next" or "OK" is pressed, after the form fields of the page are valid, e.g. according to validate of a ValidationTextBox. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the script returns a non-empty string via return, it is displayed as an error, no page change takes place and onConfirm is not executed. The string is translated using the translations of the module. The script must not produce any output, e.g. via echo

The parameters closeCaption and width always apply to the whole popup and cannot be defined per page.

PARAMETERS FOR BUTTONS

PARAMETERS	DESCRIPTION
caption	Visible caption of the button
onClick	Script that is executed when the button is clicked. If the script consists of several lines, the individual lines can also be defined as an array. No PHP-tags are required. It has the same properties as onClick of the Button. If the script does not produce any output, the popup is closed after the script has been processed. Otherwise the output is shown as a red error message at the bottom of the popup. There is an exception here if the return begins with the prefix "MESSAGE:". In this case the popup is closed and the return is displayed after the prefix in a message dialog.
enabled (optional)	(**default:** true) If true, the button can be used, otherwise it is displayed as deactivated

WIZARD

Using the parameter **pages**, a popup becomes a wizard which guides the user step by step through a process (since Symcon 9.1). Each page only shows the form fields that are relevant for the current step. Inputs can be checked per page, the order of the pages can depend on previous inputs and actions are only executed when the user confirms a step. Pages are available for the PopupButton and the PopupAlert alike.

USE CASES

A wizard is useful if

- a module has to be set up after it has been created. A PopupAlert can open the setup wizard automatically as long as the instance is not yet configured, see Setup wizard
- a configuration consists of several steps and later steps depend on previous answers, e.g. address and port for a network connection, but an interface for a serial connection
- inputs should be checked before continuing, e.g. whether a device can be reached at the entered address
- devices are searched or data is imported and the result is shown on a separate page for selection
- a summary of all inputs should be shown before applying them
- options should be requested and a confirmation obtained before an action with serious consequences, e.g. resetting a device. For a plain confirmation, a popup with buttons is sufficient

Individual settings without a fixed order, however, belong in a regular popup or directly in the configuration form.

FLOW

ACTION	BEHAVIOR
Open	The page determined by initialPage is shown, by default the first page. For the PopupButton, onClick is executed at the same time. "Previous" is deactivated

ACTION	BEHAVIOR
Next or OK	1. The form fields of the current page are checked, e.g. validate of a ValidationTextBox. 2. validate of the page is executed. If it returns an error, the page stays. 3. onConfirm of the page is executed. 4. If the page determined by nextPage exists, it is shown, otherwise the popup is closed
Previous	The page last left via "Next" is shown again. Its form fields get the values they had when the page was left, and its onUndo is executed. validate and onConfirm of the current page are not executed
Cancel	The popup is closed. No script is executed and onConfirm scripts that have already been executed are not reverted
Open again	The wizard starts at initialPage again, "Previous" is deactivated. The inputs of the last run are kept until the configuration form is reloaded

The inputs in the wizard are not saved automatically. The result is typically applied in one of the following ways:

The onConfirm of the last page sets the properties directly, e.g. via IPS_SetProperty and IPS_ApplyChanges or via a function of the module, see example 1 and 2.

The onConfirm writes the values into the fields of the configuration form via UpdateFormField. The user sees the values and saves them as usual with "Apply changes", see the example for the PopupAlert.

Each step directly executes an action, e.g. creating an instance. Then onUndo should revert this action, see example 3.

VALUES AND SCRIPTS

The scripts validate, onConfirm, onUndo, nextPage and initialPage as well as PHP code in items are executed like the onClick of a Button as a separate script and not within the module. $this is therefore not available, functions of the module are called via their prefix. The following variables are available:

VARIABLE	CONTENT
$id	ID of the instance
$(Name of an element)	The current value of every named form field of all pages of the wizard as well as of the other form fields in the same area (elements or actions). Form fields of statically defined pages are available before their page has been shown and then have their initial value. Form fields that are generated via PHP code in items are only available after their page has been shown

SCRIPT	RESULT
validate	A non-empty string via return prevents the page change and is displayed as an error
nextPage, initialPage	Index as a number or name of the page as a string via return
items	Array of form fields via return
onConfirm, onUndo	An output, e.g. via echo, is shown as a message. A return value is ignored

For validate, nextPage, initialPage and items, any output, e.g. via echo or caused by a PHP warning, makes the result invalid and is displayed as an error.
Using UpdateFormField, the module can adjust form fields of any page, even if it is not currently shown. The change becomes visible as soon as the page is shown. This way, e.g. the onChange of a Select on the first page can set the options of a Select on the second page, see example 3.

CONTROLLING PAGE CHANGES

Without nextPage, the next page of the list follows. On the last page, "OK" is shown.
With an index or name in nextPage, a fixed page is shown next, e.g. several branches lead back to the same summary via "nextPage": "summary".
For a branch, a script returns the name of the next page. The easiest way is to use the names of the pages as values of the options of a RadioButtonGroup or Select, e.g. "nextPage": "return $WizardConnection;". As the script is executed after every change of a value, it should be fast and not change anything. Time-consuming checks, e.g. a connection to the device, belong in validate.
If a page should end the wizard although further pages follow in the list, "nextPage": -1 is set. Instead of "Next", "OK" is then shown, which closes the popup. A script in nextPage can return -1 as well. This is typical for final pages at the end of the list which are reached via their name, e.g. a summary and a note when the setup is skipped.
The current page of an open wizard cannot be changed directly via UpdateFormField. If the module should decide about the path, nextPage calls a function of the module, e.g. "nextPage": "return WIZ_UIGetNextPage($id, $WizardModel);". Using popup.initialPage, the module can define the start page for the next opening. A PopupAlert can additionally be opened via UpdateFormField using visible. If it is already open, it thereby starts again on the page defined via popup.initialPage, see PopupAlert.

NOTES ON WIZARDS

The names of the form fields must be unique across all pages. Form fields with the same name on different pages share one value. The names should also differ from the properties and the other form fields, e.g. by the prefix "Wizard".
The user can cancel at any time. Permanent changes, e.g. creating an instance or sending to a device, should therefore be made in the onConfirm of the last page if possible. If they are made earlier, onUndo should revert them. However, no onUndo is executed when cancelling.
An onConfirm without onUndo prevents returning to all previous pages. This is useful if a step cannot be reverted.
Inputs are kept when the popup is opened again. If the wizard should start empty every time, the values can be reset e.g. in the onClick of the PopupButton via UpdateFormField or the form can be reloaded via ReloadForm.
The list pages cannot be changed via UpdateFormField. If the number or the content of the pages depends on the configuration, the pages are generated in GetConfigurationForm or their items via PHP code.
There is no progress indicator. A title like "Step 2 of 3" in the caption of the page shows the user where they are.

WIZARD EXAMPLES

Example 1: Linear wizard with three steps which sets up a limit monitoring. The first page checks via validate whether a variable was selected, the second page uses the validation of the ValidationTextBox. The last page generates a summary from the inputs of the previous pages via PHP code. "OK" sets the properties VariableID, Limit and Message, which the module registers in Create.

```
// Example 1: Linear wizard, e.g. in actions
{
"type": "PopupButton",
"caption": "Set up limit monitoring",
"popup": {
"caption": "Set up limit monitoring",
"width": "600px",
"pages": [
{
"caption": "Step 1 of 3: Variable",
"items": [
{ "type": "Label", "caption": "Which variable should be monitored?" },
{ "type": "SelectVariable", "name": "WizardVariableID", "caption":
"Variable", "validVariableTypes": [1, 2] }
],
"validate": "if (!IPS_VariableExists($WizardVariableID)) { return
'Please select a variable'; }"
},
{
"caption": "Step 2 of 3: Limit",
"items": [
{ "type": "NumberSpinner", "name": "WizardLimit", "caption": "Limit",
"digits": 1 },
{ "type": "ValidationTextBox", "name": "WizardMessage", "caption":
"Message when exceeded", "validate": "^.+$" }
]
},
{
"caption": "Step 3 of 3: Summary",
"items": [
"return [",
" ['type' => 'Label', 'caption' => 'Variable: ' .
IPS_GetLocation($WizardVariableID)],",
" ['type' => 'Label', 'caption' => 'Limit: ' . $WizardLimit],",
" ['type' => 'Label', 'caption' => 'Message: ' . $WizardMessage],",
" ['type' => 'Label', 'caption' => 'OK activates the monitoring.']",
"];"
],
"onConfirm": [
"IPS_SetProperty($id, 'VariableID', $WizardVariableID);",
"IPS_SetProperty($id, 'Limit', $WizardLimit);",
"IPS_SetProperty($id, 'Message', $WizardMessage);",
"IPS_ApplyChanges($id);"
]
}
]
}
}
```

Example 2: Branch depending on a selection. The values of the RadioButtonGroup correspond to the names of the pages, so nextPage can return the selection directly. Both branches lead to the same summary via "nextPage": "summary". The summary only evaluates the fields of the selected branch. "OK" sets the properties Connection, Host, Port and SerialPortID.

```
// Example 2: Branch, e.g. in actions
{
"type": "PopupButton",
"caption": "Set up connection",
"popup": {
"caption": "Set up connection",
"pages": [
{
"caption": "Connection type",
"items": [
{
"type": "RadioButtonGroup",
"name": "WizardConnection",
"caption": "How is the device connected?",
"value": "network",
"options": [
{ "caption": "Network", "value": "network" },
{ "caption": "Serial interface", "value": "serial" }
]
}
],
"nextPage": "return $WizardConnection;"
},
{
"name": "network",
"caption": "Network",
"items": [
{ "type": "ValidationTextBox", "name": "WizardHost", "caption": "Host",
"validate": "^[a-zA-Z0-9.:-]+$" },
{ "type": "NumberSpinner", "name": "WizardPort", "caption": "Port",
"value": 502, "minimum": 1, "maximum": 65535 }
],
"nextPage": "summary"
},
{
"name": "serial",
"caption": "Serial interface",
"items": [
{ "type": "SelectInstance", "name": "WizardSerialPortID", "caption":
"Serial Port", "validModules": ["{6DC3D946-0D31-450F-A8C6-
C42DB8D7D4F1}"] }
],
"validate": "if (!IPS_InstanceExists($WizardSerialPortID)) { return
'Please select a serial port'; }",
"nextPage": "summary"
},
{
"name": "summary",
"caption": "Summary",
"items": [
"if ($WizardConnection == 'network') {",
```

```
" $text = 'Network: ' . $WizardHost . ':' . $WizardPort;",
"} else {",
" $text = 'Serial: ' . IPS_GetName($WizardSerialPortID);",
"}",
"return [['type' => 'Label', 'caption' => $text]];"
],
"onConfirm": [
"IPS_SetProperty($id, 'Connection', $WizardConnection);",
"IPS_SetProperty($id, 'Host', $WizardHost);",
"IPS_SetProperty($id, 'Port', $WizardPort);",
"IPS_SetProperty($id, 'SerialPortID', $WizardSerialPortID);",
"IPS_ApplyChanges($id);"
]
}
]
}
}
```

Example 3: Adding a device, where the content is determined by the module. The vendors are filled in by GetConfigurationForm. When a vendor is selected, the module sets the models on the second page via UpdateFormField. validate of the second page checks via the module whether the device can be reached. onConfirm creates the instance and onUndo deletes it again if the user returns to this page via "Previous". The third page is generated by the module based on the created instance. Only "OK" completes the setup. If the user cancels on the third page, the created instance remains.

```
// Example 3: form.json
{
"actions": [
{
"type": "PopupButton",
"caption": "Add device",
"popup": {
"caption": "Add device",
"pages": [
{
"caption": "Step 1 of 3: Vendor",
"items": [
{
"type": "Select",
"name": "WizardVendor",
"caption": "Vendor",
"options": [],
"onChange": "WIZ_UIUpdateModels($id, $WizardVendor);"
}
],
"validate": "if ($WizardVendor == '') { return 'Please select a
vendor'; }"
},
{
"caption": "Step 2 of 3: Device",
"items": [
{ "type": "Select", "name": "WizardModel", "caption": "Model",
"options": [] },
{ "type": "ValidationTextBox", "name": "WizardHost", "caption": "IP
address" }
```

```
],
"validate": "return WIZ_UIValidateDevice($id, $WizardModel,
$WizardHost);",
"onConfirm": "WIZ_UICreateDevice($id, $WizardModel, $WizardHost);",
"onUndo": "WIZ_UIDeleteDevice($id);"
},
{
"caption": "Step 3 of 3: Finish",
"items": "return json_decode(WIZ_UIGetFinishItems($id), true);",
"validate": "if (trim($WizardName) == '') { return 'Please enter a
name'; }",
"onConfirm": "WIZ_UIFinish($id, $WizardName);"
}
]
}
}
]
}
```

```
// Example 3: module.php
<?php

declare(strict_types=1);

class DeviceWizard extends IPSModule
{
// GUID of the module whose instances are created
private const DEVICE_MODULE = '{5A1C8F2E-3B7D-4C9A-9E61-2F4D8B0C7A13}';

// In a real module, this list comes e.g. from a file with device
templates
private const VENDORS = [
'vendorA' => ['caption' => 'Vendor A', 'models' => ['A100' => 'Sensor
A100', 'A200' => 'Sensor A200']],
'vendorB' => ['caption' => 'Vendor B', 'models' => ['B10' => 'Meter
B10']]
];

public function Create()
{
//Never delete this line!
parent::Create();

$this->RegisterAttributeInteger('WizardInstanceID', 0);
}

public function GetConfigurationForm()
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);
$options = [['caption' => 'Please select', 'value' => '']];
foreach (self::VENDORS as $vendor => $data) {
$options[] = ['caption' => $data['caption'], 'value' => $vendor];
}
$form['actions'][0]['popup']['pages'][0]['items'][0]['options'] =
$options;
return json_encode($form);
```

```
}

// onChange on page 1: Sets the models on page 2, even if it is not
shown yet
public function UIUpdateModels(string $Vendor)
{
$options = [];
foreach (self::VENDORS[$Vendor]['models'] ?? [] as $model => $caption)
{
$options[] = ['caption' => $caption, 'value' => $model];
}
$this->UpdateFormField('WizardModel', 'options',
json_encode($options));
$this->UpdateFormField('WizardModel', 'value', $options[0]['value'] ??
'');
}

// validate on page 2: An empty string allows the page change
public function UIValidateDevice(string $Model, string $Host): string
{
if ($Model == '') {
return 'Please select a model';
}
if (!filter_var($Host, FILTER_VALIDATE_IP)) {
return 'Please enter a valid IP address';
}
if (!Sys_Ping($Host, 1000)) {
return 'The device does not respond';
}
return '';
}

// onConfirm on page 2
public function UICreateDevice(string $Model, string $Host)
{
$instanceID = IPS_CreateInstance(self::DEVICE_MODULE);
IPS_SetName($instanceID, $Model);
IPS_SetProperty($instanceID, 'Model', $Model);
IPS_SetProperty($instanceID, 'Host', $Host);
IPS_ApplyChanges($instanceID);
$this->WriteAttributeInteger('WizardInstanceID', $instanceID);
}

// onUndo on page 2: Reverts UICreateDevice
public function UIDeleteDevice()
{
$instanceID = $this->ReadAttributeInteger('WizardInstanceID');
if (IPS_InstanceExists($instanceID)) {
IPS_DeleteInstance($instanceID);
}
$this->WriteAttributeInteger('WizardInstanceID', 0);
}

// items on page 3: Executed when switching to the page
public function UIGetFinishItems(): string
{
$instanceID = $this->ReadAttributeInteger('WizardInstanceID');
return json_encode([
[
```

```
'type' => 'Label',
'caption' => sprintf('The device was created as instance #%d.',
$instanceID)
],
[
'type' => 'ValidationTextBox',
'name' => 'WizardName',
'caption' => 'Name',
'value' => IPS_GetName($instanceID)
]
]);
}

// onConfirm on page 3
public function UIFinish(string $Name)
{
$instanceID = $this->ReadAttributeInteger('WizardInstanceID');
IPS_SetName($instanceID, $Name);
$this->WriteAttributeInteger('WizardInstanceID', 0);
}
}
```

EXAMPLE

```
{
"type": "PopupButton",
"caption": "Open Popup",
"popup": {
"caption": "My Element Popup",
"items": [
{
"type": "SelectVariable",
"name": "VariableTest",
"caption": "Some Variable"
}
]
}
},
{
"type": "PopupButton",
"caption": "Start Wizard",
"popup": {
"caption": "A title that is never seen since every page overrides it",
"closeCaption": "Close dialog",
"items": [
{
"type": "Label",
"caption": "This element is never seen since every page has its own
items"
}
],
"pages": [
{
"caption": "Start",
"items": [
```

```
{
"type": "Label",
"caption": "Next page: Selection"
}
],
"nextPage": 2
},
{
"caption": "Never visible",
"items": [
{
"type": "Label",
"caption": "This page is never shown since the start page jumps
directly to the page at index 2, the selection"
}
]
},
{
"caption": "Selection",
"items": [
{
"type": "Select",
"name": "NextPage",
"options": [
{"value": "page1", "caption": "Go to page 1"},
{"value": "page2", "caption": "Go to page 2"},
{"value": "close", "caption": "Done, close"}
],
"caption": "Choose next page"
}
],
"nextPage": "return $NextPage;"
},
{
"name": "page1",
"caption": "Page 1",
"items": [
{
"type": "Label",
"caption": "This is page 1 with 'Undo'"
}
],
"onUndo": "echo 'Undo page 1';",
"onConfirm": "echo 'Confirm page 1';"
},
{
"name": "page2",
"caption": "Page 2",
"items": [
{
"type": "Label",
"caption": "This is page 2, there is no 'Undo'. This is also the last
page of the wizard."
}
],
"onConfirm": "echo 'Confirm page 2';"
}
]
```

```
}
}
```

ProgressBar

Require: Symcon >= 5.2
Creates a progress bar.
The progress bar is helpful to inform a user about a possibly longer process by using UpdateFormField.
It is possible to display either a progressive bar or an indefinite bar that shows a loading animation.



PARAMETERS

PARAMETER	DESCRIPTION
caption (optional)	Visible caption of the progress bar
current (optional)	(**default:** 0) The current value of the progress bar - If the bar is displayed progressively, it is unfilled if the current value corresponds to the minimum value of the progress bar, completely filled if the current value corresponds to the maximum value of the progress bar and otherwise accordingly proportionately filled.
indeterminate (optional)	(**default:** false) If true, the progress bar is indeterminate and shows an animation, otherwise the bar is displayed progressively on the basis of current, minimum and maximum
maximum (optional)	(**default:** 100) The maximum value of the progress bar
minimum (optional)	(**default:** 0) The minimum value of the progress bar
name (optional)	Name of the progress bar
type	ProgressBar
visible (optional)	(**default:** true) If true, the button is visible, otherwise it is invisible
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{
"type": "ProgressBar",
"name": "IndeterminateProgress",
"indeterminate": true,
"caption": "loading..."
},
```

```
{
"type": "ProgressBar",
"name": "DeterminateProgress",
"minimum": 0,
"maximum": 50,
"current": 20,
"caption": "20 / 50"
}
```

QrCode

Require: Symcon >= 7.0
Creates a QR Code.



PARAMETER

PARAMETER	DESCRIPTION
caption (optional)	(**default:** "") visible caption
center (optional)	(**default:** false) If true the QR Code is displayed in the middle, otherwise left-aligned
description	(**default:** true) If true, the text contained in the QR Code will be displayed below it
name (optional)	name of the QR Code/property to set
source	The string encoded in the QR Code
type	QrCode
visible (optional)	(**default:** true) If true the QR Code is visible, otherwise it is invisible

EXAMPLE

```
{ "type": "QrCode", "name": "Example", "caption": "Example text",
```

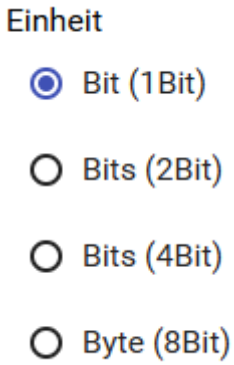
```
"source": "An example text" }
```

RadioButtonGroup

Require: Symcon >= 9.1
Creates a group of radio buttons with the heading **caption**.
If created in the "elements" area, the radio buttons set a property to the selected value **value** when accepted.
The parameter **"name"** defines which property is set.

With MultiSelect it is important to ensure that all value arrays contain the same names

If RadioButtonGroup is used within lists, there is no MultiSelect.



PARAMETER

PARAMETER	DESCRIPTION
caption	Visible caption of the group of radio buttons
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter.
enabled (optional)	(**default:** true) If true, the group can be used, otherwise it is displayed as deactivated
itemsPosition (optional)	(**default:** 0) The position of the configuration elements defined by "items" of the options 0: The elements are displayed below the respective radio button 1: The elements are displayed to the right of the respective radio button
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form.
name (optional)	Name of the group/the property to be set

PARAMETER	DESCRIPTION
options	Array containing all options
onChange (optional)	(**default:** "") Script which is executed when a radio button is selected. If the script consists of several lines, the individual lines can also be defined as an array. It has the same properties as onClick of the Button
type	RadioButtonGroup
value (optional)	(**default:** first defined option) The value of the selected radio button - If there is an associated property, this parameter is overwritten by the property in the elements area
visible (optional)	(**default:** true) If true, the group is visible, otherwise it is invisible
width (optional)	(**default:** 300px) Fixed width of the group in pixels or % as a string, e.g. "40%" or "250px"

PARAMETERS FOR OPTIONS

PARAMETER	DESCRIPTION
caption	Caption of a radio button
enabled (optional)	(**default:** true) If true, the radio button is active and can be selected, otherwise it is shown as inactive and cannot be selected
items (optional)	(**default:** []) List of configuration elements which are only displayed while this radio button is selected. The values of the elements of the other radio buttons are retained
value	Value that is transferred when selected. Can also contain another array, which transfers the values to several properties.

EXAMPLE

```
//Simple value transfer
//The "DeviceType" property is set to 0..3
//The group has 4 radio buttons
{ "type": "RadioButtonGroup", "name": "DeviceType", "caption": "Unit",
"options": [
{ "caption": "Bit (1Bit)", "value": 0 },
{ "caption": "Bits (2Bit)", "value": 1 },
{ "caption": "Bits (4Bit)", "value": 2 },
{ "caption": "Byte (8Bit)", "value": 3 }
]
}

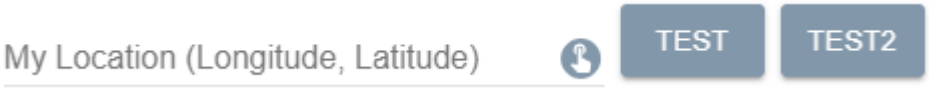
//Multiple value transfer (MultiSelect)
//"name" of the RadioButtonGroup (here: "OutputDevice") must be set,
but is not used to set the values.
//Properties "DeviceName", "DeviceChannels" and "DeviceNum" are set.
{ "type": "RadioButtonGroup", "name": "OutputDevice", "caption":
"Output device",
"options": [
{"caption" : "No sound",
```

```
"value": [
{"name": "DeviceName", "value": "No sound"},
{"name": "DeviceChannels", "value": 0},
{"name": "DeviceNum", "value": 0}
]
},
{"caption" : "Headphones",
"value": [
{"name": "DeviceName", "value": "Headphones"},
{"name": "DeviceChannels", "value": 2},
{"name": "DeviceNum", "value": 1}
]
},
{"caption" : "Speaker",
"value": [
{"name": "DeviceName", "value": "Stereo speakers"},
{"name": "DeviceChannels", "value": 2},
{"name": "DeviceNum", "value": 2}
]
}
]
}

//Configuration elements of individual radio buttons
//The elements of the radio button "Network" are only displayed while
it is selected
{ "type": "RadioButtonGroup", "name": "ConnectionType", "caption":
"Connection",
"options": [
{ "caption": "USB", "value": 0 },
{ "caption": "Network", "value": 1,
"items": [
{ "type": "ValidationTextBox", "name": "Host", "caption": "Host" },
{ "type": "NumberSpinner", "name": "Port", "caption": "Port" }
]
}
]
}
```

RowLayout

Require: Symcon >= 5.0
Creates a row in which form fields are displayed side by side.



PARAMETER

PARAMETER	DESCRIPTION
halign (optional)	(**default:** "left") Horizontal alignment of the elements within the row (since Symcon 9.1) "left": The elements are aligned to the left "center": The elements are centered "right": The elements are aligned to the right "between": The elements are distributed over the width, the first element on the left and the last one on the right
items	List with configuration elements within the row (can be changed since IP-Symcon 6.0)
name (optional)	Name of the row
type	RowLayout
valign (optional)	(**default:** "middle") Vertical alignment of the elements within the row, e.g., if the elements have different heights. For "top" and "bottom", the row uses the full available height (since Symcon 9.1) "top": The elements are aligned to the top "middle": The elements are aligned to the middle "bottom": The elements are aligned to the bottom
visible (optional)	(**default:** true) If true, the row is visible, otherwise it is invisible (since IP-Symcon 5.2)

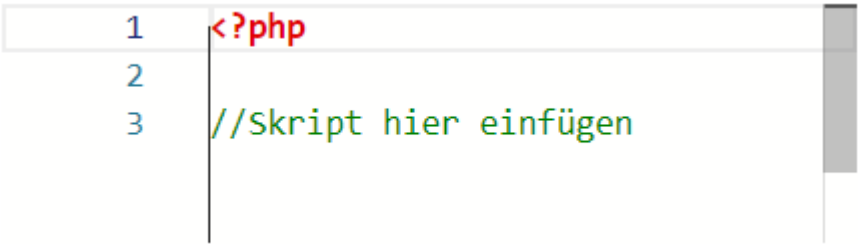
EXAMPLE

```
// row layout
{
"type": "RowLayout",
"items": [
{
"type": "SelectLocation",
"name": "Location",
"caption": "My Location"
},{
"type": "Button",
"caption": "Test",
```

```
"onClick": "TM_Update($id);"
},{
"type": "Button",
"caption": "Test2",
"onClick": "TM_Update2($id);"
}
]
}
```

ScriptEditor

Require: Symcon >= 6.0
Creates a script editor with the heading **caption**.
If created in the "elements" area, the ScriptEditor sets a property on transfer to the entered string.
The parameter **"name"** defines which property is set.



PARAMETERS

PARAMETER	DESCRIPTION
caption (optional)	(**default:** "") Visible caption of the input field
enabled (optional)	(**default:** true) If true, the button can be used, otherwise it is displayed as deactivated
name (optional)	Name of the input field/the property to be set
type	ScriptEditor
value (optional)	(**default:** Placeholder script with PHP tag) The value of the input field - If there is an associated property, this parameter is overwritten by the property in the elements area
visible (optional)	(**default:** true) If true, the input field is visible, otherwise it is invisible
width (optional)	(**default:** 100%) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px",
rowCount (optional)	(**default:** 5) Number of lines displayed in the editor. If the script itself contains more lines, a scroll bar is displayed

EXAMPLE

```
{ "type": "ScriptEditor", "name": "Script" }
```

Select

Creates a drop-down menu with the heading **caption**.
If created in the "elements" area, the drop-down menu sets a property to the selected value **value** when accepted.
The parameter **"name"** defines which property is set.

With MultiSelect it is important to ensure that all value arrays contain the same names

If Select is used within lists, there is no MultiSelect.



PARAMETER

PARAMETER	DESCRIPTION
caption	Visible caption of the drop-down menu
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.0)
enabled (optional)	(**default:** true) If true, the drop-down menu can be used, otherwise it is displayed as deactivated (since Symcon 5.2)
itemsPosition (optional)	(**default:** 0) The position of the configuration elements defined by "items" of the options (since Symcon 9.1) 0: The elements are displayed below the drop-down menu 1: The elements are displayed to the right of the drop-down menu
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 6.0)
name (optional)	Name of the drop-down menu/the property to be set
options	Array containing all options
onChange (optional)	(**default:** "") Script which is executed when the dropdown value is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported from version 6.0). It has the same properties as onClick of the Button. (since Symcon 5.2)
type	Select

PARAMETER	DESCRIPTION
value (optional)	(**default:** first defined option) The value of the drop-down menu - If there is an associated property, this parameter is overwritten by the property in the elements area (since Symcon 5.2)
visible (optional)	(**default:** true) If true, the drop-down menu is visible, otherwise it is invisible (since Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since Symcon 5.3)

PARAMETERS FOR OPTIONS

PARAMETER	DESCRIPTION
caption	Caption of an option
enabled (optional)	(**default:** true) If true, the option is active and can be selected, otherwise it is shown as inactive and cannot be selected (since Symcon 9.1)
items (optional)	(**default:** []) List of configuration elements which are only displayed while this option is selected. The values of the elements of the other options are retained (since Symcon 9.1)
~~label~~	~~Caption of an option~~ *(deprecated)*
value	Value that is transferred when selected. Can also contain another array, which transfers the values to several properties.

EXAMPLE

```
//Simple value transfer
//The "DeviceType" property is set to 0..3
//The drop-down menu has 4 entries
{ "type": "Select", "name": "DeviceType", "caption": "Unit",
"options": [
{ "caption": "Bit (1Bit)", "value": 0 },
{ "caption": "Bits (2Bit)", "value": 1 },
{ "caption": "Bits (4Bit)", "value": 2 },
{ "caption": "Byte (8Bit)", "value": 3 }
]
}

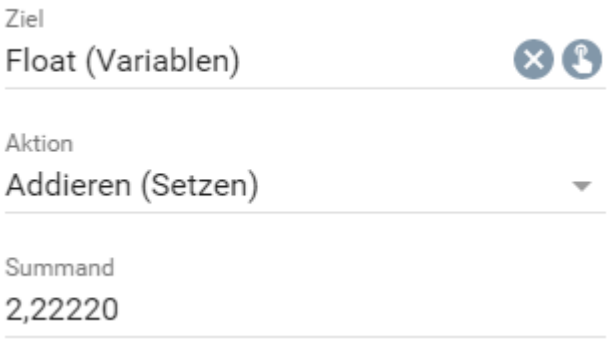
//Multiple value transfer (MultiSelect)
//"name" of the Select (here: "OutputDevice") must be set, but is not
used to set the values.
//Properties "DeviceName", "DeviceChannels" and "DeviceNum" are set.
{ "type": "Select", "name": "OutputDevice", "caption": "Output device",
"options": [
{"caption" : "No sound",
"value": [
{"name": "DeviceName", "value": "No sound"},
{"name": "DeviceChannels", "value": 0},
{"name": "DeviceNum", "value": 0}
]
```

```
},
{"caption" : "Headphones",
"value": [
{"name": "DeviceName", "value": "Headphones"},
{"name": "DeviceChannels", "value": 2},
{"name": "DeviceNum", "value": 1}
]
},
{"caption" : "Speaker",
"value": [
{"name": "DeviceName", "value": "Stereo speakers"},
{"name": "DeviceChannels", "value": 2},
{"name": "DeviceNum", "value": 2}
]
}
]
}

//Configuration elements of individual options
//The elements of the option "Network" are only displayed while it is
selected
{ "type": "Select", "name": "ConnectionType", "caption": "Connection",
"options": [
{ "caption": "USB", "value": 0 },
{ "caption": "Network", "value": 1,
"items": [
{ "type": "ValidationTextBox", "name": "Host", "caption": "Host" },
{ "type": "NumberSpinner", "name": "Port", "caption": "Port" }
]
}
]
}
```

SelectAction

Require: Symcon >= 6.0
Creates an input for an action .



If created in the "elements" area, the property **name** is set to a JSON-coded object, which contains the following parameters:

PARAMETER	DESCRIPTION
actionID	Action ID
parameters	Action parameters

PARAMETERS

PARAMETER	DESCRIPTION
caption (optional)	(**default:** "") Visible caption (since IP-Symcon 7.1)
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated
environment (optional)	(**default:** "Default") Defines the environment of the action selection, here it can be determined which types of actions are available for selection, see also Actions
highestPriorityOnly (optional)	(**default:** false) If false, the element has a selection of all suitable actions, otherwise the action with the highest priority is automatically selected
includeDefaultEnvironment (optional)	(**default:** true) If false, only actions that support the environment *environment* are offered for selection; if true, all actions of the "Default" environment are also offered
name (optional)	Name of the selection/the property to be set
targetID (optional)	(**default:** -2) With -2 the element has a target selection, otherwise the target with this ID is set as the target, where -1 stands for general actions
type	SelectAction
value (optional)	(**default:** no action selected) The value of the selection - If there is an associated property, this parameter will be overwritten by the property in the elements area

PARAMETER	DESCRIPTION
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px"
saveTarget (optional)	(**default:** true) If true, the target is saved as TARGET as part of the parameters
saveEnvironment (optional)	(**default:** true) If true, the environment is saved as ENVIRONMENT as part of the parameters
saveParent (optional)	(**default:** true) If true, the InstanceID is saved as PARENT as part of the parameters

EXAMPLE 1: TARGET, ENVIRONMENT AND INSTANCEID ARE CONSTANTLY SAVED AS PART OF THE PROPERTY

By default, all required parameters are saved as a return from SelectAction. This is the easiest way to perform the action.

```
{ "type": "SelectAction", "name": "PropertyAction", "targetID": 12345 }
// PHP code in the module
$action = json_decode($this->ReadPropertyString('PropertyAction'),
true);
IPS_RunAction($action['actionID'], $action['parameters']);
```

EXAMPLE 2: TARGET, ENVIRONMENT AND INSTANCEID ARE SET DYNAMICALLY

If the additional parameters are not saved as a return, they have to be added manually. In this way, for example, actions can be exchanged between entities and no redundant data is stored.

```
{ "type": "SelectAction", "name": "PropertyAction", "targetID": 12345,
"saveTarget": false, "saveEnvironment": false, "saveParent": false }
// PHP code in the module
$action = json_decode($this->ReadPropertyString('PropertyAction'),
true);
$parameters = $action['parameters'];
$parameters['TARGET'] = 12345;
$parameters['ENVIRONMENT'] = 'Default';
$parameters['PARENT'] = $this->InstanceID;
IPS_RunAction($action['actionID'], $parameters);
```

SelectCategory

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected category when accepted.



PARAMETER

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the category selection can be used, otherwise it is shown deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the category selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the category selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button. (since IP-Symcon 5.2)
type	SelectCategory
value (optional)	(**default:** 1) The value of the category selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the category selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectCategory", "name": "PropertyCategoryID", "caption":
"Target" }
```

SelectColor

Require: Symcon >= 4.2
Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the value as hex string of the selected color when accepted.
If the value is deleted, it is set to transparent (value = -1, `COLOR_TRANSPARENT`).



PARAMETERS

PARAMETER	DESCRIPTION
allowTransparent (optional)	(**default:** true) If set, the selection can be deleted and thereby set to "transparent". Otherwise the value cannot be deleted and therefore cannot be set to "transparent".
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.0)
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated (since Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 6.0)
name (optional)	Name of the selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the color is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button. (since Symcon 5.2)
transparentCaption	(**default:** "Transparent") Caption that is shown when the selection is set to "transparent", i.e., -1 (since Symcon 8.2)
type	SelectColor
value (optional)	(**default:** -1 if allowTransparent, otherwise 0) The value of the selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since Symcon 5.2)
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible (since Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the color selection in pixels or % as a string, e.g. "40%" or "250px" (since Symcon 5.3)

EXAMPLE

```
{ "type": "SelectColor", "name": "HexColorProperty", "caption": "Color"
}
```

SelectCondition

Require: Symcon >= 6.1
Creates an input for conditions.



If created in the "elements" area, the property **name** is set to a JSON-coded list when accepted, whereby each element represents an object with the following parameters:

PARAMETER	DESCRIPTION	
id	Condition ID	
parentID	Parent condition ID	
operation	Operation for linking the individual rules	
	0 `EVENT_CONDITION_OPERATION_AND`	AND
	1 `EVENT_CONDITION_OPERATION_OR`	OR
	2	NAND (not yet implemented)
	3	NOR (not yet implemented)
rules	Object that describes the rules of the condition	

RULES

PARAMETER	DESCRIPTION
variable	List of the variable rules
date	List of date rules
time	List of time rules
dayOfTheWeek	List of weekday rules

REPRESENTATION OF A RULE

PARAMETER	DESCRIPTION
id	Unique ID of the rule within the rule list
comparison	Comparison operator with which the current value is compared with the comparison value 0 EVENTCONDITIONCOMPARISON_EQUAL = 1 EVENTCONDITIONCOMPARISON_NOTEQUAL ≠ 2 EVENTCONDITIONCOMPARISON_GREATER > 3 EVENTCONDITIONCOMPARISON_GREATEROREQUAL ≥ 4 EVENTCONDITIONCOMPARISON_SMALLER < 5 EVENTCONDITIONCOMPARISON_SMALLEROREQUAL ≤
value	The comparison value, the type depends on the type of rule *Variable*: If "type" = 0, this is the absolute comparison value that matches the type of the variable; if "type" = 1, this is a variable ID whose value is used for comparison *Date*: The value is saved as an object with the fields "year", "month" and "day", analogous to the representation of IPS_SetEventConditionDateRule *Time*: The value is saved as an object with the fields "hour", "minute" and "second", analogous to the representation of IPS_SetEventConditionTimeRule *Weekday*: The comparison day is saved as a number (1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday, 7: Sunday)
variableID (only with variable rules)	The ID of the variable that contains the current value
type (only with variable rules)	If 0 the variable is compared with an absolute value, if 1 it is compared with the value of another variable



Instead of evaluating a configured condition independently, the IPS_IsConditionPassing function should be used. There the value of the SelectCondition can be used directly as a parameter.



If the saved condition contains nested conditions or "multi" is set to false and the condition contains several rules, the rule is displayed as "complex" and cannot be edited.

PARAMETERS

PARAMETER	DESCRIPTION
caption (optional)	(**default:** "") Visible caption (since IP-Symcon 7.1)

PARAMETER	DESCRIPTION
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated
multi (optional)	(**default:** false) If false, an individual rule is defined by the element, if true an entire condition with any number of rules
name (optional)	Name of the selection/the property to be set
type	SelectCondition
value (optional)	(**default:** no condition) The value of the selection - If there is an associated property, this parameter will be overwritten by the property in the elements area
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible
width (optional)	(**default:** 300px, if multi = false, otherwise 100%) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px"

EXAMPLE

```
{ "type": "SelectCondition", "name": "PropertyCondition", "multi": true
}
```

SelectDate

Require: Symcon >= 5.0
Creates an input for a date with the caption **caption**.



If created in the "elements" area, the property **name** is set to a JSON-coded object, which contains the following parameters:

PARAMETER	DESCRIPTION
day	Day of the date
month	Month of the date
year	Year of the date

If no time was selected, all parameters are set to 0.

PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
name (optional)	Name of the selection/the property to be set
type	SelectDate
value (optional)	(**default:** no time selected) The value of the selection - If there is an associated property, this parameter will be overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectDate", "name": "PropertyDate", "caption": "date" }
```

SelectDateTime

Require: Symcon >= 5.0
Creates an input for a date and time with the caption **caption**.



If created in the "elements" area, the property **name** is set to a JSON-coded object, which contains the following parameters:

PARAMETER	DESCRIPTION
day	Day of the date
hour	Hours of time
minute	Minutes of time
month	Month of the date
second	Seconds of time
year	Year of the date

If no time was selected, all parameters are set to 0.

PARAMETERS

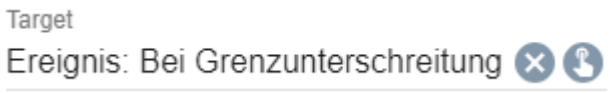
PARAMETER	DESCRIPTION
caption	Visible caption
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
name (optional)	Name of the selection/the property to be set
type	SelectDateTime
value (optional)	(**default:** no time selected) The value of the selection - If there is an associated property, this parameter will be overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectDateTime", "name": "PropertyDateTime", "caption":
"Date + Time" }
```

SelectEvent

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected event when it is accepted.



PARAMETER

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the event selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the event selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.2)
type	SelectEvent
value (optional)	(**default:** 0) The value of the event selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the event selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the event selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectEvent", "name": "PropertyEventID", "caption": "Target"
}
```

SelectFile

Require: Symcon >= 4.2
Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the value of the selected file when accepted. This means that the content of the selected file is saved as a Base64 encoded string.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the file selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
extensions (optional)	Allowed file extensions. Separated by a comma and introduced with a period Example: .jpg,.gif,.txt
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the file selection/the string property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the file selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.2)
type	SelectFile
value (optional)	(**default:** 0) The value of the file selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the file selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default** 300px) Fixed width of the file selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectFile", "name": "PropertyStringFile", "caption":
"Target", "extensions": ".jpg,.gif,.txt" }
```

SelectIcon

Require: Symcon >= 6.1
Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the name of the selected icon when accepted.
If the value is deleted, it is set to an empty string.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated
name (optional)	Name of the selection/the property to be set
type	SelectIcon
value (optional)	(**default:** "") The value of the selection - If there is an associated property, this parameter is overwritten by the property in the elements area
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible
width (optional)	(**default:** 300px) Fixed width of the icon selection in pixels or % as a string, e.g. "40%" or "250px"

EXAMPLE

```
{ "type": "SelectIcon", "name": "IconProperty", "caption": "Icon" }
```

SelectInstance

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected instance when accepted.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the instance selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the instance selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the instance selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.2)
type	SelectInstance
validModules (optional)	(**default:** []) The selection is restricted to instances that have one of the defined moduleIDs. If the list is empty, all instances are allowed. If the selected instance does not correspond to any of the moduleIDs, an error is displayed and changes cannot be accepted. (since IP-Symcon 5.5)
value (optional)	(**default:** 1) The value of the instance selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the instance selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the instance selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectInstance", "name": "PropertyInstanceID", "caption":
"Target" }
```

SelectLink

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected link when accepted.



PARAMETER

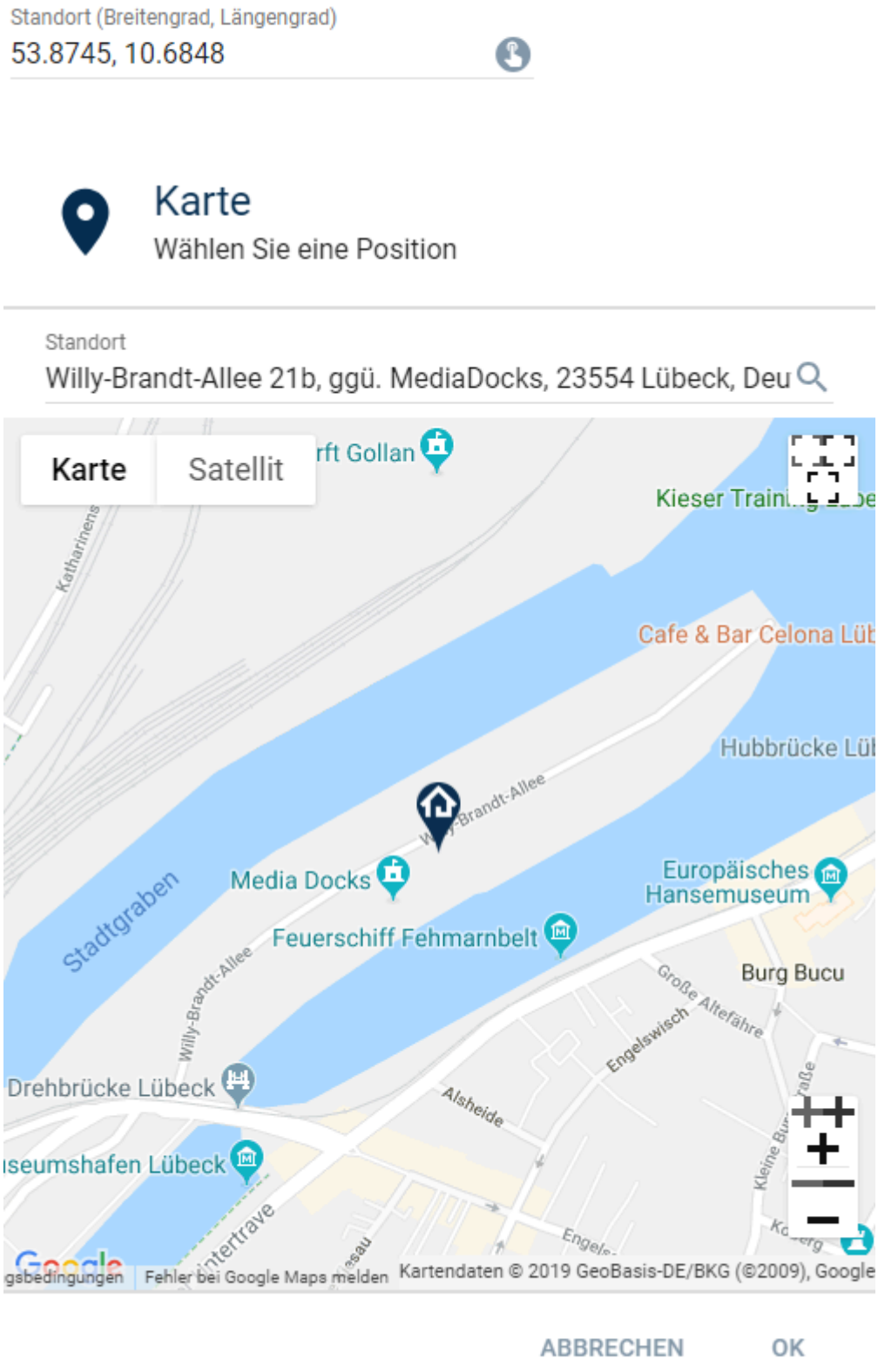
PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the link selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the link selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.2)
type	SelectLink
value (optional)	(**default:** 1) The value of the link selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the link selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the color selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectLink", "name": "PropertyLinkID", "caption": "Target" }
```

SelectLocation

Require: Symcon >= 5.0
Creates an input for a position with the caption **caption**. Here, the position can be selected via map if internet access is available. Alternatively, the longitude and latitude can also be entered manually.



If created in the "elements" area, the property **name** is set to a JSON-coded object, which contains the following parameters:

PARAMETER	DESCRIPTION
latitude	Latitude of the position
longitude	Longitude of the position

If no position was selected, all parameters are set to 0.

PARAMETERS

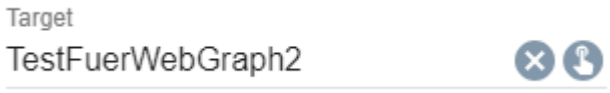
PARAMETER	DESCRIPTION
caption	Visible caption
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
name (optional)	Name of the selection/the property to be set
type	SelectLocation
value (optional)	(**default:** no position selected) The value of the selection - If there is an associated property, this parameter will be overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectLocation", "name": "Location", "caption": "Home" }
```

SelectMedia

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected medium when accepted.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the media selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the media selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the category selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.2)
type	SelectMedia
value (optional)	(**default:** 1) The value of the media selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the media selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the media selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectMedia", "name": "PropertyMediaID", "caption": "Target"
}
```

SelectModule

Require: Symcon >= 5.5
Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected instance when accepted.



PARAMETERS

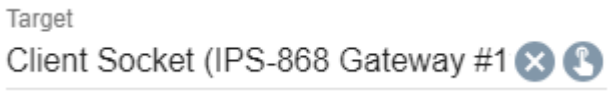
PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the module selection can be used, otherwise it is displayed as deactivated
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
moduleID	The ID of the module whose instances are to be offered for selection
name (optional)	Name of the module slection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the category selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button
type	SelectModule
value (optional)	(**default:** 1) The value of the module selection - If there is an associated property, this parameter is overwritten by the property in the elements area
visible (optional)	(**default:** true) If true, the module selection is visible, otherwise it is invisible
width (optional)	(**default:** 300px) Fixed width of the module selection in pixels or % as a string, e.g. "40%" or "250px"

EXAMPLE

```
{ "type": "SelectModule", "name": "PropertyModuleInstanceID",
"caption": "VoIP Module" , "moduleID": "{A4224A63-49EA-445F-8422-
22EF99D8F624}"}
```

SelectObject

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected object when accepted.



PARAMETER

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the object selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the object selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.2)
type	SelectObject
value (optional)	(**default:** 1) The value of the object selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the object selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of object selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectObject", "name": "ObjectID", "caption": "Target" }
```

SelectProfile

Require: Symcon >= 5.5
Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the name of the selected profile when accepted.



PARAMETERS

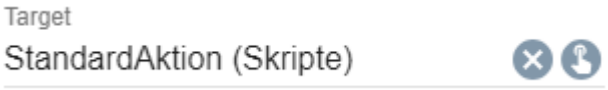
PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the profile selection can be used, otherwise it is displayed as deactivated
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the profile selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button
profileType (optional)	(**default:** -1) The type of profile which can be selected -1 All 0 `VARIABLETYPE_BOOLEAN` Boolean 1 `VARIABLETYPE_INTEGER` Integer 2 `VARIABLETYPE_FLOAT` Float 3 `VARIABLETYPE_STRING` String
type	SelectProfile
value (optional)	(**default:** "") The value of the profile selection - If there is an associated property, this parameter is overwritten by the property in the elements area
visible (optional)	(**default:** true) If true, the profile selection is visible, otherwise it is invisible
width (optional)	(**default:** 300px) Fixed width of the profile selection in pixels or % as a string, e.g. "40%" or "250px"

EXAMPLE

```
{ "type": "SelectProfile", "name": "PropertyProfileName", "caption":
"Profil - Integer" , "profileType": 1}
```

SelectScript

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected script when accepted.



PARAMETER

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the script selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the script selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the script selection is changed. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported since version 6.0). It has the same properties as onClick of the Button (since IP-Symcon 5.2)
type	SelectScript
value (optional)	(**default:** 1) The value of the script selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the script selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the script selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectScript", "name": "RXObjectID", "caption": "Target" }
```

SelectTime

Require: Symcon >= 5.0
Creates an input for a time with the caption **caption**.



If created in the "elements" area, the property **name** is set to a JSON-coded object, which contains the following parameters:

PARAMETER	DESCRIPTION
hour	Hours of time
minute	Minutes of time
second	Seconds of time

If no time was selected, all parameters are set to 0.

PARAMETERS

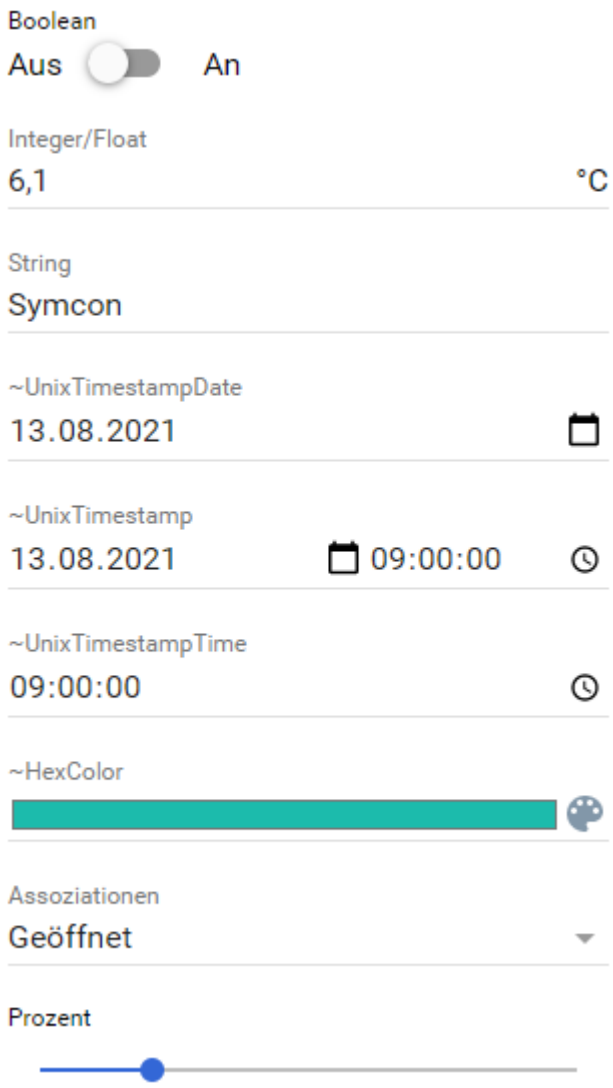
PARAMETER	DESCRIPTION
caption	Visible caption
enabled (optional)	(**default:** true) If true, the selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
name (optional)	Name of the selection/the property to be set
type	SelectTime
value (optional)	(**default:** no time selected) The value of the selection - If there is an associated property, this parameter will be overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "SelectTime", "name": "PropertyTime", "caption": "Time" }
```

SelectValue

Require: Symcon >= 6.3
Creates a selection dialog that displays a suitable form field based on the type and profile of the selected **variableID**.
If created in the "elements" area, the property **name** is set to the currently selected value when it is accepted.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
enabled (optional)	(**default:** true) If true, the value selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
name (optional)	Name of the value selection/the property to be set

PARAMETER	DESCRIPTION
onChange (optional)	(**default:** "") Script which is executed when the value of the valve selection is changed. It has the same properties as onClick of the Button
type	SelectValue
value (optional)	(**default:** null) The JSON encoded value of the choice. If there is an associated property, this parameter is overwritten by the property in the elements area
visible (optional)	(**default:** true) If true, the value selection is visible, otherwise it is invisible
variableID	(**default:** 1) The variable that is used to select the element
width (optional)	(**default:** 300px) Fixed width of the color selection in pixels or % as a string, e.g. "40%" or "250px"



The value **value** is saved as a JSON encoded string.

EXAMPLE

```
{
"type": "SelectValue",
"name": "Value",
"caption": "Value",
"variableID": 12345
}
```

SelectVariable

Creates a selection dialog with the caption **caption**.
If created in the "elements" area, the property **name** is set to the ID of the selected variable when accepted.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onChange script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since IP-Symcon 7.0)
enabled (optional)	(**default:** true) If true, the variable selection can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
link (optional)	(**default:** false) If this parameter is true, the output of the onChange script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since IP-Symcon 6.0)
name (optional)	Name of the variable selection/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the value of the variable selection is changed. It has the same properties as onClick of the Button (since IP-Symcon 5.2)
requiredAction (optional)	(**default:** 0) Indicates whether the variable must have an action (since IP-Symcon 6.0) 0: It doesn't matter whether the variable has an action or not 1: The variable must have an action 2: The variable must not have any action
requiredLogging (optional)	(**default:** 0) The way in which the variable may be logged (since IP-Symcon 6.0) 0: It doesn't matter whether the variable is logged or not 1: The variable must be logged 2: The variable must not be logged 3: The variable must be logged as standard 4: The variable must be logged as a counter
type	SelectVariable
validPresentations (optional)	(**default:** []) The GUIDs of the presentations the variable may have. With an empty array, all presentations are permitted (since IP-Symcon 8.1)

PARAMETER	DESCRIPTION
validVariableTypes (optional)	(**default:** []) The types of variables that can be selected. With an empty array, all variable types are permitted (since IP-Symcon 6.0) 0 `VARIABLETYPE_BOOLEAN` Boolean 1 `VARIABLETYPE_INTEGER` Integer 2 `VARIABLETYPE_FLOAT` Float 3 `VARIABLETYPE_STRING` String
value (optional)	(**default:** 1) The value of the variable selection - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the variable selection is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the variable selection in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)



If the variable does not correspond to the requirements defined by the parameters validVariableTypes, requiredAction or requiredLogging, an error is displayed and changes cannot be accepted

EXAMPLE

```
// Any variable can be selected
{
"type": "SelectVariable",
"name": "PropertyVariableID",
"caption": "Target"
}

// The selected variable must be an integer or float,
// and be logged and have an action
{
"type": "SelectVariable",
"name": "PropertyVariableID",
"caption": "Target",
"validVariableTypes": [1, 2],
"requiredAction": 1,
"requiredLogging": 1
}
```

Status message

Creates a status message.



Status messages can only be used in the "status" area. No other form field types can be used in the "status" area.

PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption
code	Status code (see table)
icon	Symbol which is displayed: "active", "inactive", "standby" or "error" (see overview)

OVERVIEW OF THE STATUS CODES

CODE	CONSTANT	DESCRIPTION
101	`IS_CREATING`	Instance is creating
102	`IS_ACTIVE`	Instance is active
103	`IS_DELETING`	Instance is deleting
104	`IS_INACTIVE`	Instance is inactive
105	`IS_NOTCREATED`	Instance was not created
106	`IS_STANDBY`	Instance is in standby
>=200	`IS_EBASE`	Instance is faulty



Codes >= 200 must be defined by oneself.

OVERVIEW OF THE POSSIBLE ICONS

active



error



inactive



standby (since IP-Symcon 8.1)
Displays the standby symbol.

EXAMPLE

```
// Active icon
{ "code": 102, "icon": "active", "caption": "Logged in!" }

// Error icon
{ "code": 200, "icon": "error", "caption": "Error!" }

// Inactive icon
{ "code": 104, "icon": "inactive", "caption": "Logged out!" }

// Standby icon
{ "code": 106, "icon": "standby", "caption": "Standby!" }
```

TestCenter

Require: Symcon >= 5.0
Generates a Test Center in which all switchable status variables of the instance can be switched. The operating elements are based on the type and profile of the variables and are based on the switching elements of the WebFront.



This item is not supported by the Legacy Console. If a module with this element is opened in the legacy console, the TestCenter is not displayed.



PARAMETER

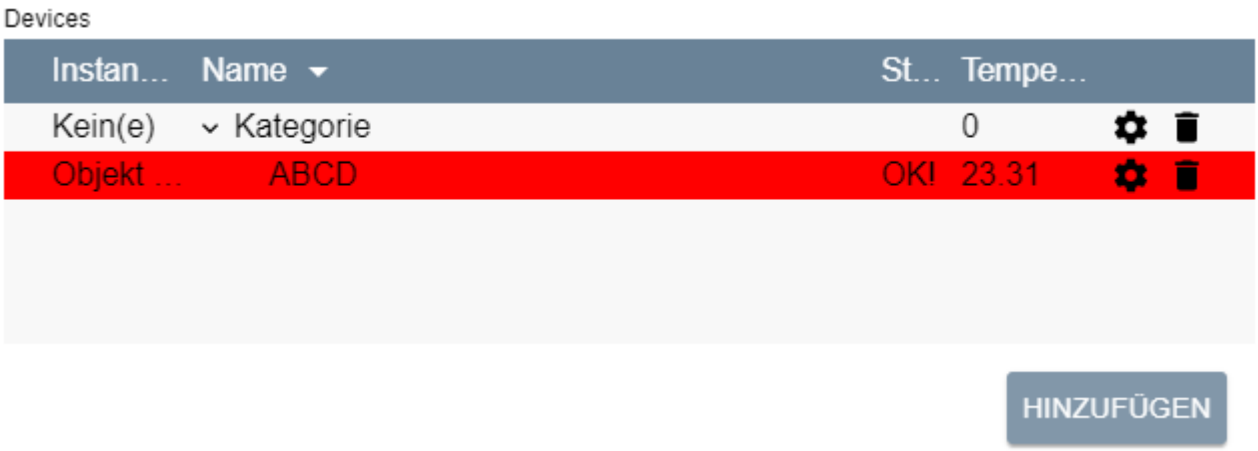
PARAMETER	DESCRIPTION
caption (optional)	(**default:** "") Visible caption (since IP-Symcon 7.2)
enabled (optional)	(**default:** true) If true, the Test Center can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
name (optional)	Name of the Test Center
type	TestCenter
visible (optional)	(**default:** true) If true, the Test Center is visible, otherwise it is invisible (since IP-Symcon 5.2)

EXAMPLE

```
{
"type": "TestCenter"
}
```

Tree

Require: Symcon >= 5.0
Creates a tree with the caption **caption**.
The displayed columns are defined in **columns**.
Initial entries in the tree can be specified via **values**.



This item is not supported by the Legacy Console. If a module with this element is opened in the legacy console, an error message appears.

GENERAL PARAMETERS

PARAMETER	DESCRIPTION
add (optional)	(**default:** false) If true, an "Add" button is displayed below the table, which adds a new element to the tree under the selected node
caption (optional)	Visible caption of the tree
columns	Columns of the tree
delete (optional)	(**default:** false) If true, a button is displayed behind each entry of the tree, which deletes the corresponding element
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onAdd, onDelete, onEdit, onExpand or onSelectionChanged script is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.0)
form (optional)	(**default:** based on the "edit" parameters of column) An individual form for editing or adding rows. The definition of the form is done analogously to the definition of the form for Actions either static or via PHP code. The usage of the form is done analogously to the same parameter of the List (since Symcon 7.0)

PARAMETER	DESCRIPTION
link (optional)	(**default:** false) If this parameter is true, the output of the onAdd, onDelete, onEdit, onExpand and onSelectionChanged script is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 6.0)
multiAdd (optional)	(**default:** true, if neither onAdd nor form is set and only a single editable column with an object selection exists, otherwise false) The types SelectObject, SelectCategory, SelectEvent, SelectInstance, SelectLink, SelectMedia, SelectScript or SelectVariable count as object selection. If this parameter is true, adding an entry directly opens the object selection of the first visible column with an object selection, in which multiple objects can be selected at once. An entry is added for every selected object, whose further values receive their add value and can be adjusted via the dialog afterwards, see List. When adding multiple objects, onAdd is still executed only once. If this parameter is false, adding an entry opens the dialog. However, if only a single editable column with an object selection exists and form is not set, the object selection for a single object is opened directly instead. (since Symcon 9.1)
name (optional)	Name of the tree
onAdd (optional)	(**default:** "") Script which is executed after adding an entry to the tree. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported from version 6.0). In this version, the tree does not offer the currently selected entry as the variable value but the added entry, see Use of the tree content in on-actions. Otherwise it has the same properties as onClick of the Button (since Symcon 5.3)
onDelete (optional)	(**default:** "") Script which is executed after deleting an entry of the tree. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported from version 6.0). In this version, the tree does not offer the currently selected entry as the variable value but the deleted entry, see Use of the tree content in on-actions. Otherwise it has the same properties as onClick of the Button (since Symcon 5.3)
onEdit (optional)	(**default:** "") Script which is executed after editing an entry of the tree. If the script consists of several lines, the individual lines can also be defined as an array (arrays are supported from version 6.0). In this version, the tree does not offer the currently selected entry as the variable value, but the edited entry, see Use of the tree content in on-actions. Otherwise it has the same properties as onClick of the Button (since Symcon 5.3)
onExpand (optional)	(**default:** "") Script which is executed after expanding an entry of the tree. If the script consists of several lines, the individual lines can also be defined as an array. In this version, the tree does not offer the currently selected entry as the variable value, but the expanded entry, see Use of the tree content in on-actions. Otherwise it has the same properties as onClick of the Button (since Symcon 6.4)

PARAMETER	DESCRIPTION
onSelectionChanged (optional)	(**default:** "") Script which is executed after the user has selected another entry of the tree via click or arrow key. If the script consists of several lines, the individual lines can also be defined as an array. The tree offers the newly selected entry as the variable value. The script is not executed when the selection is removed or when the values are changed via UpdateFormField, see Reacting to the selection. Otherwise it has the same properties as onClick of the Button (since Symcon 9.1)
rowCount (optional)	(**default:** 0) visible number of lines; If there are more lines in the tree, a scroll bar is displayed - if the value is 0, the tree fills the remaining space available in the instance editor. Only one configuration item can get all of the remaining space.
sort (optional)	Sorting of the tree. If this is not set, the tree is sorted according to the identifiers of the nodes
type	Tree
values (optional)	Initial entries in the tree
visible (optional)	(**default:** true) If true, the tree is visible, otherwise it is invisible (from Symcon 5.2)
enabled (optional)	(**default:** true) If true, the tree can be used, otherwise it is displayed as deactivated (from Symcon 6.0)
changeOrder (optional)	(**default:** false) If true, the order of the entries can be changed using drag & drop. The sorting of the list according to columns, however, is not possible in that case. (since Symcon 6.0)
loadValuesFromConfiguration (optional)	(**default:** true) This parameter specifies whether to populate the tree with the property's values, see Saving and loading trees. If the value is set to true, it is necessary to fill the tree in the interest of the user for a successful review in the Module Store, see Review guidelines Point 13 (since Symcon 6.0)

PARAMETERS FOR COLUMNS

PARAMETER	DESCRIPTION
add (optional)	Initial value in this column for a newly added entry, necessary if **add** is set to true on the tree. Alternatively, an object can be provided which generates the value automatically, see Parameters for add (since Symcon 9.1)
caption	Visible caption of a column
confirm (optional)	(**default:** "") Query text before an onClick action is performed. The action is only carried out if the query is confirmed with "Yes". If no query text is defined, the action is carried out directly (since Symcon 5.2)
download (optional)	(**default:** "") If this parameter is not an empty string and the output of the onClick script of this column is a Data-URL, the output is downloaded as a text file with the name of the download parameter. (since Symcon 7.1)
edit (optional)	Description of an editable element

PARAMETER	DESCRIPTION
~~label~~	~~Visible caption of a column~~ *(deprecated)*
link (optional)	(**default:** false) If this parameter is true, the output of the onClick script of this column is opened as a link. If it is false, the output is displayed as a dialog in the configuration form. (since Symcon 7.1)
name	Unique name of a column to refer to; the name must not be "id", "parent", "expanded", "rowColor", "editable" or "deletable", since these values are used for other functions, see parameters for values
onClick (optional)	(**default:** "") Script which is executed when a field in this column is clicked. It has the same properties as onClick of the Button. The script is only executed if this field is not empty, i.e. 0, false or "" (since Symcon 5.2)
quickFilter (optional)	(**default:** false) If true, the text of this column is considered during the search via quick filter. Should the value be false for all columns, the quick filter is hidden. (since Symcon 7.2)
save (optional)	(**default:** true, if the element is editable or **add** is an object for automatic generation, otherwise false) If true, the values of this column should be saved as module properties, see Saving and loading trees
visible (optional)	(**default:** true) If false, the column is not displayed
width	Width of the column in pixels, as a CSS string (e.g. “100px”); exactly one column must have the value “auto”, which means that the width of that column uses the rest of the available space. In addition, the expansion symbols of the tree are displayed in this column.

PARAMETERS FOR EDIT

PARAMETER	DESCRIPTION
type	Type of representation, according to the possible entries: CheckBox, IntervalBox, NumberSpinner, PasswordTextBox, Select, SelectCategory, SelectColor, SelectEvent, SelectFile, SelectInstance, SelectLink, SelectMedia, SelectObject, SelectScript, SelectVariable, ValidationTextBox, List, Tree
Further parameters	The other parameters depend on the type selected; the parameters "name" and "caption" of the type are ignored

If the "type" List or Tree is used, the value of this column is not JSON-coded again. It is only necessary to parse the value of the parent tree once. Each sub-list or subtree is then directly available as an object. Similarly, the values in "values" do not have to be coded twice.

The "add" parameter of the column is translated according to the localization, provided the "type" of "edit" is ValidationTextBox or PasswordTextBox or "edit" is not set at all

PARAMETERS FOR ADD

Instead of a fixed value, the parameter **add** of a column can also be an object which describes the strategy by which the value of a newly added entry is generated automatically (since Symcon 9.1). The value is generated when the entry is added and is treated like a regular value afterwards. The generated value is unique within the column: If a random value matches the value of an existing entry, a new value is generated. Generated values are not translated.

PARAMETER	DESCRIPTION
strategy	Strategy of the generation: "increment": The highest numeric value of this column across all entries of the tree is increased by **step**. Non-numeric values are ignored. If there is no numeric value yet, **start** is used or, without **start**, the value of **step**. If **start** is set, a value less than **start** is never generated (with a negative **step** never a greater one) "random-string": A random string of the characters a-z, A-Z and 0-9 with the length **length** "random-integer": A random integer between **min** and **max** "guid": A random GUID (UUID version 4), e.g. "{3F2504E0-4F89-4D3A-9A0C-0305E82C3301}"
step (optional)	(**default:** 1) Step by which the highest value is increased, only for the strategy "increment"; must be a number other than 0
start (optional)	Value of the first entry and lower bound of all following values (upper bound with a negative **step**), only for the strategy "increment"; must be a number
length (optional)	(**default:** 32) Number of characters of the generated string, only for the strategy "random-string"
min (optional)	(**default:** 0) Lower bound (inclusive) of the random integer, only for the strategy "random-integer"; must be less than **max**
max (optional)	(**default:** 2147483647) Upper bound (inclusive) of the random integer, only for the strategy "random-integer"; must be greater than **min**



To keep generated values, columns with such an add value are saved by default, i.e. the parameter "save" is true by default for these columns. If a loaded entry is missing the value of such a column, e.g. because the column was added later, the value is generated by the same strategy when loading. The default value that on-actions receive when no entry is selected is generated by the strategy as well. If no unique value can be generated because all possible values are already in use, e.g. with "random-integer" and a small range between min and max, adding the entry fails with an error message.

```
// Columns with automatically generated values
{
"caption": "ID",
"name": "ID",
"width": "50px",
"add": { "strategy": "increment", "step": 10 }
}, {
```

```
"caption": "Token",
"name": "Token",
"width": "auto",
"add": { "strategy": "random-string", "length": 16 },
"edit": { "type": "ValidationTextBox" }
}, {
"caption": "Random number",
"name": "Number",
"width": "100px",
"add": { "strategy": "random-integer", "min": 1, "max": 100 },
"edit": { "type": "NumberSpinner" }
}, {
"caption": "GUID",
"name": "GUID",
"width": "300px",
"add": { "strategy": "guid" }
}
```

The strategy "increment" determines the next value from the current entries. If the entry with the highest value is deleted, its value is assigned again when the next entry is added. If a value should never be reused, e.g. because it serves as a permanent ID towards an external system, the module can remember the highest value ever assigned and specify it via **start**.

```
// Deleted IDs are not assigned again
public function GetConfigurationForm(): string
{
$form = json_decode(file_get_contents(__DIR__ . '/form.json'), true);
// The attribute HighestID is updated to the highest value of the list
in ApplyChanges, but never decreases
$form['elements'][0]['columns'][0]['add'] = ['strategy' => 'increment',
'start' => $this->ReadAttributeInteger('HighestID') + 1];
return json_encode($form);
}
```

PARAMETERS FOR SORT

PARAMETER	DESCRIPTION
column	Name of the column according to which the tree is sorted
direction	The direction of the sort; "ascending" for an ascending order and "descending" for a descending order

PARAMETERS FOR VALUES

PARAMETER	DESCRIPTION
deletable (optional)	(**default:** true) If false, the line cannot be deleted (since Symcon 5.4)
editable (optional)	(**default:** true) If false, the line cannot be edited, even if the tree itself has editable columns (since Symcon 5.4)

PARAMETER	DESCRIPTION
expanded (optional)	(**default:** false) If this value is true, the entry is expanded directly when the configuration form is loaded
id	An identifier with which the entry in the tree view can be addressed as a parent node, id must be greater than 0
parent (optional)	(**default:** 0) The id of a value can be set in this field, which becomes the parent node of this value in the tree display. For this, the parent value must come before of the child in the array. If "parent" is 0, the element is inserted at the top level.
rowColor (optional)	(**default:** transparent) Background color of the line as hex code (e.g. red: "#ff0000"), empty string for transparency
icons (optional)	(**default:** {}) Icons can be defined in the same way as for List. (since Symcon 9.0)
Further parameters (optional)	(**default:** "") For each initial entry in the tree, the values for all columns are defined here. For this purpose, a parameter with the name of the column is entered for each column. The assignment of the parameter corresponds to the value of this column for the corresponding entry.

SAVING AND LOADING TREES

The saving and loading of trees is done almost analogously to Lists.
The values for **id** and **parent** are also saved in the property.
When loading, it should be noted that when using values, the identifier of the node must always be specified so that these can be assigned to the appropriate node from the property.

USE OF THE TREE CONTENT IN ON-ACTIONS

The use of the values of the tress in scripts is analogous to Lists. In addition, each entry contains the value "expanded", which defines whether the entry is expanded (true) or collapsed (false).

REACTING TO THE SELECTION

Using the parameter **onSelectionChanged**, the module can react when the user selects another entry of the tree (since Symcon 9.1). The behavior, the use cases and the difference to the onClick of a column correspond to the List. For the tree, the following applies in addition:

Changing to the parent node via left arrow key and to the first child node of an expanded entry via right arrow key also executes onSelectionChanged.
Expanding and collapsing an entry does not change the selection and only executes onExpand. If a selected entry is hidden in the process, the selection is removed without onSelectionChanged.
After a change of values via UpdateFormField, the selection is kept based on the **id** of the entry.
Trees often contain different kinds of entries, e.g. rooms and devices. Based on the values of the selected entry, e.g. $(Name of the tree)['parent'] or a custom column, the script can distinguish what was selected.

Example: A tree with rooms and devices. If a device is selected, its location is shown and the button "Show status" is enabled. If a room is selected, the button is disabled.

```
// Example: form.json
{
"actions": [
{
"type": "Tree",
"name": "Rooms",
"caption": "Rooms and devices",
"rowCount": 6,
"columns": [
{ "caption": "Name", "name": "Name", "width": "auto", "add": "" },
{ "caption": "InstanceID", "name": "InstanceID", "width": "100px",
"add": 0, "visible": false }
],
"values": [
{ "id": 1, "Name": "Living room", "InstanceID": 0, "expanded": true },
{ "id": 2, "parent": 1, "Name": "Ceiling light", "InstanceID": 12345 },
{ "id": 3, "Name": "Kitchen", "InstanceID": 0 },
{ "id": 4, "parent": 3, "Name": "Extractor hood", "InstanceID": 23456 }
],
"onSelectionChanged": "ROOM_UISelectNode($id, $Rooms['InstanceID']);"
},
{ "type": "Label", "name": "NodeDetails", "caption": "Please select a
device" },
{
"type": "Button",
"name": "StatusButton",
"caption": "Show status",
"enabled": false,
"onClick": "ROOM_UIShowStatus($id, $Rooms['InstanceID']);"
}
]
}
```

```
// Example: module.php
<?php

declare(strict_types=1);

class RoomOverview extends IPSModule
{
// onSelectionChanged of the tree Rooms: Rooms have no instance, only
devices do
public function UISelectNode(int $InstanceID)
{
$isDevice = IPS_InstanceExists($InstanceID);
$this->UpdateFormField('NodeDetails', 'caption', $isDevice ?
IPS_GetLocation($InstanceID) : 'Please select a device');
$this->UpdateFormField('StatusButton', 'enabled', $isDevice);
}

// onClick of the button "Show status"
```

```
public function UIShowStatus(int $InstanceID)
{
if (!IPS_InstanceExists($InstanceID)) {
echo 'Please select a device';
return;
}
echo (IPS_GetInstance($InstanceID)['InstanceStatus'] == IS_ACTIVE) ?
'The device is active' : 'The device is not active';
}
}
```

EXAMPLE

```
{
"type": "Tree",
"name": "Devices",
"caption": "Devices",
"rowCount": 5,
"add": true,
"delete": true,
"sort": {
"column": "Name",
"direction": "ascending"
},
"columns": [{
"caption": "InstanceID",
"name": "InstanceID",
"width": "75px",
"add": 0,
"edit": {
"type": "SelectInstance"
}
}, {
"caption": "Name",
"name": "Name",
"width": "auto",
"add": ""
}, {
"caption": "State",
"name": "State",
"width": "40px",
"add": "New!"
}, {
"caption": "Temperature",
"name": "Temperature",
"width": "75px",
"add": 20.0,
"edit": {
"type": "NumberSpinner",
"digits": 2
}
}],
"values": [{
"id": 1,
"InstanceID": 0,
```

```
"Name": "Category",
"State": "",
"Temperature": 0
},{
"id": 2,
"parent": 1,
"InstanceID": 12435,
"Name": "ABCD",
"State": "OK!",
"Temperature": 23.31,
"rowColor": "#ff0000"
}]
}
```

ValidationTextBox

Creates an input field for text with the heading **caption**.
If created in the “elements” area, the ValidationTextBox sets a property to the entered string when accepted.
The parameter **"name"** defines which property is set.



PARAMETERS

PARAMETER	DESCRIPTION
caption	Visible caption of the input field
enabled (optional)	(**default:** true) If true, the input field can be used, otherwise it is displayed as deactivated (since IP-Symcon 5.2)
multiline (optional)	(**default:** false) If true, the input field is expanded to several lines if necessary. This can happen through deliberately set line breaks, e.g. by pressing Enter, as well as in the event of an overflow. If false, the input field always remains in one line (since IP-Symcon 5.4)
name (optional)	Name of the input field/the property to be set
onChange (optional)	(**default:** "") Script which is executed when the ValidationTextBox value is changed. If the script consists of several lines, the individual lines can also be defined as an array. It has the same properties as onClick of the Button. (since IP-Symcon 8.1)
type	ValidationTextBox
validate (optional)	(**default:** "") If the value does not meet the regular expression specified here, an error is displayed and changes cannot be adopted (since IP-Symcon 5.3)
value (optional)	(**default:** "") The value of the input field - If there is an associated property, this parameter is overwritten by the property in the elements area (since IP-Symcon 5.2)
visible (optional)	(**default:** true) If true, the input field is visible, otherwise it is invisible (since IP-Symcon 5.2)
width (optional)	(**default:** 300px) Fixed width of the input field in pixels or % as a string, e.g. "40%" or "250px" (since IP-Symcon 5.3)

EXAMPLE

```
{ "type": "ValidationTextBox", "name": "IPAddress", "caption": "Host" }
```

Constants

Require: Symcon >= 5.1

CONSTANTS

DEFINE	DESCRIPTION
EVENTCONDITIONCOMPARISON	Comparison conditions for conditional events
EVENTCYCLICDATETYPE	Date type for cyclical event
EVENTCYCLICTIMETYPE	Time type of the event
EVENTTRIGGERTYPE	Trigger type for triggered event
EVENTTYPE	Type of event
MEDIATYPE	Type of mediainstance
MODULETYPE	Type of module
OBJECTTYPE	Type of object
SCRIPTTYPE	Type of script
VARIABLETYPE	Type of variable

EVENTCONDITIONCOMPARISON

Comparison conditions for conditional events

CONSTANT	VALUE	DESCRIPTION
EVENTCONDITIONCOMPARISON_EQUAL	0	Equal
EVENTCONDITIONCOMPARISON_NOTEQUAL	1	Not equal
EVENTCONDITIONCOMPARISON_GREATER	2	Greater
EVENTCONDITIONCOMPARISON_GREATEROREQUAL	3	Greater or equal
EVENTCONDITIONCOMPARISON_SMALLER	4	Smaller
EVENTCONDITIONCOMPARISON_SMALLEROREQUAL	5	Smaller or equal

EVENTCYCLICDATETYPE

Date type for cyclical event

CONSTANT	VALUE	DESCRIPTION
EVENTCYCLICDATETYPE_NONE	0	No date type
EVENTCYCLICDATETYPE_ONCE	1	Once
EVENTCYCLICDATETYPE_DAY	2	Daily
EVENTCYCLICDATETYPE_WEEK	3	Weekly

CONSTANT	VALUE	DESCRIPTION
EVENTCYCLICDATETYPE_MONTH	4	Monthly
EVENTCYCLICDATETYPE_YEAR	5	Yearly

EVENTCYCLICTIMETYPE

Time type of the event

CONSTANT	VALUE	DESCRIPTION
EVENTCYCLICTIMETYPE_ONCE	0	Once
EVENTCYCLICTIMETYPE_SECOND	1	Secondly
EVENTCYCLICTIMETYPE_MINUTE	2	Minutely
EVENTCYCLICTIMETYPE_HOUR	3	Hourly

EVENTTRIGGERTYPE

Trigger type for triggered event

CONSTANT	VALUE	DESCRIPTION
EVENTTRIGGERTYPE_ONUPDATE	0	If the variable is updated
EVENTTRIGGERTYPE_ONCHANGE	1	If the variable is changed
EVENTTRIGGERTYPE_ONLIMITEXCEED	2	If the limit value is exceeded
EVENTTRIGGERTYPE_ONLIMITDROP	3	If the limit value is exceeded
EVENTTRIGGERTYPE_ONVALUE	4	At a certain value

EVENTTYPE

Type of event

CONSTANT	VALUE	DESCRIPTION
EVENTTYPE_TRIGGER	0	Triggered event
EVENTTYPE_CYCLIC	1	Cyclical event
EVENTTYPE_SCHEDULE	2	Schedule event

MEDIATYPE

Type of mediainstance

CONSTANT	VALUE	DESCRIPTION
MEDIATYPE_IPSVIEW	0	IPSView

CONSTANT	VALUE	DESCRIPTION
MEDIATYPE_IMAGE	1	Image
MEDIATYPE_SOUND	2	Sound
MEDIATYPE_STREAM	3	Stream
MEDIATYPE_CHART	4	Chart
MEDIATYPE_DOCUMENT	5	Document

MODULETYPE

Type of module

CONSTANT	VALUE	DESCRIPTION
MODULETYPE_CORE	0	Core
MODULETYPE_IO	1	I/O
MODULETYPE_SPLITTER	2	Splitter
MODULETYPE_DEVICE	3	Device
MODULETYPE_CONFIGURATOR	4	Configurator
MODULETYPE_DISCOVERY	5	Discovery
MODULETYPE_VISUALIZATION	6	Visualization

OBJECTTYPE

Type of object

CONSTANT	VALUE	DESCRIPTION
OBJECTTYPE_CATEGORY	0	Category
OBJECTTYPE_INSTANCE	1	Instance
OBJECTTYPE_VARIABLE	2	Variable
OBJECTTYPE_SCRIPT	3	Script
OBJECTTYPE_EVENT	4	Event
OBJECTTYPE_MEDIA	5	Media
OBJECTTYPE_LINK	6	Link

SCRIPTTYPE

Type of script

CONSTANT	VALUE	DESCRIPTION
SCRIPTTYPE_PHP	0	PHP-Script
SCRIPTTYPE_FLOW	1	Flow Script

CONSTANT	VALUE	DESCRIPTION
SCRIPTTYPE_IPSWORKFLOW	2	IPSWorkflow

VARIABLETYPE

Type of variable

CONSTANT	VALUE	DESCRIPTION
VARIABLETYPE_BOOLEAN	0	Boolean
VARIABLETYPE_INTEGER	1	Integer
VARIABLETYPE_FLOAT	2	Float
VARIABLETYPE_STRING	3	String

PRESENTATIONS

GUID	KONSTANTE	NAME
{6B9CAEEC-5958-C223-30F7-BD36569FC57A}	VARIABLE_PRESENTATION_SLIDER	Slider
{4153A8D4-5C33-C65F-C1F3-7B61AAF99B1C}	VARIABLE_PRESENTATION_LEGACY	Legacy-Profile
{3319437D-7CDE-699D-750A-3C6A3841FA75}	VARIABLE_PRESENTATION_VALUE_PRESENTATION	Value Presentation
{6F477326-1683-A2FD-D2E7-477F366ECB62}	VARIABLE_PRESENTATION_VALUE_INPUT	Value Input
{9DE1D610-5106-97FB-714D-1AADEDF8377A}	VARIABLE_PRESENTATION_WEB_CONTENT	Web Content
{05CC3CC2-A0B2-5837-A4A7-A07EA0B9DDFB}	VARIABLE_PRESENTATION_COLOR	Color
{497C4845-27FA-6E4F-AE37-5D951D3BDBF9}	VARIABLE_PRESENTATION_DATE_TIME	Date/Time
{60AE6B26-B3E2-BDB1-A3A1-BE232940664B}	VARIABLE_PRESENTATION_SWITCH	Switch
{6075FC22-69AF-B110-3749-C24138883082}	VARIABLE_PRESENTATION_SHUTTER	Shuttere

GUID	KONSTANTE	NAME
{52D9E126-D7D2-2CBB-5E62-4CF7BA7C5D82}	VARIABLE_PRESENTATION_ENUMERATION	Enumeration
{08A6AF76-394E-D354-48D5-BFC690488E4E}	VARIABLE_PRESENTATION_DURATION	Duration

TEMPLATES

GUID	KONSTANTE	
{B4C70F3E-6613-DA1A-7279-5DEE8DEB1B24}	VARIABLE_TEMPLATE_DATE	Date
{362DA268-56A2-E771-5E53-17E38B5D82E6}	VARIABLE_TEMPLATE_TIME	Time
{BB0E9933-0403-BD3A-D1C9-255646934B00}	VARIABLE_TEMPLATE_DATE_TIME	Datu
{868B087E-A38D-2155-EBE0-157AFBBF9E8C}	VARIABLE_TEMPLATE_SLIDER_ROOM_TEMPERATURE	Roo Tem
{66062309-21A9-26C0-213F-775C52E1473B}	VARIABLE_TEMPLATE_SLIDER_COLOR_TEMPERATURE	Colo Tem
{BC799412-0C66-551F-CAEC-7566F5D52BD9}	VARIABLE_TEMPLATE_SLIDER_ENERGY	Ener
{8EC19DF0-89FB-A77E-ED7D-047A949CF292}	VARIABLE_TEMPLATE_SLIDER_POWER	Pow
{3BE75DE9-7D84-C082-2E77-9ED3AEE04D63}	VARIABLE_TEMPLATE_SHUTTER_LAMELLA_RIGHT	Lam
{22A0DF9C-C200-154A-641B-3A3CB096DB6D}	VARIABLE_TEMPLATE_SHUTTER_LAMELLA_LEFT	Lam

GUID	KONSTANTE	
{90AF8F8F-183F-BBFD-E078-35FAB6DCFE4F}	VARIABLE_TEMPLATE_VALUE_PRESENTATION_ROOM_TEMPERATURE	Roo Tem
{2FED3D39-073D-6037-901B-2586A1AB5569}	VARIABLE_TEMPLATE_VALUE_PRESENTATION_POWER	Pow
{C899FCFA-063E-897E-9DA4-28ADD278EED5}	VARIABLE_TEMPLATE_VALUE_PRESENTATION_ENERGY	Ener
{7BD38CF5-07F2-5B5B-8F7F-15398B823BFC}	VARIABLE_TEMPLATE_VALUE_PRESENTATION_BATTERY	Batt
{C90EF36A-165E-D0B0-032C-F468F483D42B}	VARIABLE_TEMPLATE_VALUE_PRESENTATION_BATTERY_COLOR	Batt
{0C711895-2F8E-DBFE-1700-84173491D229}	VARIABLE_TEMPLATE_COLOR_RAINBOW	Rain
{A7467E68-5C39-5BD9-C0C8-BCE6004FEEAA}	VARIABLE_TEMPLATE_COLOR_FOREST	Fore

INSTANCESTATUS

The status of an instance

CONSTANT	VALUE	DESCRIPTION
IS_CREATING	101	Instance is being created
IS_ACTIVE	102	Instance is active
IS_DELETING	103	Instance is being deleted
IS_INACTIVE	104	Instance is inactive
IS_NOTCREATED	105	Instance was not created
IS_STANDBY	106	Instance is in standby

VISUALIZATION TYPES

since 9.1

CONSTANT	VALUE	DESCRIPTION
INSTANCE_VISUALIZATION_TYPE_NONE	0	The instance does not use its own visualization
INSTANCE_VISUALIZATION_TYPE_HTML	1	The instance uses an HTML visualization
INSTANCE_VISUALIZATION_TYPE_HTML_FULLSCREEN	2	The instance uses an HTML visualization for normal and opened tile
INSTANCE_VISUALIZATION_TYPE_FORM	3	The instance uses a form as visualization

DEBUGFORMAT

Format of a debug message (IPS_SendDebug, SendDebug)

CONSTANT	VALUE	DESCRIPTION
DEBUG_FORMAT_TEXT	0	Text
DEBUG_FORMAT_BINARY	1	Binary (displayed as hex)

EVENTCONDITIONOPERATION

Combination of the rules of a condition (IPS_SetEventCondition)

CONSTANT	VALUE	DESCRIPTION
EVENT_CONDITION_OPERATION_AND	0	All rules must be met (AND)
EVENT_CONDITION_OPERATION_OR	1	At least one rule must be met (OR)

EVENTDAY

Weekdays as bitmask (IPS_SetEventCyclic, IPS_SetEventScheduleGroup). Multiple days are added.

CONSTANT	VALUE	DESCRIPTION
EVENT_DAY_MONDAY	1	Monday
EVENT_DAY_TUESDAY	2	Tuesday
EVENT_DAY_WEDNESDAY	4	Wednesday
EVENT_DAY_THURSDAY	8	Thursday
EVENT_DAY_FRIDAY	16	Friday
EVENT_DAY_SATURDAY	32	Saturday
EVENT_DAY_SUNDAY	64	Sunday
EVENT_DAY_ALL	127	All weekdays

COLORENCODING

Encoding of the presentation Color (parameter ENCODING)

CONSTANT	VALUE	DESCRIPTION
COLOR_ENCODING_RGB	0	RGB
COLOR_ENCODING_CMYK	1	CMYK
COLOR_ENCODING_HSV	2	HSV
COLOR_ENCODING_HSL	3	HSL
COLOR_ENCODING_XY	4	xy (CIE)

COLOR

Special color values

CONSTANT	VALUE	DESCRIPTION
COLOR_TRANSPARENT	-1	Transparent or no color

USERTYPE

Type of a user (IPS_CreateUser)

CONSTANT	VALUE	DESCRIPTION
USER_TYPE_LOCAL	0	Local user account
USER_TYPE_LDAP	1	LDAP user account

REPLICATIONMODE

Mode of the replication (IPS_GetReplicationMode)

CONSTANT	VALUE	DESCRIPTION
REPLICATION_MODE_PARALLEL	0	Parallel
REPLICATION_MODE_STANDBY	1	Standby

VARIABLEACTION

Special values for the custom action of a variable (IPS_SetVariableCustomAction)

CONSTANT	VALUE	DESCRIPTION
VARIABLE_ACTION_NONE	0	No custom action, the default action is used
VARIABLE_ACTION_DISABLED	1	Default action disabled

Localizations

Require: Symcon >= 4.1
The JSON file "locale.json" contains the translations for "label" and "caption" of the configuration page. Graduated language abbreviations are also possible here. For example, "de_DE" or "de_AT" can also be used.
Alternatively, the translations can also be specified directly in the form.json under "translations". If a locale.json exists, it completely replaces the translations of the form.json.
A JSON schema is available for the locale.json: localeSchema.json

EXAMPLES

Basic structure of the JSON file
The language abbreviations indicate into which language the translation is to be carried out

```
{
"translations": {
"de": {
"Word to be translated 1": "Translation into German - Word 1",
"Word to be translated 2": "Translation into German - Word 2"
},
"de_DE": {
"Word to be translated 1": "Translation into German - Word 1",
"Word to be translated 2": "Translation into German - Word 2"
},
"de_CH": {
"Word to be translated 1": "Translation into Swiss German - word 1",
"Word to be translated 2": "Translation into Swiss German - word 2"
}
}
}
```

Complete example using an SMS module (see example configuration forms)

```
{
"translations": {
"de": {
"Username": "Benutzername",
"Password": "Passwort",
"Sender": "Absender",
"Type": "Typ",
"Combi-SMS": "Kombi-SMS",
"Number": "Nummer",
"Message": "Nachricht",
"Send Message": "Sende Nachricht",
"Read Balance": "Frage Guthaben ab",
"Login information valid": "Logindaten sind gültig",
"Authentication failed": "Authentifizierung ist fehlgeschlagen",
```

```
"No credits left": "Kein Guthaben vorhanden"
}
}
}
```

Module

Require: Symcon >= 4.0

DESCRIPTION

A module basically consists of 2 files. The module.php and module.json.
A configuration page (form.json) can optionally be provided.

FORM.JSON

Further information on the configuration page is available under Configuration Forms.

LOCALE.JSON

Further information on the translation of the configuration page is available under Localizations.

MODULE.JSON

This file contains frame information essential for identification and correct integration of the module.

PARAMETER	DATA TYPE	DESCRIPTION
id	string	Unique GUID for unique identification. GUID Generator
name	string	Module name. (A-Z, a-z, 0-9, spaces, underscores are allowed characters. However, spaces and underscores may not be at the beginning or the end. An empty name is also not valid.)
type	integer	Module type 0 `MODULETYPE_CORE` Core 1 `MODULETYPE_IO` I/O 2 `MODULETYPE_SPLITTER` Splitter 3 `MODULETYPE_DEVICE` Device 4 `MODULETYPE_CONFIGURATOR` Configurator 5 `MODULETYPE_DISCOVERY` Discovery 6 `MODULETYPE_VISUALIZATION` Visualization
vendor	string	Manufacturer name and the name of the menu item under which the device can be found in "Add instance". If nothing is specified, the device is entered under " (Other)".
aliases	array [string]	Additional device names/ variants

PARAMETER	DATA TYPE	DESCRIPTION
url	string	URL to the documentation page of the module (Must start with http:// or https://. May alternatively be left "" (empty)
parentRequirements	array [string]	Data flow GUIDs, whereby compatible parent instances are determined. The parent instance must have implemented at least one of these data flow GUIDs in order to be compatible
childRequirements	array [string]	Data flow GUIDs, which determines compatible child instances. The child instance must have implemented at least one of these data flow GUIDs in order to be compatible
implemented	array [string]	Supported data flow GUIDs must be correctly evaluated and supported in the respective ReceiveData/ ForwardData functions, provided they are listed here
prefix	string	Prefix, which is assigned to the functions. Can only contain numbers and letters.

```
{
"id": "{E5AA629B-75BD-45C0-9BCB-845C102B0411}",
"name": "ModulnameXYZ",
"type": 3,
"vendor": "",
"aliases":
[
"Name1SupportedDevice",
"Name2SupportedDevice"
],
"url": "https://www.symcon.de",
"parentRequirements": [],
"childRequirements": [],
"implemented": [],
"prefix": "ABC"
}
```

MODULE.PHP

This is the actual class file, which contains the functions that process and forward data afterwards.



The class name must be identical to the "name" parameter, which was defined in module.json. The only allowed difference are spaces. These must be removed from the class name within module.php.



Function names may only consist of the following characters: "a..z", "A..Z", "0..9". Furthermore, "$InstanceID" must not be used as a parameter name.

Since IP-Symcon 8.1, the improved base class IPSModuleStrict is available.
IPSModule remains supported but should no longer be used for new modules.
The following table highlights the differences:

FEATURE	IPSMODULE	IPSMODULESTRICT
Type hints	Optional for public functions	Always required
Errors for missing type hints in public functions	Only warnings	Treated as errors
Support for outdated types	Integer/Boolean allowed	Use int/bool instead
Return value of `RegisterVariable*`	Current variable ID as int	Boolean indicating whether the variable was created (e.g., to set an initial value)
Write access to created variables	Always possible, also via SetValue from outside	Only via $this->SetValue (variables are protected as read-only)
Data flow connections	Manual via ConnectParent/RequireParent/ForceParent	Automatic via compatibility and GetCompatibleParents()
Data flow encoding	UTF-8 (utf8_encode/utf8_decode), problematic with PHP 9.0	HEX-encoded (bin2hex/hex2bin), easier to detect when encoding is needed
Hook/WebHook	Workaround via WebHookModule base class	Native support via RegisterHook, ProcessHookData and UnregisterHook
OAuth	Workaround via WebOAuthModule base class	Native support via RegisterOAuth, ProcessOAuthData and UnregisterOAuth

TEMPLATE MINIMAL

```
// class definition
class ModulnameXYZ extends IPSModuleStrict {

/**
* The following functions are automatically available if the module has
been inserted via the "Module Control".
* The functions are, with the prefix set up by oneself, made available
in PHP and JSON-RPC as follows:
```

```
*
* ABC_MyFirstOwnFunction($id);
*
*/
public function MyFirstOwnFunction(): void {
echo $this->InstanceID;
}

}
```

TEMPLATE CLASSIC

```
// class definition
class ModulnameXYZ extends IPSModuleStrict {

// Overrides the internal IPS_Create($id) function
public function Create(): void {
// Don't delete this line
parent::Create();
}

// Overwrites the internal IPS_ApplyChanges($id) function
public function ApplyChanges(): void {
// Don't delete this line
parent::ApplyChanges();
}

/**
* The following functions are automatically available if the module has
been inserted via the "Module Control".
* The functions are, with the prefix set up by oneself, made available
in PHP and JSON-RPC as follows:
*
* ABC_MyFirstOwnFunction($id);
*
*/
public function MyFirstOwnFunction(): void {
// Self-created code
}

}
```

__construct

Require: Symcon >= 4.0

```
void __construct (string $InstanceID)
```

PARAMETERS

InstanceID	ID of this instance

RETURNS

No Return

DESCRIPTION

The constructor is called with every function call and takes care of the instantiation.
The base class, for example, uses the constructor to set the InstanceID.



If individual extensions are wanted, the constructor must be overwritten by the base class.

EXAMPLE

```
// Normally, this function does not need to be overwritten.
```

ApplyChanges

Require: Symcon >= 4.0

```
void ApplyChanges ()
```

RETURNS

No Return

DESCRIPTION

Is executed when "Apply" is pressed on the configuration page and immediately after the instance has been created.



The ApplyChanges function is called by IP-Symcon. It must therefore be overwritten by the base class in order to add custom extensions

EXAMPLE

```
// IPSModuleStrict
public function ApplyChanges(): void {

// Do not delete this line
parent::ApplyChanges();

// ConnectParent/RequireParent/ForceParent is not available - but can
be replaced by the new function 'GetCompatibleParents()'.

}

// IPSModule
public function ApplyChanges() {

// Do not delete this line
parent::ApplyChanges();

// If there is no parent instance, create a new own VirtualIO instance
$this->RequireParent("{6179ED6A-FC31-413C-BB8E-1204150CF376}");

}
```

ConnectParent

Require: Symcon >= 4.0

```
bool ConnectParent (string $ParentGUID)
```

PARAMETERS

ParentGUID	GUID

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION



This function is not available for IPSModuleStrict. Please consider using the new GetCompatibleParents function

The function connects the instance with an existing compatible parent instance. If this is not available, one will be created and subsequently connected. A list of the available moduleIDs (GUID) can be found here.



The typical use case for this function is a device module that wants to connect to a splitter. A new one is only created if there is no splitter available.

EXAMPLE

```
// IPSModuleStrict
ConnectParent/RequireParent/ForceParent is not available,
but can be replaced by the new function "GetCompatibleParents()"

// IPSModule
public function Create() {

// Never remove the line!
parent::Create();
```

```
// Connect to an existing splitter or create a new one if necessary
$this->ConnectParent("{46C969BF-3465-4E3E-B2A5-E404FB969735}");

}
```

Create

Require: Symcon >= 4.0

```
void Create ()
```

RETURNS

No Return

DESCRIPTION

In contrast to Construct, this function is called only once when creating the instance and starting IP-Symcon. Therefore, status variables and module properties which the module requires permanently should be created here.
Frequently used functions:
RegisterPropertyString
RegisterPropertyInteger
RegisterPropertyFloat
RegisterPropertyBoolean



The Create function is called by IP-Symcon. It must therefore be overwritten by the base class in order to add custom extensions

EXAMPLE

```
// IPSModuleStrict
public function Create(): void {

// Do not remove this line
parent::Create();

// Module property creation
$this->RegisterPropertyString("Username", "MaxMustermann");
$this->RegisterPropertyInteger("Number", 123);
$this->RegisterPropertyFloat("Factor", 0.5);
$this->RegisterPropertyBoolean("Open", true);

}

// IPSModule
public function Create() {
```

```
// Example is identical. Please note the changed function signature.

}
```

Destroy

Require: Symcon >= 4.1

```
void Destroy ()
```

RETURNS

No Return

DESCRIPTION

This function is called when deleting the instance during operation and when updating via "Module Control". The function is not called when exiting IP-Symcon.



The Destroy function is called by IP-Symcon. It must therefore be overwritten by the base class in order to add individual extensions.

EXAMPLE

```
// IPSModuleStrict
public function Destroy(): void {

// Do not remove this line
parent::Destroy();

}

// IPSModule
public function Destroy() {

// Example is identical. Please note the changed function signature.

}
```

DisableAction

Require: Symcon >= 4.0

```
bool DisableAction (string $Ident)
```

PARAMETERS

Ident	Ident of the variable

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Deactivates the default action of the status variable. As a result, it can no longer be changed in the visualization, but can only be read. If EnableAction was not called, this function does not have to be called, since all status variables are created by default without a default action.

EXAMPLE

```
// Deactivates the default action of the status variable
$this->DisableAction("Status");
```

EnableAction

Require: Symcon >= 4.0

```
bool EnableAction (string $Ident)
```

PARAMETERS

Ident	Ident of the variable

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Activates the default action of the status variable. This means that it can be changed in the visualization and can also be written onto. This function must be called because all status variables are, by default, created without a default action. If the default action is activated, change requests must be responded to within RequestAction .

EXAMPLE

```
// Activates the default action of the status variable
$this->EnableAction("Status");
```

ForceParent

Require: Symcon >= 4.0

```
bool ForceParent (string $ModuleID)
```

PARAMETERS

ModuleID	GUID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function is not available for IPSModuleStrict. Please consider using the new GetCompatibleParents function

A connection with a compatible parent instance with the GUID **ModuleID** is forced. If the previous connection does not match the desired **ModuleID**, it will be disconnected. Following, a new instance with the parent ModuleID is created and linked.
A list of the available module IDs is available here .



The normal case for the function is a splitter module, which, depending on the configuration in the interface, requires a different I/O module.



Normally, this function is called only in the Create() or ApplyChanges() method.

EXAMPLE

```
// IPSModuleStrict
ConnectParent/RequireParent/ForceParent is not available,
but can be replaced by the new function "GetCompatibleParents()"
```

```
// IPSModule
public function ApplyChanges() {

// Never remove the line!
parent::ApplyChanges();

// Create different I/O instances depending on the configuration
switch($this->ReadPropertyInteger("GatewayMode")) {
case 0: //Create ClientSocket in mode 0
$this->ForceParent("{3CFF0FD9-E306-41DB-9B5A-9D06D38576C3}");
break;
case 1: //Create SerialPort in mode 1
$this->ForceParent("{6DC3D946-0D31-450F-A8C6-C42DB8D7D4F1}");
break;
}

}
```

ForwardData

Require: Symcon >= 4.0

```
string ForwardData (string $JSONString)
```

PARAMETERS

JSONString	Data packet in JSON format

RETURNS

Result of the function, which is returned to the calling child instance

DESCRIPTION

This function is called by IP-Symcon and processes sent data and forwards it to the parent instance.
Data can be sent using the SendDataToParent function.
Further information on data forwarding can be found under Dataflow.



The ForwardData function is called by IP-Symcon. It must therefore be overwritten by the base class in order to add individual extensions

EXAMPLE

```
// IPSModuleStrict
public function ForwardData(string $JSONString): string {

// Example within a gateway/splitter instance

// Received data from the device instance
$data = json_decode($JSONString);
IPS_LogMessage("ForwardData", utf8_decode($data->Buffer));

// The buffer would normally be processed here
// e.g., check CRC, partition into individual parts

// Forward to the I/O instance
$resultat = $this->SendDataToParent(json_encode(Array("DataID" => "
{79827379-F36E-4ADA-8A95-5F8D1DC92FA9}", "Buffer" => $data->Buffer)));
```

```
// Processing and passing on
return $resultat;

}

// IPSModule
public function ForwardData($JSONString) {

// Example is identical. Please note the changed function signature.

}
```

GetBuffer

Require: Symcon >= 4.1

```
string GetBuffer (string $Name)
```

PARAMETERS

Name	The name of the buffer

RETURNS

The content of the buffer

DESCRIPTION

This function returns the content of the buffer with the name **Name**. If the buffer is empty, an empty string is returned.



The content of the buffer is only available as long as IP-Symcon is running. After a restart all information disappears

EXAMPLE

```
// Returns the content of the "Databuffer" buffer
$Bufferdata = $this->GetBuffer("DataBuffer");
```

GetBufferList

Require: Symcon >= 5.0

```
array GetBufferList ()
```

RETURNS

List of names of all buffers. The order of the entries is not guaranteed.

DESCRIPTION

This function returns an array with the names of all buffers.

EXAMPLE

```
// Returns the array of buffer names
$BufferList = $this->GetBufferList();
```

GetCompatibleParents

Require: Symcon >= 8.2

```
string GetCompatibleParents ()
```

RETURNS

Required connection type and description of compatible instances

DESCRIPTION

The function returns a JSON-encoded object that describes compatible physical parent instances. The management console uses this information to suggest the appropriate parent instances when creating or adapting the instance.
If the function is not implemented, all modules that are compatible according to the data flow are returned. For the module types Splitter and Discovery, a new instance or an instance without connections is required (type = require); for the module types Device and Configurator, existing instances with other connections are also suggested (type = connect). The module types Core, I/O and Visualization do not require a parent instance. In most cases, this heuristic is sufficient. However, if it is not sufficient, the function can be overwritten.

PARAMETER

NAME	DESCRIPTION
type	If "require", only newly created instances and instances without other physically child instances are offered. If "connect" is selected, newly created instances and all existing compatible instances are suggested.
moduleIDs	A list of moduleIDs of compatible parent instances. If moduleIDs is used, no extended parameters can be used. Either moduleIDs or modules must be set, but not both.
modules	A description of possible compatible parent instances. For the possible parameters, see modules. Either moduleIDs or modules must be set, but not both.

MODULES

NAME	DESCRIPTION
moduleID	The moduleID of instances that are compatible

NAME	DESCRIPTION
configuration (optional)	(**default**: {}) An object that describes the required configuration of the instance. Existing instances that do not have the specified configuration are not displayed as compatible. If a new instance is created, this configuration is enforced and cannot be changed by the user.
initial (optional)	(**default**: {}) An object that contains suggested configuration parameters. If a new instance is created, these parameters are initially filled in, but can be adjusted by the user. The object has no effect on existing instances.
formOverride (optional)	(**default**: {}) An object that describes adjustments to the configuration form of the new instance, which are applied in a similar way to UpdateFormField. The keys of the object are the names of the configuration elements. The values are in turn objects that contain the names of the fields as keys and the updated parameter as values. The object has no effect on existing instances.

EXAMPLE

```
// Require a new instance, a serial port
public function GetCompatibleParents() {
return '{"type": "require", "moduleIDs": ["{6DC3D946-0D31-450F-A8C6-
C42DB8D7D4F1}"]}';
}

// Require an MQTT Client with a specific Client ID
// When creating a new gateway, the initial value for the keep alive
interval is 10 seconds, different from the usual default value.
However, it can be adjusted
public function GetCompatibleParents() {
return '{"type": "connect", "modules": [{
"moduleID": "{F7A0DD2E-7684-95C0-64C2-D2A9DC47577B}",
"configuration": {
"ClientID": "b10c75459d64cafb5d78"
},
"initial": {
"KeepAliveInterval": 10
}
}]}';
}

// Require a new instance, a serial port with Baud Rate 2400 or 9600
public function GetCompatibleParents() {
return '{"type": "require", "modules": [{
"moduleID": "{6DC3D946-0D31-450F-A8C6-C42DB8D7D4F1}",
"formOverride": {
"Baudrate": {
"options": [
{
"value": "2400",
"caption": "2400"
},
{
"value": "9600",
"caption": "9600"
```

```
  }
  ]
  }
  }
  }]}';
  }
```

GetConfigurationForm

Require: Symcon >= 4.1

```
string GetConfigurationForm ()
```

RETURNS

Content of the configuration page

DESCRIPTION

The content can be overwritten in order to transfer a self-created configuration page. This way, content can be generated dynamically. In this case, the "form.json" on the file system is completely ignored.



If this function is not defined in a module, the content of form.json is passed on by default.

EXAMPLE

```
// IPSModuleStrict
public function GetConfigurationForm() {

return '{ "actions": [ { "type": "Label", "label": "The current time is
'.date("d.m.y H:i").'" } ] }';

}

// IPSModule
public function GetConfigurationForm() {

// xxx

}
```

GetConfigurationForParent

Require: Symcon >= 4.2

```
string GetConfigurationForParent ()
```

RETURNS

Content of the configuration of the parent instance

DESCRIPTION

The configuration string is transferred to the parent instance. The instance reads the string and sets the entered values. Values read in via this function can no longer be changed via the configuration page.



If this function is not defined in its own module, nothing is changed by default and the parent instance remains freely configurable.

EXAMPLE

```
// IPSModuleStrict
public function GetConfigurationForParent() {

// Parent instance is a "SerialPort"
return "{\"BaudRate\": \"57600\", \"StopBits\": \"1\", \"DataBits\":
\"8\", \"Parity\": \"None\"}";

}

// IPSModule
public function GetConfigurationForParent() {

// xxxx

}
```

GetIDForIdent

Require: Symcon >= 4.0

```
mixed GetIDForIdent (string $Ident)
```

PARAMETERS

Ident	Ident of the object to be searched for.

RETURNS

ID of the found object, otherwise FALSE

DESCRIPTION

The function searches for the object ID of the object with the Ident **Ident** and returns it.
If the Ident is not available, an error message is generated.

EXAMPLE

```
// Output the path of the status variable with the Ident "Status"
echo IPS_GetLocation($this->GetIDForIdent("Status"));
```

GetMessageList

Require: Symcon >= 5.0

```
array GetMessageList ()
```

RETURNS

Array of all active messages. The order of the entries is not guaranteed.

DESCRIPTION

This function returns an array with all active messages that were registered via RegisterMessage.
A list of message IDs is available here: Messages



To cancel a registration, UnregisterMessage can be used.

EXAMPLE

```
// Returns an array of the active registered messages
print_r($this->GetMessageList());

// Sample output:
array(1) {
[12345]=> //ID of the Instance
array(1) {
[0]=>
int(10505) //MessageID => IM_CHANGESTATUS
}
}
```

GetReferenceList

Require: Symcon >= 5.1

```
array GetReferenceList ()
```

RETURNS

Array with integer values of the IDs of all references. The order of the entries is not guaranteed.

DESCRIPTION

This function returns an array with the IDs of all references.

EXAMPLE

```
$ReferenceList = $this->GetReferenceList();
```

GetStatus

Require: Symcon >= 5.1

```
int GetStatus ()
```

RETURNS

Current status

DESCRIPTION

This function returns the current status of the instance.
Table: Instance status

VALUE	CONSTANT	#STATUS
101	IS_CREATING	Instance is creating
102	IS_ACTIVE	Instance is active
103	IS_DELETING	Instance is deleting
104	IS_INACTIVE	Instance is inactive
105	IS_NOTCREATED	Instance was not created
106	IS_STANDBY	Instance is in standby
>=200	IS_EBASE	Instance is faulty

For more information on status codes, see Messages.

EXAMPLE

```
// Queries the current status of the instance
$currentStatus = $this->GetStatus();
```

GetTimerInterval

Require: Symcon >= 5.2

```
int GetTimerInterval (string $Name)
```

PARAMETERS

Name	The name of the timer whose interval is to be queried.

RETURNS

Currently set interval in milliseconds

DESCRIPTION

Queries the interval of the timer **Name**.



As described in RegisterTimer, the timer is stateless and when the IP-Symcon server is restarted, the timer must be set again, otherwise it takes the value of RegisterTimer.

EXAMPLE

```
// Queries the interval of the "Update" timer
echo $this->GetTimerInterval("Update");
```

GetValue

Require: Symcon >= 5.0

```
mixed GetValue (string $Ident)
```

PARAMETERS

Ident	Ident of the status variable

RETURNS

The content of the status variable

DESCRIPTION

This function supplies the value of the status variable with the Ident **Ident**.

EXAMPLE

```
// Returns the value of the status variable "Statusvariable1"
$StatusvariableValue = $this->GetValue("Statusvariable1");
```

GetVisualizationTile

Require: Symcon >= 7.1

```
string GetVisualizationTile ()
```

RETURNS

Initial display of a presentation via HTML SDK

DESCRIPTION

If the HTML-SDK is to be used, this function must be overwritten in order to return the HTML content.



For easier processing, it is usually worth defining the constant part of the display in an HTML file instead of writing it out completely in this function

EXAMPLE

```
// IPSModuleStrict
public function GetVisualizationTile(): string {

// Directly return the contents of the module.html in the same
directory
return file_get_contents('./module.html');

}

// IPSModule
public function GetVisualizationTile() {

// xxxx

}
```

HasActiveParent

Require: Symcon >= 5.1

```
bool HasActiveParent ()
```

RETURNS

TRUE, if all physical parent instances are active, otherwise FALSE

DESCRIPTION

This function checks whether all physical parent instances are active. If at least one instance in the chain is not active, the function returns FALSE. If all instances are active, the function returns TRUE.



If there is no parent, the function also returns FALSE.

EXAMPLE

```
// Abort with an error message if the splitter or I/O are not ready
if (!$this->HasActiveParent()) {
echo "error: Parent instances are not active ";
return;
}

// ...
```

LogMessage

Require: Symcon >= 5.0

```
bool LogMessage (string $Message, int $Type)
```

PARAMETERS

Message	Content of the message

Type

TYPE	VALUE	DESCRIPTION
KL_DEBUG	10206	A debug message
KL_ERROR	10205	An error message
KL_MESSAGE	10201	A standard message
KL_NOTIFY	10203	A notification
KL_WARNING	10204	A warning message

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function writes a message in the log file, specifying the type **Type** and the message **Message**. These are also displayed in the message window within the console.



Depending on the type, the messages differ in their presentation within the message window. (see messages)



KL_DEBUG are written to the log only. If the special switch "LogfileVerbose" is deactivated, these are not even written to the log.



The KL_CUSTOM message type is only available for the IPS_LogMessage function. This cannot be used in modules.

EXAMPLE

```
// Send a warning message in the message window
$this->LogMessage("This is a warning", KL_WARNING);
```

MaintainAction

Require: Symcon >= 4.0

```
bool MaintainAction (string $Ident, bool $ActivateAction)
```

PARAMETERS

Ident	Ident of the variable
ActivateAction	Enable if **True**, Disable if **False**

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

With **ActivateAction** == True the function calls EnableAction and with **ActivateAction** == False the function DisableAction with the Ident **Ident**.

EXAMPLE

```
// We only have an action if the device type == 5
$this->MaintainAction("SpecialData", $this-
>ReadPropertyInteger("DeviceType") == 5);
```

MaintainVariable

Require: Symcon >= 4.0

```
mixed MaintainVariable (string $Ident, string $Name, int $Type, string
$ProfileOrPresentation, int $Position, bool $Retain)
```

PARAMETERS

Ident	Ident of the status variable
Name	Name of the status variable
Type	Type of status variable
ProfileOrPresentation	Name of the variable profile or configuration of the Presentation to be used
Position	Position in the object tree and therefore also in the WebFront
Retain	Register if **True**, Unregister if **False**

RETURNS

CLASS	TYPE	DESCRIPTION
IPSModuleStrict	boolean	Returns whether the variable was created. The return value can be used, for example, to set an initial value.
IPSModule	integer	Variable ID of the created status variable.

DESCRIPTION

If the parameter **Retain** == true, then the matching state variable of type **type** with ident **Ident** and name **name** is created.
If the **Retain** == parameter is false, then the state variable with **Ident** ident is deleted.



The value of **type** contains the variable type

0	VARIABLETYPE_BOOLEAN	Boolean
1	VARIABLETYPE_INTEGER	Integer
2	VARIABLETYPE_FLOAT	Float
3	VARIABLETYPE_STRING	String

See also IPS_CreateVariable or IPS_GetVariable

EXAMPLE

```
// We only have this status variable if the device type is == 5

// IPSModuleStrict
$created = $this->MaintainVariable("Status", "Status of Device",
VARIABLETYPE_BOOLEAN, "Myvariablesprofileforstatus", 0, $this-
>ReadPropertyInteger("DeviceType") == 5);
if ($created) {
// Initial value should be true
$this->SetValue("Status", true);
}

// IPSModule
$variableID = $this->MaintainVariable("Status", "Status of Device",
VARIABLETYPE_BOOLEAN, "Myvariablesprofileforstatus", 0, $this-
>ReadPropertyInteger("DeviceType") == 5);
```

MessageSink

Require: Symcon >= 4.1

```
void MessageSink (int $TimeStamp, int $SenderID, int $MessageID, array
$Data)
```

PARAMETERS

TimeStamp	Continuous counter timestamp
SenderID	Sender ID
MessageID	ID of the message
Data	Data of the message

RETURNS

No Return

DESCRIPTION

The content of the function can be overwritten in order to carry out own reactions to certain messages.
The function is only called for registered MessageIDs/SenderIDs combinations.
The time stamp is incremented for every message sent.
RegisterMessage and UnregisterMessage add or remove filter rules.
A list of the MessageIDs can be found here: Messages



If this function is not defined in its own module, the default MessageSink() is executed.



The content of **Data** can vary depending on the message type.
These are not documented yet.

EXAMPLE

```
// IPSModuleStrict
public function MessageSink(int $TimeStamp, int $SenderID, int
```

```
$Message, array $Data) {

IPS_LogMessage("MessageSink", "Message from SenderID ".$SenderID." with
Message ".$Message."\r\n Data: ".print_r($Data, true));
}

// IPSModule
public function MessageSink($TimeStamp, $SenderID, $Message, $Data) {

// xxx

}
```

Migrate

Require: Symcon >= 7.0

```
string Migrate ()
```

RETURNS

JSON encoded object with the new configuration/attributes.
Empty string unless changes are needed.

DESCRIPTION

This function is only called when IP-Symcon is started or after a module update. It is not called the first time an instance is created. The delivered persistence is independent of the properties/attributes already registered in Create. As a result, it can be adjusted/corrected as desired so that it fits the new properties/attributes. This allows, for example, old properties (e.g. due to a typo) to be corrected in the name or content.
The new values are directly available in the properties/attributes in the subsequent ApplyChanges function.



The Migrate function is called by IP-Symcon. It must therefore be overwritten by the base class in order to add your own extensions

EXAMPLE

```
// IPSModuleStrict
public function Migrate(string $JSONData): string {

// Don't remove this line
parent::Migrate($JSONData);

// Example data for JSONData
/*
{
"attributes": {
"MyAttribute": "MyValue"
},
"configuration": {
"MyConfiguration": "MyValue"
}
}
*/
```

```
// Migrate Configuration/Attributes
$j = json_decode($JSONData);
$j->attributes->NewAttribut = $j->attributes->MyAttribute;
$j->configuration->NewConfiguration = $j->configuration-
>MyConfiguration;
return json_encode($j);

}

// IPSModule
public function Migrate($JSONData) {

// xxx

}
```

ProcessHookData

Require: Symcon >= 4.0

```
void ProcessHookData ()
```

RETURNS

No return

DESCRIPTION

If the WebHook registered by the function RegisterHook in the WebHook Control is called from outside, this function is executed. The output of this function (e.g. through echo) is returned to the caller. The header of the response can also be adapted using corresponding PHP functions.

EXAMPLE

```
// IPSModuleStrict
protected function ProcessHookData(): void {
// Received data
$data = json_decode(file_get_contents('php://input'), true);
$this->LogMessage('Remote Method', utf8_decode($data['method']),
KL_MESSAGE);

if ($data['method'] == 'get_version') {
header('Content-Type: application/json');

echo json_encode([
'result' => IPS_GetKernelVersion(),
'jsonrpc' => '2.0',
'id' => $request['id']
]);
}
}

// IPSModule
protected function ProcessHookData($JSONString) {

// Example is identical. Please note the changed function signature.

}
```

ProcessOAuthData

Require: Symcon >= 4.1

```
void ProcessOAuthData ()
```

RETURNS

No return

DESCRIPTION

If the OAuth handler registered by the function RegisterOAuth is called, this function is executed. The received OAuth credentials can be read, for example, via the corresponding PHP input streams.

EXAMPLE

```
// IPSModuleStrict
protected function ProcessOAuthData(): void {
// OAuth credentials from the request
$token = file_get_contents('php://input');
$this->LogMessage('OAuth token received', KL_MESSAGE);

// Save token
$this->WriteAttributeString('OAuthToken', $token);
}

// IPSModule
protected function ProcessOAuthData($Token) {

// Example is identical. Please note the changed function signature.

}
```

ReadAttributeBoolean

Require: Symcon >= 5.1

```
bool ReadAttributeBoolean (string $Name)
```

PARAMETERS

Name	Name of the attribute

RETURNS

The value of the attribute

DESCRIPTION

The function returns the value of the attribute with the name **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->ReadAttributeBoolean("CurrentState");
```

ReadAttributeFloat

Require: Symcon >= 5.1

```
float ReadAttributeFloat (string $Name)
```

PARAMETERS

Name	Name of the attribute

RETURNS

The value of the attribute

DESCRIPTION

The function returns the value of the attribute with the name **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->ReadAttributeFloat("Median");
```

ReadAttributeInteger

Require: Symcon >= 5.1

```
int ReadAttributeInteger (string $Name)
```

PARAMETERS

Name	Name of the attribute

RETURNS

The value of the attribute

DESCRIPTION

The function returns the value of the attribute named **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->ReadAttributeInteger("SequenceCounter")
```

ReadAttributeString

Require: Symcon >= 5.1

```
string ReadAttributeString (string $Name)
```

PARAMETERS

Name	Name of the attribute

RETURNS

The value of the attribute

DESCRIPTION

The function returns the value of the attribute named **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->ReadAttributeString("Token");
```

ReadPropertyBoolean

Require: Symcon >= 4.0

```
bool ReadPropertyBoolean (string $Name)
```

PARAMETERS

Name	Name of the property

RETURNS

The value of the property

DESCRIPTION

The function returns the value of the property named **Name**.



Changed values are only available when IPS_ApplyChanges has been called

EXAMPLE

```
$this->ReadPropertyBoolean("EmulateStatus");
```

ReadPropertyFloat

Require: Symcon >= 4.0

```
float ReadPropertyFloat (string $Name)
```

PARAMETERS

Name	Name of the property

RETURNS

The value of the property

DESCRIPTION

The function returns the value of the property with the name **Name**.



Changed values are only available when IPS_ApplyChanges has been called

EXAMPLE

```
$this->ReadPropertyFloat("Factor");
```

ReadPropertyInteger

Require: Symcon >= 4.0

```
int ReadPropertyInteger (string $Name)
```

PARAMETERS

Name	Name of the property

RETURNS

The value of the property

DESCRIPTION

The function returns the value of the property with the name **Name**.



Changed values are only available when IPS_ApplyChanges has been called

EXAMPLE

```
$this->ReadPropertyInteger("GatewayMode");
```

ReadPropertyString

Require: Symcon >= 4.0

```
string ReadPropertyString (string $Name)
```

PARAMETERS

Name	Name of the property

RETURNS

The value of the property

DESCRIPTION

The function returns the value of the property with the name **Name**.



Changed values are only available when IPS_ApplyChanges has been called

EXAMPLE

```
$this->ReadPropertyString("Username");
```

ReceiveData

Require: Symcon >= 4.0

```
string ReceiveData (string $JSONString)
```

PARAMETERS

JSONString	Data package in JSON format

RETURNS

Optional response to the parent instance

DESCRIPTION

This function is called by IP-Symcon and processes sent data and, if necessary, forwards it to all child instances.
Data can be sent using the SendDataToChildren function.
Further information on data forwarding can be found under Dataflow.



The ReceiveData function is called by IP-Symcon. It must therefore be overwritten by the base class in order to add individual extensions



Up to IP-Symcon 5.3 this function had no return value.

EXAMPLE

```
// IPSModuleStrict
public function ReceiveData(string $JSONString): string {

// Example within a gateway/splitter instance

// Received data from I/O
$data = json_decode($JSONString);
IPS_LogMessage("ReceiveData", utf8_decode($data->Buffer));
```

```
// This is where the data is processed

// Forwarding to all device-/device-instances
$results = $this->SendDataToChildren(json_encode(Array("DataID" => "
{66164EB8-3439-4599-B937-A365D7A68567}", "Buffer" => $data->Buffer)));

// If a child instance delivers a result, this can be used.
foreach($results as $result) {
IPS_LogMessage("IOSplitter RECV-RES", $result);
}
}

// Example within a device-/device-instance
public function ReceiveData(string $JSONString): string {

// Received data from the gateway/splitter
$data = json_decode($JSONString);
IPS_LogMessage("ReceiveData", utf8_decode($data->Buffer));

// Data processing and writing of the values in the status variables
SetValue($this->GetIDForIdent("Value"), $data->Buffer);

// Send result back to the gateway/splitter
return "OK from " . $this->InstanceID;
}

// IPSModule
public function ReceiveData($JSONString) {

// xxxx

}
```

RegisterAttributeBoolean

Require: Symcon >= 5.1

```
bool RegisterAttributeBoolean (string $Name, bool $DefaultValue)
```

PARAMETERS

Name	Name of the attribute
DefaultValue	Default value of the attribute

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function creates an attribute of the type boolean with name **Name** and default value **DefaultValue**



This function can only be called in the Create method!

EXAMPLE

```
public function Create() {

// Don't delete or change this line.
parent::Create();

$this->RegisterAttributeBoolean("CurrentState", true);

}
```

RegisterAttributeFloat

Require: Symcon >= 5.1

```
bool RegisterAttributeFloat (string $Name, float $DefaultValue)
```

PARAMETERS

Name	Name of the attribute
DefaultValue	Default value of the attribute

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates an attribute of the type float with the name **Name** and the default value **DefaultValue**



This function can only be called in the Create method!

EXAMPLE

```
public function Create() {

// Don't delete or change this line.
parent::Create();

$this->RegisterAttributeFloat("Median", 0.5);

}
```

RegisterAttributeInteger

Require: Symcon >= 5.1

```
bool RegisterAttributeInteger (string $Name, int $DefaultValue)
```

PARAMETERS

Name	Name of the attribute
DefaultValue	Default value of the attribute

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates an attribute of the type integer with the name **Name** and the default value **DefaultValue**



This function can only be called in the Create method!

EXAMPLE

```
public function Create() {

// Don't delete or change this line.
parent::Create();

$this->RegisterAttributeInteger("SequenceCounter", 0);
}
```

RegisterAttributeString

Require: Symcon >= 5.1

```
bool RegisterAttributeString (string $Name, string $DefaultValue)
```

PARAMETERS

Name	Name of the attribute
DefaultValue	Default value of the attribute

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates an attribute of the type string with the name **Name** and the default value **DefaultValue**



This function can only be called in the Create method!

EXAMPLE

```
public function Create() {

// Don't delete or change this line.
parent::Create();

$this->RegisterAttributeString("Token", "");

}
```

RegisterHook

Require: Symcon >= 8.1

```
bool RegisterHook (string $Address)
```

PARAMETERS

Address	Address of the hook

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

This function registers a webhook with the address in Webhook Control. This can then be accessed from outside via 127.0.0.1:3777/hook/*address*, where *address* describes the address parameter. If this registered webhook is called, the function ProcessHookData of this module is called.
The registered webhook is volatile. On a restart, it is removed until the RegisterHook function is executed again by the instance, which typically happens as part of the restart in Create. This means that the module does not have to clean up when it is deleted and the WebHook is automatically removed when the instance is deleted.



This function can normally only be used for IPSModuleStrict. For the old IPSModule, however, the base class WebHookModule can be downloaded and extended. However, the functionality of the base class is not recommended for IPSModuleStrict, as a webhook registered in this way is not volatile and must therefore be cleaned up manually.

EXAMPLE

```
// IPSModuleStrict
public function Create(): void {

// Do not remove this line
parent::Create();

$this->RegisterHook('my-module');
```

```
}

// IPSModule
// This function can only be used natively for IPSModuleStrict
// Using the base class WebHookModule enables a comparable use in
IPSModule
```

RegisterMessage

Require: Symcon >= 4.1

```
bool RegisterMessage (int $SenderID, int $MessageID)
```

PARAMETERS

SenderID	ID of the Sender
MessageID	ID of the message

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function registers the module for the receipt of the message **MessageID** and the ID **SenderID**.
A list of the MessageIDs can be found here: Messages



Multiple calls with the same parameters only register once

EXAMPLE

```
$this->RegisterMessage(12345 /* InstanzID */, IM_CHANGESTATUS);
```

RegisterOAuth

Require: Symcon >= 8.1

```
bool RegisterOAuth (string $Identifier)
```

PARAMETERS

Identifier	Identifier of the OAuth handler

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

This function registers an OAuth handler with the identifier in the OAuth Control. When the OAuth process is completed, the function ProcessOAuthData of this module is called.
The registered OAuth handler is volatile. On a restart, it is removed until the RegisterOAuth function is executed again by the instance, which typically happens as part of the restart in Create. This means that the module does not have to clean up when it is deleted and the OAuth handler is automatically removed when the instance is deleted.



This function can normally only be used for IPSModuleStrict. For the old IPSModule, however, the base class WebOAuthModule can be downloaded and extended. However, the functionality of the base class is not recommended for IPSModuleStrict, as an OAuth handler registered in this way is not volatile and must therefore be cleaned up manually.

EXAMPLE

```
// IPSModuleStrict
public function Create(): void {

// Do not remove this line
parent::Create();

$this->RegisterOAuth('my-module');
}

// IPSModule
```

```
// This function can only be used natively for IPSModuleStrict
// Using the base class WebOAuthModule enables a comparable use in
IPSModule
```

RegisterOnceTimer

Require: Symcon >= 5.5

```
bool RegisterOnceTimer (string $Name, string $ScriptContent)
```

PARAMETERS

Name	Name of the timer
ScriptContent	PHP script without PHP tags (<?php ...)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates a timer with name **name** and content **ScriptContent**. The timer is executed once and immediately.



The content of **script content** is overwritten every time, therefore the correct content can always be guaranteed, even after changes.

EXAMPLE

```
// Creates a timer called "Update".
$this->RegisterOnceTimer("Update", "echo 'Hallo World';");
```

RegisterPropertyBoolean

Require: Symcon >= 4.0

```
bool RegisterPropertyBoolean (string $Name, bool $DefaultValue)
```

PARAMETERS

Name	Name of the property
DefaultValue	Default value of the property

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates a module property of the type Boolean with name **Name** and default value **DefaultValue**



This function can only be called in the Create method!

EXAMPLE

```
public function Create(): void {

// Don't delete or change this line.
parent::Create();

$this->RegisterPropertyBoolean("EmulateStatus", true);

}
```

RegisterPropertyFloat

Require: Symcon >= 4.0

```
bool RegisterPropertyFloat (string $Name, float $DefaultValue)
```

PARAMETERS

Name	Name of the property
DefaultValue	Default value of the property

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates a property of type float with name **Name** and default value **DefaultValue**



This function can be called in the Create method only!

EXAMPLE

```
public function Create(): void {

//Don't delete or change this line.
parent::Create();

$this->RegisterPropertyFloat("Factor", 0.5);

}
```

RegisterPropertyInteger

Require: Symcon >= 4.0

```
bool RegisterPropertyInteger (string $Name, int $DefaultValue)
```

PARAMETERS

Name	Name of the property
DefaultValue	Default value of the property

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates a property of type Integer with name **Name** and default value **DefaultValue**



This function can be called in the Create method only!

EXAMPLE

```
public function Create(): void {

// Don't delete or change this line.
parent::Create();

$this->RegisterPropertyInteger("GatewayMode", 0);
}
```

RegisterPropertyString

Require: Symcon >= 4.0

```
bool RegisterPropertyString (string $Name, string $DefaultValue)
```

PARAMETERS

Name	Name of the property
DefaultValue	Default value of the property

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function creates a property of type String with name **Name** and default value **DefaultValue**



This function can be called in the Create method only!

EXAMPLE

```
public function Create(): void {

// Don't delete or change this line.
parent::Create();

$this->RegisterPropertyString("Username", "MaxMustermann");

}
```

RegisterReference

Require: Symcon >= 5.1

```
bool RegisterReference (int $ID)
```

PARAMETERS

ID	ID of the object

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function registers the ID ID as referenced, enabling the module to be found during a reference search.



A repeated call with the same ID still only registers once

EXAMPLE

```
$this->RegisterReference(10505 /* ObjectID */);
```

RegisterScript

Require: Symcon >= 4.0

```
mixed RegisterScript (string $Ident, string $Name, string $Contents,
int $Position)
```

PARAMETERS

Ident	Ident of the script
Name	Name of the script
Contents	Content to be entered in the script. **Default** == "<?php //Autogenerated script"
Position	Position in the object tree and thus also in the visualization. **Default** == 0

RETURNS

CLASS	TYPE	DESCRIPTION
IPSModule	integer	Returns the object ID of the created or existing script.
IPSModuleStrict	boolean	Returns whether the script was created. The return value can be used, for example, to perform initial configurations.

DESCRIPTION

Creates a script with the Ident **Ident**, name **Name**, content **Content** and the position **Postion**, if the specified Ident does not already exist.
In both cases, the ObjectID of the script that has been created or that already exists is returned.



The content of **Content** is overwritten every time, always guaranteeing the correct content, even after changes. For scripts which are to be processed, it is recommended to use a configuration form with SelectScript-selection

EXAMPLE

```
// IPSModuleStrict
$created = $this->RegisterScript("TestScript", "My TestScript");
if ($created) {
// Do things, after initially creating the script
}

// IPSModule
$scriptID = $this->RegisterScript("TestScript", "My TestScript");
```

RegisterTimer

Require: Symcon >= 4.0

```
bool RegisterTimer (string $Name, int $Interval, string $ScriptContent)
```

PARAMETERS

Name	Name of the timer
Interval	Interval in milliseconds with which the timer should be created. 0 = never
ScriptContent	PHP script without PHP tags (<?php ...)

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function creates a timer with the name **Name**. It can also be specified which **interval** is to be set when creating and the **script content** will be defined.
The timer is stateless. Therefore, following each restart the timer is set anew with the **interval** previously set here. Interval settings must be reloaded in ApplyChanges().



The timer can only be created in the "Create()" function. If the timer should not start when created, the **Interval** parameter must be set to 0.



To change the interval at a later point, SetTimerInterval can be used.



The content of **script content** is overwritten every time, therefore the correct content can always be guaranteed, even after changes.

EXAMPLE

```
// Creates a timer named "Update" with an interval of 5 seconds.
$this->RegisterTimer("Update", 5000, "echo 'Hello World';");
```

RegisterVariableBoolean

Require: Symcon >= 4.0

```
mixed RegisterVariableBoolean (string $Ident, string $Name, array
$Presentation, int $Position)
```

PARAMETERS

Ident	Ident of the status variable
Name	Name of the status variable
Presentation	The configuration of the presentation as an array. A profile can be set via the Legacy profile. **Default** == ""
Position	Position in the object tree and therefore also in the visualization. **Default** == 0

RETURNS

CLASS	TYPE	DESCRIPTION
IPSModuleStrict	boolean	Returns whether the variable was created. The return value can be used, for example, to set an initial value.
IPSModule	integer	Variable ID of the created status variable.

DESCRIPTION

This function creates a status variable of type Boolean with the ident **Ident** and the name **Name**. You can also specify which **Presentation** is to be used and on which **Position** the status variable is to be placed.
The structure of the presentation is described in more detail here.



Presentations can be used from version 8.0. In previous versions, profiles can be set as a string via the **Presentation** parameter.

This function can also be called if the variable already exists. In this case, only the presentation is updated if it is different. If the variable type is different, the variable is deleted without prompting and recreated with the correct variable type.

The variable ID can be determined via GetIDForIdent so that it does not have to be saved anywhere.

EXAMPLE

```
// IPSModuleStrict
$created = $this->RegisterVariableBoolean("Switch", "Light Switch",
["PRESENTATION" => VARIABLE_PRESENTATION_SWITCH]);
if ($created) {
// Set initial value
$this->SetValue("Switch", true);
}

// IPSModule
$variableID = $this->RegisterVariableBoolean("Switch", "Light Switch",
["PRESENTATION" => VARIABLE_PRESENTATION_SWITCH]);
```

RegisterVariableFloat

Require: Symcon >= 4.0

```
mixed RegisterVariableFloat (string $Ident, string $Name, array
$Presentation, int $Position)
```

PARAMETERS

Ident	Ident of the status variable
Name	Name of the status variable
Presentation	The configuration of the display as an array. A profile can be set via the Legacy profile. **Default** == ""
Position	Position in the object tree and therefore also in the visualization. **Default** == 0

RETURNS

CLASS	TYPE	DESCRIPTION
IPSModuleStrict	boolean	Returns whether the variable was created. The return value can be used, for example, to set an initial value.
IPSModule	integer	Variable ID of the created status variable.

DESCRIPTION

This function creates a status variable of type Float with the ident **Ident** and the name **Name**. You can also specify which **Presentation** is to be used and at which **Position** the status variable is to be placed.
The structure of the presentations is described in more detail here.



Presentations can be used from version 8.0. In previous versions, profiles can be set as a string via the **Presentation** parameter.

This function can also be called if the variable already exists. In this case, only the presentation is updated if this is different. If the variable type is different, the variable is deleted without prompting and recreated with the correct variable type.

The variable ID can be determined via GetIDForIdent so that it does not have to be saved anywhere.

EXAMPLE

```
// IPSModuleStrict
$created = $this->RegisterVariableFloat("Factor", "Zoom Factor",
["PRESENTATION"=> VARIABLE_PRESENTATION_SLIDER, 'SUFFIX' => ' %']);
if ($created) {
// Set initial value
$this->SetValue("Factor", 5.8);
}

// IPSModule
$variableID = $this->RegisterVariableFloat("Factor", "Zoom Factor",
["PRESENTATION"=> VARIABLE_PRESENTATION_SLIDER, 'SUFFIX' => ' %']);
```

RegisterVariableInteger

Require: Symcon >= 4.0

```
mixed RegisterVariableInteger (string $Ident, string $Name, array
$Presentation, int $Position)
```

PARAMETERS

Ident	Ident of the status variable
Name	Name of the status variable
Presentation	The configuration of the display as an array. A profile can be set via the Legacy profile. **Default** == ""
Position	Position in the object tree and therefore also in the visualization. **Default** == 0

RETURNS

CLASS	TYPE	DESCRIPTION
IPSModuleStrict	boolean	Returns whether the variable was created. The return value can be used, for example, to set an initial value.
IPSModule	integer	Variable ID of the created status variable.

DESCRIPTION

This function creates a status variable of type integer with the ident **Ident** and the name **Name**. You can also specify which **Presentation** is used and on which **Position** the status variable should be placed.
The structure of the presentations is described in more detail here.



Presentations can be used from version 8.0. In previous versions, profiles can be set as a string via the **Presentation** parameter.

This function can also be called if the variable already exists. In this case, only the presentation is updated if this is different. If the variable type is different, the variable is deleted without prompting and recreated with the correct variable type.

The variable ID can be determined via GetIDForIdent so that it does not have to be saved anywhere.

EXAMPLE

```
// IPSModuleStrict
$created = $this->RegisterVariableInteger("Brightness", "Lamp
Brightness", ["PRESENTATION"=> VARIABLE_PRESENTATION_SLIDER, 'SUFFIX'
=> ' lx']);
if ($created) {
// Set initial value
$this->SetValue("Brightness", 50);
}

// IPSModule
$variablenID = $this->RegisterVariableInteger("Brightness", "Lamp
Brightness", ["PRESENTATION"=> VARIABLE_PRESENTATION_SLIDER, 'SUFFIX'
=> ' lx']);
```

RegisterVariableString

Require: Symcon >= 4.0

```
mixed RegisterVariableString (string $Ident, string $Name, array
$Presentation, int $Position)
```

PARAMETERS

Ident	Ident of the status variable
Name	Name of the status variable
Presentation	The configuration of the display as an array. A profile can be set via the Legacy profile. **Default** == ""
Position	Position in the object tree and therefore also in the visualization. **Default** == 0

RETURNS

CLASS	TYPE	DESCRIPTION
IPSModuleStrict	boolean	Returns whether the variable was created. The return value can be used, for example, to set an initial value.
IPSModule	integer	Variable ID of the created status variable.

DESCRIPTION

This function creates a status variable of type String with the ident **Ident** and the name **Name**. You can also specify which **Presentation** is to be used and at which **Position** the status variable is to be placed.
The structure of the presentations is described in more detail here.



Presentations can be used from version 8.0. In previous versions, profiles can be set as a string via the **Presentation** parameter.

This function can also be called if the variable already exists. In this case, only the presentation is updated if this is different. If the variable type is different, the variable is deleted without prompting and recreated with the correct variable type.

The variable ID can be determined via GetIDForIdent so that it does not have to be saved anywhere.

EXAMPLE

```
// IPSModuleStrict
$created = $this->RegisterVariableString("Name", "My Name",
["PRESENTATION"=> VARIABLE_PRESENTATION_VALUE_INPUT]);
if ($created) {
// Set initial value
$this->SetValue("Name", "Peter");
}

// IPSModule
$variablenID = $this->RegisterVariableString("Name", "My Name",
["PRESENTATION"=> VARIABLE_PRESENTATION_VALUE_INPUT]);
```

ReloadForm

Require: Symcon >= 5.2

```
bool ReloadForm ()
```

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function reloads the instance configuration form in each open instance configuration. Current entries will get lost.

EXAMPLE

```
// Reload the form
$this->ReloadForm();
```

RequestAction

```
void RequestAction (string $Ident, mixed $Value)
```

PARAMETERS

Ident	Ident of the variable
Value	The value to be set

RETURNS

No Return

DESCRIPTION

Is called when, for example, a button is clicked in the visualization.



The RequestAction function is called by IP-Symcon. It must therefore be overwritten by the base class in order to add one's own extensions

EXAMPLE

```
// IPSModuleStrict
public function RequestAction(string $Ident, mixed $Value): void {

switch($Ident) {
case "TestVariable":
// An action, e.g. switching, would normally be carried out here
// Outputs via 'echo' are returned to the visualization

// Write new value to the status variable
SetValue($this->GetIDForIdent($Ident), $Value);
break;
default:
throw new Exception("Invalid Ident");
}

}

// IPSModule
public function RequestAction($Ident, $Value) {
```

```
// xxxxx

}
```

RequireParent

Require: Symcon >= 4.0

```
bool RequireParent (string $ParentGUID)
```

PARAMETERS

ParentGUID	GUID

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION



This function is not available for IPSModuleStrict. Please consider using the new GetCompatibleParents function

If there is no connection, a parent instance with the GUID **ParentGUID** is created. Even if there was a compatible parent instance.



The normal case for the function is a splitter module, which requires its own I/O instance when it is created. Alternatively, ForceParent may also be suitable.



Normally, this function is called only in the Create() or ApplyChanges() method.

EXAMPLE

```
// IPSModuleStrict
ConnectParent/RequireParent/ForceParent is not available,
but can be replaced by the new function “GetCompatibleParents()”.

// IPSModule
```

```
public function Create() {

// Never remove the line!
parent::Create();

// Connect to the newly created splitter if there is no connection yet
$this->RequireParent("{46C969BF-3465-4E3E-B2A5-E404FB969735}");

}
```

SendDataToChildren

Require: Symcon >= 4.0

```
array SendDataToChildren (string $Data)
```

PARAMETERS

Data	JSON encoded string

RETURNS

Results which ReceiveData delivers from the parent instances

DESCRIPTION

This function sends data to all connected child instances.
Further information is available under Data flow.



Up to IP-Symcon 5.3 this function had no return value.

EXAMPLE

```
// The example can be found at ReceiveData
```

SendDataToParent

Require: Symcon >= 4.0

```
string SendDataToParent (string $Data)
```

PARAMETERS

Data	JSON encoded string

RETURNS

Result which ForwardData delivers from the parent instance

DESCRIPTION

This function sends data to a connected parent instance.
Further information can be found under Data flow.

EXAMPLE

```
// Example can be found at ForwardData
```

SendDebug

Require: Symcon >= 4.0

```
bool SendDebug (string $MessageName, string $Data, int $Format)
```

PARAMETERS

MessageName	Name/title of the debug output
Data	Content of the debug message
Format	Presetting for automatic formation selection (0 = DEBUG_FORMAT_TEXT, 1 = DEBUG_FORMAT_BINARY)

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Outputs the debug message **Data** with the message name **MessageName** and the format **Format**.
Format indicates whether text or hex output is selected for automatic display.

EXAMPLE

```
// Example from the presence simulation
$this->SendDebug("Fetch", "Fetched day -".$day." with
".sizeof($data['Data'])." valid device(s)", DEBUG_FORMAT_TEXT);

// Sample output automatically configured as text output in the debug
window
"Fetch" - "Fetched day -28 with 2 valid device(s)"
```

SetBuffer

Require: Symcon >= 4.1

```
bool SetBuffer (string $Name, string $Data)
```

PARAMETERS

Name	Name of the buffer
Data	Data which should be packed into the buffer.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function writes data **Data** into the buffer with the name **Name**.



The buffer has a soft limit of 256kB and a hard limit of 1024kB. Therefore, from a size of 256kB, IP-Symcon dumps a "Warning" in the messages and the maximum buffer size is 1024kB. Data larger than 1024kB will be cut off.



The content of the buffer is only available as long as IP-Symcon is running. After a restart all information disappears.

EXAMPLE

```
// Writes "Hello World" to the "Databuffer" buffer
$this->SetBuffer("DataBuffer", "Hello World");
```

SetForwardDataFilter

Require: Symcon >= 4.1

```
bool SetForwardDataFilter (string $RequiredRegexRule)
```

PARAMETERS

RequiredRegexRule	RegexRule which should be used as a filter

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sets the filter using the **RequiredRegexRule** regular expression. ForwardData() is only called if the set filter is matching.



The page www.regex101.com can be used to test the RegEx-expression.



In certain special cases PHP json_encode() may change from upper to lower case. For example, the special character encoded JSON_string ""\u001D"" becomes ""\u001d"" after a json_decode.

EXAMPLE

```
// Add filter for ForwardData
public function ApplyChanges(): void {

[...]
$this->SetForwardDataFilter(".*Hello.*");
[...]

}

// Only called if “Hello” is found in the $JSONString
public function ForwardData(string $JSONString): string {
```

```
return "OK";

}
```

SetReceiveDataFilter

Require: Symcon >= 4.1

```
bool SetReceiveDataFilter (string $RequiredRegexRule)
```

PARAMETERS

RequiredRegexRule	Regexrule which should be used as a filter

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

Sets the filter using the **RequiredRegexRule** regular expression. RecieveData() is only called if the set filter is true.



The page www.regex101.com can be used to test the RegEx-expression.



In certain special cases PHP json_encode() may change from upper to lower case. For example, the special character encoded JSON_string ""\u001D"" becomes ""\u001d"" after a json_decode.

EXAMPLE

```
// Add filters for ReceiveData
public function ApplyChanges(): void {

[...]
$this->SetReceiveDataFilter(".*Hello.*");
[...]

}

// Only called if “Hello” is found in the $JSONString
public function ReceiveData(string $JSONString): string {
```

```
return "";

}
```

SetStatus

Require: Symcon >= 4.0

```
bool SetStatus (int $StatusValue)
```

PARAMETERS

StatusValue	The status value to which the instance should be set.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the status of an instance to the value of **Status value**.
Table: Instance status

VALUE	CONSTANT	STATUS
101	IS_CREATING	Instance is creating
102	IS_ACTIVE	Instance is active
103	IS_DELETING	Instance is deleting
104	IS_INACTIVE	Instance is inactive
105	IS_NOTCREATED	Instance was not created
106	IS_STANDBY	Instance is in standby
>=200	IS_EBASE	Instance is faulty

An error status over 200 can be self-defined.
This also has a direct influence on the display in the configuration forms in the "status"-area

EXAMPLE

```
// sets the status to "inactive"
$this->SetStatus(IS_INACTIVE);
```

SetSummary

Require: Symcon >= 4.1

```
bool SetSummary (string $ShortInfo)
```

PARAMETERS

ShortInfo	Content to be set

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the short information **ShortInfo** of the object, which is displayed in the administration console and can be read out again using IPS_GetObject.

EXAMPLE

```
// puts a possible IP address in the short description to ensure quick
recognition.
$this->SetSummary($this->ReadPropertyString("IPAddress"));
```

SetTimerInterval

Require: Symcon >= 4.0

```
bool SetTimerInterval (string $Name, int $Interval)
```

PARAMETERS

Name	The name of the timer whose interval is to be set.
Interval	The interval, at which the timer should be set, in milliseconds.

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

Sets the interval of the timer **Name** to the value of **Interval**.



As described in RegisterTimer, the timer is stateless and when the IP-Symcon server is restarted, the timer must be set again, otherwise it takes the value of RegisterTimer.

EXAMPLE

```
// Sets the interval of the "Update" timer to 5 seconds
$this->SetTimerInterval("Update", 5000);
```

SetValue

Require: Symcon >= 5.0

```
bool SetValue (string $Ident, mixed $Value)
```

PARAMETERS

Ident	Ident of the status variable
Value	Value to be written into the status variable.

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function writes the Value **Value** into the status variable with the Ident **Ident**.

EXAMPLE

```
// Writes 123 into the status variable "Statusvariable1"
$this->SetValue("Statusvariable1", 123);
```

SetVisualizationType

Require: Symcon >= 7.1

```
SetVisualizationType (int $VisualizationType)
```

PARAMETERS

VisualizationType	Visualization type for individual visualization 0 (INSTANCE_VISUALIZATION_TYPE_NONE): no individual visualization 1 (INSTANCE_VISUALIZATION_TYPE_HTML): visualization via HTML SDK 2 (INSTANCE_VISUALIZATION_TYPE_HTML_FULLSCREEN): Visualization via HTML SDK in normal tile view and full-screen mode (since 9.1) Constants

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function can be used to display the instance individually. Depending on the VisualizationType specified, the instance does not offer an individual display at 0 (INSTANCE_VISUALIZATION_TYPE_NONE). Otherwise, an individual display is offered, which is implemented at 1 (INSTANCE_VISUALIZATION_TYPE_HTML) via HTML-SDK.

EXAMPLE

```
// Acivate HTML-SDK
$this->SetVisualizationType(INSTANCE_VISUALIZATION_TYPE_HTML);
```

Translate

Require: Symcon >= 4.3

```
string Translate (string $Text)
```

PARAMETERS

Text	Text to be translated

RETURNS

Translated text

DESCRIPTION

Translates the string **Text** using the selected system language and locale.json.



If no locale.json is available or the string cannot be found in locale.json, the original text is returned.

EXAMPLE

```
// module.php
$label = sprintf($this->Translate("The current time is %s"),
date("d.m.y H:i"));

// locale.json
{
"translations": {
"de": {
"The current time is %s": "Die aktuelle Zeit ist %s"
}
}
}
```

UnregisterHook

Require: Symcon >= 8.2

```
bool UnregisterHook (string $Address)
```

PARAMETERS

Address	Address of the hook

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

This function unregisters a previously registered WebHook with the address in WebHook Control. Since registered WebHooks are transient, it is normally not necessary to use UnregisterHook. Unless, for example, you want to change the hook in ApplyChanges during runtime. In this case, the old hook can be unregistered and the new one registered.



This function can only be used for IPSModuleStrict.

EXAMPLE

```
// IPSModuleStrict
public function ApplyChanges(): void {
...
$this->UnregisterHook('my-module');
...
}
```

UnregisterMessage

Require: Symcon >= 4.1

```
bool UnregisterMessage (int $SenderID, int $MessageID)
```

PARAMETERS

SenderID	ID of the sender
MessageID	ID of the message

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function removes a registration of the module for receiving the message MessageID and the ID SenderID.
A list of the MessageIDs can be found here: Messages

EXAMPLE

```
// The module no longer "listens" for messages from instance 12345 with
MessageID IM_CHANGESTATUS
$this->UnregisterMessage(12345 /* InstanceID */, IM_CHANGESTATUS);

// All messages from the module should be deleted
foreach ($this->GetMessageList() as $senderID => $messages) {
foreach ($messages as $message) {
$this->UnregisterMessage($senderID, $message);
}
}
```

UnregisterOAuth

Require: Symcon >= 8.2

```
bool UnregisterOAuth (string $Identifier)
```

PARAMETERS

Identifier	Identifier of the OAuth handler

RETURNS

If the command could be executed successfully, the result is TRUE, otherwise FALSE.

DESCRIPTION

This function unregisters a previously registered OAuth handler with the identifier in the OAuth Control. Since registered OAuth handlers are transient, it is normally not necessary to use UnregisterOAuth. Unless, for example, you want to change the handler in ApplyChanges during runtime. In this case, the old handler can be unregistered and the new one registered.



This function can only be used for IPSModuleStrict.

EXAMPLE

```
// IPSModuleStrict
public function ApplyChanges(): void {
...
$this->UnregisterOAuth('my-module');
...
}
```

UnregisterReference

Require: Symcon >= 5.1

```
bool UnregisterReference (int $ID)
```

PARAMETERS

ID	ID of the object

RETURNS

If the command succeeds, it returns TRUE, otherwise FALSE.

DESCRIPTION

This function removes the ID ID as referenced for the module.

EXAMPLE

```
$this->UnregisterReference(10505 /* ObjectID */);
```

UnregisterVariable

Require: Symcon >= 4.0

```
bool UnregisterVariable (string $Ident)
```

PARAMETERS

Ident	Ident of the status variable

RETURNS

CLASS	TYPE	DESCRIPTION
IPSModuleStrict	boolean	Returns whether the variable has been removed. The return value can be used, for example, to clean up other things.
IPSModule	boolean	If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The command deletes the status variable with the Ident **Ident**

EXAMPLE

```
$this->UnregisterVariable("Temperature");
```

UpdateFormField

Require: Symcon >= 5.2

```
bool UpdateFormField (string $Field, string $Parameters, mixed $Value)
```

PARAMETERS

Field	Name of the form field to be changed
Parameters	Name of the parameter to be changed
Value	New value of the parameter

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sets the parameter **Parameter** of the form field named **Field** to the value **Value** in every open instance configuration of this instance. If a form field with this name exists in the element and action areas, both are updated.



It is not possible to change the **Name** and **Type** parameters of form fields. Other parameters that cannot be changed are listed in the respective documentation.



For complex parameters, i.e. lists or objects, the new value must be passed in JSON-encoded form.

EXAMPLE

```
// hide the button
$this->UpdateFormField("MyButton", "visible", false);
```

UpdateVisualizationValue

Require: Symcon >= 7.1

```
bool UpdateVisualizationValue (mixed $Data)
```

PARAMETERS

Data	Any data that is sent to the visualization

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

This function sends **data** to the visualization via HTML SDK. The visualization must in turn have implemented the function handleMessage for receiving. This function is then called with **Data** as a parameter.



There is no fixed definition of how the data must be coded. Therefore, when developing a module, you can choose your own definition that suits the module well

EXAMPLE

```
// In this example, a simple number is passed as the count
$this->UpdateVisualizationValue(5);
```

WriteAttributeBoolean

Require: Symcon >= 5.1

```
bool WriteAttributeBoolean (string $Name, bool $Value)
```

PARAMETERS

Name	Name of the attribute
Value	Value of the attribute

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function writes the value **Value** in the attribute with the name **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->WriteAttributeBoolean("CurrentState", false);
```

WriteAttributeFloat

Require: Symcon >= 5.1

```
bool WriteAttributeFloat (string $Name, float $Value)
```

PARAMETERS

Name	Name of the attribute
Value	Value of the attribute

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function writes the value **Value** in the attribute with the name **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->WriteAttributeFloat("Median", 5.5);
```

WriteAttributeInteger

Require: Symcon >= 5.1

```
bool WriteAttributeInteger (string $Name, int $Value)
```

PARAMETERS

Name	Name of the attribute
Value	Value of the attribute

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function writes the value **Value** in the attribute with the name **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->WriteAttributeInteger("SequenceCounter", 4);
```

WriteAttributeString

Require: Symcon >= 5.1

```
bool WriteAttributeString (string $Name, string $Value)
```

PARAMETERS

Name	Name of the attribute
Value	Value of the attribute

RETURNS

If the command succeeds, it returns **TRUE**, otherwise **FALSE**.

DESCRIPTION

The function writes the value **Value** in the attribute with the name **Name**.



This function cannot be called in the Create method



Changed values, as opposed to properties, are available immediately

EXAMPLE

```
$this->WriteAttributeString("Token",
"08da50bd109c7fb1bec49d15ae86e55f");
```

Messages

Require: Symcon >= 4.1

DESCRIPTION

All messages have their own ID, which describes the type of message.
This is a five-digit number and is made up as follows:
Base + category + type = total value



With version 5.0 and above the values are available as constants automatically

OVERVIEW OF ALL MESSAGEIDS

MESSAGE BASE

Messages from IP-Symcon have the value base 10000, those from module instances (still undocumented) have the value base 20000.

DEFINE	VALUE	DESCRIPTION
IPS_BASE	10000	Value base for the kernel
IPS_MODULBASE	20000	Value base for modules

MESSAGE CATEGORIES

DEFINE	VALUE	DESCRIPTION
IPS_BASE	0	Service messages
IPS_KERNELMESSAGE	100	Kernel Manager
IPS_LOGMESSAGE	200	Message Manager
IPS_MODULEMESSAGE	300	Module Manager
IPS_OBJECTMESSAGE	400	Object Manager
IPS_INSTANCEMESSAGE	500	Instance Manager
IPS_SEARCHMESSAGE	510	Search Manager
IPS_VARIABLEMESSAGE	600	Variable Manager
IPS_SCRIPTMESSAGE	700	Script Manager
IPS_EVENTMESSAGE	800	Event Manager
IPS_MEDIAMESSAGE	900	Media Manager

DEFINE	VALUE	DESCRIPTION
IPS_LINKMESSAGE	1000	Link Manager
IPS_FLOWMESSAGE	1100	Flow Manager
IPS_ENGINEMESSAGE	1200	Script Engine
IPS_PROFILEMESSAGE	1300	Profile Pool
IPS_TIMERMESSAGE	1400	Timer Pool
IPS_ACTIONMESSAGE	1500	Action Pool
IPS_LICENSEMESSAGE	1600	License Pool

MESSAGE TYPES

IPS_BASE

Value base 0

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
IPS_KERNELSTARTED (ab 4.2)	1	10001	Is sent after KR_READY and processed synchronously
IPS_KERNELSHUTDOWN (ab 4.2)	2	10002	Is sent before KR_UNINIT and processed synchronously

IPS_KERNELMESSAGE

Value base 100

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
KR_CREATE	1	10101	Kernel was created
KR_INIT	2	10102	Kernel components are initialized, modules are loaded and settings are read
KR_READY	3	10103	Kernel is ready and running
KR_UNINIT	4	10104	"Shutdown"-command received, finalize everything loaded
KR_SHUTDOWN	5	10105	Finalization complete, remove kernel

IPS_LOGMESSAGE

Value base 200

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
KL_MESSAGE	1	10201	Normal message
KL_SUCCESS	2	10202	Success

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
KL_NOTIFY	3	10203	Change Notification
KL_WARNING	4	10204	Warning
KL_ERROR	5	10205	Error Message
KL_DEBUG	6	10206	Debug Information
KL_CUSTOM	7	10207	Other Messages

IPS_MODULEMESSAGE

Value base 300

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
ML_LOAD	1	10301	Module loaded
ML_UNLOAD	2	10302	Module unloaded

IPS_OBJECTMESSAGE

Value base 400

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
OM_REGISTER	1	10401	Object created
OM_UNREGISTER	2	10402	Object removed
OM_CHANGEPARENT	3	10403	Parent object has changed
OM_CHANGENAME	4	10404	Name has changed
OM_CHANGEINFO	5	10405	Description has changed
OM_CHANGETYPE	6	10406	Type has changed
OM_CHANGESUMMARY	7	10407	Short information has changed
OM_CHANGEPOSITION	8	10408	Type has changed
OM_CHANGEREADONLY	9	10409	"Read-only" status has changed
OM_CHANGEHIDDEN	10	10410	Visibility has changed
OM_CHANGEICON	11	10411	Icon has changed
OM_CHILDADDED	12	10412	Child added
OM_CHILDREMOVED	13	10413	Child removed
OM_CHANGEIDENT	14	10414	Ident has changed
OM_CHANGEDISABLED	15	10415	Usability has changed
OM_CHANGELOCKED	16	10416	Locked was Changed

IPS_INSTANCEMESSAGE

Value base 500

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
IM_CREATE	1	10501	Instance created
IM_DELETE	2	10502	Instance removed
IM_CONNECT	3	10503	Instance interface available
IM_DISCONNECT	4	10504	Instance interface no longer available
IM_CHANGESTATUS	5	10505	Status has changed
IM_CHANGESETTINGS	6	10506	Settings have changed
IM_CHANGEATTRIBUTE	7	10507	Attribute has changed
IM_ADDATTRIBUTE	8	10508	Attribute added
IM_REMOVEATTRIBUTE	9	10509	Attribute removed

IPS_SEARCHMESSAGE

Value base 510

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
IM_SEARCHSTART	1	10511	Search started
IM_SEARCHSTOP	2	10512	Search stopped
IM_SEARCHUPDATE	3	10513	Search has new results

IPS_FORMMESSAGE

Value base 520

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
IM_FORMFIELDCREATE	1	10521	Create new form field
IM_FORMFIELDDELETE	2	10522	Delete existing form field
IM_FORMFIELDUPDATE	3	10523	Updating existing form field
IM_FORMRELOAD	4	10524	Reload form

IPS_VARIABLEMESSAGE

Value base 600

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
VM_CREATE	1	10601	Variable was created
VM_DELETE	2	10602	Variable was removed
VM_UPDATE	3	10603	Variable was updated
VM_CHANGEPROFILENAME	4	10604	Variable profile name has been changed

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
VM_CHANGEPROFILEACTION	5	10605	Variable profile action was changed
VM_CHANGEDLOCKED	6	10606	Locked was Changed

IPS_SCRIPTMESSAGE

Value base 700

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
SM_CREATE	1	10701	Script was created
SM_DELETE	2	10702	Script was removed
SM_CHANGEFILE	3	10703	Script was attached to file
SM_BROKEN	4	10704	Script error status has changed
SM_UPDATE	5	10704	Script was updated

IPS_EVENTMESSAGE

Value base 800

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
EM_CREATE	1	10801	Event was created
EM_DELETE	2	10802	Event was removed
EM_UPDATE	3	10803	Event was updated
EM_CHANGEACTIVE	4	10804	Event activation has changed
EM_CHANGELIMIT	5	10805	Event call limit has changed
EM_CHANGESCRIPT	6	10806	Event script content has changed
EM_CHANGETRIGGER	7	10807	Event trigger has changed
EM_CHANGETRIGGERVALUE	8	10808	Event limit has changed
EM_CHANGETRIGGEREXECUTION	9	10809	Event limit triggering has changed
EM_CHANGECYCLIC	10	10810	Cyclic event has changed

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
EM_CHANGECYCLICDATEFROM	11	10811	Start date has changed
EM_CHANGECYCLICDATETO	12	10812	End date has changed
EM_CHANGECYCLICTIMEFROM	13	10813	Start time has changed
EM_CHANGECYCLICTIMETO	14	10814	End time has changed
EM_ADDSCHEDULEACTION	15	10815	Entry in the action table of the schedule has been added
EM_REMOVESCHEDULEACTION	16	10816	Entry in the action table of the schedule has been removed
EM_CHANGESCHEDULEACTION	17	10817	Entry in the action table of the schedule has changed
EM_ADDSCHEDULEGROUP	18	10818	Group of the schedule days was added
EM_REMOVESCHEDULEGROUP	19	10819	Group of the schedule days has been removed
EM_CHANGESCHEDULEGROUP	20	10820	Group of the schedule days has changed
EM_ADDSCHEDULEGROUPPOINT	21	10821	Switching point of a group has been added
EM_REMOVESCHEDULEGROUPPOINT	22	10822	Switching point of a group has been removed
EM_CHANGESCHEDULEGROUPPOINT	23	10823	The switching point of a group has changed
EM_ADDCONDITION	24	10824	Condition was added
EM_REMOVECONDITION	25	10825	Condition was removed
EM_CHANGECONDITION	26	10826	Condition has changed
EM_ADDCONDITIONVARIABLERULE	27	10827	Condition variable rule has been added

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
EM_REMOVECONDITIONVARIABLERULE	28	10828	Variable rule of the condition has been removed
EM_CHANGECONDITIONVARIABLERULE	29	10829	Variable rule of the condition has changed
EM_ADDCONDITIONDATERULE	30	10830	Date rule of condition was added
EM_REMOVECONDITIONDATERULE	31	10831	Date rule of condition was added
EM_CHANGECONDITIONDATERULE	32	10832	Condition date rule has changed
EM_ADDCONDITIONTIMERULE	33	10833	Condition time rule was added
EM_REMOVECONDITIONTIMERULE	34	10834	Time rule of the condition has been removed
EM_CHANGECONDITIONTIMERULE	35	10835	Time rule of the condition has changed
EM_ADDCONDITIONDAYOFTHEWEEKRULE	36	10836	Day of the week rule was added
EM_REMOVECONDITIONDAYOFTHEWEEKRULE	37	10837	Day of the week rule of the condition has been removed
EM_CHANGECONDITIONDAYOFTHEWEEKRULE	38	10838	Day of the week rule of the condition has changed

IPS_MEDIAMESSAGE

Value base 900

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
MM_CREATE	1	10901	Media object was created
MM_DELETE	2	10902	Media object has been removed
MM_CHANGEFILE	3	10903	Media asset file has changed
MM_AVAILABLE	4	10904	The availability of the media asset has changed
MM_UPDATE	5	10905	Media asset has been updated
MM_CHANGECACHED	6	10906	Media object cache option has changed

IPS_LINKMESSAGE

Value base 1000

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
LM_CREATE	1	11001	Link was created
LM_DELETE	2	11002	Link was removed
LM_CHANGETARGET	3	11003	The destination of the link has changed

IPS_FLOWMESSAGE

Value base 1100

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
FM_CONNECT	1	11101	Instance has been connected
FM_DISCONNECT	2	11102	Instance was disconnected
FM_CHILDADDED	3	11103	Child instance was connected to this instance
FM_CHILDREMOVED	4	11104	Child instance was separated from this instance

IPS_ENGINEMESSAGE

Value base 1200

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
SE_UPDATE	1	11201	Script engine was reloaded
SE_EXECUTE	2	11202	Script was executed
SE_RUNNING	3	11203	Script is running
SE_FLOWSCRIPT	4	11204	On FlowScript Status (either all steps finished or was aborted)

IPS_PROFILEMESSAGE

Value base 1300

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
PM_CREATE	1	11301	Profile was created
PM_DELETE	2	11302	Profile has been removed
PM_CHANGETEXT	3	11303	Profile prefix/ profile suffix has changed
PM_CHANGEVALUES	4	11304	Profile values have changed

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
PM_CHANGEDIGITS	5	11305	Profile decimal places have changed
PM_CHANGEICON	6	11306	Profile icon has changed
PM_ASSOCIATIONADDED	7	11307	Profile association was added
PM_ASSOCIATIONREMOVED	8	11308	Profile association was removed
PM_ASSOCIATIONCHANGED	9	11309	Profile association has changed

IPS_TIMERMESSAGE

Value base 1400

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
TM_REGISTER	1	11401	Timer was created
TM_UNREGISTER	2	11402	Timer has been removed
TM_CHANGEINTERVAL	3	11403	Timer interval has changed

IPS_ACTIONMESSAGE

Value base 1500

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
AM_FORMFIELDCREATE	1	11501	Currently unused
AM_FORMFIELDDELETE	2	11502	Currently unused
AM_FORMFIELDUPDATE	3	11503	Form element needs an update
AM_FORMRELOAD	4	11504	Reload configuration form

IPS_LICENSEMESSAGE

Value base 1600

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
LP_UPDATELICENSE	1	11601	On license update
LP_UPDATESUBSCRIPTION	2	11602	On subscription update

MESSAGE PARAMETERS

Values of various message parameters

SEARCH HANDLING

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
IF_UNKNOWN	0	0	Unknown value
IF_NEW	1	1	Device is created but not configured
IF_OLD	2	2	Device is configured and should have an InstanceID
IF_CURRENT	3	3	The device is configured and the InstanceID belongs to the searching device
IF_UNSUPPORTED	4	4	Device is not supported by the module

STATUS CODES

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
IS_SBASE	100	100	Value base for status codes
IS_CREATING	1	101	Instance is creating
IS_ACTIVE	2	102	Instance is active
IS_DELETING	3	103	Instance is deleting
IS_INACTIVE	4	104	Instance is inactive
IS_NOTCREATED	5	105	Instance was not created
IS_STANDBY	6	106	Instance is in standby

ERROR CODES

DEFINE	VALUE	TOTAL VALUE	DESCRIPTION
IS_EBASE	200	200	General error

References

Require: Symcon >= 5.1

DESCRIPTION

References indicate which objects require oneself to function properly. For example, during a reference search for a variable, a triggered event is listed which is triggered when the variable changes. Without the variable, the event cannot function properly.
The references are called up by default via the context menu in the object tree. One can then jump to the objects in the displayed list.



MANAGE REFERENCES

References can be added and removed in the module.

FUNCTION	DESCRIPTION
GetReferenceList	This function returns an array with all references.
RegisterReference	This function registers a reference using the TargetobjectID, which is then found during a reference search.
UnregisterReference	This function removes a reference for the module.

Store

Since version 5.1 there is a module store in IP-Symcon in which developed modules can be offered. This is based on a library which can be found and installed conveniently.



General information about the Module Store can be found here:
Module Store

Up to three different channels can be offered for each library: Stable, Beta and Testing
Stable is the channel for the stable productive version of the library. Publications on this channel are checked by the Symcon team.
Beta is the open test channel. Every user has access to it and can install the beta version.
Testing describes an internal test channel. For this purpose, users must be invited explicitly, granting them access to the channel. If one is not invited, no insight into this version is possible.

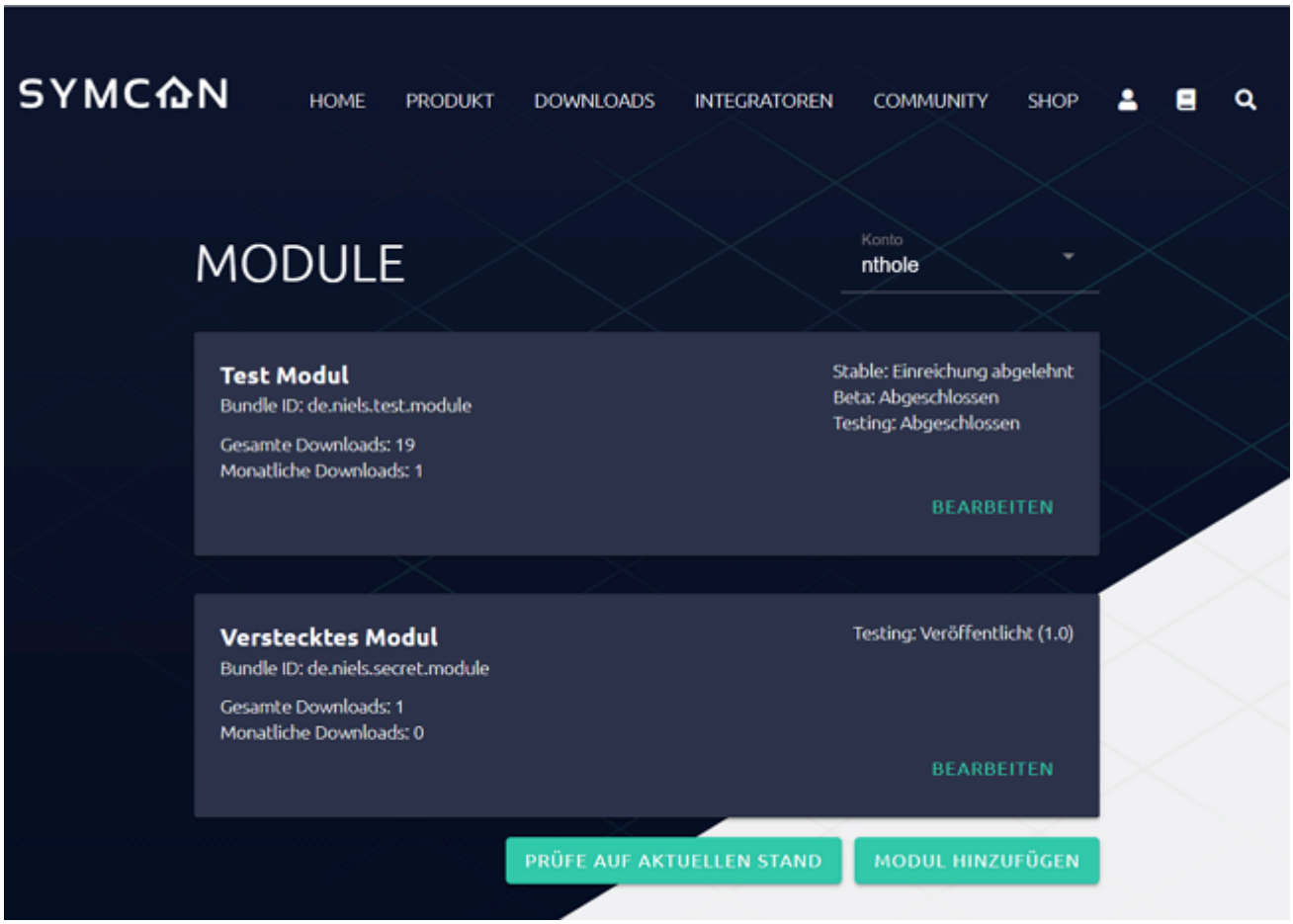
Submit

DESCRIPTION

If a developed library is to be offered in the Module Store, this must be submitted via the developer area on the account page .

DEVELOPER AREA

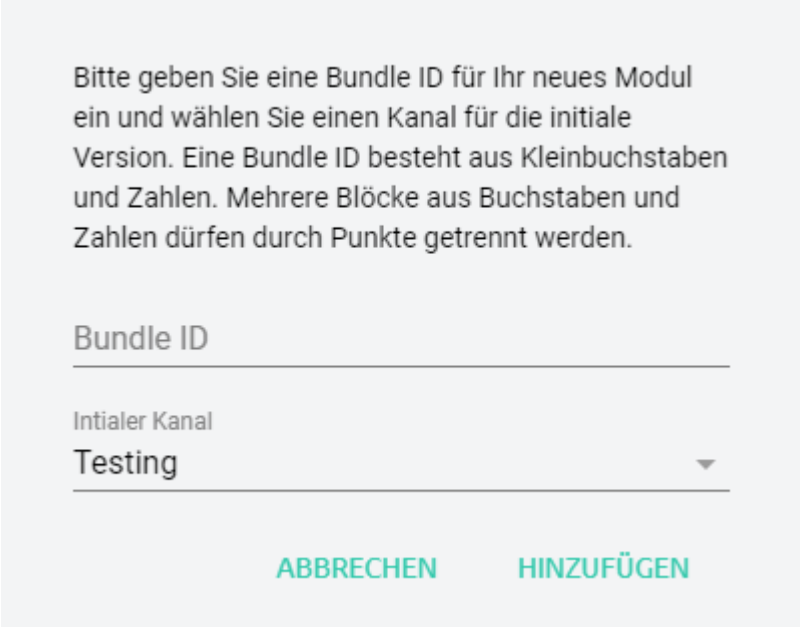
The developer area can be accessed on the account page under "developer area".



All previous submissions are listed here. The number of downloads as well as the state of the individual channels is displayed here. With the "Edit" button a module can be adjusted. The button "Check up to date" automatically checks which modules are currently at the state of the newest commit of its main branch. This function is only supported for GitHub repositories. For a bigger number of modules correspondingly many requests are made towards the GitHub API, which could become exhausted if the account from the personal area is not connected with a GitHub account. If the account is connected to GitHub, the limit is significantly increased. New modules can be prepared for submission using the "Add module" button. If your own account has been granted access to other accounts, the submissions of these accounts can be checked and edited by selecting "Account".

ADD MODULE

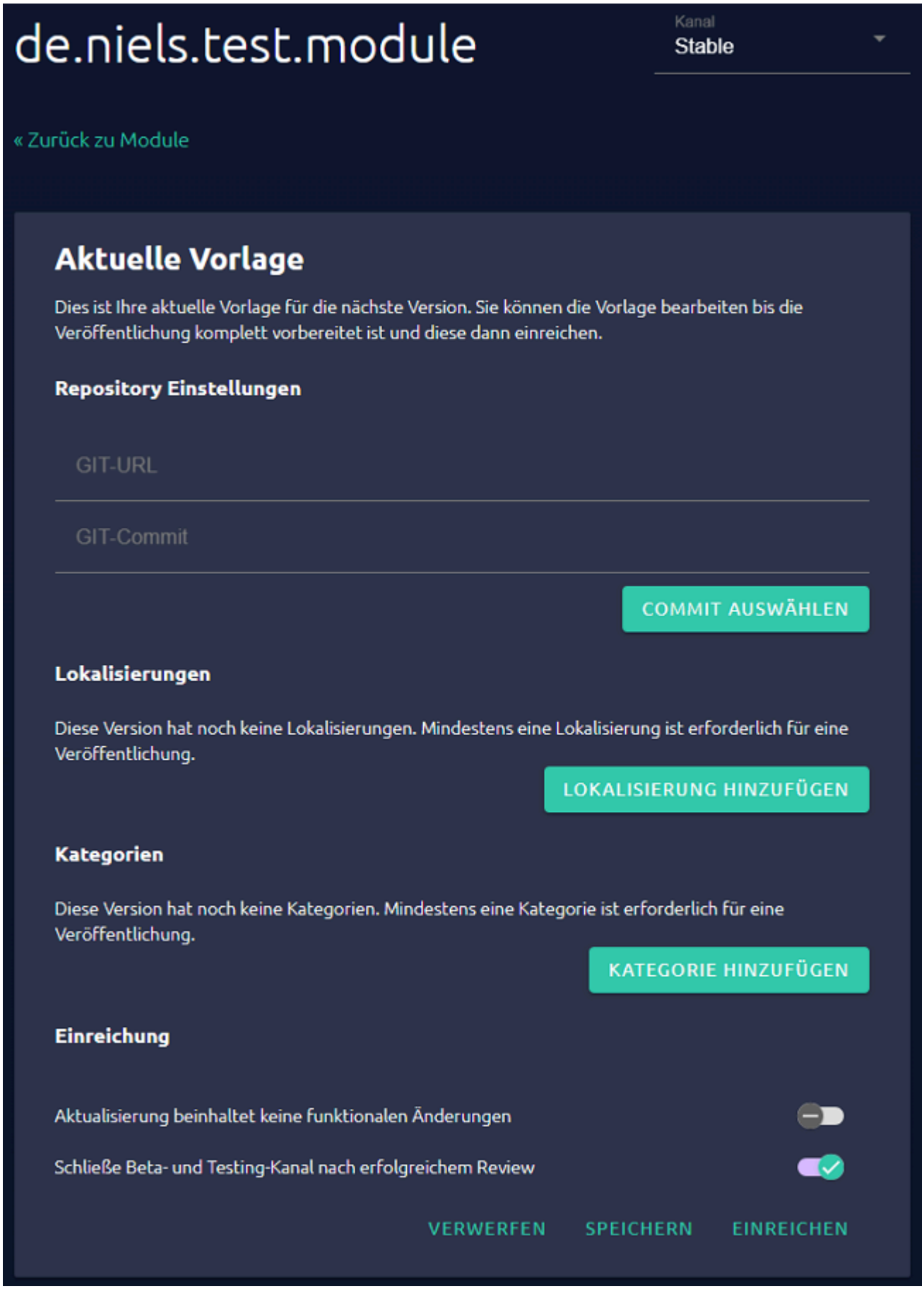
After clicking on "Add module", a dialog appears in which a Bundle ID and an initial channel can be selected.



The Bundle ID is a unique ID, which remains the same for new versions, and consists of lowercase letters and dots. It is recommended to use the reverse domain notation, e.g. de.symcon.alexa. The initial channel is the channel for the first submission. Additional channels can easily be added later.

SUBMIT MODULE

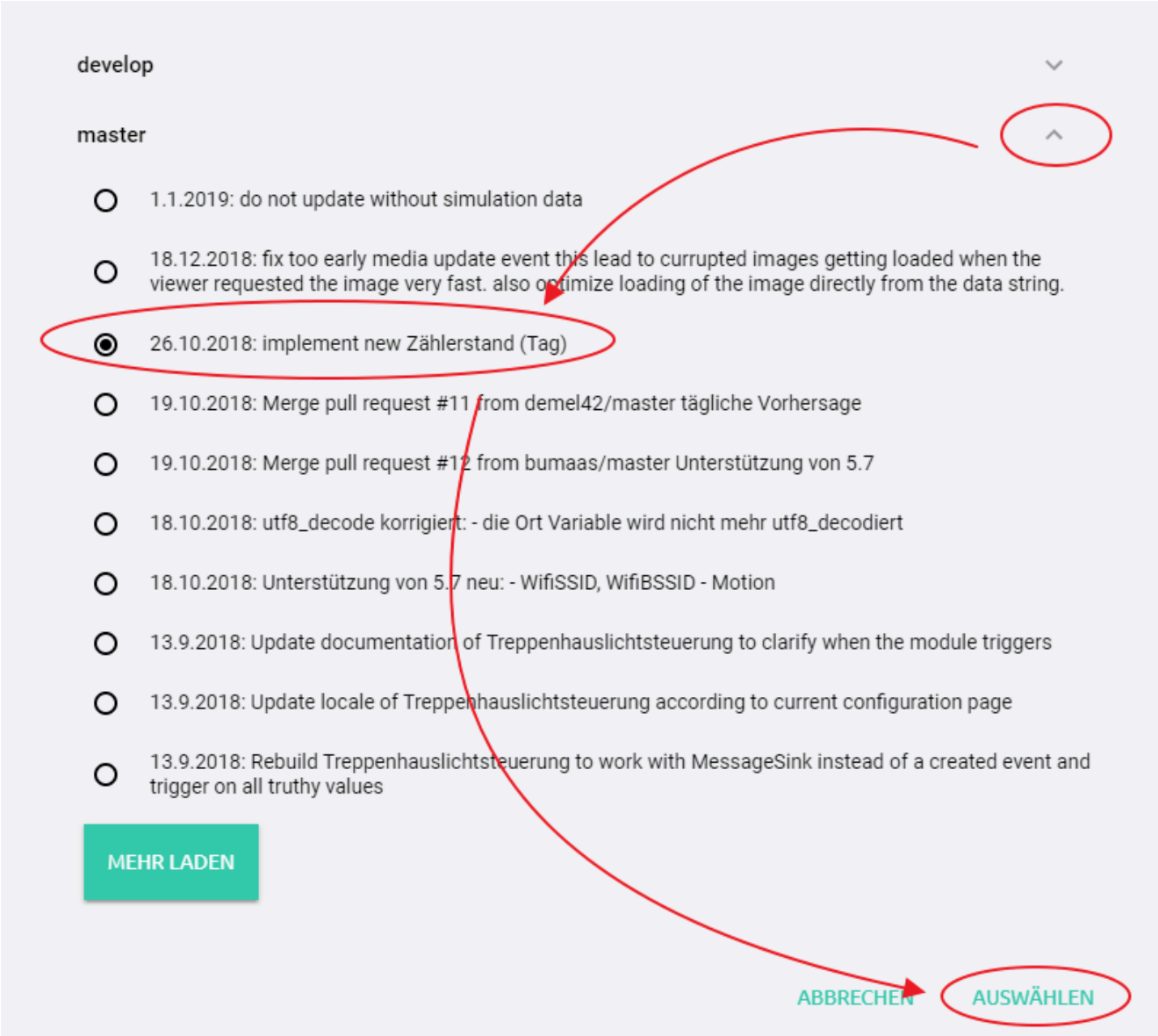
Under "Edit module" one can check the current status according to channels and prepare the next submission.



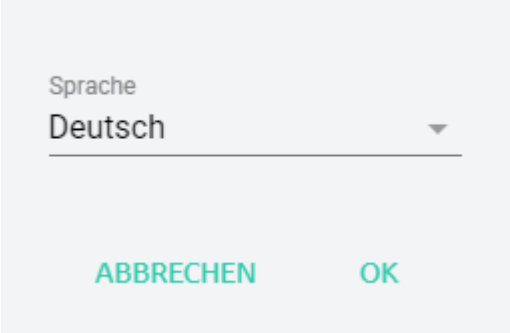
The address of the Git repository under which the current library can be found is entered under GIT-URL. To use private repositories, the account of the personal area needs to be connected to a GitHub account that has access to that private repository.

Currently private repositories are only supported on GitHub.

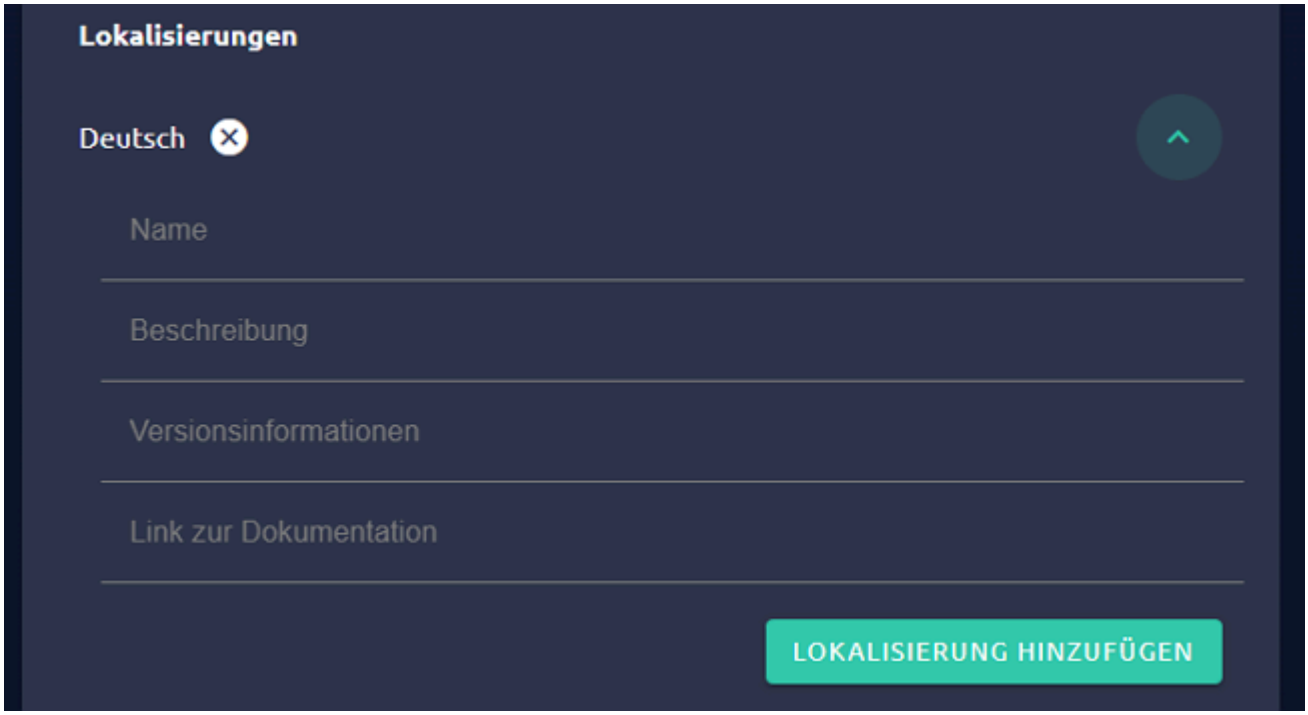
After the URL, the commit that is to be published is selected. To do this, one can click on the gear in the "GIT-Commit" line, click on the desired commit and confirm via "Select". The dialog offers the commits divided according to branches. If the desired commit is not displayed, more commits of a branch can be loaded with "Load more". Alternatively, the ID of the commit can also be entered manually. If the repository is not on GitHub, the dialog is limited and can only select a branch. The newest commit from that branch is automatically selected.



Localization provides names and descriptions of the module in one language. Multiple localizations for multiple languages can be added, but at least one is required. The module must be usable in all localized languages. Once a module has been published with a localization, this language must also be offered in future publications.
Clicking on "Add localization" opens a dialog in which the language of the new localization can be selected.



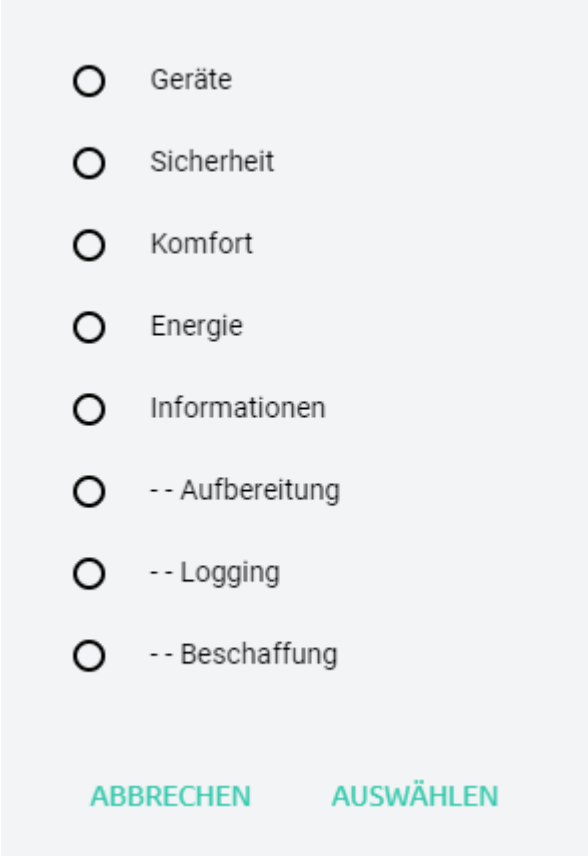
After the selection has been confirmed with "OK", a new empty localization of the selected language appears. A localization can be expanded with a click on the arrow on the right.



The following fields must be filled in for localization:

FIELD	DESCRIPTION
Name	The displayed name of the module
Description	A description of the module
Version information	Information on what's new about this version
Link to documentation	A link to explain the module. This can be, for example, a readme in the repository or an explanatory post in the forum

A localization can be removed by clicking on the X next to the language.
Finally, suitable categories are selected for the submission. Via "Add Category" a dialog appears in which a new category can be selected and confirmed with "Select".



Categories can be removed by clicking on the associated X. At least one category is required for a submission, selecting multiple categories is also possible.
Below "Submission" further settings are possible.

SETTING	DESCRIPTION
Update includes no functional changes	This setting can be activated, if the current submission does not include any programaticcal changes that are relevant for the user, e.g., only the style was updated. The new version is not offered to users. The submission of non-functional changes can be helpful, to properly recognize the module as up to date
Close beta and testing channels after succesful review	If this setting is activated, all lower channels are closed after a succesful publication. Closed channels are not available in the Module Store any more. Users that are on a closed channel now, are forwarded towards the next highest channel for updates. This setting is not available if there are no publications on lower channels

Finally, the current template offers three buttons. "Discard" will delete the template. If only the template exists for the module, the entire bundle will be deleted. "Save" saves the current status of the template. Not all elements are required to be filled in for this. A module without localizations and categories can therefore be saved, but not submitted. "Submit" is used to submit the module for deployment in the Module Store. The module is offered directly in the module store on the testing and beta channels. When submitting to Stable, the module goes through a review process by the Symcon-Team before it is published. If the module is accepted, it will be published in the Module Store. If it is rejected, one will be informed why the submission was rejected, both in the developer area and via e-mail.

REVIEW

When reviewing a module, various points are checked on the basis of our Best Practices .
These are the following points in particular:

- Does the module access local files and could read and manipulate them?
  Access to local files is permitted in exceptional cases, especially within the IP-Symcon folder. However, a more detailed check is carried out here in order to prevent misuse.
- Is there a corresponding translation for each localization of the module?
- Can the instances of the module be created without errors, even if not all requirements may have been met?
  Under all circumstances it must be possible to create a module as an instance. The module must therefore independently check whether the requirements are met. If the requirements are not met, this should be shown appropriately in the instance configuration.
- Does any communication via splitter/IO run as intended via the data flow via SendDataToChildren/SendDataToParent/ReceiveData/ForwardData?
- Does a module only access its own objects or ones that have been assigned to it?
  Own objects are below the instance.
  Using SelectVariable and similar elements, further objects can be assigned to a module, which may be used within the expected framework.
  External objects that have not been assigned to the module must not be manipulated.
- New objects may only be created outside of the instance if this has been explicitly confirmed by the user. A conforming example of this is the use of the "Configurator" configuration element.
  This also prohibits the creation of events to switch internal functions. For this purpose RegisterTimer or MessageSink and the associated functions are to be used.
- Deviations are possible with prior agreement. However, this should be discussed in a suitable place when submitting, for example by e-mail or as a comment at the relevant point in the code.
- Code quality or absence of errors are not checked during the review.
- "IPSymcon", "IPS" or similar may not be part of a name chosen in a localization.
- Modules that require more than the current stable version as a condition, e.g. ones that only work with the current beta, may not be submitted to the stable channel.
- Some object properties are in the sovereignty of the user even in the case of module-specific objects. These may initially be specified, but cannot be further manipulated without explicit confirmation from the user. These properties are:
  The name
  All visual settings: Show object, object active and icon
  The description
  The position
  All event-specific properties
  User-defined profile, user-defined action and logging of variables must never be set without explicit confirmation, not even initially! Action or profile can of course be defined using a standard action or profile.
- PHP files cannot use the short PHP-tag (<?) And should use the long PHP-tag (<?php) instead.
- After the initial creation of an instance, the user is in the sovereignty of the properties.
  If the module wants to change something here, other values can be specified via the dynamic function UpdateFormField and subsequently be confirmed by the user.
  Thus the functions IPS_SetProperty and IPS_ApplyChanges should never be used.
  If the parameter loadValuesFromConfiguration is set to false in a List or a Tree, the property still has to be filled in the interests of the user when loading. The parameter should only make it possible to adjust, for example, the order or to add new values in the middle.

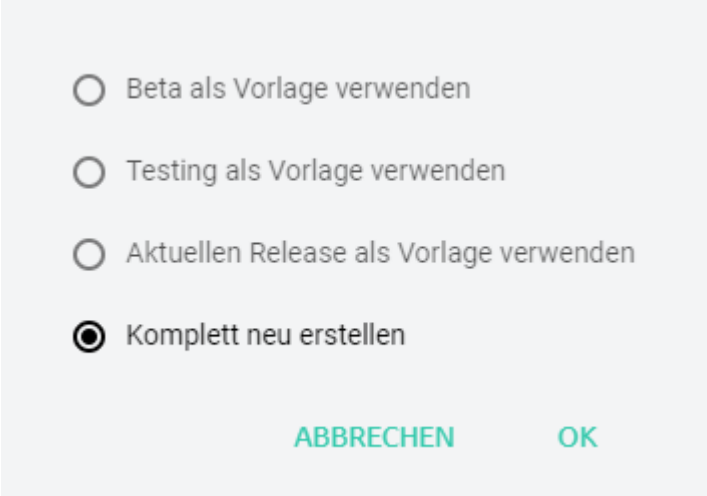
A submission requires a descriptive name, a suitable description and version information as well as a documentation that explains the configuration and use of the module.
If configurators are connected to a splitter or I/O, they may only manage devices that are also connected to this splitter or I/O

- Instances, that are connected to the same splitter or I/O as the configurator, should be listed in the configurator regardless of whether they still exist physically
- Instances, that are connected to another splitter or I/O, must not be listed in a configurator with a splitter or I/O

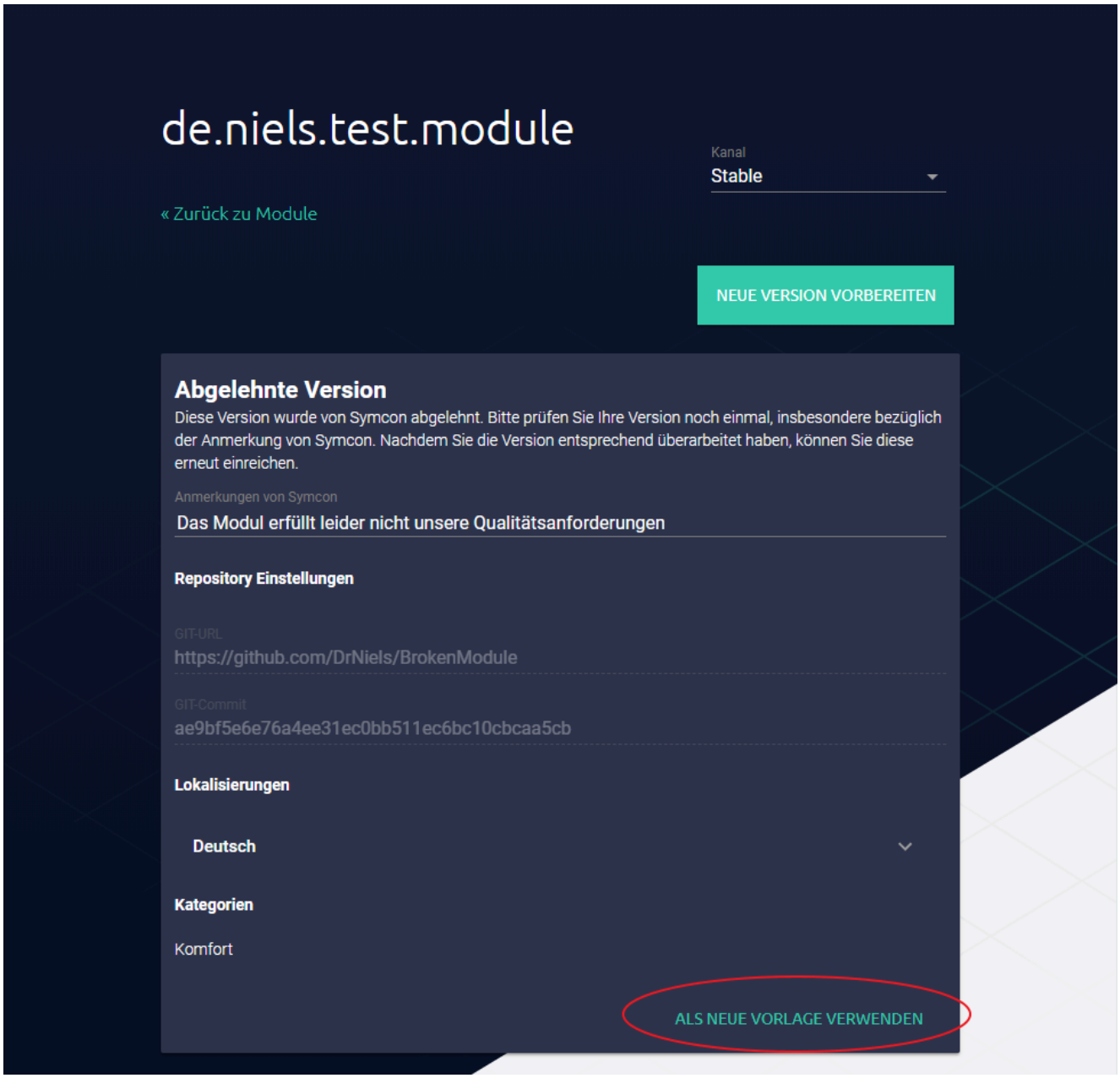
If a module offers a discovery instance, it must work without configuration
Since logging can smoothly be implemented via $this->LogMessage and these messages are correctly linked to the sending instance, IPS_LogMessage may no longer be used for logging

EDIT MODULE

If there is no current template on a channel under "Edit module", a new template can be created by clicking on "Prepare new version". In the dialog one can choose whether one wants to create a completely new template or use a current publication as a template.



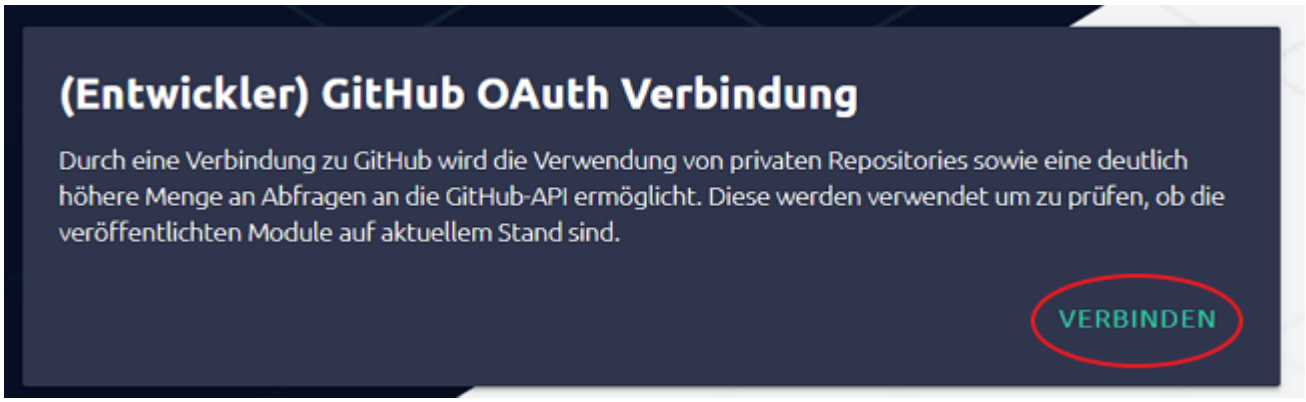
Alternatively, when publishing there is an option to move these to a higher channel. If a submission has been rejected, it can also be used as a new template to compensate for any deficiencies without having to configure everything from scratch again.



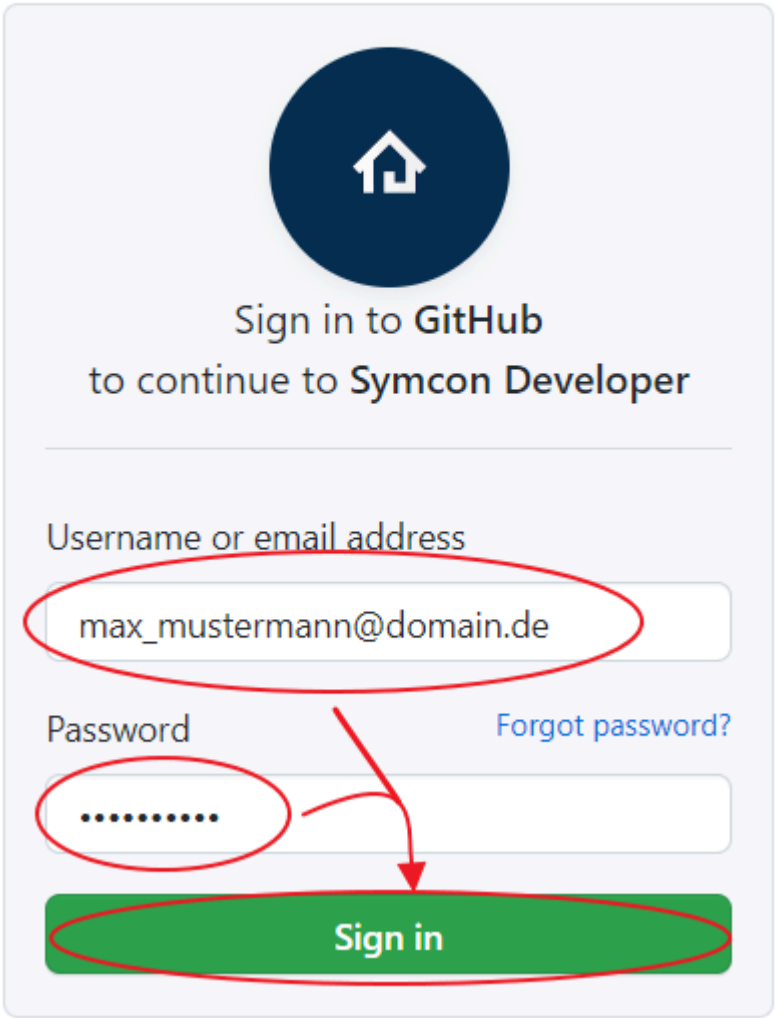
If a module, which is currently being reviewed, is to be withdrawn, this can be done by clicking on "Withdraw". If a module is in the module store, it can no longer be withdrawn.

CONNECT WITH GITHUB

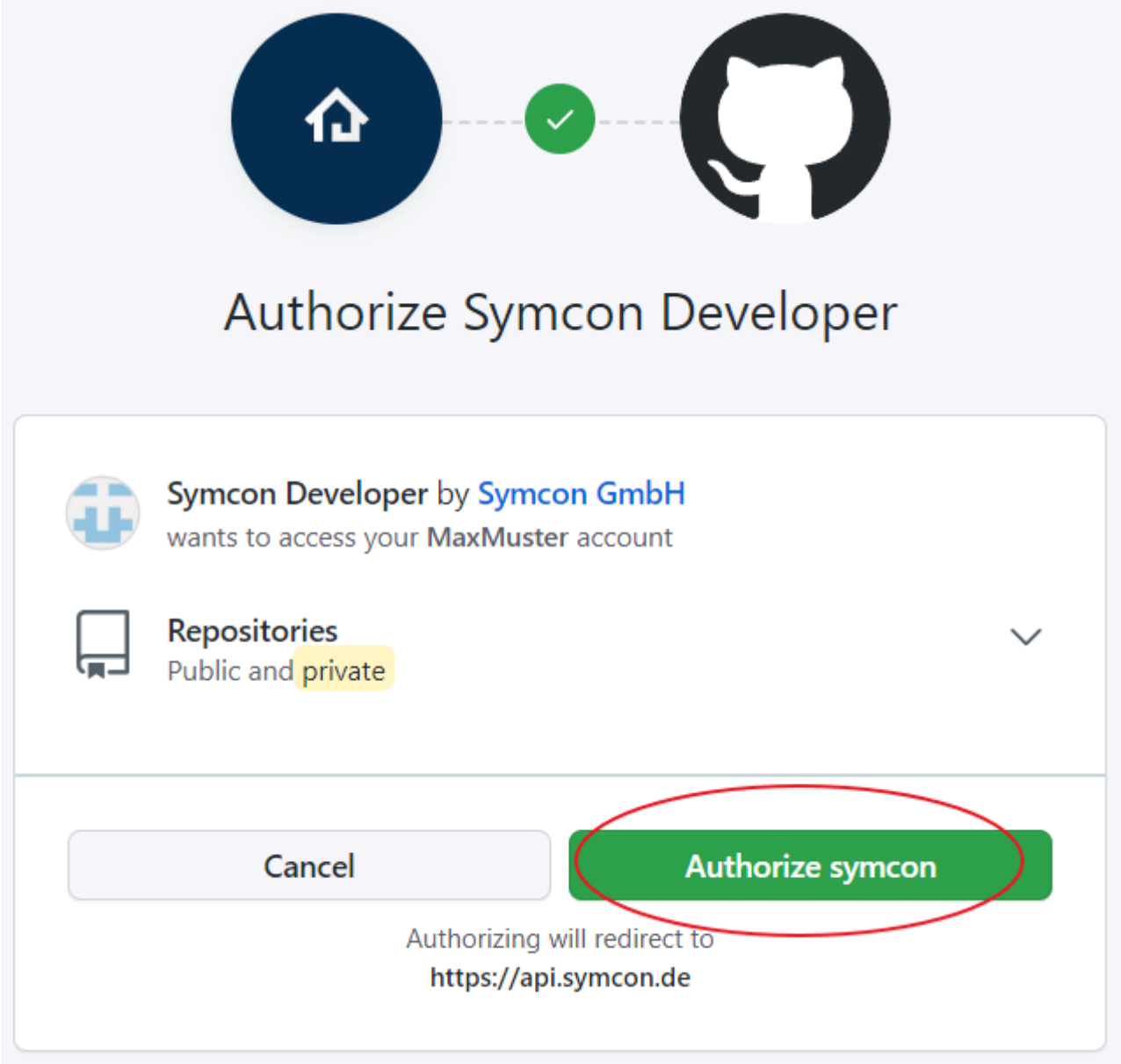
The account of the personal area can be connected to GitHub. This enables the use of private repositories as well as a significantly higher number of requests towards the GitHub API. This can be especially relevant for checking if published modules are up to date. The connection can be done in the Settings of the Personal Area. This is done by clicking "Connect" in the area "(Developer) GitHub OAuth Connection".



If no GitHub user is currently logged in within the browser, this is done within the next step. Here, the email as well as the password of the GitHub account need to be entered. If an account is already logged in, this step is skipped.



Finally, the access via Symcon needs to be authorized. This is done by clicking "Authorize symcon".



Now, the account from the personal area is connected with the GitHub account. The connection can be undone by clicking "Disconnect".

Structure

DESCRIPTION

The required structure is necessary for correct functioning within IP-Symcon. Integration via Module Control only works if the structure is adhered to.
The "library.json" is the core of every module development.
The "module.php" and "module.json" together form the actual module.
Form.json can be used to set up the configuration page.
Locale.json is used for possible translations.
The module folder name (here: Module1, Module2) should have the same name as the class name in module.php.
Folders that do not contain any module.json are marked as faulty.
Exceptions are the folders:
libs/ (since version 4.2)
docs/ (since version 4.2)
imgs/ (since version 4.2)
tests/ (since version 4.4)
actions/ (since version 6.0)



Point-folders (e.g. .github, .style) are also ignored and are not necessary for the module to function correctly. These are important, for example, for the correct functioning of the repository.

These folders are not integrated as a module and offer the possibility of making external libraries, documents, images and tests available.

DIRECTORY STRUCTURE

```
Library
|
- Module1
| |
| - module.php
| |
| - module.json
| |
| - form.json (optional)
| |
| - locale.json (optional)
|
- Module2
| |
| - module.php
| |
| - module.json
```

```
| |
| - form.json (optional)
| |
| - locale.json (optional)
|
- actions (optional)
| |
| - Definitions of actions
|
- libs (optional)
| |
| - any libraries
|
- docs (optional)
| |
| - any documents
|
- imgs (optional)
| |
| - any media files
|
- tests (optional)
| |
| - any test files
|
- library.json
|
- README.txt (optional)
```

VALIDATION

JSON schemas are available for the JSON files of a module. With them, editors like Visual Studio Code or PhpStorm can offer autocompletion and check the files while editing. To do this, the schema is specified in the respective file via the parameter "$schema", e.g. `"$schema": "https://www.symcon.de/assets/files/validation/librarySchema.json"`. IP-Symcon ignores this parameter.

FILE	JSON SCHEMA
library.json	librarySchema.json
module.json	moduleSchema.json
form.json	formSchema.json (see Configuration forms)
locale.json	localeSchema.json

SDK (Skins) (deprecated)

Require: Symcon >= 3.0

Skins are only available for the WebFront. Since version 7.0, there has been the Tile Visualization, whose design can be changed directly in the visualization for both web and apps.

DESCRIPTION

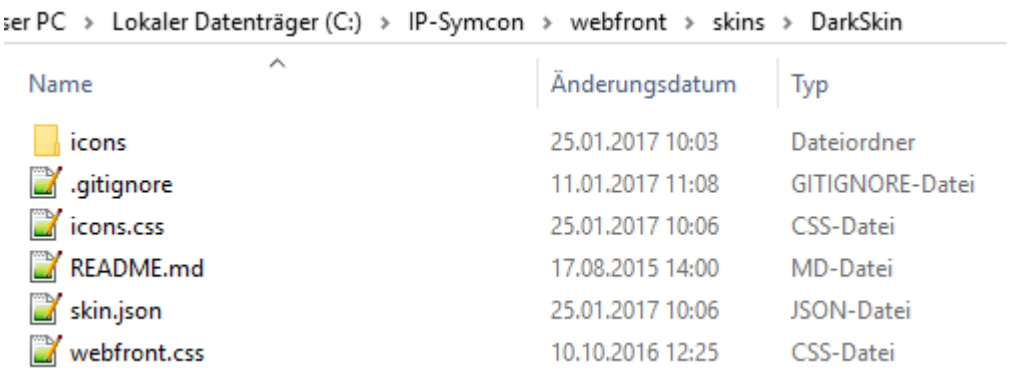
Since version 3.0, individual skins can be created.
The creation and change of icons is quick and easy, but if complete changes to the style are to be made, additional knowledge of CSS is required.

This only works for the WebFront. If a skin or adaptation of the mobile apps is desired, this can be implemented via the Enterprise program.
Further information can be found under Products for companies and an individual branding can be requested via direct contact.

REQUIREMENTS

IP-Symcon 3.0 or above
CSS knowledge (if more than the icons are to be changed)
Download sample skin to have a basic structure (recommended)

FOLDER STRUCTURE




```
Skinname
|
- icons (optional)
| |
```

```
| - Aircraft.png
| |
| - Bird.png
|
- img (optional)
| |
| - Background.png
|
- font (optional)
| |
| - WebFont.ttf
|
- skin.json (will be generated)
|
- webfront.css (required, even if empty)
|
- icons.css (will be generated)
|
- README.md (optional)
```

ICONS-FOLDER

Icons includes all icons. If there are standard icons with the same name, the custom icons will be preferred.
Contains all icons in PNG format and size 32x32.
Icon names may only consist of letters, numbers, minus (-) and underscore (_). Spaces, umlauts and special characters are not permitted.
The icons must all start with capital letters (e.g. Arrow.png, Right.png)
In the case of word chains, it is recommended that these are also written with a capital letter (e.g. ArrowRight.png). This is for better readability and consistency with the IP-Symcon icons.
If an icon has the same name as one of the IP-Symcon icons, the custom icon is preferred in the WebFront. The upper and lower case letters are to be observed. (This function is an elegant way of effortlessly equipping a third-party WebFront with a skin.)

IMAGE-FOLDER

Img contains additional graphics for e.g. possible backgrounds.
Contains additional images, e.g. for special backgrounds

FONT-FOLDER

Contains the fonts required for the skin.
Contains other WebFonts
The CSS elements of the WebFont should be added in the webfront.css

INDIVIDUAL FILES

Files which are in the main folder.

SKIN.JSON

Meta file that is automatically created by the skin builder. The fields Author, Version, URL, GIT and Compatible may be changed after creation. The URL should be the address for the skin topic in the forum. GIT should be the address for the GitHub repository so that the skin control can download updates accordingly. The icon array is created automatically and is used by the WebFront editor as an aid to make the custom icons selectable. Compatible must have the value 1. This is strictly checked so that changes to the WebFront, which make the skins incompatible, can be deactivated.

WEBFRONT.CSS

CSS file containing changes to the WebFront CSS style. Must be present. Can also be empty. CSS changes must comply with the above rules.

ICONS.CSS

CSS file that contains the links to the custom icons.
Is generated automatically by the skin builder and should not be changed.

README.MD

Readme which GitHub automatically displays in the browser in the repository overview. Can be used for documentation.

SKIN BUILDER

The skin manager is called with http//host:port/skins/. (Example: http://localhost:3777/skins/)
As soon as a skin has been selected, the icons currently available in the CSS/meta file are displayed.
If new icons have been added/removed, a "Build Skin!" must be executed so that the icons.css/skin.json are regenerated accordingly. No action is required when changing the icons themselves or the webfront.css. The browser cache may have to be cleared so that changes can be seen.



Up to version 4.4 the skin manager could be found under http//host:port/user/skins/. (Example: http://localhost:3777/user/skins/)

PROVIDE SKINS

Skins can be uploaded to GitHub.
The GitHub repository name should match the name of the skin.
The repository should only contain the necessary files such as icons.css, skin.json, webfront.css and optional folders such as icons, font or img. A README.md and a LICENSE file are also

recommended. The files must not be nested in an additional subfolder, otherwise the installer will not work correctly.
When publishing, the additional size (number in kB/mB) the skin requires when loading must be specified. (This will be displayed after the build process)

INSTALL SKINS

This works via Skin-Control and is described there.

SELECT SKINS

In the WebFront configurator there is a "Skin" field under the Presentation tab, in which the skin can be selected.

Tools

NETWORK-CONFIGURATION-TOOL

All IPS gateways in the network can be found with the network-configuration-tool.
Instructions for use can be found HERE .
To download

RECOVERYTOOL

The recovery tool can write an image to the SymBox.
Instructions for use can be found HERE under point "5.2. Recovery using the recovery tool ".
To download

MODULE DEVELOPMENT

The following tools are helpful in developing modules for the Module Store and help to successfully submit them.

GUID GENERATOR

The GUID Generator generates a GUID in the correct format for the module.

MODULE GENERATOR

The Module Generator is a wizard which helps to create the basic structure for a module and sets up all the basic facilities for a functioning data flow.

MODULE VALIDATOR

The Module Validator checks the developed files for correct syntax and whether all required information is included.

EXTERNAL IDES

External IDEs can insert and display function declarations.

VISUAL STUDIO CODE

Instructions for generating and using in Visual Studio Code can be found HERE.

GUID generator

GENERATE NEXT GUID

COPY TO CLIPBOARD

Module Generator

(info: For use in Visual Studio Code, it is recommended to use the Symcon Module Helper extension from pitti instead of this tool. More information can be found [here] (https://community.symcon.de/t/symcon-modul-helfer-eine-vs-code-extension-fuer-modul-entwickler-ehem-forminator/141324)).

LIBRARY

GUID

Author

Name

URL

7.1

1.0

0

0

ADD MODULE

GENERATE MODULE TEMPLATE

Module Validator

library.json

SELECT FILE (OPTIONAL) VALIDATE

Visual Studio Code

VisualStudioCode has some useful extensions for module development.

INSTALL EXTENSION FOR IP-SYMCON

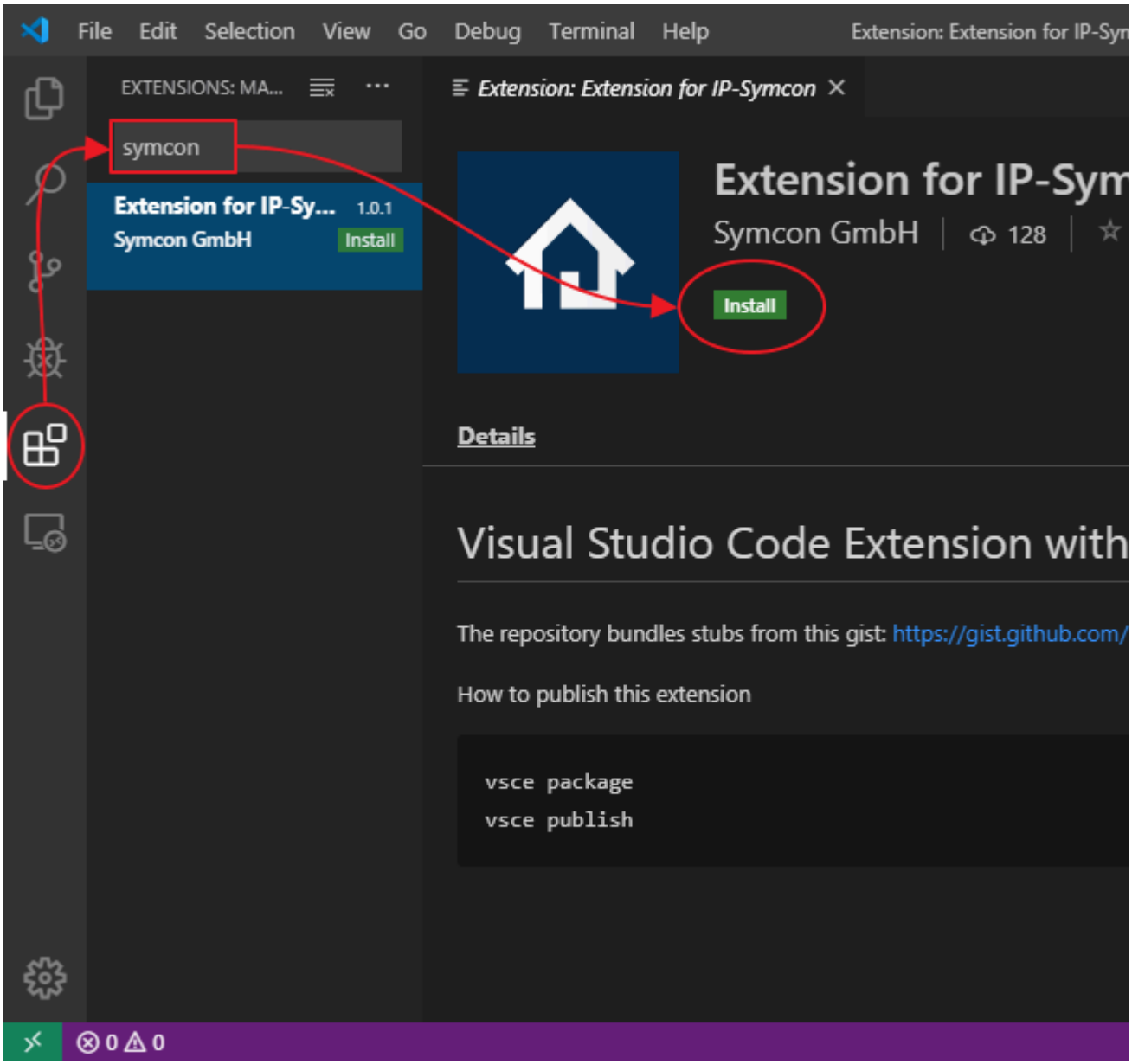
A function list has been created in this extension to display it in the source code.

MARKETPLACE

Call up this Link to Marketplace and click on "Install".

EXTENSION BROWSER IN VSC

Call up "Extensions"
Search for "Symcon"
Select the "Extension for IP-Symcon" and click on "Install" to install it
In the Extensions area under "Installed" -> "Extension for IP-Symcon" should now appear



DOWNLOAD PHP

PHP is required for the extensions "CS Fixer" and "PHPUnit".
Here is a link to download PHP. PHP 8 or higher is recommended.
Call up "Settings" by clicking on the cogwheel (Manage).
Select "PHP" under "Extensions".
There is the item "PHP > Validate: Executable Path" where the "settings.json" can be called.
The path to php.exe is now specified after "php.validate.executablePath".
The "\" must be escaped with a second "\".

```
"php.validate.executablePath": "Path\\To\\php.exe"
```

INSTALL CS FIXER

Needed are:
The PHAR file, which can be downloaded from this link . Version 3 is supported with PHP 8.

A .style folder. It is recommended to include this Repository as a submodule in the own repository
The extension to use cs-fixer in Visual Studio Code, which can be downloaded from the Marketplace or subsequently embed directly into Visual Studio Code

EXTENSION BROWSER IN VSC

Call up "Extensions"
Search for "php cs fixer".
Select the "php cs fixer" from junstyle and install it by clicking on "Install".
The extensions area should now appear under "Installed" -> "php cs fixer".

CONFIGURE CS FIXER

Call up "Settings" by clicking on the cogwheel (Manage)
Select "PHP CS Fixer" under "Extensions"
The item "Allow Risky" is set to true
Under the item "PHP-cs-fixer: Executable Path" specifies the path to "php-cs-fixer.phar".
Alternatively, these settings can be entered in "settings.json".

```
"php-cs-fixer.executablePath": "Path\\To\\php-cs-fixer-v3.phar",
"php-cs-fixer.allowRisky": true
```

USE CS FIXER

If everything is correctly integrated and set up, the extension can be used. Either by right-clicking in the file to be formatted and selecting Format Document or by pressing the shortcut "Shift+Alt+F".
If the entire project should adopt the style, the following command can be entered via the command line.

```
php "Path\\To\\php-cs-fixer-v3.phar" fix --config=.style/.php_cs -v --
allow-risky=yes
```

INSTALL PHP UNIT

Needed are:
PHPUnit's PHAR file, which can be downloaded from this link .
A test to run
The extension to use PHPUnit in Visual Studio, which can be downloaded from the Marketplace or subsequently embed directly into Visual Studio Code

EXTENSION BROWSER IN VSC

Call up "Extensions"
Search for "PHPUnit".
Select the "PHPUnit Test Explorer" and install it by clicking on "Install".
The extensions area should now appear under "Installed" -> "PHPUnit Test Explorer" and "Test Explorer UI".

CONFIGURE PHP UNIT

Call up "Settings" by clicking on the cogwheel (Manage)
Select "PHPUnit configuration" under "Extensions".
At the item "Phpunit: Files" is entered "{test,tests}/*/Test.php".
Under the point "Phpunit: PHP" the path to PHP.exe is entered.
At the item "Phpunit: Phpunit" the path to the Phpunit Phar file is entered
Therefore, under "Phpunit: Args" ["--configuration", "tests/phpunit.xml"] must be entered.
Alternatively, these settings can be entered in "settings.json".

```
"phpunit.phpunit": "Path\\To\\phpunit-9.5.10.phar",
"phpunit.args": [
"--configuration", "tests/phpunit.xml"
],
"phpunit.php": "Path\\To\\php.exe",
"phpunit.files": "{test,tests}/**/*Test.php",
```

Now PHPUnit can be used. However, errors can still occur. So the following error can appear:

```
"PHPUnit requires the "mbstring" extension."
```

Navigate to the folder where "php.exe" is located
Look for the "php.ini" file in this folder. If it doesn't exist, search for "php.ini-development" and rename it
Open the "php.ini" file in any text editor
Search for "mbstring" in the "Dynamic Extensions" section
Remove the semicolon in front of "extension=mbstring".
Save file
Close and reopen Visual Studio Code
Now, another error may occur:

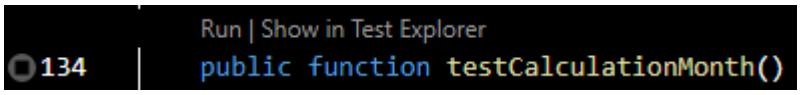
```
PHP Warning: PHP Startup: Unable to load dynamic library 'mbstring'
(tried: C:\php\ext\mbstring (Das angegebene Modul wurde nicht
gefunden.), )
C:\php\ext\php_mbstring.dll (Das angegebene Modul wurde nicht
gefunden.) in
Unknown on line 0
```

Navigate to the folder where "php.exe" is located

Look for the "php.ini" file in this folder. If it doesn't exist, search for "php.ini-development" and rename it
Open the "php.ini" file in any text editor
Look for "extension_dir" in the "Paths and Directories" section
Remove the semicolon in front of "extension_dir = "ext"" which is under "On windows:".
Save file
Close and reopen Visual Studio Code

USE PHPUNIT

Two buttons should now appear above the function with the test. The test is started by clicking on Run.

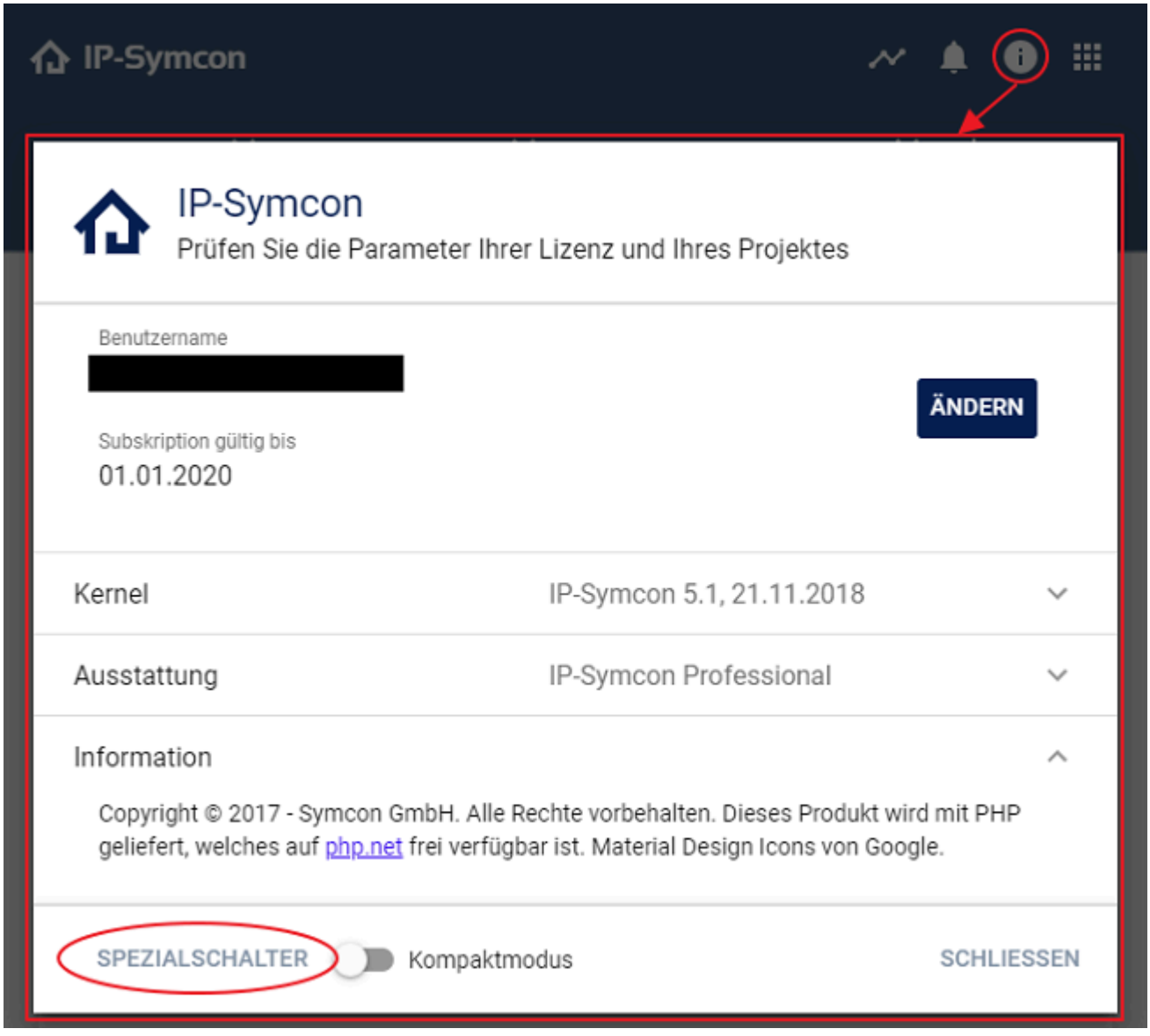


Special Switches

Require: Symcon >= 2.6
In order to be able to better control some functions of IP-Symcon, it is possible to change some switches within the Management Console.



Please note that changes usually only take effect as soon as IP-Symcon has been restarted. Exceptions are noted in the description of the respective switch.



The following switches are available:

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
ArchiveCommitInterval (since 5.0)	Archive time interval	**Default = 60 (in seconds).** which logged values are w

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
ArchiveRecordLimit (since 4.3)	Number of data points used in the archive	**Default = 50000 (up to 8.0** the limit parameters of th correlates directly with th for rendering and aggrega Symcon 8.1, smaller value 50000. Attention: Higher r rendering. Only suitable fo systems.
ArchiveRecordLimitBool (since 5.0, up to 7.0)	Number of raw data points used in the archive	**Default = 500**. Maximum v parameters of the archive directly with the maximum rendering and aggregating number slows down graph high-performance systems 7.1, ArchiveRecordLimit is
ArchiveWatch (since 6.0)	Enable extended protocol for the archive	**Default = Disabled**. If **activ** the archive are written to concerns problematic time or not continuous) that are writing and aggregating d
BackupCount	Number of backups	**Default = 25**. Number of B archived in the 'backup' fo created at every start and number in the file name is creation time. If the value unlimited number of back (Attention, storage consur
CompatibilityDeprecations (since 7.0)	Show deprecations in PHP	**Default = Disabled**. If **activ** messages are output (erro **deactivated**, they are supp interesting for developers so that the planned switch Requires IPSModuleStrict utf8_encode/utf8_decode
CompatibilityRequired (since 4.0)	Compatibility Features	**Default = Disabled (Windo** the compatibility function since 2.x / 3.x are available Activating this special swit every PHP script and shou possible. This feature is en Windows due to backward
CompatibilitySloppyUTF8 (since 6.3)	Correct UTF-8 errors automatically	**Default = Enabled**. If **activ** corrected automatically w strings via PHP, so that onl shown as broken instead o with an error message. A c immediately.
ConnectCheck (since 6.4)	Check interval from the Connect service	**Default = 30 (in minutes)**. the connection is actively service.

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
ConnectLimit (since 5.4, up to 6.4)	Connection attempt limit	**Default = 25** (maximum 1(number of connection att Connect-Control. All inqui with an error message anc limit is reset every 24 hou 7.0, the limit is fixed at 10(
ConnectWatch (since 4.0)	Log Connect service	**Default = Disabled**. If **activ** from processed "Connect to the logfile.
DataFlowWatch (since 6.1)	Logs runtimes and statistics of the data flow	**Default = Disabled**. If **activ** data flow are measured. P takes longer than DataFlo DataFlowWatchRxLimit ar extended statistics of the using the function IPS_Get
DataFlowWatchRxLimit (since 6.1)	Delay in milliseconds	**Default = 250 (in millisecc** from which delayed packe logged to the logfile. Bloc Only evaluated if DataFlov value zero (0) is specified,
DataFlowWatchTxLimit (since 6.1)	Delay in milliseconds	**Default = 250 (in millisecc** from which delayed packe logged to the logfile. Bloc Only evaluated if DataFlov value zero (0) is specified,
DebugDisableSnapshot (since 6.1)	Disable snapshots	**Default = Disabled**. For de **activated**, IPS_GetSnapsho IPS_GetSnapshotObject at Clients that rely on snapsh
DebugDisableSnapshotChanges (since 6.1)	Disable snapshot changes	**Default = Disabled**. For de **activated**, messages are n message buffer (see Mess IPS_GetSnapshotChanges message.
DebugDisableStreams (since 6.1)	Disable streams	**Default = Disabled**. For de **activated**, requests to the RTSP/MJPEG streams of ca answered by the web serv
DebugDisableWebSocket (since 6.1)	Disable WebSocket return channel	**Default = Disabled**. For de **activated**, no messages ar via the WebSocket return
DebugProtectMemory (since 8.1)	Enable memory protection for PHP	**Default = Disabled**. For de **activated**, the number of F and the memory protectio (opcache.protect_memory memory errors easier to tr productive operation.

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
DefaultVisualization (since 7.0)	Specify which visualization is on the front page	**Default = 1**. Specifies whic the base URL of the web s http://127.0.0.1:3777/). If shown, if **0** the WebFront- systems upgrading to 7.0
DiscoveryWatch (since 4.0)	Enable extended protocol for SSDP	**Default = Disabled**. If **activ** related to the SSDP servic The SSDP service is, for ex automatic detection of IP- management console and
Locale (since 6.4)	Set language/region	**Default = ""**. Overrides the language or region of the "de_AT". If empty, the sett is used.
LogfileCount	Number of logfiles	**Default = 25**. Number of l the 'logs' folder. A log file shortly after midnight. The UnixTimeStamp of the cre (0) is specified, an unlimite (Attention, storage consu
LogfileDisable (since 6.4)	Disable logfile	**Default = Disabled**. If **activ** are written to the logfile a are still shown in the mess
LogfileFilter (since 5.4)	RegEx for logfile messages	**Default = ""**. RegEx to filte not empty and the filter ru not be displayed in the log window. The text to which found in the logfile, exclu
LogfileVerbose	Activate extended protocol	**Default = Enabled**. If **activ** written to the logfile. If **de** except KL_DEBUG are writ
LogMessageCount (since 4.1)	Number of messages per message type	**Default = 25**. Number of m that are saved for the stat
MatterWatch (since 9.0)	Enable extended protocol for Matter	**Default = Disabled**. If **activ** the Matter stack are writte messages.
MaxLoginAttemptsBeforeLockdown (since 5.1)	Login attempts until blocked	**Default = 15**. Number of i address is completely bloc
MaxLoginAttemptsBeforeSlowdown (since 5.1)	Login attempts until delay	**Default = 5**. Number of inc logins from this IP address down is defined by MaxLoginAttemptsSlowdo
MaxLoginAttemptsLockdownDuration (since 5.1)	Duration of the login lock	**Default = 900 (in seconds)** for the affected IP addres
MaxLoginAttemptsSlowdownDuration (since 5.1)	Duration of the login delay	**Default = 300 (in seconds)** removed for the affected
MaxLoginAttemptsSlowdownWaitTime (since 5.1)	Time of delay	**Default = 5000 (in millisec** milliseconds for slowed do

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
MessageQueueWatch	Log processing queue	**Default = Disabled**. If **activ** of the internal processing Messages whose processir MessageQueueWatchLimi and statistics can be retrie IPS_GetInstanceMessageS these are not logged.
MessageQueueWatchLimit (since 6.1)	Delay in milliseconds	**Default = 50 (in millisecor** milliseconds from which a logfile if MessageQueueW zero (0) is specified, nothir
MessageRingBufferSize (since 4.1)	Number of buffer messages	**Default = 8192**. Indicates are buffered. If the maxim first (oldest) messages are used for IPS_GetSnapshot
NATPublicIP (since 5.4)	Set NAT Public IP	**Default = ""**. If NATSuppor automatically used everyw the public IP can be set ma instances.
NATSupport (since 5.0)	Activate NAT extensions	**Default = Disabled**. If **activ** are activated, e.g. in the K Socket, in order to specify behind the NAT. This is par operation in Docker conta bridge mode by default.
NotificationLimit (since 5.4)	Limit of push notifications	**Default = 250** (maximum maximum number of push hours. All inquiries beyonc error message and writter reset every 24 hours, giver reached. If the limit has be be corrected and the servi push messages can be sen
NotificationWatch (since 5.5)	Log push notifications	**Default = Disabled**. If **activ** push notification service is
OAuthWatch (since 4.3)	Enable OAuth endpoint protocol	**Default = Disabled**. If **activ** OAuth endpoint are logge
ObjectsLimit (since 6.0)	Maximum number of objects	**Default = 50000**. Minimun number of available objec are only applied if the licer higher variable limit, other
OPcacheSupport (since 5.0)	Activate PHP's OPcache	**Default = Enabled (up to 5** PHP scripts are saved in m and parsing the scripts eac
ProxyConnectLimitFrameRate (since 5.5)	Number of frames in the stream via Connect service	**Default = 2**. Specifies whe per second via the Connec in order to conserve the d
ProxyConnectReduceQuality (since 5.5)	Reduce the quality of the stream	**Default = Enabled**. If **activ** stream via the Connect se conserve the data volume

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
ProxyConnectReduceResolution (since 5.5)	Limit the resolution of the stream	**Default = Enabled**. If **activ** stream via the Connect se maximum of 720p.
ProxyInterface (since 7.0)	Setting the IP address for Proxy/VoIP	**Default = ""**. IP address of be used for the Proxy/VoI set if the automatic detec
ProxyLimitFrameRate (since 5.5)	Number of frames in the local stream	**Default = 0**. Specifies whe per second for local conne order to conserve the data
ProxyPassthrough (since 7.0)	Forward streams without transcoding	**Default = Enabled**. If **activ** WebRTC are forwarded wi possible (passthrough), wh
ProxyPort (since 9.0)	Port for SIP (VoIP/DoorIP)	**Default = 5060**. Local port for VoIP/DoorIP are accep
ProxyRTSPBuffer (since 5.5)	Size of the RTSP buffer	**Default = 524288 (in byte** buffer for RTSP streams. T more latency the stream h picture can be with bad co
ProxyRTSPEnforceTCP (since 6.1)	Enforce RTSP via TCP	**Default = Disabled**. If **acti** always retrieved via TCP in helpful, e.g. in case of pac
ProxySTUN (since 8.0)	Use STUN for WebRTC	**Default = Enabled**. If **activ** Symcon is used for WebRT streams) to determine the ProxyTURN is enabled as v instead.
ProxyThreadCount (since 8.0)	Number of threads for Proxy/VoIP	**Default = 4**. Number of th processing Proxy/VoIP cor
ProxyTURN (since 8.0)	Use TURN for WebRTC	**Default = Enabled**. If **activ** Symcon is used for WebRT streams), which can relay t connection is possible.
ProxyUseHWAccel (since 5.5)	Hardware acceleration index for RTSP	**Default = -1 (automatic)**. S available hardware acceler decoding. 0 = never, 1..x = acceleration.
ProxyWatch (since 5.1)	Log RTSP/MJPEG streams	**Default = Disabled**. If **acti** information about called F logged.
SaveInterval	StorageInterval	**Default = 10 (in minutes)**. the cycle in which the sett to the hard disk. When exi are written regardless of t (0) is specified, the setting (not recommended).
ScriptOutputBufferLimit (since 5.2)	Define script output length	**Default = 1048576 (in byt** script output can have. If t script is immediately abort "Output-Buffer exceeds Li

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
ScriptWatch	Log script executions	**Default = Disabled (up to** script executions are logg**deactivated**, no script exe and outputs (e.g. via echo)
ServerContentTimeout (since 6.2)	Time limit for transfers	**Default = 300 (in seconds)** which a connection is imm transferring a request and completed by then. Only a connections (HTTP), there the value zero (0) is specifi
ServerHardQueueBytesLimit (since 5.3)	Byte limit of the queue	**Default = 33554432 (in by** of all messages in the que is terminated immediately
ServerHardQueueSizeLimit (since 5.3)	Message limit of the queue	**Default = 65536**. When th queue is reached, the conn immediately.
ServerHardQueueTransferTimeout (since 5.3)	Queue timeout	**Default = 300 (in seconds)** which an inactive connecti terminated.
ServerLogging (since 5.0)	Activate server logging	**Default = Disabled**. If **activ** the web server (port = 377 only affects the server on servers the setting in the r into account. The log file i depending on the operati logfiles folder. See Installa
ServerLoggingFilter (since 5.0)	Activate filter for server logging	**Default = Enabled**. If **activ** (IPS_GetSnapshotChanges are not written to the "acc ServerLogging is active. If written to the logfile.
ServerMaxPostSize (since 5.1)	Maximum bytes for POST	**Default = 25165824 (in by** bytes that can be sent to t
ServerSecurity (since 4.1)	Activate CORS protection	**Default = Enabled**. If **activ** the web server is activated
ServerSoftQueueBytesLimit (since 5.3)	Bytes soft limit of the queue	**Default = 8388608 (in byt** of all messages in the que is disconnected if the ServerSoftQueueTransfer exceeded.
ServerSoftQueueSizeLimit (since 5.3)	Message soft limit of the queue	**Default = 32768**. When th queue is reached, the conn ServerSoftQueueTransfer exceeded.
ServerSoftQueueTransferTimeout (since 5.3)	Queue time soft limit	**Default = 10 (in seconds)**. successful transmission af apply to the size and numb

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
ServerTokenExpiration (since 8.0)	Timeout for internal tokens	**Default = 3600 (in second** RBAC feature, username a for a short-lived token, wh specified number of secor required.
ServerUserFolderPassword (since 5.1)	Basic authentication password	**Default = ""**. Password for the web server to secure t
ServerUserFolderUsername (since 5.1)	Basic authentication username	**Default = ""**. Username fo the web server to secure t
SettingsWatch (since 4.0)	Log saving of settings	**Default = Disabled**. If **activ** to the logfile as soon as th in the SaveInterval.
SOAPEnabled (up to 3.4)	Activate SOAP	**Default = Disabled**. If **activ** can be reached via port 37
StoreDownloadTimeout (since 8.1)	Timeout for downloads from the Module Store	**Default = 60000 (in millise** that the download of a mo may take before a timeout aborted with an error mes
ThreadCount	Number of threads	**Default = 50**. Minimum = 2 PHP threads that are avail thread consumes a few me percentage of the CPU loa value can be determined f Limitations.
ThreadHotReserve (since 9.1)	Number of PHP threads kept ready	**Default = 5**. Number of PH keep their PHP state so th without delay. All further t when needed and release ThreadIdleTimeout. The va ThreadCount.
ThreadIdleTimeout (since 9.1)	Idle time of PHP threads	**Default = 300 (in seconds)** which an unused PHP thre more than ThreadHotRese the value zero (0) is specifi their memory.
ThreadMemoryCleanup (since 6.2)	Memory cleanup in PHP threads	**Default = 10 (in MB)**. If the execution in a PHP thread MB, the internally used me is cleaned up afterwards t consumption of the PHP t
ThreadQueueLimit (since 4.0)	Thread queue limit	**Default = 50**. Specifies the requests that are waiting i execution. Any further inq an error message. By defa not be changed, but instea scripts have a short runtim
TimerThreadHotReserve (since 9.1)	Number of timer threads kept ready	**Default = 10**. Minimum = 1 executing the timers of ins kept ready. If needed, furt which are terminated agai TimerThreadIdleTimeout.

OPTION-NAME	SPECIAL SWITCH FUNCTION	DESC
TimerThreadIdleTimeout (since 9.1)	Idle time of timer threads	**Default = 300 (in seconds)** which an additionally start terminated. If the value ze threads are never termina
VariableWatch	Activate extended variable log	**Default = Disabled (up to** value changes/updates of the logfile. If **deactivated**, logfile.
VoIPInterface (since 5.2, up to 6.4)	Setting the IP address for VoIP	**Default = ""**. IP address of be used for the VoIP conn the automatic detection d Superseded by the new Pr that is available since IP-Sy
VoIPLogLevel (since 5.2, up to 6.4)	Set debug output level for VoIP	**Default = 0 (disabled)**. The and 3 (Recommended: 2). means the highest level of determines the level of de in the message log. Supers special switch that include with IP-Symcon 7.0.
WebSocketWatch (since 5.1)	Activate extended WebSocket protocol	**Default = Disabled**. If **activ** information is output on t channel, such as Open, Clo limits have been exceedec