TAPKO
TECHNOLOGIES GMBH

SIMip

IP Interface for KNX
IP/KNX Data Server

Technical & Application Description

This document is property of the company named at the last page.
Without written approval, it may not be reproduced or commercialised, distributed or presented to other individuals for commercial purpose.
Details and information contained within may be subject to change without notice. For the accuracy of the document no warranty is given.
All rights reserved.

Content

1 Product Description

The SIMip data server opens the possibility to connect arbitrary IP products to a KNX system. Non-KNX devices with suitable IP connectivity (TCP/IP) can be controlled via the KNX TP line. Regarding IP end devices, SIMip hereby serves as extension for each kind of IP unit that was not originally designed for KNX connection.

IP side and KNX TP are coupled having a galvanic isolation in between. Extended frames and long telegrams with up to 240 bytes APDU length are supported.

The SIMip device supports up to 1024 objects. Operating modes RAW Mode, Interoperability Mode and Transparent Mode are available. The KNX-certified communication system including all necessary functions for conversion of KNX datapoint types is already contained. Transmission properties can be configured for every group object separately. Actual object values are stored in the device internal memory. With the Function button, the IP device(s) linked to SIMip can be disconnected to start a linkage reset (short button press). With a long Function button press, SIMip can be reset to its factory default state.

SIMip is a rail-mounted device for installation in distribution boards on 35 mm DIN rails. Supplied by the KNX bus line, the device does not require an additional power supply. By a standard RJ45 connector, SIMip can be connected to an Ethernet network to establish the necessary IP connections to the concerned IP end devices.

A comfortable web front-end shows actual information about device details, like it´s state and used addresses (individual address, IP connections of SIMip). IP configuration settings can be set here. Programming Mode, Linkage Reset and the firmware update process (to update the firmware via IP) can also be activated here.

1.1 Front Panel

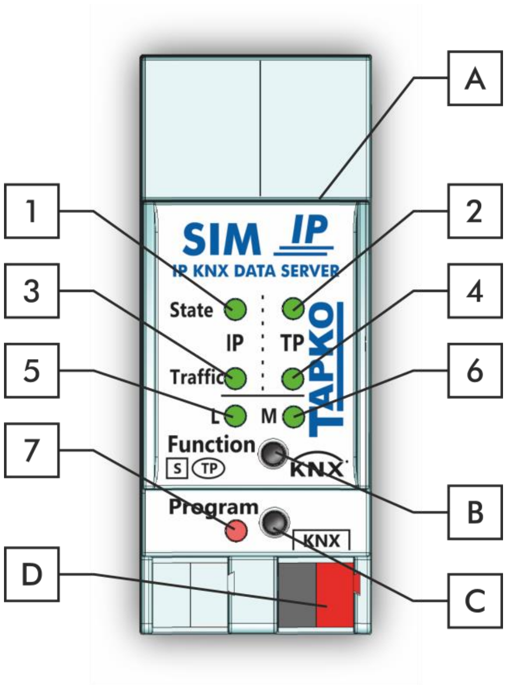


Figure 1: Front View

Table 1: Front Panel Elements

LEDs		Buttons / Connectors	
1	State IP	A	Ethernet Connector
2	State KNX TP	B	Function Button
3	Telegram Traffic IP	C	Programming Button
4	Telegram Traffic KNX TP	D	KNX TP Connector
5	IP Linkage		
6	Operating Mode		
7	Programming LED		

1.2 LED Indication

Following table gives a general description of the LED display indication intended for normal operation. Constellations of LED lighting during active special functions are further described in next chapter.

Table 2: Normal LED Display

Number	LED	Color	Explanation / Range
1	**State IP**	green	IP connection OK (connection established)
		orange	Factory Reset
		< off >	No IP connection
2	**State KNX TP**	green	KNX connection OK
		orange	Disconnect IP device(s) / Linkage Reset
		< off >	KNX TP line not connected
3	**Telegram Traffic IP**	blinking green	Telegram traffic extent indicated by blinking
		< off >	No telegram traffic
4	**Telegram Traffic KNX TP**	green	Telegram traffic extent indicated by blinking
		blinking red	Transmission error (BUSY, NACK, missing IACK)
		< off >	No telegram traffic
5	**IP Linkage**	green	IP device(s) connected
		orange	Four IP devices connected
		< off >	No IP linkage
6	**Operating Mode**	green	RAW/Interoperability Mode
		red	Transparent Mode
7	**Programming LED**	red	Programming Mode active
		< off >	Programming Mode not active

1.3 LED Indication of Special Functions

During an active special function, only LEDs described here are lighting. Other LEDs are off.

Table 3: LED Status Display for Linkage Reset

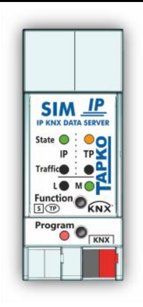
Number	LED	Color	Comment	
1	**State IP**	green	light is off if not connected	
2	**State KNX TP**	orange		
5	**IP Linkage**	< off >	turns to off	
6	**Operating Mode**	green	lights red in Transparent Mode	

Table 4: LED Status Display for Factory Reset after first Function Button Press

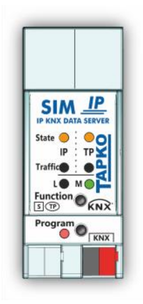
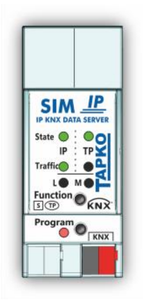
Number	LED	Color	Comment	
1	**State IP**	orange	lights red if not connected	
2	**State KNX TP**	orange		
5	**IP Linkage**	< off >	lights green (or orange) in case of connected IP device(s)	
6	**Operating Mode**	green	lights red in Transparent Mode	

Table 5: LED Status Display for Firmware Update

Number	LED	Color	Comment	
1	**State IP**	green		
2	**State KNX TP**	blinking green		
3	**Telegram Traffic IP**	green		
5	**IP Linkage**	< off >		
7	**Operating Mode**	< off >		

1.4 Commissioning

Please note for commissioning with default settings:

- Individual Address is 15.15.255
- Connection port is 12004

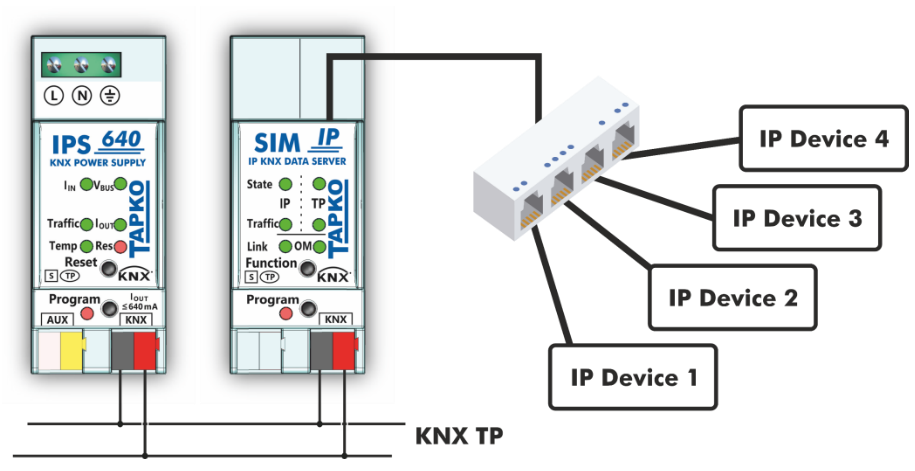


Figure 2: Connection Scheme

Please also read chapter 1.5 Important Notes before putting the device into operation.

1.5 Important Notes

It is recommended to participate the standardized courses of a KNX-certified training center before installing, programming, and commissioning a KNX system. Here, the participant gains the necessary knowledge and skills, also required for troubleshooting, by practical exercises.

Please read this chapter carefully before first use and installation:

1.5.1 Installation and Commissioning

- In the case of damage (at storage, transport) no repairs may be carried out by unauthorized persons
- After connection to the KNX bus system, the device works with its default settings
- Warning: Do not connect to 230 V. The device is supplied by the KNX bus and does not require any additional external power supply
- The device may only be installed and put into operation by a qualified electrician or authorized person
- For planning and construction of electric installations the appropriate specifications, guidelines and regulations in force of the respective country have to be complied
- For configuring, use the ETS (or ETS Inside)

1.5.2 Mounting and Safety

- For mounting use an appropriate equipment according to IEC60715
- Installation on a 35 mm DIN rail (TH35)
- Connect the KNX bus line as for common KNX bus connections with a KNX bus cable, to be stripped and plugged into a KNX TP connector
- Do not damage electrical insulations during connecting
- Installation only in dry locations

1.5.3 Maintenance

- Accessibility of the device for operation and visual inspection must be provided
- The housing must not be opened
- Protect the device from moisture, dirt and damage
- The device needs no maintenance
- If necessary, the device can be cleaned with a dry cloth

1.6 Feature Summary

Application Interface:

- configurable transmission parameters
- access to KNX group communication objects (runtime communication)
- access to KNX interface objects (configuration)
- configurable indication when group communication value was received

KNX features (RAW Mode + Interoperability Mode):

- integrated mechanism for configuration via KNX
- read requests from KNX are internally processed
- up to 1024 group objects available

KNX group communication objects (RAW Mode):

- transparent transmission of group communication object data
- data conversion not active
- telegram generation is controlled via IP
- configuration via IP

KNX group communication objects (Interoperability Mode):

- support of KNX datapoint types (DPT)
- data conversion for group object values (e.g. temperature -> DPT5)
- configurable sending conditions for all group communication objects
- configuration via ETS database entry
- indication when data received, value changed, positive/negative edges (DPT1)
- cyclic (time can be configured from 1 to 255 seconds, 1 to 255 minutes and 1 to 255 hours)
- complex sending conditions are available for certain datapoint types:
  - send on value difference
  - integrated threshold switch
    (to trigger a group communication object on threshold value passing)

Transparent Mode:

- receiving of all group-oriented messages
- sending to all group addresses
- no filtering of messages
- no data type restriction to a specific group address for sending
- unnecessary protocol-oriented information is removed
- also suitable for tracing and logging of messages

1.7 Resources in RAW and Interoperability Mode

Number of group addresses: 4096

Number of associations: 4096

Number of communication objects: 1024

Size of application parameters: 4 kbyte

Table 6: SIMip Communication Objects

Communication Object	Max. Size	Comments
1 – 128	14 bytes	Support of complex sending conditions for DPT5, DPT6, DPT7, DPT8, DPT9, DPT12, DPT13, DPT14
129 – 256	14 bytes	
257 – 1024	4 bytes	DPT16, DPT19, DPT235 are not supported

1.8 Command Overview

This chapter gives an overview of the commands that can be processed by SIMip. Their usage is also stated. Complete information concerning a command is shown by tables in chapter 4 Commands on IP side (grey hyperlinks guide to the command table of interest).

Certain commands modify the device internal memory. It is strongly recommended to not use these commands on a permanent basis. Command tables indicate the relevant commands.

1.8.1 General Commands

Table 7: General Commands

Command	Usage
dag	Get Individual Address
das *physicalAddress*	Set Individual Address
dpg	Get programming mode
dps *progMode*	Set programming mode
dr	Restart device
dsg	Get actual SIMip states
dvg	Get version
dts *data*	Device transparent set
gci	Reset to manufacturer default settings
pdg *(index count)*	Get parameter data
cpg	Get connection port
ctg	Get connection timeout
cts	Set connection timeout
idg *(ioIndex propertyID)* **idg** *(ioIndex propertyID elementIndex)*	Get interface object data
ids *(ioIndex propertyID) data* **ids** *(ioIndex propertyID elementIndex) data*	Set interface object data

1.8.2 Commands in RAW Mode and Interoperability Mode

Table 8: RAW/Interoperability Mode Commands

Command	Usage
ods *(objectNr) data*	Set object data (RAW Mode)
odg *(objectNr)*	Get object data (RAW Mode)
odt *(objectNr)*	Send group telegram
odr *(objectNr)*	Send group read telegram
ofg *(objectNr)*	Get RAM flags
ovs *(objectNr) data*	Set object value (Interoperability Mode)
ovg *(objectNr)*	Get object value (Interoperability Mode)
ogs *(objectNr) group*	Set sending group address
oga *(objectNr) group*	Add group address
ogd *(objectNr) group*	Delete group address
ogg *(objectNr)*	Get group addresses
ocs *(objectNr) DPT objectType comFlags sendConfig rcvConfig time*	Set object configuration
ocg *(objectNr)*	Get object configuration
dus	Set event generation
gug	Return the update flag of all group object
gcg	Return the valueChanged flag of all group object
gtg	Return the timeout flag of all group object

Table 9: RAW/Interoperability Mode Indications

gui	Receive the global update flag
oui	Receive update data for a communication object
dsi	Get actual states of SIMip

1.8.3 Commands in Transparent Mode

Table 10: Transparent Mode Commands

Command	Usage
dts *data*	Device transparent set
tds *(dest) data* **tds** *(dest length) data*	Send data by GroupValueWrite telegram
trs *(dest)*	Send request by GroupValueRead telegram
tes *(dest) data* **tes** *(dest length) data*	Send read response by GroupValueResponse telegram
tdi *(source dest length) data*	Send GroupValueWrite indication
tri *(source dest)*	Send GroupValueRead indication
tei *(source dest length) data*	Send GroupValueResponse indication
tdc *(source dest length) data*	Send GroupValueWrite confirmation
trc *(source dest)*	Send GroupValueRead confirmation
tec (source dest length) data	Send GroupValueResponse confirmation
tdn *(source dest length) data*	Send GroupValueWrite negative confirmation
trn *(source dest)*	Send GroupValueRead negative confirmation
ten *(source dest length) data*	Send GroupValueResponse negative confirmation

1.8.4 Explanation of Command Table Buildup

The command tables contained in chapter 4 have following general buildup.

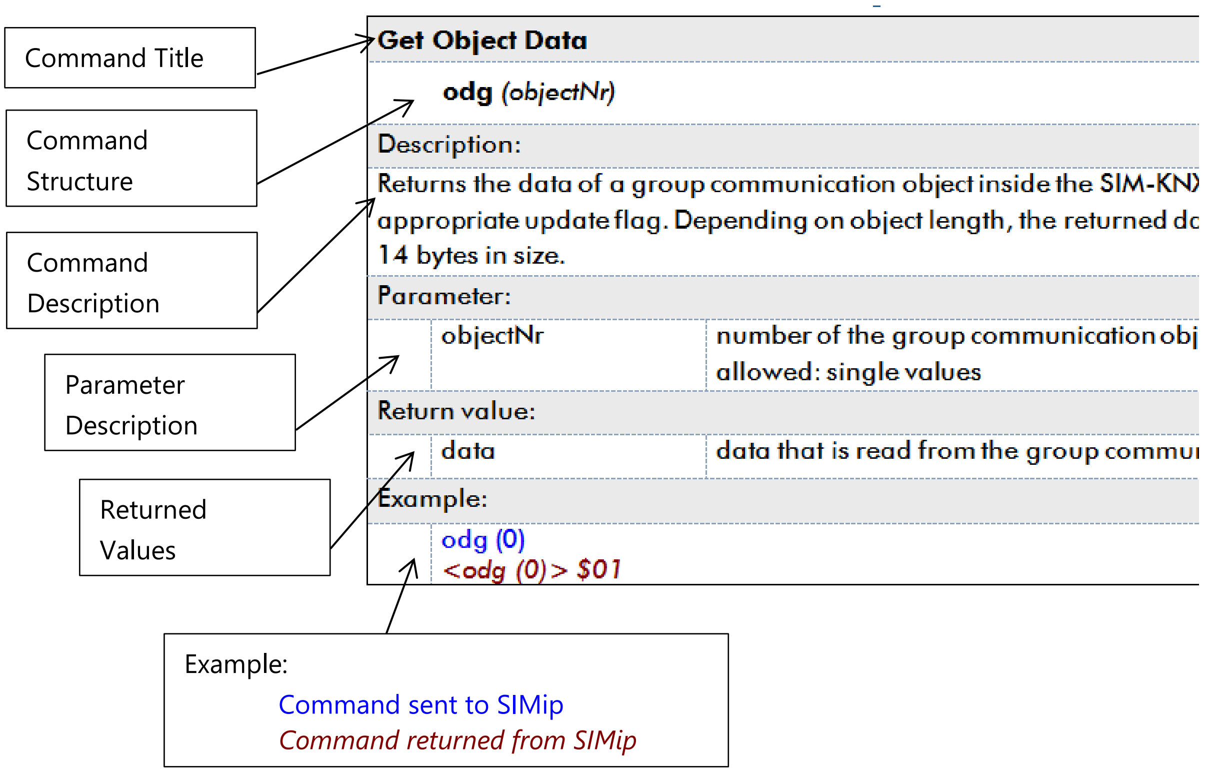


Figure 3: Command Table Example

2 Operational Description

In network installations, SIMip is used as IP/KNX data server for interfacing of IP devices to KNX. After connecting to the KNX TP bus line, SIMip operates with its default settings.

2.1 Introduction

In general, there are two ways of operating with SIMip:

- SIMip behaves like a normal device having its own Individual Address, an address table, communication objects and full device management.
  (→ RAW/Interoperabilty Mode)

- SIMip has an additional bypass for group-oriented communication. It is active in the lower layers. All group-oriented traffic is routed to IP without filtering and without any communication object linkage.
  (→ Transparent Mode)

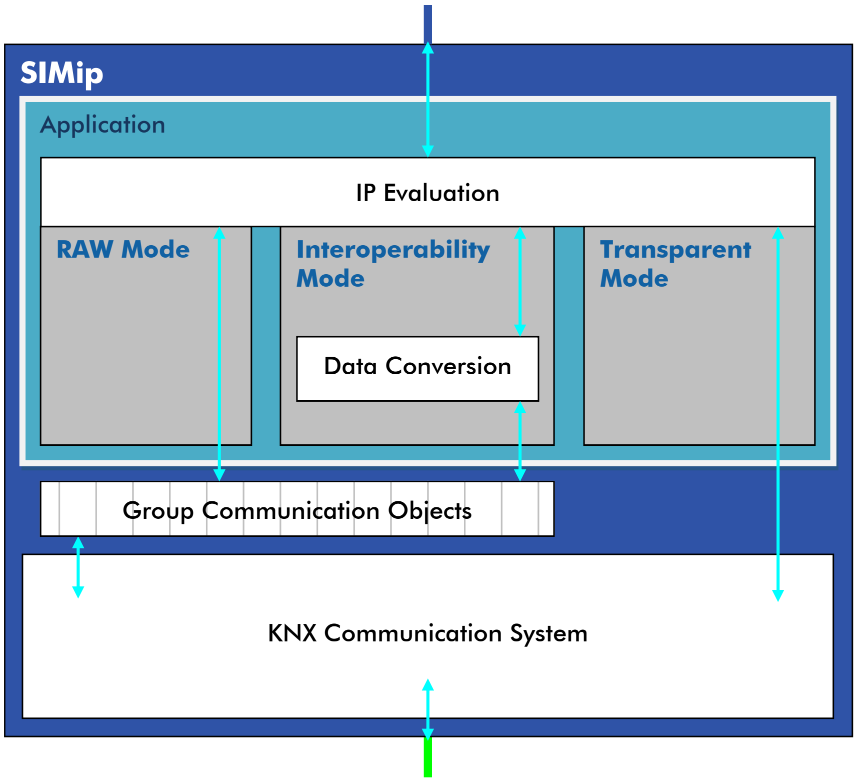


Figure 4: Block Diagram of SIMip Software

2.2 Operating Modes

2.2.1 RAW Mode

In RAW Mode, SIMip transparently transfers data between IP and KNX TP. Data transmission is controlled via IP. No external tool is additionally required.

2.2.2 Interoperability Mode

In Interoperability Mode, SIMip converts the data sent from IP to KNX TP to a KNX data format. According to the configuration settings, data transmission can be controlled via IP. Configuring Interoperability Mode can also be done by ETS.

For some datapoint types complex sending conditions are available.

2.2.3 Transparent Mode

Transparent Mode enables sending to and receiving from all group addresses without filtering. There are no data type limitations for sending.

Maximum object data size is 14 bytes.

Transparent Mode is suitable for tracing the group-oriented telegram traffic.

2.3 IP Network

2.3.1 DHCP/AutoIP

SIMip is designed for use in 10/100 BaseT networks compliant to IEEE802.3. The AutoSensing function sets the baud rate (10 Mbit or 100 Mbit) automatically. The IP address can be received from a DHCP server. For this, the automatic assignment of the IP address can be set in ETS (“obtain an IP address automatically”). If set so and no DHCP server was found, SIMip starts an AutoIP procedure and autonomously assigns the IP address. The IP address that SIMip receives during its start-up (via DHCP or AutoIP) is retained until next start-up (e.g. due to power off or reprogramming). If the IP address is supposed to have a fixed assignment a fixed IP address (as well as subnet mask and standard gateway) can be set by ETS and via the web-frontend.

If problems occur for IP address assignment, please ask your network administrator.

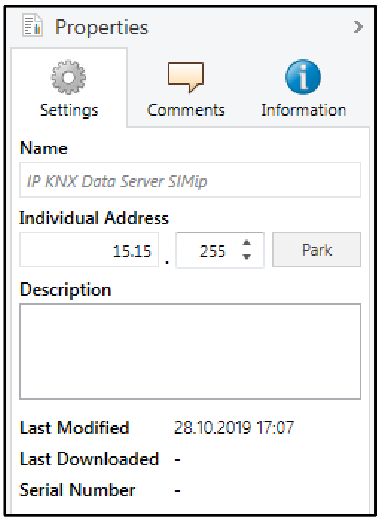
2.3.2 IP Firmware Update

To provide updating the firmware remotely via IP, SIMip has a bootloader functionality integrated. This function is called IP Firmware Update and can be executed in the web front-end. The download process for rewriting the program memory content is independent from ETS and replaces both communication stack and application software.

2.4 Programming

2.4.1 Programming of Individual Address and Application

The Individual Address (IA) can be assigned to SIMip by setting the desired address in the properties window of ETS. After downloading it into the device, SIMip can be addressed and identified by its new Individual Address.


Properties

Settings | Comments | Information

Name: IP KNX Data Server SIMip

Individual Address: 15.15 . 255 Park

Description

Last Modified 28.10.2019 17:07

Last Downloaded -

Serial Number -

Figure 5: ETS Properties Window

To download the Individual Address into the device, Programming Mode must be active. Successive pressing the Programming Button switches Programming Mode on and off. LED 7 lighting red indicates Programming Mode is on. Once the download is started in ETS, the Programming Button has to be pressed. After that, the new Individual Address becomes stored in the memory of the device.

To program devices of a line different to which the device used as ETS Current Interface is connected, a correct topology is mandatory.

The device is supplied with the Individual Address 15.15.255 (Factory Default Setting). It is recommended not to use this address for normal operation and to assign a different address when commissioning.

2.5 Special Functions

The Function Button activates SIMip´s special functions Linkage Reset and Factory Reset. IP devices linked to SIMip can be disconnected by activating the Linkage Reset and with the Factory Reset, SIMip can be reset to its manufacturer default state. Pressing the Function Button is also necessary during the Firmware Update process. The active special function status is indicated by the LED display (see chapter 1.3 LED Indication of Special Functions).

2.5.1 Linkage Reset

During normal operation, a rather short press (≈ 3 sec) disconnects the linked IP device(s). LED 2 signals the period of disconnection by lighting orange.

Table 11: Activation of Linkage Reset

Step	Linkage Reset
1	Hold Function button for 3 seconds
2	LED 2 now is orange indicating Linkage Reset is active
3	After switch-off, normal operation is indicated by LED 2 lighting green

2.5.2 Factory Reset

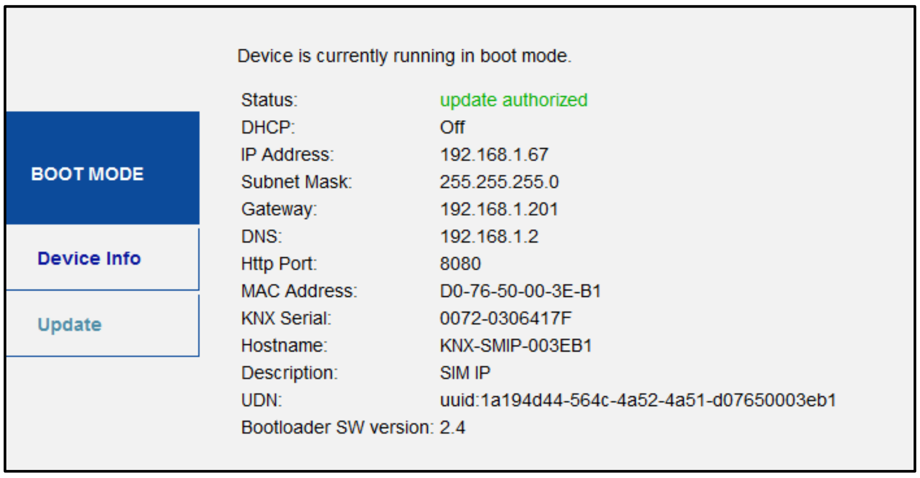
A long press (≈ 15 sec) of the Function Button soon followed by a short press (≈ 3 sec) executes the Factory Reset. After the first press, the LED display lights like described in Table 4: LED Status Display for Factory Reset after first Function Button Press. After the second press, all parameters (incl. Individual Address) will be set to factory default. Subsequently, LEDs show the normal operation display again.

Table 12: Activation of Factory Reset

Step	Factory Reset
1	Hold Function button for 15 seconds
2	LEDs 1/2 now are orange
3	Hold Function button for 3 seconds
4	Device restarts

2.5.3 IP Firmware Update Request

To start the Firmware Update download process, a short press on the Function Button is necessary when Programming Mode is activated. After a click on the “request update” button in the web front-end, SIMip switches to its boot mode (see chapter 7.5 IP Firmware Update) and ‘Status: update authorized’ is indicated.



Device is currently running in boot mode.

```
Status:                 update authorized
DHCP:                   Off
IP Address:             192.168.1.67
Subnet Mask:            255.255.255.0
Gateway:                192.168.1.201
DNS:                    192.168.1.2
Http Port:              8080
MAC Address:            D0-76-50-00-3E-B1
KNX Serial:             0072-0306417F
Hostname:               KNX-SMIP-003EB1
Description:            SIM IP
UDN:                    uuid:1a194d44-564c-4a52-4a51-d07650003eb1
Bootloader SW version: 2.4
```

Figure 6: Authorization of Update Request

Table 13: Activation of Firmware Update

Step	Firmware Update
1	Short press on Program button
2	Short press on Function button
3	Click on “request update” in the web front-end
4	LED2 is blinking green
5	Firmware file can be selected
6	Device restarts

2.6 Operation with Communication Objects (RAW Mode/Interoperability Mode)

Runtime communication on KNX is accomplished via communication objects. The SIMip memory contains a set of communication objects. Group Telegrams are received and values of associated group addresses are stored in the corresponding communication objects. SIMip always stores the last received communication object value. This value can be read out via IP.

Regarding the other direction, the IP device transfers a value to a communication object of SIMip and SIMip sends the value, according to its sending conditions, to the KNX bus.

It is also possible to read values from KNX side. Therefore, SIMip can be triggered by the IP device to send a read request on the bus. When a 'value read' was received from KNX, the response is handled inside SIMip and the stored communication object values are sent to the bus.

When an object value was received or changed, SIMip is able to create an indication and send it to IP.

For data exchange via communication objects SIMip offers two modes, RAW Mode and Interoperability Mode. By choosing the command, the operating mode can be selected for each communication object independently.

2.6.1 Communication Objects

All configuration data of SIMip is stored in non-volatile memory. A device restart or power down cannot cause configuration data changes.

Communication object values are stored in volatile memory. After a restart or power down of SIMip all present object values are lost.

Configuring communication, i.e. the communication objects of table 3, can be done

- via IP
- with the application specific ETS database entry from KNX side.

2.6.1.1 Feature Summary

- Standard features of communication objects as defined by the device model type:
  1. Enable sending and receiving of communication object values
  2. Enable receiving of values read from KNX side
  3. Setting of priorities
- Specific features of all SIMip communication objects:
  1. Format conversion in Interoperability Mode
  2. Object values can be sent to KNX
     - after receiving a value from IP
     - after changing a value on IP
     - for DPT1: on changes of positive edge, negative edge and/or both
     - cyclically
  3. Indications to IP can be generated
     - on receiving a value from KNX
     - on changing a received value from KNX
     - when the receive-timeout was elapsed
- Complex sending conditions are available for communication objects 1-128:
  1. Only configurable by ETS
  2. Usage with DPT5, DPT6, DPT7, DPT8, DPT9, DPT12, DPT13, DPT14
  3. Sending to KNX after a value change of certain amount
  4. Threshold switch for triggering a communication object after value crossing
  5. Setting threshold values for switching on and off (to set a hysteresis).

2.6.1.2 Configuring Objects by ETS Database Entry

Building an application specific ETS database can be done by adapting the generic ETS database that is made available. Required information to build the ETS database entry can be found in chapter 6 ETS.

Settings that can be set via IP can also be set by ETS.

With an ETS database entry, complex sending conditions can be used.

2.6.1.3 Configuring Objects via IP

Configuring communication objects is divided in 3 parts:

1. Assignment of group addresses
2. Setting the communication parameters
3. Configuring the indications

Assignment of group addresses

Group addresses for the communication objects are configured with the commands:

- **ogs** (set sending group address)
- **oga** (add group address)
- **ogd** (delete group addresses)

Each communication object can be associated with several group addresses. One of these associated group addresses always is the sending group address for sending object values (on the bus). All other associated group addresses are used to receive the object values.

Receiving group addresses are set by the command **oga** (e.g. **oga (1) 1/0/0**). Group addresses can be transferred as hex number ($1000) or in ETS format (2/0/0).

The sending group address is set by **ogs**. When there was one sending group address present before sending this command, this sending group address remains the receiving group address. When the sending group address was deleted or no group address is marked for sending, one of the receiving group addresses automatically becomes the sending group address.

Single group addresses can be deleted by the command **ogd**. To delete all group addresses associated to one communication object the command **ogd (0) "all"** can be used.

The number of group addresses which can be associated to the communication objects is limited by the global address resources and the association table.

For each group address one address table entry is used, no matter how many communication objects are associated to this group address.

For each association between group address and communication object one entry in the association table is used.

Setting the communication parameters

Parameters for communication objects are configured by the command **ocs**. The current communication object configuration can be retrieved via **ocg**.

Commands have following parameters:

• ***DPT***	data point type
• ***objectType***	type of the communication object
• ***flags***	configuration flags
• ***sendConfig***	configuration when the value is sent
• ***rcvConfig***	configuration when the indications will be sent
• ***time***	cycle time

The parameters ***DPT*** and ***objectType*** set operation mode and size/format of a communication object.

Setting the data format for Interoperability Mode is done by selecting the desired data point type with the parameter ***DPT***. Then the parameter ***objectType*** is not used.

To set the size of a communication object in RAW Mode, the parameter ***DPT*** must be set to 0 and the parameter ***objectType*** must be set to the required size.

The total object data size for communication objects is limited.

The ***flags*** parameter contains the configuration flags of a communication object which are also displayed in ETS. With this flags, sending and receiving of communication object values can be controlled.

The ***sendConfig*** parameter configures the behavior, when to send an object value on the KNX bus. Depending on this parameter, the object value is sent when a value was received from the IP, or only when it was changed, or e.g. cyclically.

Via the ***rcvConfig*** parameter it can be configured which indication is sent by the IP.

The ***time*** parameter sets the time period for cyclic sending and for the receive-timeout. If this parameter is set to 0, cyclic sending and receive-timeout are switched off.

When one of the parameters not needs to be modified by the command **ocs**, the relevant single value can be replaced by the wildcard character '*'.

2.6.2 Transferring Communication Object Data via IP

The data of a communication object is set with the commands

- **ods** in RAW Mode,
- **ovs** in Interoperability Mode.

For receiving the communication object data, the commands are

- **odg** in RAW Mode,
- **ovg** in Interoperability Mode.

In RAW Mode, the format in which the data is transferred to is hex bytes. In Interoperability Mode, the data format depends on the DPT that is used and can be, for example, a simple number, a float value or a string. Data formats for most used DPTs are described in this chapter. The complete list of DPTs supported by SIMip can be found in chapter 4.3.5.2 Group Communication Objects.

2.6.2.1 RAW Mode

In RAW Mode, raw data is exchanged between SIMip and the IP device(s). SIMip has no knowledge about format and semantics of exchanged data. Only the communication object size is known by SIMip.

Usually, RAW Mode is used for configuring SIMip via IP.

Configuring communication objects in RAW Mode can be done by setting the DPT to 0 with the command **ocs** and then setting the object size with the object type parameter.

2.6.2.2 Interoperability Mode

In Interoperability Mode, SIMip knows communication object size and datapoint type (DPT). As they are standardized to guarantee interworking of all devices, DPTs are defined by format and usage.

SIMip supports a wide range of datapoint types.

Configuring communication objects in Interoperability Mode can be done by setting the DPT with the command **ocs**. The object type parameter is then ignored by SIMip.

Most commonly used KNX Datapoint Types

KNX datapoint types and their usage are standardized. All definitions of these datapoint types can be found in the KNX handbook "Volume 3 / Part 7 / Chapter 2: Datapoint Types" available from KNX Association.

- **DPT1 (1-BIT) → SWITCH**

These single bit DPTs are defined to be independent of any application. For example, DPT1.001 is used for switch on (1) and off (0). DPT1.008 is used for move up (0) and down (1). Further usages like enable/disable are also defined. SIMip transfers the data on IP as single values 0 or 1.

- **DPT3 (3-BIT) → DIMMING**

This DPT is defined as a 4-bit communication object. It is used for control of dimming (DPT3.007) and blinds (DPT3.008). The values of this DPT are interpreted as *c Stepcode*. Data format for SIMip: *c* {0,1}: control (0=off, 1=on)
Stepcode {000b...111b}: value

- **DPT5 (8-BIT) → SCALING**

This DPT is used for absolute dimming, absolute blind position, valve position, etc. Values are transmitted as 1-byte values. 100% is coded as 100 (DPT5.001), 360(DPT5.003) and 255 (DPT5.004). In Interoperability Mode, values are transmitted as single values in the range of 0 to 100. (Due to data conversion of values, it may happen that the value that is read back from SIMip is not exactly the original one that was written.)

- **DPT9 (2-OCTET) → TEMPERATURE**

This DPT is defined as a 16-bit floating point value. Data format for SIMip (Interoperability Mode): single value. (Due to data conversion of values, it may happen that the value that is read back from SIMip is not exactly the original one that was written.)

- **DPT10 (3-OCTET) → TIME**

On KNX, time is transmitted as a 3-byte value. Data format of SIMip: 4 kinds of value formats (*weekday hour minute second*)

- **DPT11 (3-OCTET) → DATE**

This DPT is similar to DPT10. Data format of SIMip: 3 kinds of value formats: *day month year*

2.7 Operation without Communication Objects (Transparent Mode)

In Transparent Mode, SIMip activates a bypass to channel all group-oriented messages from the lower layers directly to the IP side without filtering. All requests coming from IP side are directly sent to the lower layers.

Care must be taken when Transparent Mode is switched on because the bypass activation won´t deactivate the object handling. Erasing all communication objects with the command **gci** prevents unpredictable interference between objects and incoming messages.

Transparent Mode can be switched on or off with the command **dts**.

2.7.1 Transparent Mode Communication

To use Transparent Mode, it is essential to understand the communication mechanism for group-oriented communication. In the following, three communication situations are described in detail. The appropriate ASCII commands for IP are indicated by green letters.

Situation1:One device wants to distribute a value (simplest situation)

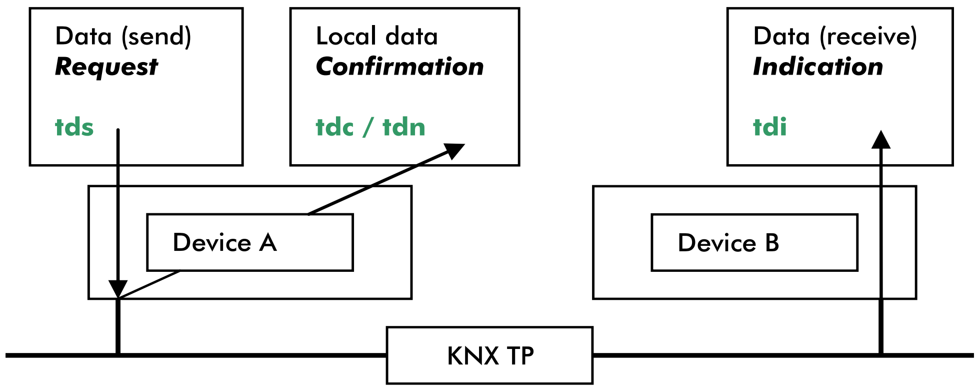


Figure 7: Communication Situation Case1

A ‘send data request’ is sent to SIMip. This request is forwarded to the bus by device A. When local confirmations are enabled, messages plus the information of successful or faulty transmission are returned to IP. (This procedure bases on the result of immediate acknowledge reception on the KNX bus after transmission of the message.) The remote device B receives the message from device A and indicates this fact to its own IP side by sending a data indication.

Situation2: A device is requested to send its actual value

This request for data is called a 'read request'. A local confirmation is sent via IP once the transmission is completed. The remote device B receives the request and sends a read indication to its own IP side.

This step alone does not yet result in getting the requested value.

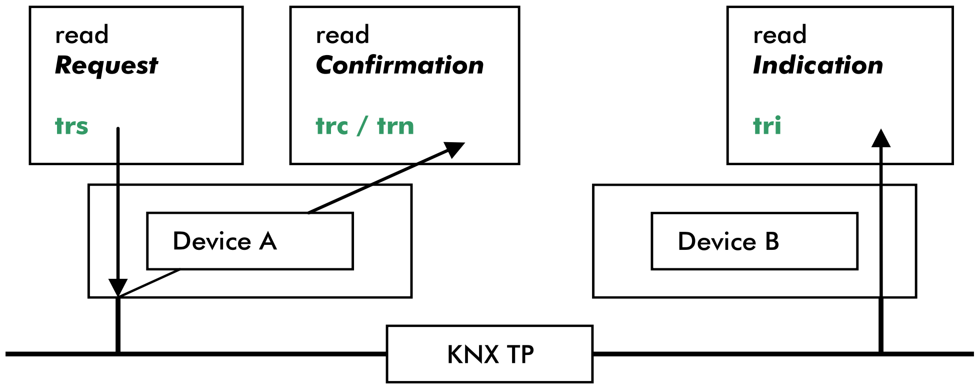


Figure 8: Communication Situation Case2

Situation3: A device sends the response as a result to the request of Situation2

The remote device B sends a response message from its IP side to the KNX bus. In the remote device B it is locally confirmed and the confirmation message is distributed to all other devices connected with the KNX bus system. Device A receives this confirmation message and sends a response indication to its IP side.

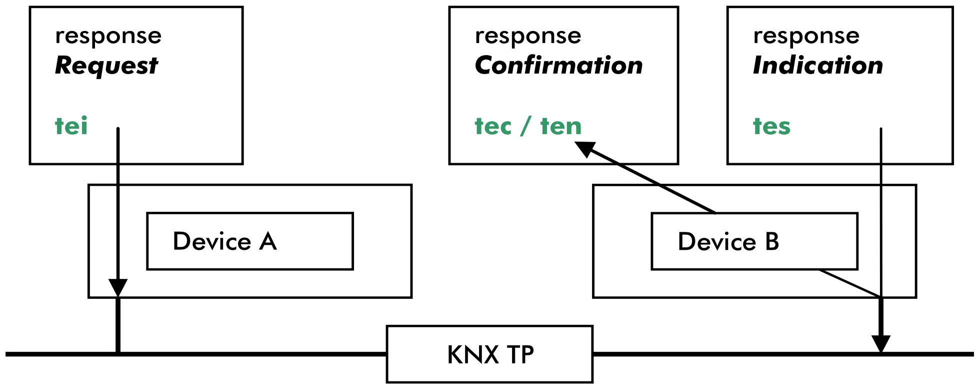


Figure 9: Communication Situation Case3

2.7.2 Transparent Mode Configuration

In Transparent Mode, the local confirmations can be suppressed by an appropriate setting of the **dts** command parameters.

2.8 Device Information

2.8.1 Connection Port

The connection port number can be read by the command **cpg**.

For connecting to SIMip the connection port must be known by the client (IP devices, IP units, terminal emulator software etc.).

The default setting for the connection port can be found here: 9.1 State of Delivery.

2.8.2 Connection Timeout

With activation of the connection timeout the time after receiving a command is counted. When there is no further IP communication during the connection timeout interval, the connection will be disconnected as soon as the pre-set value for the interval is reached. The connection timeout interval can be read and set by commands **ctg** and **cts**.

2.8.3 Individual Address

Individual Address and its assignment are completely handled inside SIMip. Usually, there is no need to set or read the Individual Address from IP. If required anyway, there is the possibility to have it read and set by the commands **dag** and **das**.

2.8.4 Programming Mode

Programming Mode, Programming LED and Programming Button are handled inside SIMip. Usually, there is no need to set or read the Programming Mode state from IP. If required anyway, there is the possibility to have it read and set by the commands **dpg** and **dps**.

2.8.5 Other Device Information

SIMip also provides other device information like version (**dvg**) and state of the internal application (**dsg**).

3 Examples for using SIMip

This chapter gives a short info how SIMip can be used with and without ETS database entry, and in Transparent Mode.

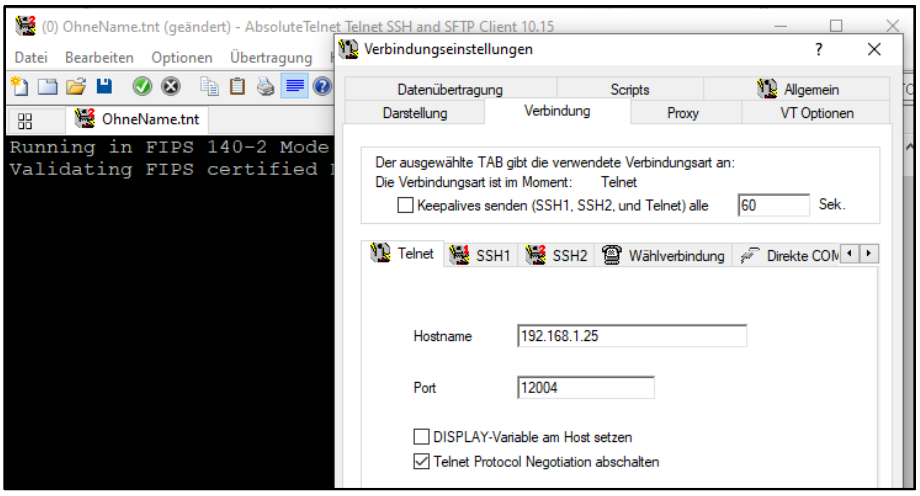
3.1 without ETS Database Entry

In this case the complete configuration of SIMip can be done via IP:

- Configuration of communication objects
- Assignment of Group Addresses

Example: Connection via Absolute Telnet

To establish an IP connection to SIMip, a terminal emulator like AbsoluteTelnet can be used. IP address and Connection Port have to be entered. Local Echo must be set to active.



```
Running in FIPS 140-2 Mode
Validating FIPS certified
```

Der ausgewählte TAB gibt die verwendete Verbindungsart an:
Die Verbindungsart ist im Moment: Telnet
☐ Keepalives senden (SSH1, SSH2, und Telnet) alle 60 Sek.

Hostname 192.168.1.25
Port 12004
☐ DISPLAY-Variable am Host setzen
☑ Telnet Protocol Negotiation abschalten

Figure 10: Connection Configuring in AbsoluteTelnet

3.2 with ETS Database Entry

In this case, all configurations of SIMip can be done by using the ETS database entry. Configuring via IP is not necessary.

During development, it may be necessary that SIMip is configured via IP.

3.3 in Transparent Mode

After switching on Transparent Mode, e.g. with the command **dts**, from KNX bus to IP all group-oriented messages are passed through in the form of *Transparent Data Indication* (normal writing coming from the bus), *Transparent Read Indication* (asking a sensor its value) or *Transparent Response Indication* (the answer on a request from the sensor). The information passed to IP is the Individual Address of the originating device (source), the destination Group Address and the data (containing length information) itself.

From the IP a *Transparent Data Send* (normal writing to the bus), a *Transparent Response Send* (answer to a request from the bus) and a *Transparent Read Send* (sending a request for a value to the bus) can be sent.

In this example no local confirmations are generated and the hex format is used for displaying received messages.

For getting the Transparent Mode status, see chapter 5 Implemented Application Interface Object and property ID 51.

4 Commands on IP side

4.1 Syntax

4.1.1 General Syntax (command based)

Syntax is command based and uses the ASCII char set. Commands are terminated by <CR>.

General syntax for commands

command<CR>
command (parameter)<CR>
command (parameter) data<CR>

A command contains up to 3 parts

- *command*
  The command itself. It defines the kind of operation that is executed.
- *parameter*
  specifies the element to be manipulated. For example, one of the group communication objects.
- *data*
  contains the values that are transmitted to SIMip. For example, the value of a group communication object or the Individual Address.

4.1.2 Values

There are different types of values. For every command, their usage is defined.

- **Single value** can either be a 1-byte value, a 2-byte value or a 4-byte value specified in one of following formats:

decimal:	*1234*
hexadecimal:	*$1234*
binary:	*%10101*

- **Hex stream** defines a sequence of hexadecimal bytes:
  #12345678
- **String** must be enclosed in quotation marks:
  "Hello"
- **Wildcard**
  In some commands it is allowed to use '*' as wildcard character.
- **Group address in ETS format** is used for group address manipulation:
  1/234
  1/2/45

4.1.3 Strings from SIMip

Three types of strings are sent from SIMip: Responses, Indications, and Error messages.

4.1.3.1 Responses

Responses follow the general syntax for commands.

General syntax for responses

If echoing the received command string is active, response syntax is:

<commandstring>returnValues<CR>

If echoing the received command string is not active, response syntax is:

returnValues<CR>

A response contains 2 parts

- *commandstring*
  is a general generic response for all commands. It can be configured whether the received command string will be returned or not.
- *returnValues*
  are the data that were requested. Values and their data format depend on the executed command.

Changing the responses in RAW/Interoperability Mode

The settings are stored in memory. They can be changed by ETS (see chapter 6 ETS) and with the Application Interface object ID52 (see chapter 5 Implemented Application Interface Object). Following settings are possible:

Bit 0:	0 = response without command string
	1 = response with command string
Bit 1:	1 = print "ok" if no data are following

Table 14: Examples for Changing the Responses in RAW/Interoperability Mode

Setting: 0x00	Setting: 0x01	Setting: 0x02	Setting: 0x03
odg (0) *$01*	odg (0) *<odg (0)>$01*	odg (0) *$01*	odg (0) *<odg (0)>$01*
odt (0)	odt (0) *<odt (0)>*	odt (0) *ok*	odt (0) *<odt (0)>* *ok*

4.1.3.2 Indications

Indications are sent without requests. They are independent from the response syntax. Indications are terminated by <CR><LF>. For details see chapter 4.3.6 Indications.

4.1.3.3 Error Messages

An error message is sent after an invalid command was received. Explanations of error codes can be found in chapter 4.5 Error Codes.

An error message contains 3 parts

- Keyword *"!error"*
- *error number*
- *received command*

Example of an Error Message

!error $0215 : <abc>

4.2 Command Reference (General)

4.2.1 Accessing Interface Objects

Get interface object data	
idg *(ioIndex propertyID)* **idg** *(ioIndex propertyID elementIndex)*	
Description:	
Get the data of one interface object property. The property is selected via ioIndex and propertyID. If the property is implemented as an array the elements are selected via elementIndex.	
Parameter:	
ioIndex	index to the interface object allowed: single values
propertyID	ID of the property allowed: single values
elementIndex	element of the property allowed: single values is set to 1, if skipped
Return value:	
data	Data, which are read from the property.
Example	
idg (5 52) *<idg (5 52)>$01*	

Set interface object data

ids *(ioIndex propertyID) data*
ids *(ioIndex propertyID elementIndex) data*

Description:

Set the data of one interface object property. The property is selected via *ioIndex* and *propertyID*. If the property is implemented as an array the elements are selected via *elementIndex*. This command modifies the internal memory. It is recommended not to use this command on a permanent basis.

Parameter:

ioIndex	index of the interface object allowed: single values
propertyID	ID of the property allowed: single values
elementIndex	element of the property allowed: single values is set to 1, if skipped
data	data that is written to the property, only for one element. allowed: single values, hex stream for elements of size >1 byte

Example:

```
ids (7 52) 1
<ids (7 52) 1>
```

4.2.2 Device Settings

Get Individual Address	
dag	
Description:	
Get the Individual Address of SIMip.	
Parameter:	
-	
Return value:	
physicalAddress	Individual Address as one single value
Example:	
dag *<dag>$ffff*	

Set Individual Address	
das *physicalAddress*	
Description:	
Set the Individual Address of SIMip. This command modifies the internal memory. It is recommended not to use this command on a permanent basis.	
Parameter:	
physicalAddress	Individual Address allowed: single values
Example:	
das $1508 *<das $1508 >*	

Get programming mode	
dpg	
Description:	
Get the Programming Mode state of the device. This state is also indicated by the LED.	
Parameter:	
-	
Return value:	
progMode	state of Programming Mode 0: off 1: on
Example:	
dpg *<dpg>$01*	

Set programming mode	
dps *progMode*	
Description:	
Set the state of Programming Mode of the device.	
Parameter:	
progMode	state of Programming Mode 0: off 1: on allowed: single values
Example:	
dps 1 *<dps 1>*	

Restart device
dr
Description:
Execute a restart of SIMip.
Parameter:
-
Example:
dr *<dr>* *gui $01*[1]

[1] if the global indication for restart is set

Get device state	
dsg	
Description:	
Get various states of the SIMip device.	
Parameter:	
-	
Return value:	
bit0	0: normal operation
	1: Transparent Mode
bit1	1: application loaded
bit2	1: application is running
bit7	1: device is in Programming Mode
Example:	
dsg	
<dsg>$06	

Get version

dvg

Description:

Get the version of SIMip.

Parameter:

-

Return value:

device version	single value, which shows the kind and the version of this device. high byte: general type of the device: 00: bus coupling via TP1 media low byte: version of this device
protocol version	version of the software protocol high byte: main version of this protocol. When this version changes, the protocol may have incompatible changes. low byte: sub version of this protocol. Higher sub versions are always upward compatible.
active Objects	number of activated communication objects

Example:

```
dvg
<dvg>$0001 $0001 $80
```

Reset to manufacturer default settings

gci

Description:

Clear all settings that were done by ETS and via IP, and execute a restart. This command modifies the internal memory. It is recommended not to use this command on a permanent basis.

Example:

```
gci
```

Get connection port

cpg

Description:

Get the connection port number of SIMip.

Parameter:

-

Return value:

connectionPort	connection port as one single value

Example:

```
cpg
<cpg>$2EE4
```

Get connection timeout

ctg

Description:

Get the connection timeout of SIMip.

Parameter:

-

Return value:

connectionTimeout	connection timeout as one single value

Example:

```
ctg
<ctg>$3E8
```

Set connection timeout

cts

Description:

Set the connection timeout of SIMip.

Parameter:

-

Example:

```
cts $2EE0
<cts $2EE0>
```

4.2.3 Parameter

Get parameter data

pdg *(index)*
pdg *(index count)*

Description:

Get the parameter settings that can only be written by ETS.

Parameter:

index	index of the parameter
count	number of parameters to be read out is set to 1 if skipped

Return value:

data

Example:

pdg (10 4)
<pdg (10 4)>$01 $02 $03 $04

pdg (11)
<pdg (11)>$02

4.3 Command Reference (RAW/Interoperability Mode)

4.3.1 Configuration

Set event generation	
dus *globalEvent*	
Description:	
Set the configuration of the global event generation. This command modifies the internal memory. It is recommended not to use this command on a permanent basis.	
Parameter:	
globalEvent	send global Event bit 0: at restart bit 3: if global update flag is set bit 4: if global changed flag is set bit 6: if timeout on IP has occured bit 7: if global receive-timeout is set
Example:	
dus $01 *<dus $01>*	

For getting the status see property ID 128 in chapter 5 Implemented Application Interface Object.

4.3.2 Accessing Group Communication Objects (RAW Mode)

Set object data

ods *(objectNr) data*

Description:

Set the data of a group communication object inside SIMip. Depending on the sending condition, transmission is automatically initiated. This command deletes the appropriate update flag. Depending on object length, the data size of this command is from 1 byte up to 14 bytes.

Parameter:

objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
data	data that is written to the group communication object. allowed: single values, hex stream (Take care that the correct length of the data is set.)

Example:

```
ods (0) 1
<ods (0) 1>

ods ($1) $0 $1 $2
<ods ($1) $0 $1 $2>

ods (1) #000102
<ods (1) #000102>
```

Get object data

odg *(objectNr)*

Description:

Get the data of a group communication object inside SIMip. This command deletes the appropriate update flag. Depending on object length, the returned data can be from 1 byte up to 14 bytes in size.

Parameter:

objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)

Return value:

data	data that is read from the group communication object.

Example:

```
odg (0)
<odg (0)> $01
```

4.3.3 Accessing Group Communication Objects (Interoperability Mode)

Set object value

ovs *(objectNr) data*

Description:

Set the value of a group communication object inside SIMip. Depending on the sending condition, transmission is automatically initiated. The data and its format depend on the configured datapoint type.

Parameter:

objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
data	data that is written to the group communication object. allowed: single values, hex stream (depends on data point type)

Example:

```
ovs (0) 1
<ovs (0) 1>

ovs ($1) 2100
<ovs ($1) 2100>
```

Get object value

ovg *(objectNr)*

Description:

Get the value of a group communication object inside SIMip. The data and its format depend on the configured datapoint type.

Parameter:

objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)

Return value:

data	data that is read from the group communication object. Its data format depends on the datapoint type.

Example:

```
ovg (0)
<ovg (0) >1

ovg ($1)
<ovg ($1)>2100>
```

4.3.4 Accessing Group Communication Objects (RAW/ Interoperability Mode)

Send group telegram	
odt *(objectNr)*	
Description:	
Start the transmission of the group communication object value on KNX as 'A_ValueWrite' without checking the sending condition.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
Example:	
odt (0) *<odt (0)>*	

Send group read telegram	
odr *(objectNr)*	
Description:	
Start the request of the group communication object value. An 'A_ValueRead' is transmitted on KNX.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
Example:	
odr (1) *<odr (1)>*	

Get RAM flags	
ofg *(objectNr)*	
Description:	
Get the RAM flags of the object.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
Return value:	
RAM flags	RAM flags of the object.
Example:	
ofg (1) *<ofg (1)> $01*	

Structure of the RAM flags

The RAM flags contain the information about the communication status of the communication object. Usually, only the three indications update, value changed and receive-timeout are of interest to the user application.

Table 15: Structure of RAM Flags

Bit	7	6	5	4	3	2	1	0
Flag	receive-timeout	not used	not used	value changed	update	read	transmission	transmission
	I	0	0	C	U	R	T	T

Table 16: Flag Description

Flag	**Flag Name**	**Description**
T	transmission	Information about the state of value transmission on KNX.
R	read	Flag to trigger the sending of 'value read request'. This flag is used in combination with the transmission state.
U	update	This flag indicates a value was received from the bus.
C	value changed on update	This flag indicates the value that was received from the bus changed the communication object value.
I	receive-timeout	This flag is set, when no value was received within the configured time.
0	-	-not used-

Return the update flags of all group objects	
gug	
Description:	
Get the update flags of all objects.	
Return value:	
updateFlags	packed update flags
Example:	
gug *<gug>$01 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00*	

Return the valueChanged flags of all group objects	
gcg	
Description:	
Get the valueChanged flags of all objects.	
Return value:	
valueChangedFlags	packed valueChanged flags
Example:	
gcg *<gcg>$04 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00*	

Return the timeout flags of all group objects	
gtg	
Description:	
Get the timeout flags of all objects.	
Return value:	
timeoutFlags	packed timeout flags
Example:	
gtg *<gtg>$10 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00 $00*	

4.3.5 Configuring Group Communication

4.3.5.1 Group Addresses

Set sending group address	
ogs *(objectNr) group*	
Description:	
Set the sending group address of a group communication object. The previous sending group address, if existing, will further be used, but as the receiving group address. This command modifies the internal memory. It is recommended not to use this command on a permanent basis.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
group	sending group address (as one single value) allowed: single values and ETS format
Example:	
ogs (0) $affe *<ogs (0) $affe>* ogs (0) 21/2046 *<ogs (0) 21/2046>* ogs (0) 21/7/254 *<ogs (0) 21/7/254>* ogs (0) 21/7/$fe *<ogs (0) 21/7/$fe>*	

Add group address	
oga *(objectNr) group*	
Description:	
Add a group address to a group communication object as the receiving group address. When there was no group address associated with the object before, this group address automatically becomes the sending group address. This command modifies the internal memory. It is recommended not to use this command on a permanent basis.	
Parameter:	
objectNr	number of the group communication object which is manipulated. allowed: single values (according to Table 6)
group	group address (as one single value) allowed: single values and ETS format
Example:	
oga (0) $affe *<oga (0) $affe>* oga (0) 21/2046 *<oga (0) 21/2046>* oga (0) 21/7/254 *<oga (0) 21/7/254>* oga (0) 21/7/$fe *<oga (0) 21/7/$fe>*	

Delete group address

ogd *(objectNr) group*

Description:

Delete one or all group addresses of a group communication object. When a sending group address is deleted, the next group address automatically becomes the sending group address. This command modifies the internal memory. It is recommended not to use this command on a permanent basis.

Parameter:

objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
group	group address (as one single value) allowed: single values and string "all"

Example:

```
ogd (0) $4711
<ogd (0) $4711>

ogd (0) 8/1809
<ogd (0) 8/1809>

ogd (0) 8/7/17
<ogd (0) 8/7/17>

ogd (1) "all"
<ogd (1) "all">
```

Get group addresses

ogg *(objectNr)*

Description:

Get the group addresses of a group communication object. The first group address is the sending group address.

Parameter:

objectNr	number of the group communication object that is read. allowed: single values (according to Table 6)

Return value:

group	group addresses as single values in hex format

Example:

```
ogg (0)
<ogg (0)>$4711 $affe
```

4.3.5.2 Group Communication Objects

Set object configuration

ocs *(objectNr) DPT objectType comFlags sendConfig rcvConfig time*

Description:

Set the configuration of a group communication object.
To configure the communication object in Interoperability Mode,
set the DPT and have the object type set to 0.
To configure the communication object in RAW Mode,
set the DPT to 0 and set the length by the object type parameter.
When the wildcard character '*' is used, the parameter will not be changed.

Parameter:

objectNr	number of the group communication object that is manipulated allowed: single values (according to Table 6)
DPT	datapoint type allowed: single values, * see also part Supported Datapoint Types (DPT)
objectType	object type allowed: single values, * see also part Group Object Types (objectType)
comFlags	configuration flags of the group communication object allowed: single values, * see also part Structure of Configuration Flags (comFlags)
sendConfig	configuration when value is sent allowed: single values, * see also part Send Configuration (sendConfig)
rcvConfig	configuration when indications is sent allowed: single values, * see also part Receive Configuration (rcvConfig)
time	delay time When time is set to 0, the send/receive timeout is disabled. The time base can be changed by sendConfig. allowed: single values, *

Example:

```
ocs (0) 1 0 $df $0001 $0001 0
<ocs (0) 1 0 $df $0001 $0001 0>

ocs ($1) 9 0 $df $0002 $0004 120
<ocs ($1) 9 0 $df $0002 $0004 120>

ocs ($1) 9 * * $0002 * 120
<ocs ($1) 9 * * $0002 * 120>
```

Get object configuration	
ocg *(objectNr)*	
Description:	
Get the configuration of the group communication object.	
Parameter:	
objectNr	number of the group communication object that is manipulated. allowed: single values (according to Table 6)
Return value:	
DPT	data point type see also part Supported Datapoint Types (DPT)
objectType	object type see also part Group Object Types (objectType)
comFlags	configuration flags of the group communication object see also part Structure of Configuration Flags (comFlags)
sendConfig	configuration when the value is sent see also part Send Configuration (sendConfig)
rcvConfig	configuration when indication is sent see also part Receive Configuration (rcvConfig)
time	delay time
Example:	
ocg(0) *<ocg(0)> 1 0 $df $0001 $0001 0*	

Supported Datapoint Types (DPT)

The following table gives an overview of the DPTs supported by SIMip. Complete usage information can be found in the Interworking Datapoint Types System Specifications.

Table 17: DPTs supported by SIMip

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
1	DPT 1 "1-bit"	Format:	*b* *b* {0,1}: value (0=off, 1=on)
		Object Size:	1 bit
		Usage:	switch on / off, move up / down, enable / disable, …
		Example:	ovg (0) *<ovg (0) >1*
2	DPT 2 "1-bit controlled"	Format:	*c v* *c* {0,1}: control (0=off, 1=on) *v* {0,1}: value (0=off, 1=on)
		Object Size:	2 bit
		Usage:	control
		Example:	ovg (1) *<ovg (1)>1 1*
3	DPT 3 "3-bit controlled"	Format:	*c StepCode* *c* {0,1}: control (0=off, 1=on) *StepCode* {000b…111b}: value Interval: $2^{(StepCode-1)}$
		Object Size:	4 bit
		Usage:	dimming control
		Example:	ovg (9) *<ovg (9)>1 5*
4	DPT 4 "character set"	Format:	A_8 character string
		Object Size:	1 byte
		Usage:	text, single charaters, ASCII
		Example:	ovg (120) *<ovg (120)>"h"*

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
5	DPT 5 "8-bit unsigned value"	Format:	*U*U_8 unsigned value {0...100}, allowed: single values (DPT5.001 DPT_Scaling)
		Object Size:	1 byte
		Usage:	scale (0...100 %), absolute dimming
200		Format:	U_8 unsigned value {0...360}, allowed: single values (DPT5.003 DPT_Angle)
		Object Size:	1 byte
		Usage:	angle (0...360°)
201		Format:	U_8 unsigned value {0...255} (DPT5.010 DPT_Value_1_Ucount)
		Object Size:	1 byte
		Usage:	counter pulses
		Example:	ovg (211) *<ovg (211)>54*
	can be used with complex sending conditions (communication objects 1 – 128)		
6	DPT 6 "8-bit signed value"	Format:	V_8 signed value {-128...127}
		Object Size:	1 byte
		Usage:	percent, counter pulses
		Example:	ovg (288) *<ovg (288)>127*
	can be used with complex sending conditions (communication objects 1 – 128)		
7	DPT 7 "2-octet unsigned value"	Format:	U_{16} unsigned value {0...65535}
		Object Size:	2 byte
		Usage:	counter pulses
		Example:	ovg (76) *<ovg (76)>45698*
	can be used with complex sending conditions (communication objects 1 – 128)		
8	DPT 8 "2-octet signed value"	Format:	V_{16} signed value {-32768...32767}
		Object Size:	2 byte
		Usage:	counter pulses, percent
		Example:	ovg (86) *<ovg (86)>-25789*
	can be used with complex sending conditions (communication objects 1 – 128)		

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
9	DPT 9 "2-octet float value"	Format:	signed float value {-671088.64...670760.96}
		Object Size:	2 byte
		Usage:	temperature, temperature change, brightness, wind speed, pressure, humidity, air quality, air flow, time, voltage, current, power, power density, ...
		Example:	ovg (99) *<ovg (99)>92815.27*
	can be used with complex sending conditions (communication objects 1 – 128)		
10	DPT 10 "time"	Format:	$N_3\ U_5\ U_6\ U_6$ N_3 {0...7}: weekday (1=Monday, 7=Sunday, 0=no day) U_5 {0...23}: hour U_6 {0...59}: minute U_6 {0...59}: second
		Object Size:	3 byte
		Example:	ovg (111) *<ovg (111)>4 21 39 11*
11	DPT 11 "date"	Format:	$U_5\ U_4\ U_7$ U_5 {1...31 }: day (valid number of days for the month) U_4 {1...12}: month U_7 {0...99}: year (<90 interpreted as 21th century)
		Object Size:	3 byte
		Example:	ovg (37) *<ovg (37)>25 10 2087*
12	DPT 12 "4-octet unsigned value"	Format:	U_{32} unsigned value {0...4294967295}
		Object Size:	4 byte
		Usage:	counter pulses
		Example:	ovg (66) *<ovg (66)>1234567*
	can be used with complex sending conditions (communication objects 1 – 128)		
13	DPT 13 "4-octet signed value"	Format:	V_{32} signed value {-2147483648...2147483647} (range is not validated)
		Object Size:	4 byte
		Usage:	counter pulses, flow rate, energy, long time
		Example:	ovg (51) *<ovg (51)>-1234327*
	can be used with complex sending conditions (communication objects 1 – 128)		

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
14	DPT 14 "4-octet float value"	Format:	F_{32} float value
		Object Size:	4 byte
		Usage:	value
		Example:	ovg (92) *<ovg (92)>5642.736*
	can be used with complex sending conditions (communication objects 1 – 128)		
16	DPT 16 "string"	Format:	A_{112} character string {maximum length: 14 characters} Response always has 14 characters (unused characters were set to 0 <NULL>)
		Object Size:	14 byte
		Usage:	Text, string, fixed length, ASCII
		Example:	ovg (123) *<ovg (123)>"Hello"*
17	DPT 17 "scene number"	Format:	U_6 U_6 {0...63}: scene number
		Object Size:	1 byte
		Example:	ovg (29) *<ovg (29)>54*
18	DPT 18 "scene control"	Format:	C U_6 C {0,1}: 0=control, 1=learn U_6 {0...63}: scene number
		Object Size:	1 byte
		Example:	ovg (870) *<ovg (870)>1 36*
19	DPT 19 "date time"	Format:	U_8 U_4 U_5 U_3 U_5 U_6 U_6 B_9 U_8 {0...255}: year (0 = 1900) U_4 {1...12}: month U_5 {1...31}: day of month U_3 {0...7}: day of week (1=Monday, 7=Sunday, 0=no day) U_5 {0...24}: hour of day U_6 {0...59}: minutes U_6 {0...59}: seconds B_9 {0,1}
		Object Size:	8 bytes
		Example:	ovg (19) *<ovg (19)>5 3 2020 2 9 8 4 0000001 00000001*

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
20	DPT 20 "absolute value"	Format:	*N_8* *N_8* {0…255}: unsigned value
		Object Size:	1 byte
		Usage:	Encoding absolute value
		Example:	ovg (10) *<ovg (10)>225*
21	DPT 21 "8-bit set"	Format:	*B_8* *B_8* {0,1}
		Object Size:	1 byte
		Usage:	General status and device control
		Example:	ovg (77) *<ovg (77)>10110010*
22	DPT 22 "16-bit set"	Format:	*usage not recommended, reserved for Functional Block*
		Object Size:	2 byte
		Usage:	DHW control, HVAC status
26	DPT 26 "scene info"	Format:	*B_1 U_6* *B_1* {0,1}: 0=scene is active, 1=scene is inactive *U_6* {0…63}: scene number
		Object Size:	1 byte
		Usage:	scene status, scene number
		Example:	ovg (1000) *<ovg (1000)>1 55*
217	DPT 217 "version"	Format:	*U_5 U_5 U_6* *U_5* {0…31}: magic number *U_5* {0…31}: version number *U_6* {0…63}: revision number
		Object Size:	2 byte
		Usage:	incrementation/compatibility of version numbers
		Example:	ovg (1011) *<ovg (1011)>23 30 63*

Value (code)	Datapoint Type (DPT)	Expected Values / Response Format	
232	DPT 232 "colour RGB"	Format:	U_8 U_8 U_8 U_8 {0...255}: R U_8 {0...255}: G U_8 {0...255}: B
		Object Size:	3 byte
		Usage:	colour control
		Example:	ovg (555) *<ovg (555)>200 155 98*
235	DPT 235 "tariff & active energy" (see also DPT13.010 ActiveEnergy and DPT5.006 Tariff)	Format:	V_{32} U_8 B_8 V_{32} {-2147483648...2147483647}: ActiveElectricalEnergy U_8 {0...254}: Tariff (associated to ActiveElectricalEnergy) B_8 {0,1}: Tariff/ActiveElectricalEnergy data 0 = is valid, 1 = is not valid
		Object Size:	6 byte
		Usage:	electrical energy, tariff, validity
		Example:	ovg (199) *<ovg (199)>253637 250 0 0*

DPT Examples

Table 18: Dimming Control Examples

Command	Value binary (Control Value)	Action
ovs(9) $1 1	1 001	1/1 brighter (dim to on)
ovs(9) 0 $1	0 001	1/1 darker (dim to off)
ovs(9) 1 8	1 100	⅛ brighter
ovs(9) 0 $3	0 011	¼ darker
ovs(9) 1 $0	1 000	stop dimming

$$Value = \frac{1}{2^{stepcode-1}}$$

Table 19: Temperature[2] Examples

Command
ovs(9) –12.75
ovs(9) 37

Table 20: Scaling Examples

Command	Usage	Action
ovs(9) $13	Wind direction	26.82 °
ovs(9) $A5	Relative Brightness	64.71 %
ovs(9) $DC	Counter	220

Table 21: Time Examples

Command	Action
ovs(9) 1 12 45 59	Monday, 12:45:59
ovs(9) 5 $A $1F $2F	Friday, 10:31:47

Table 22: Date Examples

Command	Action
ovs(9) 24 12 56	24.12.2056
ovs(9) $F $B $5C	15.11.1992

[2] Due to value conversion, it may happen that the value read back from SIMip not exactly equals the originally written one.

Group Object Types (objectType)

Table 23: Group Object Types according to the KNX Handbook Resource Definition

Value (code)	Size	Used Memory
0	1 bit	1 byte
1	2 bit	1 byte
2	3 bit	1 byte
3	4 bit	1 byte
4	5 bit	1 byte
5	6 bit	1 byte
6	7 bit	1 byte
7	1 octet	1 byte
8	2 octets	2 byte
9	3 octets	3 byte
10	4 octets	4 byte
11	6 octets	6 byte
12	8 octets	8 byte
13	10 octets	10 byte
14	14 octets	14 byte

Structure of Configuration Flags (comFlags)

The Configuration Flags contain the information about communication. They are identical to the „Edit Object" dialog flags in ETS.

Table 24: Structure of Configuration Flags

Bit	7	6	5	4	3	2	1	0
Flag	read response enable	transmit enable	not used	write enable	read enable	communication enable	priority	priority
	U	T	0	W	R	C	P	P

Table 25: Flag Description

Flag	**Flag Name**	**Description**
P	priority	Possible values: 3 = low 2 = urgent 1 = high 0 =system (do not use!)
C	communication enable	Enable communication of the object.
R	read enable	Object value can be read from the KNX bus.
W	write enable	Enable receiving from KNX.
T	transmit enable	Enable sending to KNX.
U	read response enable	Object value is updated by a 'Read Response'. In ETS, this flag is called "Update Flag".
0	-	-not used-

Default configuration setting is "0xD3".

Defining the direction of telegrams:

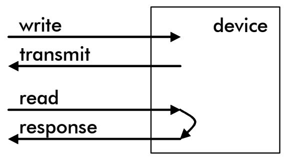


Figure 11: Telegram Direction Definition

Table 26: Default Configuration of the Flag Settings

Bit	7	6	5	4	3	2	1	0
Flag	U	T	0	W	R	C	P	P
Set	yes	yes	-	yes	no	no	yes	yes

Send Configuration (sendConfig)

Table 27: Byte containing Send Conditions Configuration

Bit	Explanation
0	send on receive 1: sends the object value when the value was received from the IP
1	send on change (not DPT1) 1: sends the object value when the value received from the IP is different than the current value
1	send on falling edge (DPT1 only) 1: sends the object value when the value changes from 1 to 0
2	send on rising edge (DPT1 only) 1: sends the object value when the value changes from 0 to 1
3	reserved (0)
4	reserved (0)
5	reserved (0)
6	select the timer usage 0: send timer 1: receive timer
7	select the time base (timer is activated when time > 0) 0: time in seconds 1: time in minutes
8	reserved (0)
9	reserved (0)
10	reserved (0)
11	reserved (0)
12	reserved (0)
13	reserved (0)
14	reserved (0)
15	reserved (0)

Receive Configuration (rcvConfig)

Table 28: Byte containing Receive Conditions Configuration

Bit	Explanation
0	set usage of indication 0: single indication 1: global indication
1	set format of single indication 1: send object value with indication
2	reserved (0)
3	reserved (0)
4	reserved (0)
5	reserved (0)
6	reserved (0)
7	reserved (0)
8	reserved (0)
9	reserved (0)
10	reserved (0)
11	indication on received value
12	indication on changed value
13	reserved (0)
14	reserved (0)
15	indication on timeout

4.3.6 Indications

Following commands are sent from the module without any requests. They are independent from the response syntax.

Receive the global update flag	
gui	
Description:	
Get the global update flag.	
Return value:	
global updateFlag	bit 0: application restart bit 3: update flag bit 4: changed flag bit 6: timeout on IP has occured bit 7: receive-timeout
Example:	
gui $01	

Receive the update data for a communication object	
oui	
Description:	
Indicate data changes on a communication object. Update flag, transmit flag and timeout flag are cleared after indication.	
Return value:	
groupObjectNo	number of the group communication object that was updated
flags	RAM flags of the communication object
(Object Value)	actual communication object value; sending this value depends on rcvConfig.
Example:	
oui $01 $10 oui $01 $10 50	

Get device state

dsi

Description:

Get actual states of SIMip.

Return value:

bit0	0: normal operation
	1: Transparent Mode
bit1	1: application loaded
bit2	1: application is running
bit7	1: device is in Programming Mode

Example:

```
dsi $06
```

4.4 Command Reference (Transparent mode)

Device transparent set		
dts *data*		
Description:		
Configure Transparent Mode.		
Parameter:		
data	bit0:	1 = enable Transparent Mode
		0 = disable Transparent Mode
	bit1:	1 = send Ack on every group telegram (recommended)
		0 = no Ack on every group telegram
	bit2:	1 = generate confirmation on IP
		0 = no confirmation on IP
	bit3:	not used
	bit4:	format of address
		0 = hex format
		1 = format depends on bit5
	bit5:	1 = address as 2-level group address
		0 = address as 3-level group address
Example:		
dts 7		

Transparent data send	
tds *(dest) data* **tds** *(dest length) data*	
Description:	
Send a 'Value Write'.	
Parameter:	
dest	ETS group address
length	0 = 1...6 bit (default)
	1 = 1 byte
	...
data	data that is sent to destination;
	allowed: single values
Example:	
tds (2/0/0) 2	

Transparent response send

tes *(dest) data*
tes *(dest length) data*

Description:

Send a 'Value Response'.

Parameter:

dest	ETS group address
length	0 = 1...6 bit (default)
	1 = 1 byte
	...
data	data that is sent to destination;
	allowed: single values

Example:

```
tes ($1000) $01
tes (2/0/1 2) $01 $02
```

Transparent send read request

trs *(dest)*

Description:

Send a 'Value Read'.

Parameter:

dest	ETS group address

Example:

```
trs ($1000)
```

Transparent read indication

tri *(source dest)*

Description:

Get a 'Value Read' indication.

Parameter:

source	Individual Address
dest	ETS group address

Example:

```
tri ($ffff $1000)
```

Transparent data indication	
tdi *(source dest length) data*	
Description:	
Get a 'Value Write' indication.	
Parameter:	
source	Individual Address
dest	ETS group address
length	0 = 1…6 bit
	1 = 1 byte
	…
data	data that is received from destination
Example:	
tdi ($ffff $17d0 $06) $01 $00 $00 $00 $00 $01	

Transparent response indication	
tei *(source dest length) data*	
Description:	
Get a 'Value Response' indication.	
Parameter:	
source	Individual Address
dest	ETS group address
length	0 = 1…6 bit
	1 = 1 byte
	…
data	data that is received from destination
Example:	
tei ($ffff $17d0 $00) $01	

Transparent read confirmation	
trc *(source dest)*	
Description:	
Get a 'Value Read' confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	Individual Address
dest	ETS group address
Example:	
trc ($ffff $1000)	

Transparent data confirmation	
tdc *(source dest length) data*	
Description:	
Get a 'Value Write' confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	Individual Address
dest	ETS group address
length	0 = 1...6 bit
	1 = 1 byte
	...
data	data that is received from destination
Example:	
tdc ($1508 $0002 $00) $01	

Transparent response confirmation	
tec *(source dest length) data*	
Description:	
Get a 'Value Response' confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	Individual Address
dest	ETS group address
length	0 = 1...6 bit
	1 = 1 byte
	...
data	data that is received from destination
Example:	
tec ($1508 $0002 $00) $01	

Transparent read negative confirmation	
trn *(source dest)*	
Description:	
Get a 'Value Read' negative confirmation – only if enabled in ETS (bit2).	
Parameter:	
source	Individual Address
dest	ETS group address
Example:	
trn ($1508 $1000)	

Transparent data negative confirmation

tdn *(source dest length) data*

Description:

Get a 'Value Write' negative confirmation – only if enabled in ETS (bit2).

Parameter:

source	Individual Address
dest	ETS group address
length	0 = 1...6 bit
	1 = 1 byte
	...
data	data that is received from destination

Example:

tdn ($1508 $1000 $00) $01

Transparent response negative confirmation

ten *(source dest length) data*

Description:

Get a 'Value Response' negative confirmation – only if enabled in ETS (bit2).

Parameter:

source	Individual Address
dest	ETS group address
length	0 = 1...6 bit
	1 = 1 byte
	...
data	data that is received from destination

Example:

ten ($1508 $1000 $00) $01

4.5 Error Codes

Table 29: Error Codes Explanation

errCode	Internal Name	Explanation	Possible Reasons
$0010	TRANSMIT_FAIL	requested transmission of an object failed	• object is already transmitting
$0101	APPL_STOP	requested command is denied, because application was stopped	• illegal memory access via KNX bus • incomplete ETS download • active ETS download
$0102	NO_OBJ	requested object number is invalid	
$0103	NUMBER_EXP	number format of the expected value is missing	
$0104	VALUE_RANGE	value is out of range	
$0105	COMMAND_END	end of command expected	
$0106	TYPE_RANGE	given object type is not allowed	• requested object type exceeds the object size
$0109	TO_MUCH_VALUES	bytestream is too long	
$010b	NUMBER_OR_COMMA	number or comma is expected	
$010d	NUMBER_OF_INDEX	too many indices are given	
$010e	LINK_WRITE	modification of group addresses failed	• deleting a non-existing address • maximum number of addresses reached
$010f	NO_OF_VALUES	number of values in bytestream is incorrect	
$0110	SYNTAX_ERROR	invalid syntax	
$0111	STRING_EXP	expected parameter is a string	
$0112	STRING_SIZE	size of string is too large	
$0113	SOE_GROUPADDR_RANGE	invalid group address	
$0114	SOE_KEY_ALREADY_EXIST	security key is already set	
$0210	TYPE_EXPECTED	command expected	• serial string does not start by command
$0215	ILLEGAL_OPERATION	unknown command	
$0216	NO_PROP	interface object property not found	

errCode	Internal Name	Explanation	Possible Reasons
$0217	IO_ELEMENT	element index is too large for the property	
$0218	RO	property is 'Read Only'	
$0219	TYPE_MISMATCH	object type and DPT are not compatible	
$0220	DPT_NOT_FOUND	given DPT is not supported	
$0221	WRONG_PARAMETER	given parameter is invalid	
$0222	ACCESS_DENIED	accessing the given command is denied	• some commands are disabled after an ETS download
$0223	NO_PARAMETER	the requested 'User Parameter' does not exist	
$0224	TO_MUCH_PARAMETER	too many 'User Parameters' are requested	
$0225	IO_ERROR	property access has failed	

5 Implemented Application Interface Object

The interface object of SIMip can be accessed from KNX side with standard property access mechanisms and from the IP side with the commands **ids** (*ioIndex propertyID elementIndex*) *data*) and **idg** (*ioIndex propertyID elementIndex*). For access from the KNX bus, tools like the "Device editor" provided with the ETS can be used.

To access the application interface object for both commands **ids** and **idg** the *ioIndex* always is 5.

Table 30: Object Index: 5

Property ID	Type Read/Write	Element Index	Usage
1	PDT_UNSIGNED_INT R		**Object Type** (0xF000)
50	PDT_GENERIC_02 RW		**Parameter of SIMip** For a detailed description of configuring SIMip see chapter 4.2.1 Accessing Interface Objects. *Example:* *Set default values to SIMip* ids (5 50) $31 $00
51	PDT_GENERIC_02 RW		**Parameter for Transparent Mode** (see chapter 3.3 in Transparent Mode) *Example:* *Read the Transparent Mode setting* idg (5 51)
52	PDT_GENERIC_01 RW		**Syntax options** (see chapter 4.1.3 Strings from SIMip) *Example:* *Enable responding of "OK if response contains no data"* ids (5 52) $02
53	PDT_GENERIC_02 R		Reserved

Property ID	Type Read/Write	Element Index	Usage
54	PDT_GENERIC_01 RW		**Control replacement of text on receiving from the IP** 0 = disable text replacement 1 = enable text replacement Original strings and replaced strings are defined in property 136. *Example:* *Enable* replacement of text ids (5 54) $01
55	PDT_GENERIC_01 RW		**Control the sending of user-defined strings on the IP** (standard indication replacement) after receiving group-oriented communication objects. 0 = disable 1 = enable
60	Array PDT_GENERIC_06 RW		**Object configuration** Every *elementIndex* configures one communication object. Element structure is as follows: • byte 0: Supported Datapoint Types (DPT) • byte 1: delay time • byte 2: upper byte of Send Configuration (sendConfig) • byte 3: lower byte of Send Configuration (sendConfig) • byte 4: upper byte of Receive Configuration (rcvConfig) • byte 5: lower byte of Receive Configuration (rcvConfig)
		1	**Configuration for communication object no. 1**
		…	…
		1024	**Configuration for communication object no. 1024)**
96	Array PDT_GENERIC_06 RO		**Manufacturer serial number** *Example:* *Get the serial number (123456789ABC)* idg (5 96) *<idg (5 96)> $00 $72 $01 $02 $03 $04*

Property ID	Type Read/Write	Element Index	Usage
128	PDT_GENERIC_01 RW		**Global event generation** (see chapter 4.3.1 Configuration) *Example:* *Read the global event generation setting* idg (5 128)
130	PDT_GENERIC_01 RW		Reserved
132	Array PDT_GENERIC_06 RW		Reserved
133	Array PDT_GENERIC_01 RW		Reserved
134	Array PDT_GENERIC_01 RW		**User-defined parameter (ETS and local) in non-volatile memory** (See also command **pdg**) *Example:* *Read byte 4 of parameter* idg (5 134 4) *Example:* *Write byte 4 with $D6* ids (5 134 4) $D6
135	Array PDT_GENERIC_01 RW		**Complex sending conditions**
136	Array PDT_GENERIC_01 RW		**String replacements** **(Original strings and replacement strings)** Replacement of complete strings and string fragments that were received from IP before command string is processed for execution. If string is shorter than 32 bytes it is terminated by '0'.
		1…32	1st original string that is to be replaced
		33…64	1st replacement string that is inserted instead of the original one
		65…96	2nd original string that is to be replaced
		97…128	2nd replacement string that is inserted instead of the original one

Property ID	Type Read/Write	Element Index	Usage
		…	…
		1985…2016	32nd original string that is be replaced
		2017…2048	32nd replacement string that is inserted instead of the original one
137	Array PDT_GENERIC_01 RW		**Replacements of indication text on reception** Each replacement can be described as a structure having 35 bytes: • bytes 1-32: string that is sent • byte 33: object number where indication is replaced • byte 34: value=0: send string regardless of which value was received; value ≠ 0: send string if received value value equals the reference value • byte 35: reference value
		1…35	Replacement structure 1
		36…70	Replacement structure 2
		…	…
		1086…1120	Replacement structure 32

Property ID	Type Read/Write	Element Index	Usage
140	PDT_GENERIC_01 RW		**Control of the communication timeout supervision of the IP - time** • value = 0: disable timeout supervision • value ≠ 0: enable timeout supervision • value is the timeout value *Example:* ids (5 140) $10 If enabled, a timeout is detected as soon the given time expires and no communication on IP side takes place. The timeout flag is set and the given object with the given value is sent to the bus. After communication resumes, it depends on configuration if this changed situation is sent to the bus or not. A global update indication that informs about the previous lost communication is sent to IP.
141	PDT_GENERIC_02 RW		**Configuration 1 of the communication timeout supervision of the SIMip behaviour** 1st byte: not used 2nd byte: • bit 7: 0 = time (value of byte 1) in seconds 1 = time (value of byte 1) in minutes • bit 6, 5, 4, 3, 2, 1 not used • bit 0: 0 = do nothing when communication resumes 1 = clear timeout flag when communication resumes and after timeout restart After communication resumes, timeout supervision starts automatically, if enabled (property 140). After a value change of the timeout flag (number in property 142) the object with the value of property 143 is sent.
142	PDT_GENERIC_02 RW		**Configuration 2 of the communication timeout supervision of the SIMip object number** 1st byte: not used 2nd byte: number of communication object that is used for sending the alarm message “communication lost”.

Property ID	Type Read/Write	Element Index	Usage
143	PDT_GENERIC_02 RW		**Configuration 3 of the communication timeout supervision of the SIMip object value** 1st byte: this value is used when timeout flag is set 2nd byte: this value is used when timeout flag is cleared
144	PDT_GENERIC_01 RW		**Device State Generation** Configuration of automatic sending by command **dsi**.
224	PDT_GENERIC_01 RW		Reserved

6 ETS

For creating ETS databases, please contact the manufacturer to get detailled information about device-internal data.

Also, a generic database is provided (TAPKO_ETS5_SIMip.knxprod) to start with development without having to create an individual database first. Details about the database parameters can be found in the corresponding Application Supprt Document (TAPKO_SD_EN_SIMip.pdf)

7 WEB Front-End

The web front-end can be used to readout SIMip´s actual device parameters (HTTP port, IP address, MAC address, …), to update its firmware and to reset the connection to the linked IP devices. For identifying a certain SIMip in a KNX network, Programming Mode can be remotely switched on and off without having to press the on-device Programming Button.

To switch back from boot mode to normal operation it is necessary to run the firmware update procedure, then press abort, or wait for the 10 min timeout.

7.1 Accessing the SIMip Web-Frontend

There are three ways to access the SIMip web front-end. It can be accessed via Windows explorer directly, or by a web browser. For access via web browser, either the IP address or the MAC address, together with the HTTP port, have to be known. How to use IP address and MAC address with the browser´s URL bar is described in the following.

For access via web browser, the correct HTTP port must be used.

Factory default HTTP port is 8080.

7.1.1 via Windows Explorer

Due to UPnP discovery, SIMip appears in the local network window. A double click on SIMip (the host name that is set in ETS), opens the web front-end in the standard web browser.

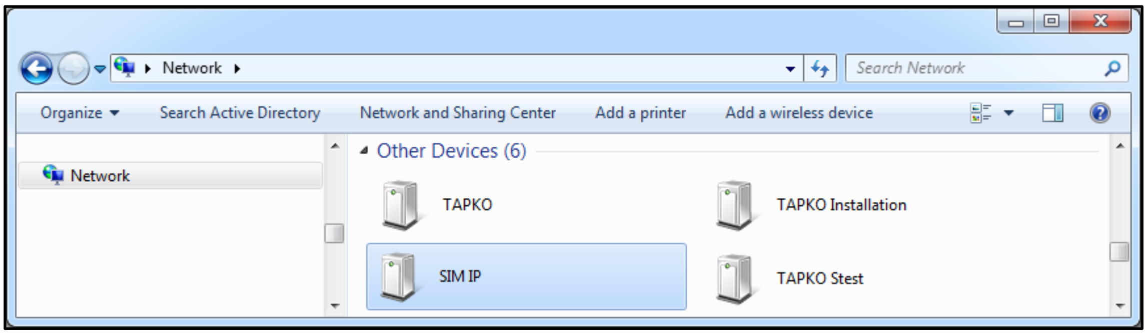
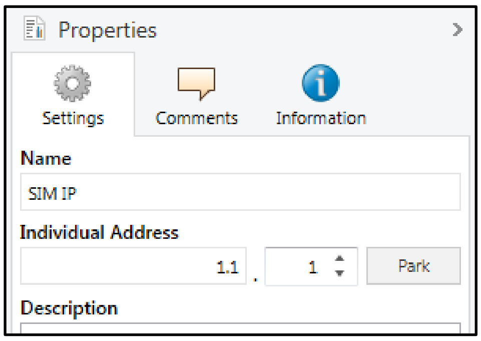


Figure 12: Windows Explorer showing SIMip

Changing the indicated name that is shown in the network can be done by setting a new Host Name at the IP configuration of SIMip in ETS. With downloading the changed data to the device, the name of the network device becomes actualized.



Name: SIM IP
Individual Address: 1.1 . 1 Park
Description

Figure 13: Product Name Setting

7.1.2 via IP Address

When IP address and HTTP port (80 or 8080) are known, this information is sufficient to access the SIMip web front-end by a web browser. According to SIMip´s pre-set IP configuration (HTTP port, IP address and DHCP, respectively) in the URL bar has to be entered (without brackets):

http://[IP address]:[HTTP port]

Example1: DHCP is not used. With the latest ETS download the IP address was set to 192.168.1.32 and HTTP port was set to 80. In the browser´s URL bar has to be entered "http://192.168.1.32:80/".

Example2: With the web-frontend DHCP was activated. The DHCP server assigned a free IP address to SIMip and the web-frontend shows IP address and HTTP port to be 192.168.1.201 and 8080. In the browser´s URL bar has to be entered "http://192.168.1.201:8080/".

7.1.3 via MAC Address

When NetBIOS is installed (by default on Windows systems), the MAC address that is printed on a label on the side of the SIMip housing can be used. The MAC address is also shown in the properties window in the Windows explorer. Due to name resolution, it is mandatory to establish communication by Host name. Hereby, activation of NetBIOS is necessary.

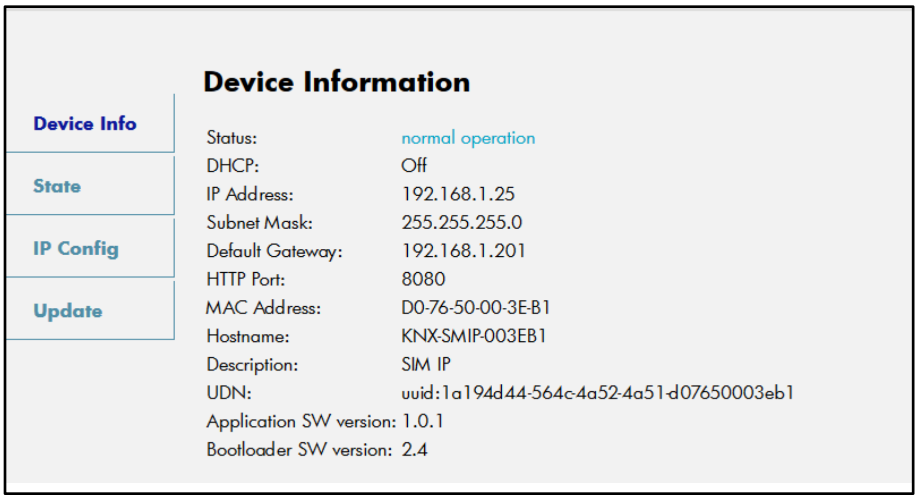
Use the MAC address AA-BB-CC-XX-YY-ZZ and the pre-set HTTP port and enter both in the browser´s URL bar, as described here (without brackets):

http://knx-smip-[XXYYZZ]:[HTTP port]

Example: On side of its housing, SIMip is labelled with MAC address D0-76-50-11-22-33 and the pre-set HTTP port is 8080. Then, in the web browser´s URL bar has to be entered "http://knx-smip-112233:8080/".

7.2 Device Information

After accessing the web front-end, the Device Info tab appears. General information about actual device state, current settings, device parameters (like addresses, names, etc.), and software versions are shown.



Device Information

Status:	normal operation
DHCP:	Off
IP Address:	192.168.1.25
Subnet Mask:	255.255.255.0
Default Gateway:	192.168.1.201
HTTP Port:	8080
MAC Address:	D0-76-50-00-3E-B1
Hostname:	KNX-SMIP-003EB1
Description:	SIM IP
UDN:	uuid:1a194d44-564c-4a52-4a51-d07650003eb1
Application SW version:	1.0.1
Bootloader SW version:	2.4

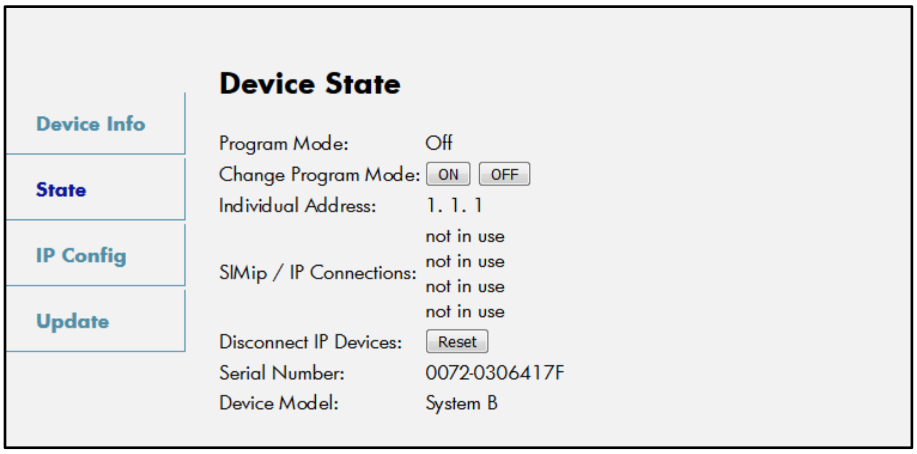
Figure 14: Device Info Tab

7.3 Device State

KNX-specific addresses are shown here. Settings can easily be checked. With a click on “On”, Programming Mode can be activated (same as a Programming Button press). Together with the Device Info tab, this function is useful to distinguish the regarded SIMip device (having a certain IP address, MAC address and serial number) from other SIMip devices used in the installation network.

The Individual Address of SIMip is indicated and the number of IP devices linked to SIMip is shown at SIMip/IP Connections. Each SIMip/IP Connection must have the same IP address like SIMip.

Up to four IP devices can be linked to SIMip. With a click on “Reset” the linked IP device(s) become disconnected by the Linkage Reset function.



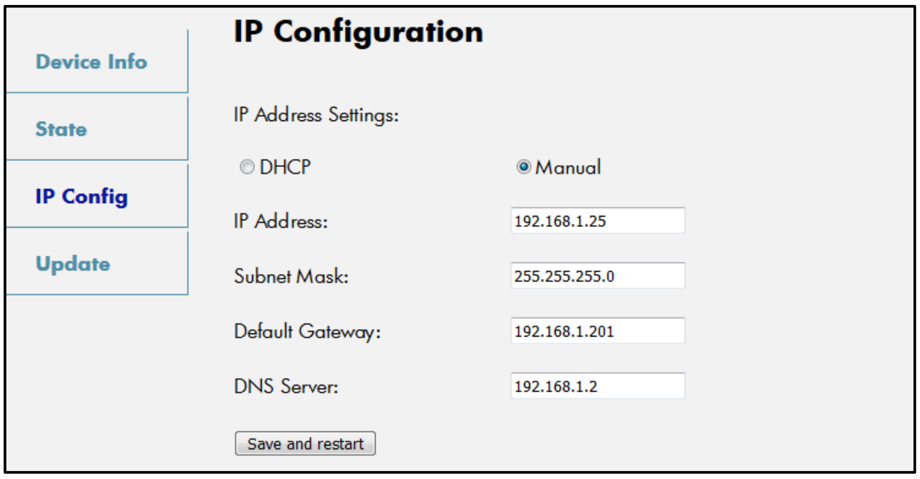
Device State

Program Mode:	Off
Change Program Mode:	ON OFF
Individual Address:	1. 1. 1
SIMip / IP Connections:	not in use not in use not in use not in use
Disconnect IP Devices:	Reset
Serial Number:	0072-0306417F
Device Model:	System B

Figure 15: State Tab

7.4 IP Configuration

When DHCP is deactivated, the IP Address Settings can be set manually. After changing one or more settings, a click on the "Save and restart" button downloads the values to SIMip.



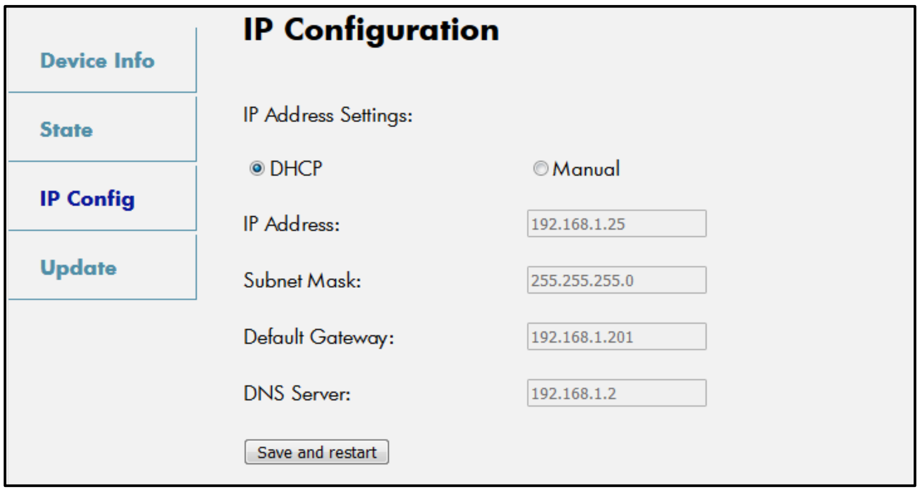
IP Configuration

IP Address Settings:

DHCP [✓ Manual]

IP Address:	192.168.1.25
Subnet Mask:	255.255.255.0
Default Gateway:	192.168.1.201
DNS Server:	192.168.1.2

Figure 16: IP Config Tab



IP Configuration

IP Address Settings:

[✓ DHCP] Manual

IP Address:	192.168.1.25
Subnet Mask:	255.255.255.0
Default Gateway:	192.168.1.201
DNS Server:	192.168.1.2

Figure 17: IP Config Tab with DHCP deactivated

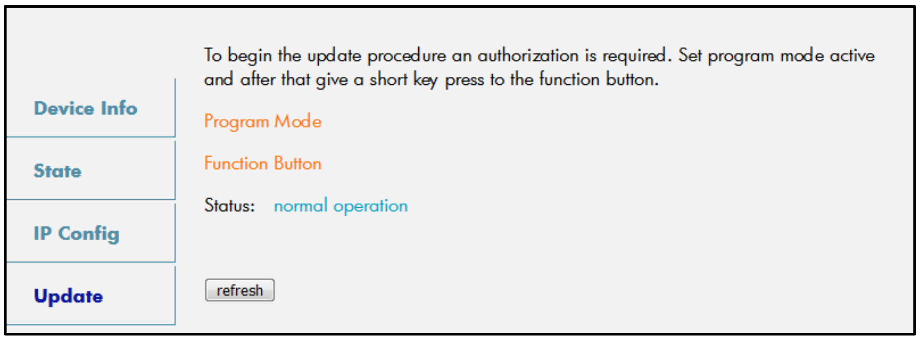
7.5 IP Firmware Update

Under the Update tab the MECip-Sec firmware can be updated via IP i.e. the Ethernet network. The complete remote update process is described in following steps. During this process, SIMip enters its boot mode. Then LEDs 1, 2, 3, 5 and 7 light as described in Table 5: LED Status Display for Firmware Update.

If boot mode is already active only the web front-end instructions from step 3 to step 5 must be followed (refresh, request update).

To exit boot mode, it is necessary to enter the Update tab of the web front-end. Then, either the firmware update has to be completed (like shown by steps 1 to 5) or the firmware update process has to be stopped by a click on the "Abort" button (see step 5, Figure 22). After that, SIMip restarts and continues with normal operation.

Step 1: Open the Update tab of the web front-end.

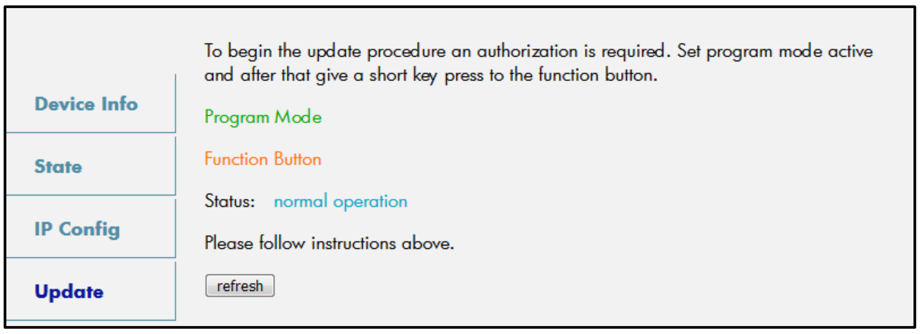


To begin the update procedure an authorization is required. Set program mode active and after that give a short key press to the function button.

Status: normal operation

Figure 18: Update Tab

Step 2: Activate Programming Mode (KNX tab or Programming Button).



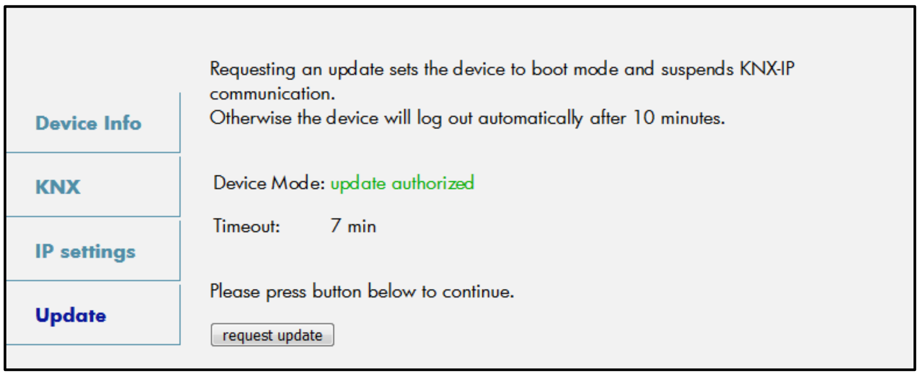
To begin the update procedure an authorization is required. Set program mode active and after that give a short key press to the function button.

Status: normal operation

Please follow instructions above.

Figure 19: Update Tab and activated Programming Mode

Step 3: **After Programming Mode activation, give a short press to the Function Button. Then click on the “refresh” button.**



Requesting an update sets the device to boot mode and suspends KNX-IP communication.
Otherwise the device will log out automatically after 10 minutes.

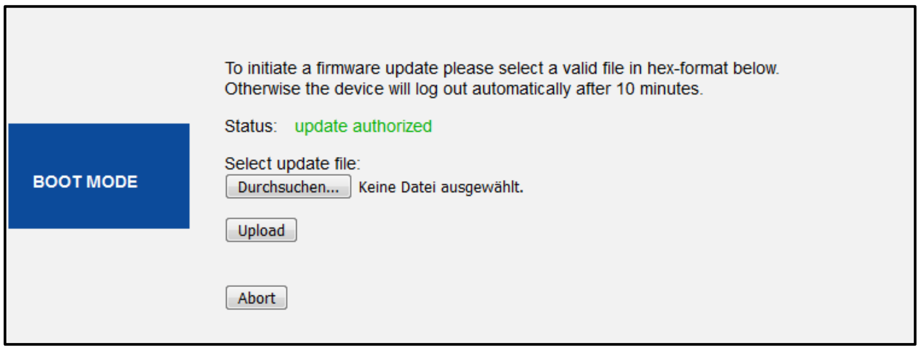
Device Mode: update authorized

Timeout: 7 min

Please press button below to continue.

Figure 20: Update Authorized

Step 4: **When the „request update“ button appears, it has to be pressed to select the update file and enter boot mode.**



To initiate a firmware update please select a valid file in hex-format below.
Otherwise the device will log out automatically after 10 minutes.

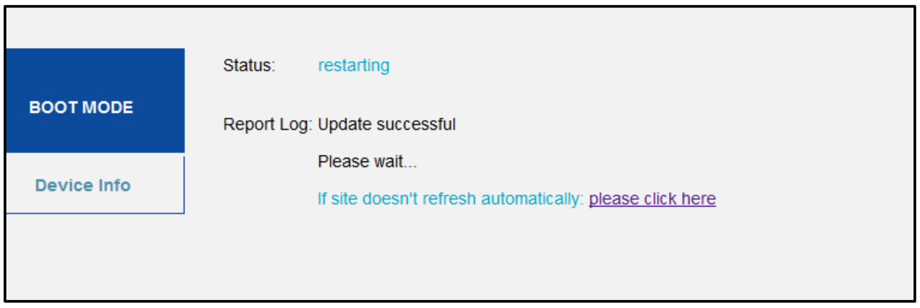
Status: update authorized

Select update file:

Keine Datei ausgewählt.

Figure 21: Request Update

Step 5: **The update file can be selected and be uploaded by a click on „Upload”. After that, the device exits boot mode and restarts. Clicking on the „Abort“ button cancels the firmware update procedure and the device exits boot mode.**



Status: restarting

Report Log: Update successful

Please wait...

If site doesn't refresh automatically: please click here

Figure 22: Select Update File

8 Glossary

ACK	An ACK is a positive IACK frame. If the sender detects an ACK, then the sender´s data has been received correctly, meaning the data has been successfully transmitted to the receiver.
Acknowledgement frames	Acknowledgment on the KNX Link Layer is also called Immediate ACK (IACK) in KNX jargon, presumably to differentiate it from other ack methods on the upper layers. Regarding sender and receiver, IACK frames are used to confirm to the sender the transmitted data was received correctly by the receiver (ACK) or not (BUSY/NACK). Also, a receiver cannot respond by sending back an IACK when a frame is damaged or incorrectly addressed (missing IACK). The IACK confirmation is a mechanism within a KNX TP line or segment. For communication across different lines or segments, the couplers connecting the lines generate the relevant IACKs.
BUSY	A BUSY is a negative IACK frame. If the sender detects a BUSY, then the receiver was not able to process the received frame. Thereafter, the sender waits for a short time period and retries to send the frame.
Communication Object	same as Group Object
Data Point Type (DPT)	Standardized data format for transmitting values via KNX. The complete list of DPTs is available at KNX Association.
Extended Frames format	An extended frame has a maximum APDU length of 254 octets and a maximum length of 263 octets (incl. checksum).
Filtering	Filtering of telegrams by couplers can be accomplished according to the topology via Individual Addresses (Physical Telegrams) and according to filter tables for group communication via Group Addresses (Group Telegrams).
Group Address	Group addresses are used to link group communication objects.

Group Communication Object	Group communication objects contain the datapoints which are transmitted via runtime communication. One or more group addresses are assigned to one group communication object. One of these assigned group addresses is the sending group address (to send the group communication object value to the bus). The remaining assigned group addresses, if available, then receive the value.
Group Object	same as Group Communication Object. A data point in KNX can be called shortly a 'Group Object' or just 'Object'.
Group Telegram	Group-oriented telegrams are named Group Telegrams. Filtering of Group Telegrams by couplers is accomplished according to their built-in filter tables for group communication.
IACK	see Acknowledgement frames
Individual Address	The Individual Address of a device defines the location of the device within the topology.
Long Telegrams	Long telegrams or long frames are telegrams having an APDU length that exceeds 15 octets. Long telegrams use the extended frame format.
NACK	A NACK is a negative IACK frame. When the sender detects a NACK, then the sender´s data has not been received correctly by at least one device meaning it has not been successfully transmitted to one or more receivers. Thereafter, the sender waits for a short time period and retries to send the frame.
Physical Address	same as Individual Address
Physical Telegram	Individually addressed telegrams are named Physical Telegrams.
Repetition of telegrams	When there is no positive IACK on the regarded TP line (e.g. NACK, BUSY, missing IACK), couplers usually repeat messages up to three times. For all MEC couplers, the number of repetitions on TP is configurable.

Security functions	For using ETS Security functions, a minimum ETS version is necessary. Security functions have been available since ETS version 5.7.2 (ETS Inside 1.4.0).
Short Telegrams	Short telegrams or short frames are telegrams having an APDU length that is not exceeding 15 octets. Short telegrams use the standard frame format.
Standard Frame format	A standard frame has a maximum APDU length of 15 octets and a maximum length of 23 octets (incl. checksum).

9 Technical

9.1 State of Delivery

Table 31: Factory Default Setting

General	
Individual Address	15.15.255

IP configuration	
IP address assignment	DHCP/AutoIP
Connection port	12004

KNX	
Group addresses	deactivated
Transparent Mode	deactivated

9.2 Datasheet

Marking/Design	SIMip	
Current consumption	< 20 mA	
Connections	IP:	RJ45 socket for 100 Mbit and 10 Mbit BaseT, IEEE 802.3 networks
	KNX TP line:	KNX TP connector (red/black), screwless, for single-core cable Ø 0.6...0.8 mm
LED Display elements	State (IP and KNX TP) Traffic (IP and KNX TP) IP Linkage Operating Mode Programming LED	
Control elements	Function Button Programming Button	
Mounting	35 mm top-hat rail (TH35) according to IEC60715	
Protection type	IP20 according to IEC60529	
Pollution degree	2 according to IEC60664-1	
Protection class	III according to IEC61140	
Overvoltage category	II according to IEC60664-1	
Approbation	KNX-certified according to ISO/IEC14543-3	
CE Marking	In compliance with directives 2014/35/EU (LVD), 2014/30/EU (EMC), 2011/65/EU (RoHS)	
Standards	EN IEC 61000-6-2, EN IEC 61000-6-3, EN IEC 62368-1, EN IEC 63000, EN IEC 63044-3, EN IEC 63044-5-1, EN IEC 63044-5-2, EN IEC 63044-5-3	
Voltage supply	KNX: 21...30V DC (SELV)	
Housing color	Plastic PA66 housing, grey (similar to RAL9018 Papyrus white)	
Housing dimensions	H = 90 mm, W = 36 mm (2 modules), D = 71 mm	
Mounting depth	64 mm	
Weight	68 g	
Operating temperature	-5...45 °C	
Storage temperature	-20...60 °C	
Ambient humidity	5...93 %, non-condensing	

9.3 Drawings

Dimensions shown here are specified in mm.

The total device width is 2 modules at 18 mm.

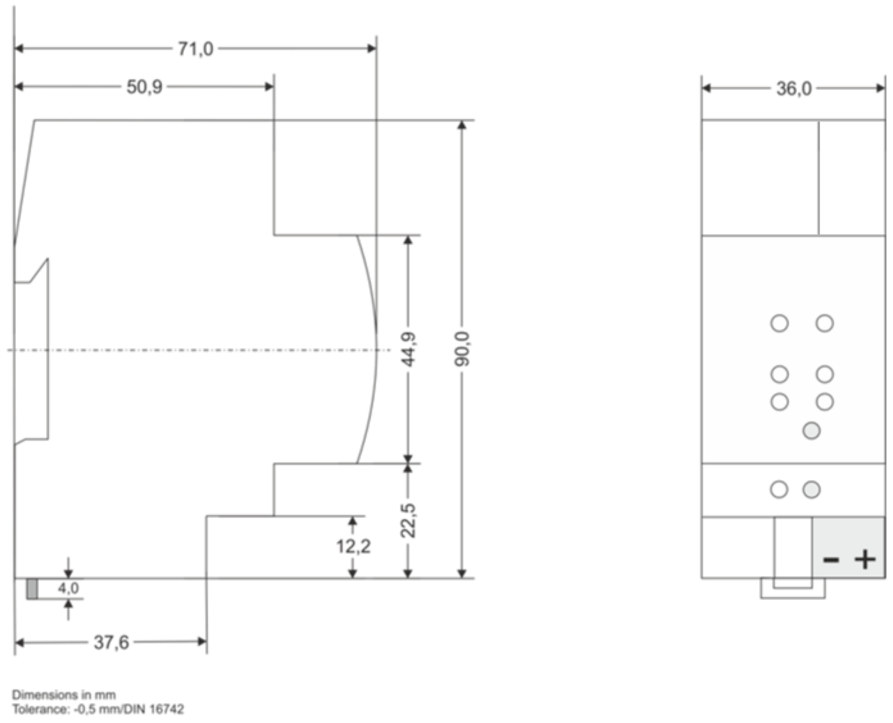


Figure 23: Dimension drawings

10 Legal Notice

lwIP is used in developing the SIMip.

lwIP is licenced under the BSD licence.

Copyright (c) 2001-2004 Swedish Institute of Computer Science. All rights reserved.

Providing that the following conditions are met redistribution and use in source and binary forms, with or without modification, are permitted:

- Redistributions of the source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR `` AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

11 FAQ

11.1 Starting with SIMip

- **What knowledge do I need before starting with SIMip?**

  It is not necessary having detailed knowledge about KNX and protocols.

  But it is very helpful knowing KNX from the installer´s side. Then, you know the basic principles of KNX and the general installation tool ETS. Such knowledge is required to develop your product.

- **Can I use SIMip without connection to KNX?**

  No. SIMip is bus-powered and needs the KNX connection to be functional.

- **Which tools are required to combine IP devices with SIMip?**

  No specific tools are required. But it is recommended to use the ETS – then you have the same possibilities as your customer or the installer: changing parameters, monitor telegrams, ...

  If you want to develop the database entry by yourself, you need the manufacturer extension of ETS.

- **What do I need to start developing with SIMip?**

  1. SIMip device
  2. An IP device that is connected to SIMip, or a PC with a terminal program running, or a microcontroller.
  3. A device which reacts on and creates KNX telegrams. This may be an arbitrary KNX device.
  4. It is recommended to use the ETS together with SIMip for monitoring bus telegrams and configuring.

- **Where can I get introduced to KNX?**

  Comprehensive information about KNX can be found in the download area of www.knx.org.

- **How can I find the correct datapoint type (DPT)?**

  A complete list of defined DPTs and their usage is provided by KNX Association. DPTs for common applications are listed in this document (see also next chapter).

11.2 Configuring SIMip

- **How can SIMip be configured?**

  SIMip can be configured via its web front-end for IP and by the ETS.

- **Is it required to configure SIMip each time after a restart?**

  No. SIMip stores the configuration data in its non-volatile memory.

- **Is it necessary to configure SIMip via IP, after I used the ETS for configuring?**

  No. All configuration settings can be downloaded by ETS.

- **How are devices configured by an installer?**

  Mainly the ETS is used by installers to configure KNX devices.

- **Where can I find a complete list of DPTs?**

  Complete information about DPTs is contained in the KNX System Specifications, Interworking, Datapoint Types (Chapter 03_07_02).

- **What are the most common used DPTs?**

  DPT 1 “1-bit” is used for switching on and off, enable/disable, …

  DPT 5 “8-bit unsigned value” is used for procentual dimming (0-100%), …

  DPT 9 “2-octet float value” is used for temperature in units of °C.

- **The firmware update finished successfully but the device doesn´t work.**

  To restart turn the power off and on again (dis-/reconnection of KNX TP line).

- **Is it Ok to connect and disconnect the Ethernet cable quickly?**

  No! Don’t do this. Before reconnection, wait for a few seconds.

- **What can be a transmission error when LED 2 Bus State KNX TP is lighting red?**

  For every telegram sent out on KNX TP, SIMip waits for an acknowledgement on the TP line. When the receiver was busy (BUSY) or received an incorrect telegram (NACK) or SIMip didn´t receive an acknowledgement (missing IACK), LED 2 is lighting red to show a transmission error exists on the line.

- **LED 2 Bus State KNX TP is continuously blinking green. Why?**

  This indicates SIMip is waiting for his firmware file download. For more information and how to switch back to normal operation please refer to chapter 7.5.

- **I disabled DHCP and assigned a correct IP configuration, but I cannot access the web front-end. Why?**

  Reset the SIMip and try again. More information about changing the IP network configuration can be found in chapter 7.4.

- **I try to access the web front-end but I'm not successful. What can I do?**

  Make sure the URL bar entry matches the correct IP address together with the right HTTP port or use the MAC address in exactly the way as explained (see chapter 7.1). Then wait for few seconds, refresh the browser and try again. Or check IP configuration via TP by ETS.

- **Is it possible to reach the web front-end when boot mode is active?**

  Yes, it is. The web front-end is accessible. When boot mode is active, the web front-end looks like illustrated in Figure 6. To run a firmware update procedure and/or exit boot mode the Update tab of the web front-end must be used. Otherwise, after 10 min the update procedure is aborted and boot mode is switched off automatically.

- **I was able to reach the web front-end. But when I try to use its tabs or buttons, my browser only shows a connection error to me. What can I do?**

  Refresh the browser window. The web front-end will show up again.

- **How can I find out the actual IP address of my SIMip?**

  In the web front-end, the Device Info tab shows the actual IP address.

  The command **dag** can be used.

  In Windows, with a right click on the network device the properties window can be opened. MAC address, IP address, HTTP port, serial number and version of firmware (application software version) can be found here.

11.3 KNX Certification / KNX Membership

- **What are the benefits of KNX certification?**

  In general, you gain a much higher acceptance on the market.

  You are allowed to use the KNX logo on your product, in related documents and for advertising and commercial material.

  You can distribute the ETS database entry as product database.

- **Why is the SIMip module/device not certified?**

  To be precise, SIMip is not a complete KNX end device.

  The communication system that is contained in SIMip is KNX-certified. It is used in various KNX products in high volume.

  Due to KNX rules, not only the communication system but also the application interworking is KNX-certified. Also, the datapoint types that are used will be checked during a certification process. The correct usage of datapoint types cannot be guaranteed by SIMip. This is part of the individual application buildup.

- **What is necessary if I want to certify a product that makes use of SIMip?**

  According to the KNX rules, following requirements have to be fulfilled:

  1. You must be a KNX member.
  2. You need a quality management system.
  3. The product must be fully consistent with the KNX specification.

- **Where can I get information about KNX membership and KNX certification?**

  General information about KNX membership and KNX certification can be found on the KNX homepage (www.knx.org) or can be retrieved from the certification department of KNX Association in Brussels.

  The TAPKO KNX Testlab offers full support for KNX certification. A document that describes all the necessary testing and TAPKO services in detail is available.

11.4 SIMip and ETS

- **How can I use SIMip together with the ETS?**

  There are various possibilities to use SIMip together with the ETS. The choice depends on the effort and certification stage you want to reach for your product.

  Stage 1:generic project database provided by TAPKO
  Stage 2:specific project database provided by yourself
  Stage 3:specific KNX-certified product database

  If you decide to use a database at an early stage, you are not bound to it. You can change it to another stage later. For a quick start in developing with SIMip, TAPKO also provides a registered database that can be easily customized to project-/device-specific needs by parameters.

- **What is the difference between product database entry and project database?**

  The product database is only related to certified KNX products. This is the usual way for manufacturers to distribute the KNX database for their products.

  Project databases are used to export/transfer projects between ETS installations. Product databases can be contained in uncertified products. This way can be used to distribute database entries for uncertified products.

- **Are there differences to be regarded by the installer for product/project database handling?**

  Yes, the product databases can be selected in the ETS product catalogue by choosing the manufacturer among further criteria.

  With project databases the user has to import the distributed project, and then copy the device of interest into his project.

- **What tool do I need to create a database entry for the ETS?**

  To create an ETS database entry, you need the ETS manufacturer extension. It is available for KNX members and can be retrieved from KNX Association.

11.5 KNX Information and Product Support

- **Who offers support for my development?**

  Of course, TAPKO offers full support for your developments, no matter at what stage.

  One of TAPKO´s core businesses is universal support for all kinds of KNX developments, problems and tasks.

- **Where can I get further information about the KNX system?**

  You can get further information from the KNX Association in Brussels (www.knx.org) and the KNX National organisations (e.g. ZVEI in Germany: www.knx.de).

SIMip

Product:	IP Interface for KNX IP/KNX Data Server	
Doctype:	Technical & Application Description	
Release Number / Release Date:	R1.2 / December 2022	
TAD is intended for: (x = 0,1,2, ... and y = a,b,c, ...)	Firmware ETS version	1.0.x ETS5 and higher
Weblink to actual ETS Database:	https://www.tapko.de/simip	
Contact:	sales@tapko.de	
Telephone:	+49 941 30747-0	

© 1999-2022 TAPKO Technologies GmbH
Im Gewerbepark A15
93059 Regensburg
Germany