zenon driver manual

KabaDPServer

v.8.20

Contents

1 Welcome to COPA-DATA help 5

2 KabaDPServer 5

3 KabaDPServer - data sheet 6

4 Driver history 7

5 Requirements 8

5.1 PLC 8

6 Configuration 8

6.1 Creating a driver 9

6.2 Settings in the driver dialog 12

6.2.1 General 13

6.2.2 KABA server connection 17

7 Creating variables 18

7.1 Creating variables in the Editor 18

7.2 Addressing 21

7.3 Driver objects and datatypes 23

7.3.1 Driver objects 23

7.3.2 Mapping of the data types 24

7.3.3 Alarms and messages 25

7.4 Creating variables by importing 35

7.4.1 XML import 36

7.4.2 DBF Import/Export 37

7.4.3 Online import 42

7.5 Communication details (Driver variables) 43

8 Driver-specific functions 49

9 Driver command function 49

10 Error analysis 54

10.1 TCP API error numbers 54

10.2 Analysis tool 62

10.3 Driver monitoring 63

10.4 Check list 64

1 Welcome to COPA-DATA help

ZENON VIDEO TUTORIALS

You can find practical examples for project configuration with zenon in our YouTube channel (https://www.copadata.com/tutorial_menu). The tutorials are grouped according to topics and give an initial insight into working with different zenon modules. All tutorials are available in English.

GENERAL HELP

If you cannot find any information you require in this help chapter or can think of anything that you would like added, please send an email to documentation@copadata.com.

PROJECT SUPPORT

You can receive support for any real project you may have from our customer service team, which you can contact via email at support@copadata.com.

LICENSES AND MODULES

If you find that you need other modules or licenses, our staff will be happy to help you. Email sales@copadata.com.

2 KabaDPServer

The driver is for displaying the status of alarms of doors; the monitoring of these is implemented with a Kaba system.

3 KabaDPServer - data sheet

General:	
Driver file name	KabaDPServer.exe
Driver name	Kaba data point server driver
PLC types	Kaba exos 9300
PLC manufacturer	Kaba

Driver supports:	
Protocol	Kaba Server XML-Frames
Addressing: Address-based	Name based
Addressing: Name-based	--
Spontaneous communication	X
Polling communication	--
Online browsing	X
Offline browsing	--
Real-time capable	--
Blockwrite	--
Modem capable	--
RDA numerical	--
RDA String	--
Hysteresis	--
extended API	--
Supports status bit **WR-SUC**	--
alternative IP address	--

Requirements:	
Hardware PC	--
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	--

Platforms:	
Operating systems	Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

4 Driver history

Date	Build number	Change
31.10.14	15380	Created driver documentation

DRIVER VERSIONING

The versioning of the drivers was changed with zenon 7.10. There is a cross-version build number as of this version. This is the number in the 4th position of the file version,
For example: **7.10.0.4228** means: The driver is for version **7.10** service pack **0**, and has the build number **4228**.

Expansions or error rectifications will be incorporated into a build in the future and are then available from the next consecutive build number.

Example

A driver extension was implemented in build **4228**. The driver that you are using is build number **8322**. Because the build number of your driver is higher than the build number of the extension, the extension is included. The version number of the driver (the first three digits of the file version) do not have any significance in relation to this. The drivers are version-agnostic

5 Requirements

This chapter contains information on the requirements that are necessary for use of this driver.

5.1 PLC

The driver communicates with a Kaba data point server. The access data (IP address, port, user and password) are issued by the server administrator.

6 Configuration

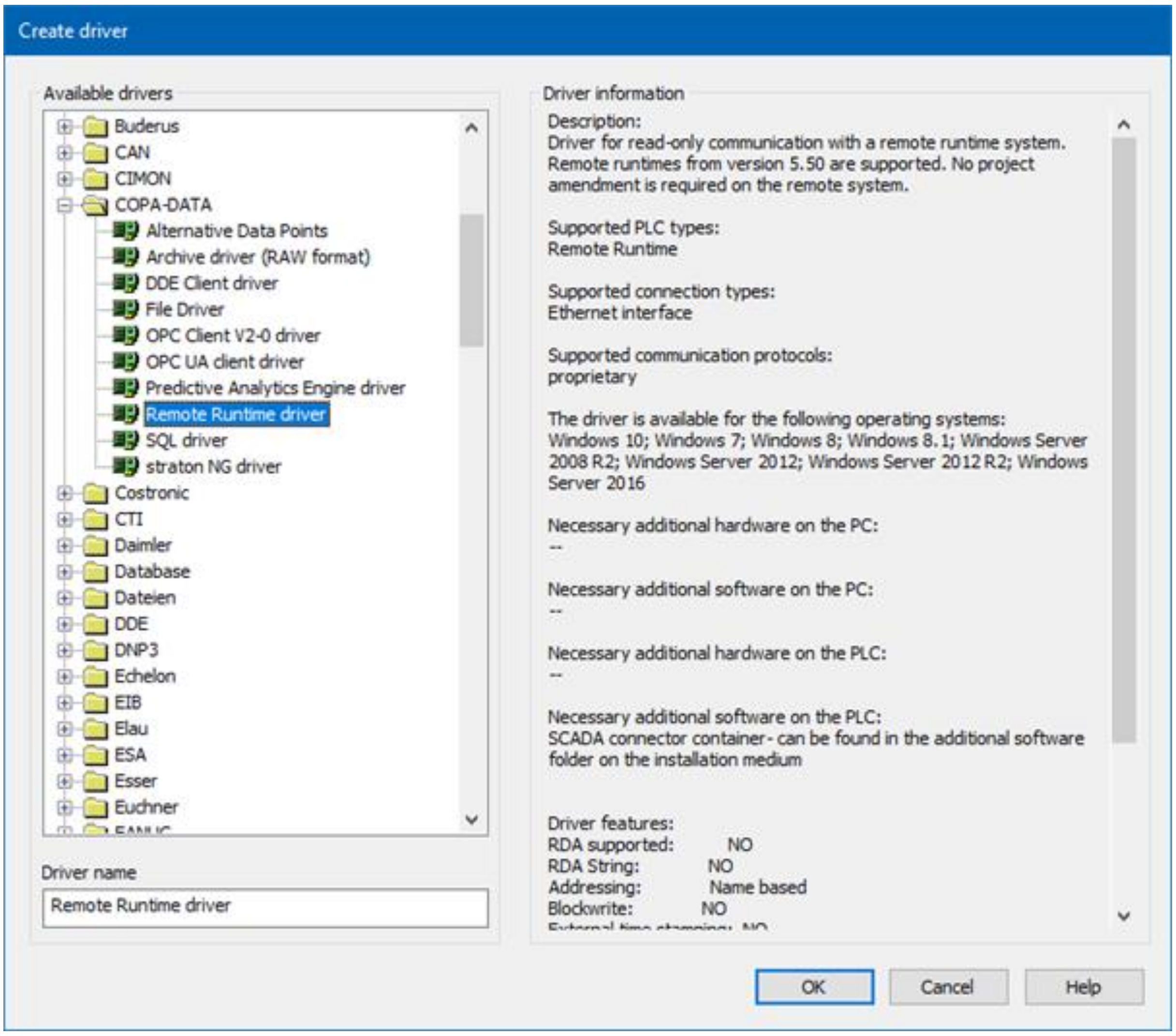
In this chapter you will learn how to use the driver in a project and which settings you can change.

Information

Find out more about further settings for zenon variables in the chapter Variables of the online manual.

6.1 Creating a driver

In the **Create driver** dialog, you create a list of the new drivers that you want to create.


Create driver

Available drivers

- Buderus
- CAN
- CIMON
- COPA-DATA
  - Alternative Data Points
  - Archive driver (RAW format)
  - DDE Client driver
  - File Driver
  - OPC Client V2-0 driver
  - OPC UA client driver
  - Predictive Analytics Engine driver
  - Remote Runtime driver
  - SQL driver
  - straton NG driver
- Costronic
- CTI
- Daimler
- Database
- Dateien
- DDE
- DNP3
- Echelon
- EIB
- Elau
- ESA
- Esser
- Euchner

Driver name

Remote Runtime driver

Driver information

Description:
Driver for read-only communication with a remote runtime system. Remote runtimes from version 5.50 are supported. No project amendment is required on the remote system.

Supported PLC types:
Remote Runtime

Supported connection types:
Ethernet interface

Supported communication protocols:
proprietary

The driver is available for the following operating systems:
Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

Necessary additional hardware on the PC:
--

Necessary additional software on the PC:
--

Necessary additional hardware on the PLC:
--

Necessary additional software on the PLC:
SCADA connector container - can be found in the additional software folder on the installation medium

Driver features:

RDA supported:	NO
RDA String:	NO
Addressing:	Name based
Blockwrite:	NO

Parameter	Description
Available drivers	List of all available drivers. The display is in a tree structure: *[+]* expands the folder structure and shows the drivers contained therein. [-] reduces the folder structure Default: *No selection*
Driver name	Unique **Identification** of the driver. Default: *empty* The input field is pre-filled with the pre-defined

Parameter	Description
	Identification after selecting a driver from the list of available drivers.
Driver information	Further information on the selected driver. Default: *empty* The information on the selected driver is shown in this area after selecting a driver.

CLOSE DIALOG

Option	Description
OK	Accepts all settings and opens the driver configuration dialog of the selected driver.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

Information

The content of this dialog is saved in the file called Treiber_[Language].xml. You can find this file in the following folder: *C:\ProgramData\COPA-DATA\zenon[version number]*.

CREATE NEW DRIVER

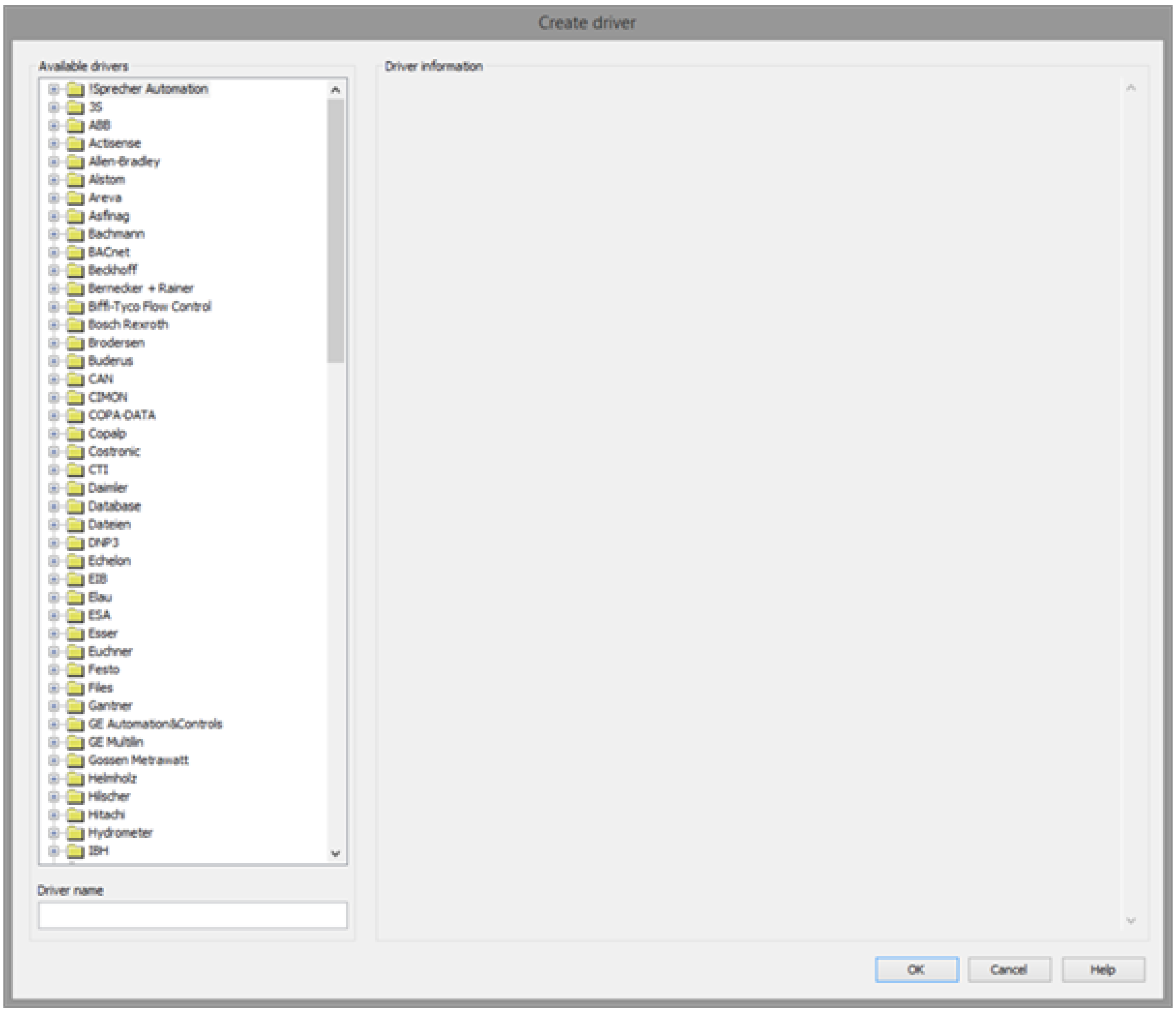
In order to create a new driver:

1. Right-click on **Driver** in the Project Manager and select **New driver** in the context menu.

   Optional: Select the **New driver** button from the toolbar of the detail view of the **Variables**.The Create driver dialog is opened.

   The **Create simple data type** dialog is opened.

2. The dialog offers a list of all available drivers.

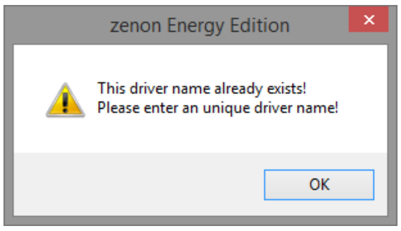


3. Select the desired driver and name it in the **Driver name** input field.
   This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.
   The following is applicable for the **Driver name**:
   - The **Driver name** must be unique.
     If a driver is used more than once in a project, a new name has to be given each time.
     This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
   - The **Driver name** is part of the file name.
     Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (_).
   - **Attention:** This name cannot be changed later on.
4. Confirm the dialog by clicking on the **OK** button.
   The configuration dialog for the selected driver is opened.

Note: The language of driver names cannot be switched. They are always shown in the language in which they have been created, regardless of the language of the Editor. This also applies to driver object types.

DRIVER NAME DIALOG ALREADY EXISTS

If there is already a driver in the project, this is shown in a dialog. The warning dialog is closed by clicking on the **OK** button. The driver can be named correctly.



This driver name already exists!
Please enter an unique driver name!

ZENON PROJECT

The following drivers are created automatically for newly-created projects:

- **Intern**
- **MathDr32**
- **SysDrv**

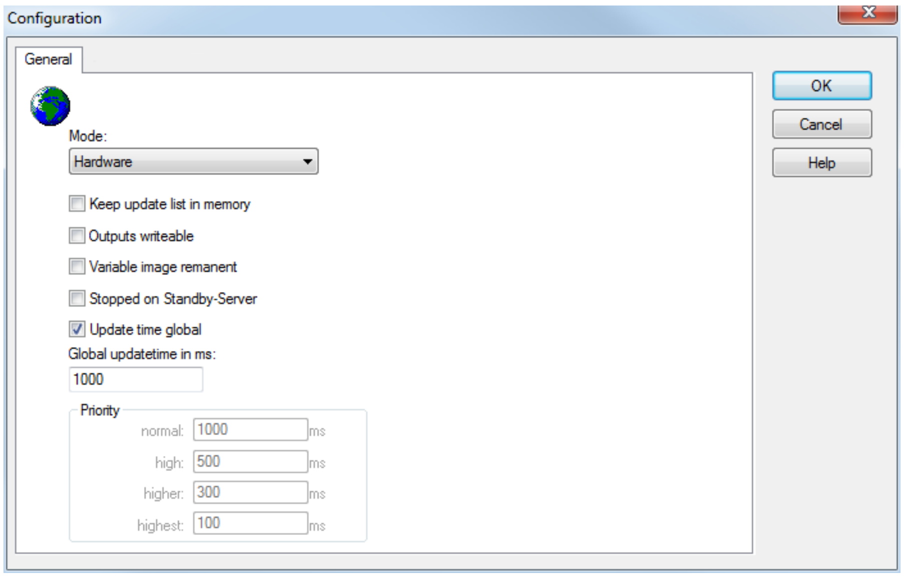
> **Information**
>
> Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

The configuration dialog is opened when a driver is created. In order to be able to open the dialog later for editing, double click on the driver in the list or click on the **Configuration** property.



Mode: Hardware

- [ ] Keep update list in memory
- [ ] Outputs writeable
- [ ] Variable image remanent
- [ ] Stopped on Standby-Server
- [x] Update time global

Global updatetime in ms: 1000

Priority

normal:	1000	ms
high:	500	ms
higher:	300	ms
highest:	100	ms

Option	Description
Mode	Allows to switch between hardware mode and simulation mode ▶ *Hardware*: A connection to the control is established. ▶ Simulation - static: No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▶ *Simulation - counting*: No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically. ▶ *Simulation - programmed*: No communication is established to the PLC. The

Option	Description
	values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation.
Keep update list in the memory	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	- *Active*: Outputs can be written. - *Inactive*: Writing of outputs is prevented. **Note**: Not available for every driver.
Variable image remanent	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in hardware mode if one of these statuses is active: - User status *M1* (*0*) to *M8* (*7*) - *REVISION(9)* - *AUS(20)* - *ERSATZWERT(27)* The variable image is always saved if: - the variable is of the **Communication details** object type - the driver runs in simulation mode. (not programmed simulation) The following states are not restored at the start of the Runtime:

Option	Description
	▸ *SELECT(8)* ▸ *WR-ACK(40)* ▸ *WR-SUC(41)* The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.
Stop on Standby Server	Setting for redundancy at drivers which allow only one communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. **Attention:** If this option is active, the gapless archiving is no longer guaranteed. ▸ *Active*: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **switched off** but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian. Default: *inactive* **Note:** Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.
Global Update time	Setting for the global update times in milliseconds: ▸ *Active*: The set **Global update time** is used for all variables in the project. The priority set at the variables is not used. ▸ *Inactive*: The set priorities are used for the individual variables. **Exceptions:** Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the **Spontaneous driver update time** section.

Option	Description
Priority	The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time. The variables are allocated separately in the settings of the variable properties. The communication of the individual variables can be graded according to importance or required topicality using the priority classes. Thus the communication load is distributed better. **Attention:** Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.

CLOSE DIALOG

Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

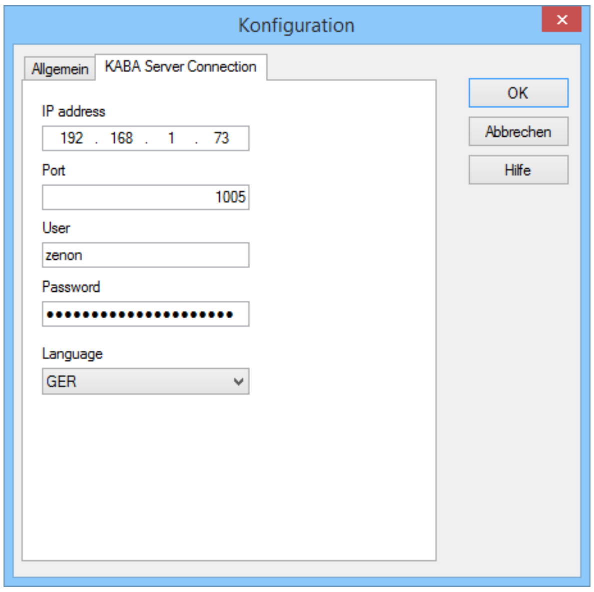
UPDATE TIME FOR SPONTANEOUS DRIVERS

With spontaneous drivers, for **Set value**, **advising** of variables and **Requests**, a read cycle is triggered immediately - regardless of the set update time. This ensures that the value is immediately available for visualization after writing. The update time is generally 100 ms.

Spontaneous drivers are **ArchDrv**, **BiffiDCM**, **BrTcp32**, **DNP3**, **Esser32**, **FipDrv32**, **FpcDrv32**, **IEC850**, **IEC870**, **IEC870_103**, **Otis**, **RTK9000**, **S7DCOS**, **SAIA_Slave**, **STRATON32** and **Trend32**.

6.2.2 KABA server connection

You configure the connection to the KABA server in this tab.



IP address
192 . 168 . 1 . 73
Port
1005
User
zenon
Password
••••••••••••••••••••
Language
GER

Parameter	Description
IP address	IP address of the server.
Port	Port that is used for communication. Default: *1005*
User	Login name.
Password	Password.
Language	Language in which the texts for string variables are displayed. Select from drop-down list: ▶ *GER*: German ▶ *FRA*: French ▶ *ENG*: English ▶ *ITA*: Italian

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

Options	Description
Help	Opens online help.

7 Creating variables

This is how you can create variables in the zenon Editor:

7.1 Creating variables in the Editor

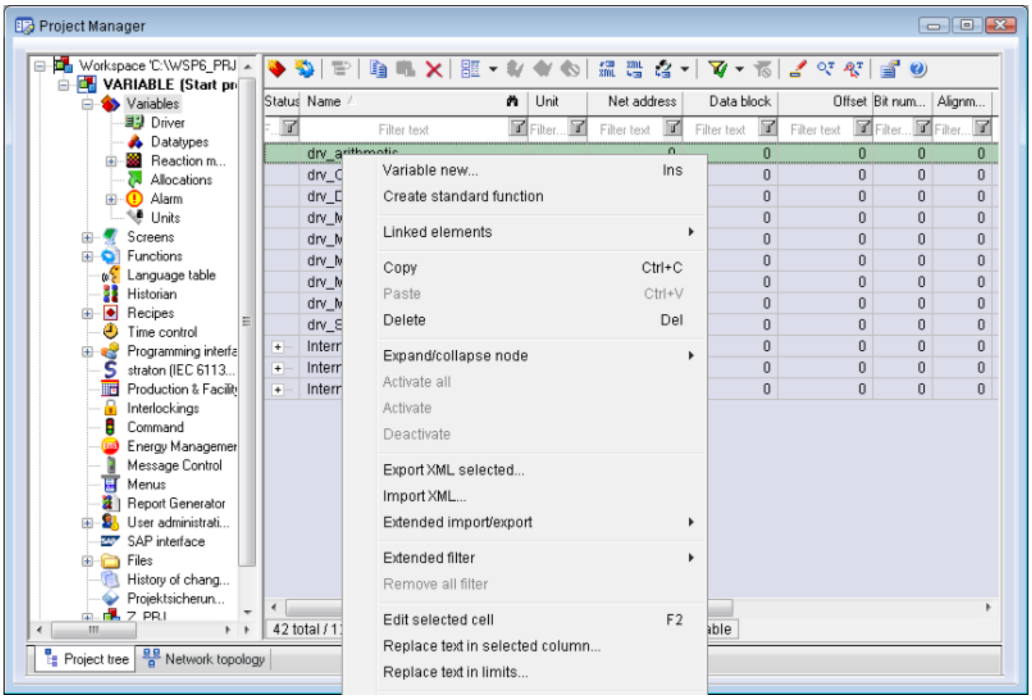
Variables can be created:

- as simple variables
- in arrays
- as structure variables

VARIABLE DIALOG

To create a new variable, regardless of which type:

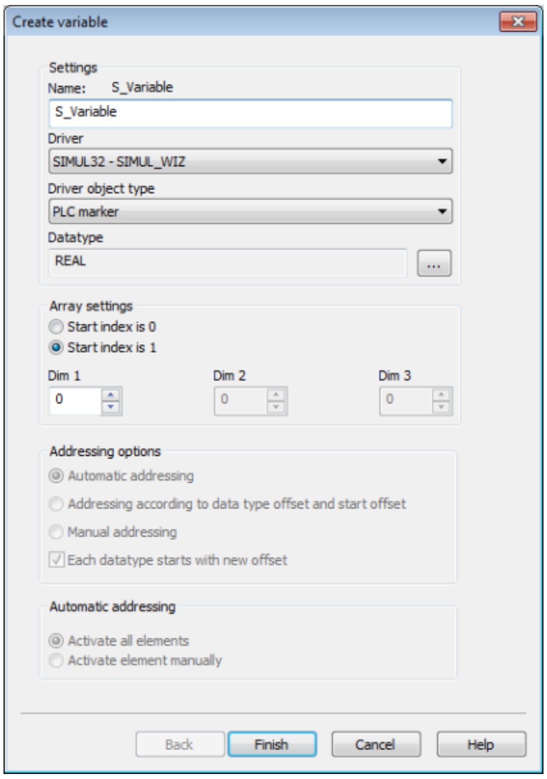
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

2. Configure the variable
3. The settings that are possible depend on the type of variables

CREATE VARIABLE DIALOG


Create variable

Settings
Name: S_Variable
S_Variable
Driver
SIMUL32 - SIMUL_WIZ
Driver object type
PLC marker
Datatype
REAL

Array settings
Start index is 0 [✓ Start index is 1]
Dim 1: 0 Dim 2: 0 Dim 3: 0

Addressing options
[✓ Automatic addressing] Addressing according to data type offset and start offset Manual addressing
[✓ Each datatype starts with new offset]

Automatic addressing
[✓ Activate all elements] Activate element manually

Property	Description
Name	Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name. Maximum length: 128 characters **Attention:** the characters **#** and **@** are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive. **Note:** Some drivers also allow addressing using the **Symbolic address** property.
Driver	Select the desired driver from the drop-down list. **Note:** If no driver has been opened in the project, the driver for internal variables (**Intern.exe**) is automatically loaded.
Driver Object Type	Select the appropriate driver object type from the drop-down list.

Property	Description
Data Type	Select the desired data type. Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the Arrays chapter.
Addressing options	Expanded settings for arrays and structure variables. You can find details in the respective section.
Automatic element activation	Expanded settings for arrays and structure variables. You can find details in the respective section.

SYMBOLIC ADDRESS

The **Symbolic address** property can be used for addressing as an alternative to the **Name** or **Identification** of the variables. Selection is made in the driver dialog; configuration is carried out in the variable property. When importing variables of supported drivers, the property is entered automatically.

Maximum length: *1024* characters.

The following drivers support the **Symbolic address**:

- 3S_V3
- AzureDrv
- BACnetNG
- IEC850
- KabaDPServer
- OPCUA32
- Phoenix32
- POZYTON
- RemoteRT
- S7TIA
- SEL
- SnmpNg32
- PA_Drv
- EUROMAP63

INHERITANCE FROM DATA TYPE

Measuring range, **Signal range** and **Set value** are always:

- derived from the datatype
- Automatically adapted if the data type is changed

Note for signal range: If a change is made to a data type that does not support the set **signal range**, the **signal range** is amended automatically. For example, for a change from **INT** to **SINT**, the **signal range** is changed to *127*. The amendment is also carried out if the **signal range** was not inherited from the data type. In this case, the **measuring range** must be adapted manually.

7.2 Addressing

Group/Property	Description
General	Property group for general settings.
Name	Freely definable name. **Attention:** For every zenon project the name must be unambiguous.
Identification	Freely definable identification. E.g. for Resources label, comments, ...
Addressing	Properties for the address of the variables.
Net address	not used for this driver
Data block	not used for this driver
Offset	not used for this driver
Alignment	not used for this driver
Bit number	not used for this driver

Group/Property	Description
Symbolic address	The **Symbolic address** property can be used for addressing as an alternative to the **Nar** **Identification** of the variables. Selection is made in the driver dialog; configuration is car variable property. When importing variables of supported drivers, the property is enterec Maximum length: *1024* characters. The real address of the variables for the driver consists of: - Prefix of the Kaba address - Attribute in square brackets Example: *I01000201[AlarmID]* Possible attributes: - *State*: Status of the door - *AlarmID*: Alarm-ID - *AlarmText*: Alarm-Text **Hint:** Import the variables from the Server into the editor, in order to avoid typing errors
String length	Length of the alarm text. Default: *256 characters*
Driver connection	Properties for driver connection.
Driver Object Type	The following object types are available: - *State of the door* - *Alarm ID of the door* - *Alarm text of the door*
Data Type	There are fixed assignments of data types to object types: - *State of the door*: *INTEGER* - *Alarm ID of the door*: *UINT* - *Alarm text of the door*: *WSTRING*
Driver connection/Priority	not used for this driver The driver does not support cyclically-poling communication in p

7.3 Driver objects and datatypes

Driver objects are areas available in the PLC, such as markers, data blocks etc. Here you can find out which driver objects are provided by the driver and which IEC data types can be assigned to the respective driver objects.

7.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Description
Alarm ID of the door	*0x0041* *(65)*	X	--	*UINT*	Variable to display the alarm ID of a door.
Alarm text of the door	*0x0042* *(66)*	X	--	*WSTRING*	Variable to display the alarm text of a door.
State of the door	*0x0040* *(64)*	X	--	*INT*	Variable to display the status of a door.
Communication details	35	X	X	*BOOL, DINT, INT, REAL, SINT, STRING, UDINT, UINT, USINT*	Variables for the static analysis of the communication; Values are transferred between driver and Runtime (not to the PLC). **Note**: The addressing and the behavior is the same for most zenon drivers. You can find detailed information on this in the Communication details (Driver variables) (on page

Driver Object Type	Channel type	Read	Write	Supported data types	Description
					43) chapter.

Key:

X: supported

--: not supported

CHANNEL TYPE

The term **Kanaltyp** is the internal numerical name of the driver object type. It is also used for the extended DBF import/export of the variables.

"**Kanaltyp**" is used for advanced CSV import/export of variables in the "**HWObjectType**" column.

7.3.2 Mapping of the data types

All variables in zenon are derived from IEC data types. The following table compares the IEC datatypes with the datatypes of the PLC.

PLC	zenon	Data type
-	BOOL	8
-	USINT	9
-	SINT	10
Alarm ID of the door	UINT	2
Status of the door	INT	1
-	UDINT	4
-	DINT	3
-	ULINT	27
-	LINT	26
-	REAL	5
-	LREAL	6
-	STRING	12

PLC	zenon	Data type
Alarm text of the door	WSTRING	21
-	DATE	18
-	TIME	17
-	DATE_AND_TIME	20
-	TOD (Time of Day)	19

DATA TYPE

The term **data type** is the internal numerical identification of the data type. It is also used for the extended DBF import/export of the variables.

7.3.3 Alarms and messages

Alarms and messages can be displayed using the driver.

Alarms are displayed using the **AlarmID**. With system messages, a distinction is made between messages about the current state of a system component (**State**) and messages about individual parameter settings of substations (**Mode**).

Alarms and messages are displayed using whole-number IDs.

7.3.3.1 Alarms

Alarms are displayed using the **AlarmID**. The **AlarmID** is a whole number that specifies whether a variable has an alarm and which alarm it is.
AlarmID:

- *0*: no alarm
- >*0*: Alarm.

The sentence types comprise the type of sentence and the alarm number (for example *A100*).

The following sentence types are used:

Definition	Type	Identification	Can be selected in the Control Panel dialog, event/alarm administration under
Alarm log book	*E*	Event	Alarm log book
Alarm handling	*A*	Alarm	Alarm handling (+alarm log book)

Definition	Type	Identification	Can be selected in the Control Panel dialog, event/alarm administration under
	C	Acknowledged alarm	
	R	Deleted alarm	

Each message can be defined as type *A* or *E*. The numbers correspond to those of the alarm message list in the alarm administration dialog.

ALARM NUMBERS

Value (text ID)	Description + possible additional value (= argument)
1	Company code incorrect.
2	User medium unknown or blocked.
3	Error in block check digit.
4	Version number incorrect.
5	User medium blocked.
6	Medium not valid.
8	No room zone authorization.
9	Week day / time of day incorrect.
10	Biometrics error.
11	PIN code not entered correctly.
12	PIN code incorrect.
13	Max. number of incorrect PIN code entries reached, user medium blocked.
14	Silent alarm.
15	Room sequence incorrect.
16	Double access block.
17	Access granted.
18	Access not granted.
19	Biometrics verification incorrect.

Value (text ID)	Description + possible additional value (= argument)
20	Max. person check of area breached.
22	Min. person check of area breached.
23	Sabotage alarm.
24	End of sabotage alarm.
25	Access authorized.
26	Special function.
27	Standard user medium.
28	Access permitted with special user medium.
29	Number of loaded people.
30	Firmware version.
31	Last parameter download number of sets.
34	Start of download
35	End of download.
36	Insufficient security level.
37	Advance warning: max. person check of area breached.
38	Advance warning: min. person check of area breached.
39	Reader configuration.
40	AMC was restarted.
43	Load database into the AMC.
50	Door open too long.
51	Unauthorized door opening.
52	Door closed again.
53	Sabotage alarm.
54	End of sabotage alarm.
60 - 69	(empty field)

Value (text ID)	Description + possible additional value (= argument)
70	Booking memory full.
71	Booking memory empty.
72	Start of setting parameters.
73	End of setting parameters.
74	Start of diagnosis.
75	End of diagnosis.
76	Content of set incorrect.
77	Command cannot currently be executed.
78	Command cannot be executed.
79	Unknown data set.
81	Operating mode of the access control center / terminal changed.
83	Bar setting incorrect.
90	Subterminal is not responding.
91	Subterminal is responding again.
92	Access control center / access manager / terminal is not responding.
93	Access control center / access manager / terminal is responding again.
94	Communication control center is not responding.
95	Communication control center is responding again.
96	Communication control center has been restarted.
97	Port is not responding.
98	Port is responding again.
99	GID/DID used twice.
100	Communication to port could not be initialized.
101	Access control center / access manager / terminal is responding.
102	Port is responding.

Value (text ID)	Description + possible additional value (= argument)
103	Connection to the port is established.
104	Communication control center is being restarted.
105	Communication control center is being loaded.
106	Connection to branch is established.
107	Connection to branch is disconnected.
108	Subterminal is responding.
109	Sabotage alarm.
110	End of sabotage alarm.
111	Contact short circuit.
112	Contact interruption.
115	Depot open too long.
116	CardLink: Validation failed.
117	CardLink: Low battery.
118	CardLink: Event buffer full.
120	Booked to storage space.
121	Booked from storage space.
122	No parking space / storage space found.
123	Parking space / storage space occupied/full.
124	Car already parked there.
125	Parking space / storage space not occupied.
126	Cannot assign the reader in the parking area.
127	Access authorized - operating mode pass: unlocked.
128	Access authorized - operating mode pass: blocked.
143	Access authorized and passenger mode activated.
144	Passenger mode deactivated.

Value (text ID)	Description + possible additional value (= argument)
145	Passenger mode blocked.
146	User medium blocked by **staff trigger** function.
166	Bar setting correct again.
170	Access manager was restarted.
171	Inconsistent data is possible with the access manager.
172	Fewer **Kaba exos OC8** extension modules than are configured in **Kabaexos** are connected.
198	Short circuit.
199	Interruption.
200	Off.
201	Off.
600	On.
601	On.
9101	Fabrication key was successfully replaced by application key.
9102	Multiple conversion of the fabrication key in one application key.
9103	CardLink: Validation stamp sequence breached.
9105	Conversion of the fabrication key to an application key with unknown medium.
9970	Switch off.
9971	Switch on.
9980	Monitor.
9981	Open for access.
9982	Lock.
9983	Open statically (not monitored).
9996	Service mode on.
9997	Service mode off.
9998	Alarm acknowledged.

Value (text ID)	Description + possible additional value (= argument)
9999	Confirm alarm acknowledgement.

7.3.3.2 Messages

With system messages, a distinction is made between:

- Status (**State**): System message about the current status of the system components.
- Mode (**Mode**): Message about individual parameter settings of substations and passes. The operating mode of a pass is only reported if it is closed.

MESSAGES

STATE

Optional.

A whole number that represents the variable status. The value *-1* means that the status could not be determined.

Concerns	Description	Value
Front Server	Unknown.	*-1*
	Is not responding.	*0*
	Normal operation.	*1*
	Download in progress.	*2*
	Starting up.	*3*
Port	Unknown.	*-1*
	Not connected.	*0*
	Connected.	*1*
	Connection is being established.	*2*
Substation	Unknown.	*-1*
	Not connected.	*0*

Concerns	Description	Valu e
	Connected.	*1*
	Connection is being established.	*2*
Subdevice door manager/reader	Unknown.	*-1*
	Not connected.	*0*
	Connected.	*1*
Pass	Unknown.	*-1*
	open.	*0*
	closed.	*1*
	Locked.	*2*
Binary outputs	Unknown.	*-1*
	Off.	*0*
	On.	*1*
Binary inputs	Unknown.	*-1*
	Off.	*0*
	On.	*1*
	Interruption.	*2*
	Short circuit.	*3*

STATE TEXT

Optional.

A character sequence that describes the status.

MODE

Optional.

A whole number that represents the variable mode. The value *-1* means that the mode could not be determined.

Concerns	Description	Valu e
Substation (access control center/manager)	Unknown	*-1*
	Central decision.	*0*
	Local decision.	*1*
Pass	Unknown.	*-1*
	Access profile checking.	*0*
	Unlocked once.	*1*
	Locked.	*2*
	Unlocked on a permanent basis.	*3*
Turnstile	Unknown.	*-1*
	Access profile checking.	*0*
	Unlocked once.	*1*
	Locked.	*2*
	Unlocked on a permanent basis.	*3*
	Unlocked once externally.	*4*
	Unlocked once internally.	*5*
Lift	Unknown.	*-1*
	Access profile checking.	*0*
	Locked.	*2*
	Unlocked on a permanent basis.	*3*
Depot	Unknown.	*-1*
	Access profile checking.	*0*
	Locked.	*2*

PRIORITY

Optional.

A whole number that specifies the alarm priority.

ALARMTEXT

Optional.

A character sequence that describes the alarm.

CHECKLIST

Optional.

A character sequence that describes what is to be done when this alarm occurs.

DATE

Optional.

A character sequence that provides the data on which the change occurred.

Syntax: **YYYYMMDD**

TIME

Optional.

A character sequence that provides the time at which the change occurred.

Syntax: **hhmmss**

DOORTYPE

Optional.

Is only transferred if **Type** has the value *DR*.

Value	Description
0	Normal pass
1	Rotation block

ONLY FOR SUBSCRIBERS

COMMAND

Optional.

A whole number that specifies the variable command number. A command can be sent to the variable by stating this argument.

Command	Value Type	Description
-1	*DR*	Normal operation.
0	*DR*	Monitor.
1	*DR*	Open for access.
2	*DR*	Lock.
3	*DR*	Statically open.
4	*DR*	Open for access externally.
5	*DR*	Open for access internally.
0	*PR*	Disconnect connection (with remote connection).
1	*PR*	Establish connection (with remote connection).
0	*BO*	Switch off.
-1	*BO*	Switch on.

Information

Inputs and outputs that are used in a *DR* door can only be actuated with a *DR* command.

This means: A door opener can only be triggered with the **1** (**open for access**) command. The status of a frame contact can only be found out using the status pass *open* or *closed*.

7.4 Creating variables by importing

Variables can also be imported by importing them. The XML and DBF import is available for every driver.

Information

You can find details on the import and export of variables in the Import-Export manual in the Variables section.

7.4.1 XML import

During XML import of variables or data types, these are first assigned to a driver and then analyzed. Before import, the user decides whether and how the respective element (variable or data type) is to be imported:

- *Import*:
  The element is imported as a new element.
- *Overwrite*:
  The element is imported and overwrites a pre-existing element.
- *Do not import*:
  The element is not imported.

Note: The actions and their durations are shown in a progress bar during import. The import of variables is described in the following documentation. Data types are imported along the same lines.

REQUIREMENTS

The following conditions are applicable during import:

- **Backward compatibility**

  At the XML import/export there is no backward compatibility. Data from older zenon versions can be taken over. The handover of data from newer to older versions is not supported.

- **Consistency**

  The XML file to be imported has to be consistent. There is no plausibility check on importing the file. If there are errors in the import file, this can lead to undesirable effects in the project.

  Particular attention must be paid to this, primarily if not all properties exist in the XML file and these are then filled with default values. E.g.: A binary variable has a limit value of *300*.

- **Structure data types**

  Structure data types must have the same number of structure elements.
  Example: A structure data type in the project has 3 structure elements. A data type with the same name in the XML file has 4 structure elements. Then none of the variables based on this data type in the file are imported into the project.

Hint

You can find further information on XML import in the **Import - Export** manual, in the **XML import** chapter.

7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

Information

Import and Export via CSV or dBase supported; no driver specific variable settings, such as formulas. Use export/import via XML for this.

IMPORT DBF FILE

To start the import:

1. right-click on the variable list.
2. In the drop-down list of **Extended export/import...** select the **Import dBase** command.
3. Follow the instructions of the import assistant.

The format of the file is described in the chapter File structure.

Information

Note:

- Driver object type and data type must be amended to the target driver in the DBF file in order for variables to be imported.
- dBase does not support structures or arrays (complex variables) at import.

EXPORT DBF FILE

To start the export:

1. right-click on the variable list.
2. In the drop-down list of **Extended export/import...** select the **Export dBase...** command .
3. Follow the instructions of the import assistant.

Attention

DBF files:

- must correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
- must not have dots (.) in the path name.
  e.g. the path *C:\users\John.Smith\test.dbf* is invalid.
  Valid: *C:\users\JohnSmith\test.dbf*
- must be stored close to the root directory in order to fulfill the limit for file name length including path: maximum 255 characters

The format of the file is described in the chapter File structure.

Information

dBase does not support structures or arrays (complex variables) at export.

FILE STRUCTURE OF THE DBASE EXPORT FILE

The dBaseIV file must have the following structure and contents for variable import and export:

Attention

dBase does not support structures or arrays (complex variables) at export.

DBF files must:

- conform with their name to the 8.3 DOS format (8 alphanumeric characters for name, 3 characters for extension, no space)
- Be stored close to the root directory (Root)

STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered

Identification	Type	Field size	Comment
			manually). The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_D	Log	1	The variable is deleted with the *1* entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager
LES_SCHR	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTTYP and DATENTYP

Identification	Type	Field size	Comment
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix
ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists.

Identification	Typ e	Field size	Comment
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
ADJWVBA	R	1	Non-linear value adaption: *0*: Non-linear value adaption is used *1*: Non-linear value adaption is not used
ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
ZWREMA	N	16	Linked counter REMA.
MAXGRAD	N	16	Gradient overflow for counter REMA.

Attention

When importing, the driver object type and data type must be amended to the target driver in the DBF file in order for variables to be imported.

LIMIT VALUE DEFINITION

Limit definition for limit values *1* to *4*,   or status *1* to *4*:

Identification	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	technical value or ID number of a linked variable for a dynamic limit value (see VARIABLEx) (if VARIABLEx is *1* and here it is *-1*, the existing variable linkage is not overwritten)
SCHWWERT1	F	16	Threshold value for limit value
HYSTERESE1	F	14	Is not used
BLINKEN1	R	1	Set blink attribute
BTB1	R	1	Logging in CEL
ALARM1	R	1	Alarm

Identification	Type	Field size	Comment
DRUCKEN1	R	1	Printer output (for CEL or Alarm)
QUITTIER1	R	1	Must be acknowledged
LOESCHE1	R	1	Must be deleted
VARIABLE1	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	R	1	Functions linking
ASK_FUNC1	R	1	Execution via Alarm Message List
FUNC_NR1	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/Event Group
A_KLASSE1	N	10	Alarm/Event Class
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	C	66	Limit value text
A_DELAY1	N	10	Time delay
INVISIBLE1	R	1	Invisible

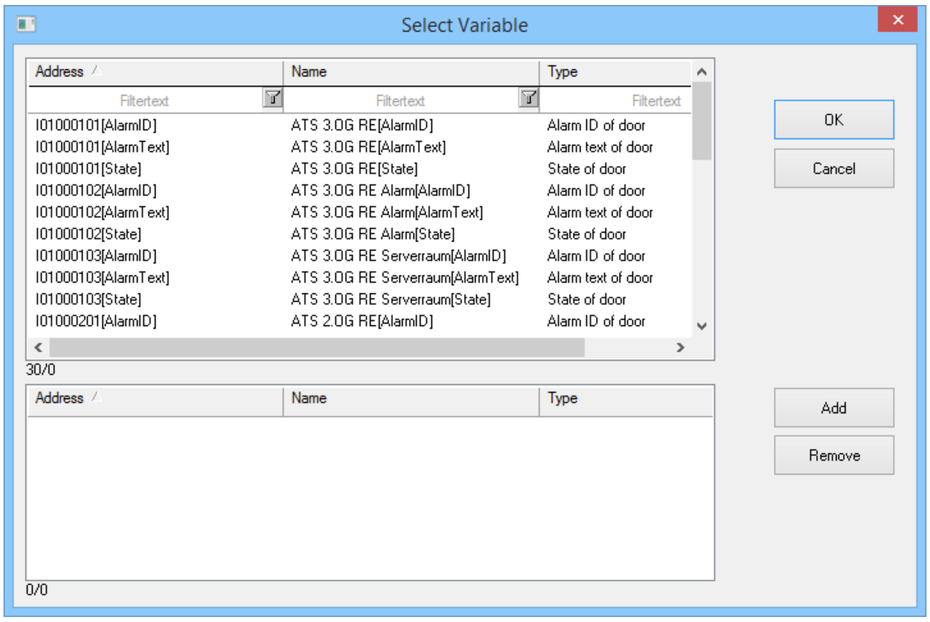
Expressions in the column "Comment" refer to the expressions used in the dialog boxes for the definition of variables. For more information, see chapter Variable definition.

7.4.3 Online import

To import variables online from the server:

1. Select the Kaba driver.
2. Select **Import variables from driver** in the toolbar or in the context menu

3. The dialog for variable selection is opened:



Address	Name	Type
I01000101[AlarmID]	ATS 3.0G RE[AlarmID]	Alarm ID of door
I01000101[AlarmText]	ATS 3.0G RE[AlarmText]	Alarm text of door
I01000101[State]	ATS 3.0G RE[State]	State of door
I01000102[AlarmID]	ATS 3.0G RE Alarm[AlarmID]	Alarm ID of door
I01000102[AlarmText]	ATS 3.0G RE Alarm[AlarmText]	Alarm text of door
I01000102[State]	ATS 3.0G RE Alarm[State]	State of door
I01000103[AlarmID]	ATS 3.0G RE Serverraum[AlarmID]	Alarm ID of door
I01000103[AlarmText]	ATS 3.0G RE Serverraum[AlarmText]	Alarm text of door
I01000103[State]	ATS 3.0G RE Serverraum[State]	State of door
I01000201[AlarmID]	ATS 2.0G RE[AlarmID]	Alarm ID of door

30/0

Address	Name	Type

0/0

4. Select the desired variables (multiple selection is possible).
5. Add selected variables via click on button **Add** to the list of the variables to be imported.
6. You can also deselect variables again by clicking on **Remove**.
7. Start the import by clicking on the **OK** button.

The selected variables are generated automatically during import in the zenon project and are assigned the KABA driver.

7.5 Communication details (Driver variables)

The driver kit implements a number of driver variables. This variables are part of the driver object type *Communication details*. These are divided into:

- Information
- Configuration
- Statistics and
- Error message

The definitions of the variables implemented in the driver kit are available in the import file **DRVVAR.DBF** and can be imported from there.
Path to file: *%ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables*

Note: Variable names must be unique in zenon. If driver variables of the driver object type *Communication details* are to be imported from **DRVVAR.DBF** again, the variables that were imported beforehand must be renamed.

Information

Not every driver supports all driver variables of the driver object type *Communication details*.

For example:

- Variables for modem information are only supported by modem-compatible drivers.
- Driver variables for the polling cycle are only available for pure polling drivers.
- Connection-related information such as **ErrorMSG** is only supported for drivers that only edit one connection at a a time.

INFORMATION

Name from import	Type	Offset	Description
MainVersion	*UINT*	0	Main version number of the driver.
SubVersion	*UINT*	1	Sub version number of the driver.
BuildVersion	*UINT*	29	Build version number of the driver.
RTMajor	*UINT*	49	zenon main version number
RTMinor	*UINT*	50	zenon sub version number
RTSp	*UINT*	51	zenon Service Pack number
RTBuild	*UINT*	52	zenon build number
LineStateIdle	*BOOL*	24.0	TRUE, if the modem connection is idle
LineStateOffering	*BOOL*	24.1	TRUE, if a call is received
LineStateAccepted	*BOOL*	24.2	The call is accepted
LineStateDialtone	*BOOL*	24.3	Dialtone recognized
LineStateDialing	*BOOL*	24.4	Dialing active
LineStateRingBack	*BOOL*	24.5	While establishing the connection
LineStateBusy	*BOOL*	24.6	Target station is busy
LineStateSpecialInfo	*BOOL*	24.7	Special status information received
LineStateConnected	*BOOL*	24.8	Connection established

Name from import	Type	Offset	Description
LineStateProceeding	*BOOL*	24.9	Dialing completed
LineStateOnHold	*BOOL*	24.10	Connection in hold
LineStateConferenced	*BOOL*	24.11	Connection in conference mode.
LineStateOnHoldPendConf	*BOOL*	24.12	Connection in hold for conference
LineStateOnHoldPendTransfe r	*BOOL*	24.13	Connection in hold for transfer
LineStateDisconnected	*BOOL*	24.14	Connection terminated.
LineStateUnknow	*BOOL*	24.15	Connection status unknown
ModemStatus	*UDINT*	24	Current modem status
TreiberStop	*BOOL*	28	Driver stopped For *driver stop*, the variable has the value *TRUE* and an **OFF** bit. After the driver has started, the variable has the value *FALSE* and no **OFF** bit.
SimulRTState	*UDINT*	60	Informs the state of Runtime for driver simulation.
ConnectionStates	*STRING*	61	Internal connection status of the driver to the PLC. Connection statuses: ▸ *0:* Connection OK ▸ *1:* Connection failure ▸ *2:* Connection simulated Formating: **<Net address>:<Connection status>;…;…;** A connection is only known after a variable has first signed in. In order for a connection to be contained in a string, a variable of this connection must be signed in once. The status of a connection is only updated if a variable of the connection is signed in. Otherwise there is no communication with

Name from import	Type	Offset	Description
			the corresponding controller.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	*BOOL*	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	*BOOL*	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	*BOOL*	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings **PhoneNumberSet** and **ModemHwAdrSet**.
PhoneNumberSet	*STRING*	38	Telephone number, that should be used
ModemHwAdrSet	*DINT*	39	Hardware address for the telephone number
GlobalUpdate	*UDINT*	3	Update time in milliseconds (ms).
BGlobalUpdaten	*BOOL*	4	TRUE, if update time is global
TreiberSimul	*BOOL*	5	TRUE, if driver in sin simulation mode
TreiberProzab	*BOOL*	6	TRUE, if the variables update list should be kept in the memory
ModemActive	*BOOL*	7	TRUE, if the modem is active for the driver
Device	*STRING*	8	Name of the serial interface or name of the modem
ComPort	*UINT*	9	Number of the serial interface.
Baudrate	*UDINT*	10	Baud rate of the serial interface.
Parity	*SINT*	11	Parity of the serial interface
ByteSize	*USINT*	14	Number of bits per character of the serial

Name from import	Type	Offset	Description
			interface Value = *0* if the driver cannot establish any serial connection.
StopBit	*USINT*	13	Number of stop bits of the serial interface.
Autoconnect	*BOOL*	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	*STRING*	17	Current telephone number
ModemHwAdr	*DINT*	21	Hardware address of current telephone number
RxIdleTime	*UINT*	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	*UDINT*	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	*UDINT*	20	Number of ringing tones before a call is accepted
ReCallIdleTime	*UINT*	53	Waiting time between calls in seconds (s).
ConnectTimeout	*UINT*	54	Time in seconds (s) to establish a connection.

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	*UDINT*	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	*UDINT*	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	*UDINT*	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	*UDINT*	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	*UDINT*	33	Number of writing errors
ReadSucceedCount	*UDINT*	35	Number of successful reading attempts

Name from import	Type	Offset	Description
MaxCycleTime	*UDINT*	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	*UDINT*	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	*UDINT*	26	Number of writing attempts
ReadErrorCount	*UDINT*	34	Number of reading errors
MaxUpdateTimeNormal	*UDINT*	56	Time since the last update of the priority group **Normal** in milliseconds (ms).
MaxUpdateTimeHigher	*UDINT*	57	Time since the last update of the priority group **Higher** in milliseconds (ms).
MaxUpdateTimeHigh	*UDINT*	58	Time since the last update of the priority group **High** in milliseconds (ms).
MaxUpdateTimeHighest	*UDINT*	59	Time since the last update of the priority group **Highest** in milliseconds (ms).
PokeFinish	*BOOL*	55	Goes to *1* for a query, if all current pokes were executed

ERROR MESSAGE

Name from import	Type	Offset	Description
ErrorTimeDW	*UDINT*	2	Time (in seconds since 1.1.1970), when the last error occurred.
ErrorTimeS	*STRING*	2	Time (in seconds since 1.1.1970), when the last error occurred.
RdErrPrimObj	*UDINT*	42	Number of the PrimObject, when the last reading error occurred.
RdErrStationsName	*STRING*	43	Name of the station, when the last reading error occurred.
RdErrBlockCount	*UINT*	44	Number of blocks to read when the last reading error occurred.
RdErrHwAdresse	*DINT*	45	Hardware address when the last reading error occurred.

Name from import	Type	Offset	Description
RdErrDatablockNo	*UDINT*	46	Block number when the last reading error occurred.
RdErrMarkerNo	*UDINT*	47	Marker number when the last reading error occurred.
RdErrSize	*UDINT*	48	Block size when the last reading error occurred.
DrvError	*USINT*	25	Error message as number
DrvErrorMsg	*STRING*	30	Error message as text
ErrorFile	*STRING*	15	Name of error log file

8 Driver-specific functions

The driver supports the following functions:

Display of the status and alarms of doors that are monitored by a Kaba server.

9 Driver command function

The zenon **Driver commands** function is to influence drivers using zenon.
You can do the following with a driver command:

- Start
- Stop
- Shift a certain driver mode
- Instigate certain actions

Note: This chapter describes standard functions that are valid for most zenon drivers.
Not all functions described here are available for every driver. For example, a driver that does not, according to the data sheet, support a modem connection also does not have any modem functions.

Attention

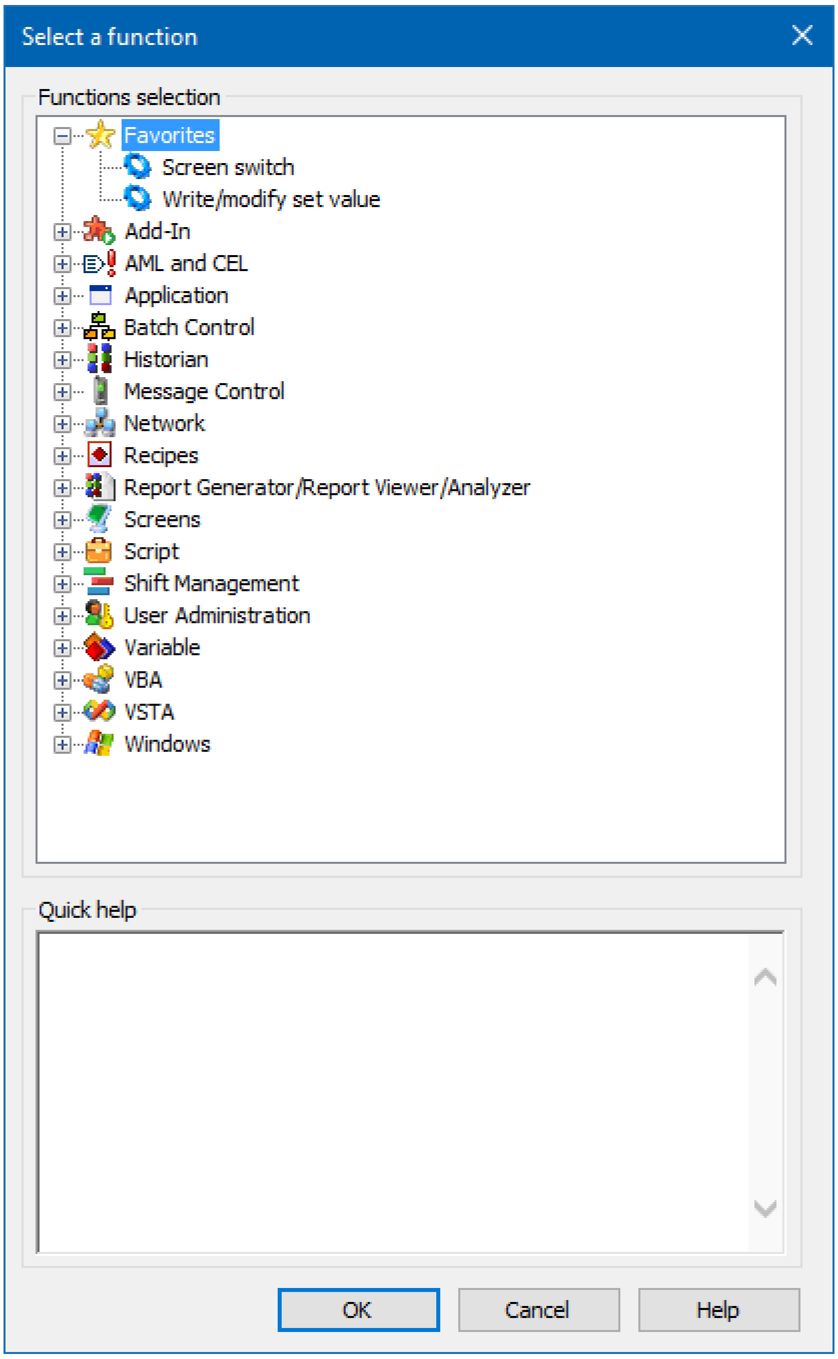
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

Configuration is carried out using the **Driver commands** function.
To configure the function:

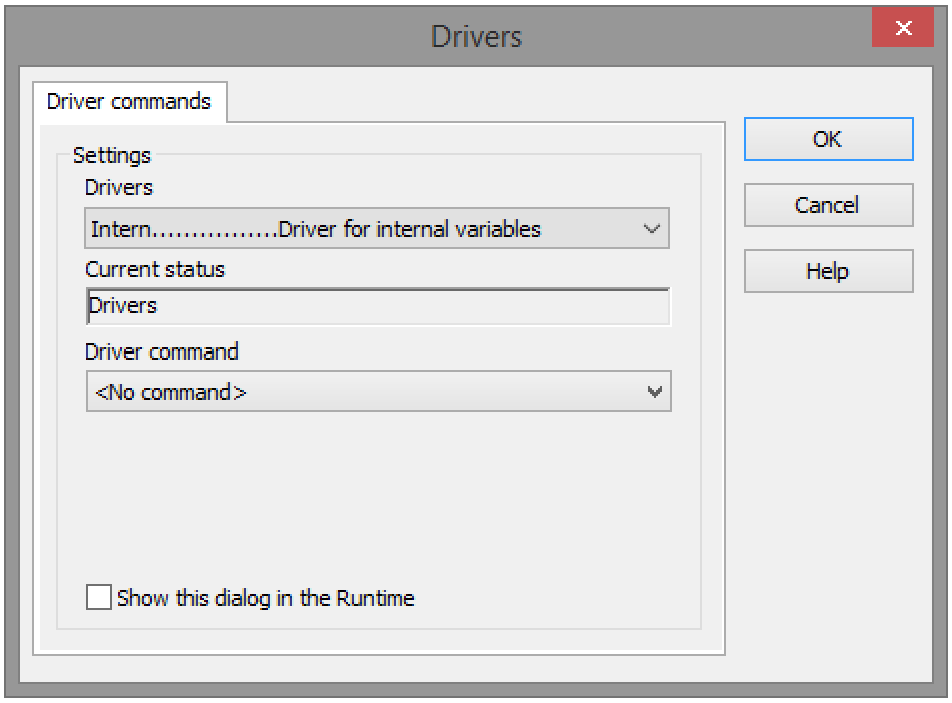
1. Create a new function in the zenon Editor.

   The dialog for selecting a function is opened



2. Navigate to the node **Variable.**
3. Select the **Driver commands** entry.

The dialog for configuration is opened



Drivers

Driver commands

Settings

Drivers: Intern...............Driver for internal variables

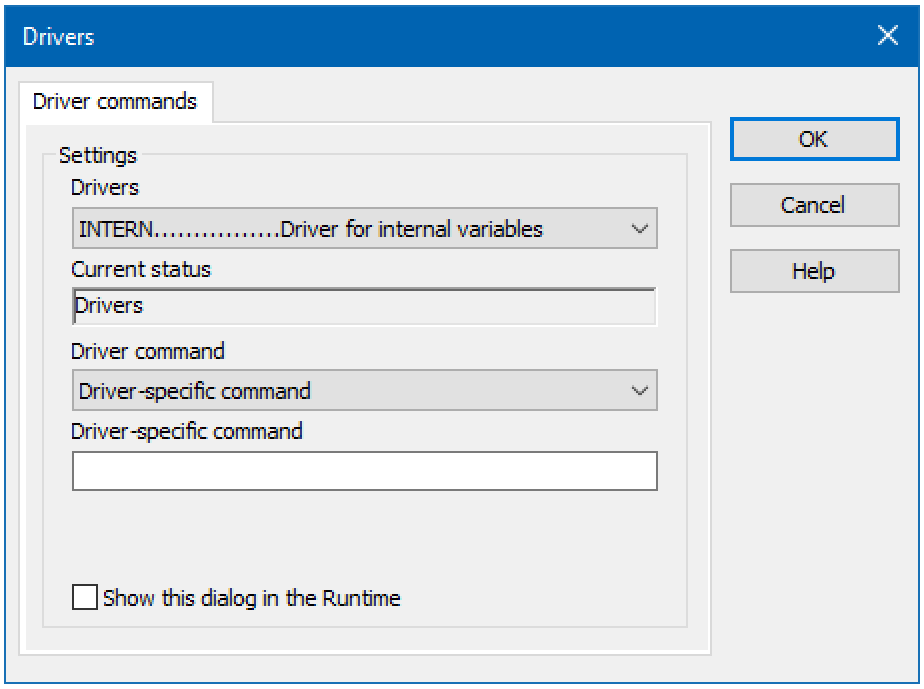
Current status: Drivers

Driver command: <No command>

☐ Show this dialog in the Runtime

4. Select the desired driver and the required command.
5. Close the dialog by clicking on **OK** and ensure that the function is executed in the Runtime. Heed the notices in the **Driver command function in the network** section.

DRIVER COMMAND DIALOG



Drivers

Driver commands

Settings

Drivers: INTERN...............Driver for internal variables

Current status: Drivers

Driver command: Driver-specific command

Driver-specific command:

☐ Show this dialog in the Runtime

Option	Description
Driver	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
Current condition	Fixed entry that is set by the system. no function in the current version.
Driver command	no function in the current version. For details on the configurable driver commands, see the **available driver commands** section.
Driver-specific command	Entry of a command specific to the selected driver. **Note:** Only available if, for the **driver command** option, the *driver-specific command* has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime: - *Active*: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. - *Inactive*: The Editor configuration is applied in the Runtime when executing the function. Default: *inactive*

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

AVAILABLE DRIVER COMMANDS

These driver commands are available - depending on the selected driver:

Driver command	Description
No command	No command is sent. A command that already exists can thus be removed from a configured function.

Driver command	Description
Start driver (online mode)	Driver is reinitialized and started. **Note:** If the driver has already been started, it must be stopped. Only then can the driver be re-initialized and started.
Stop driver (offline mode)	Driver is stopped. No new data is accepted. **Note:** If the driver is in offline mode, all variables that were created for this driver receive the status *switched off* (*OFF*; Bit *20*).
Driver in simulation mode	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
Driver in hardware mode	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
Driver-specific command	Entry of a driver-specific command. Opens input field in order to enter a command.
Driver - activate set setpoint value	Write set value to a driver is possible.
Driver - deactivate set setpoint value	Write set value to a driver is prohibited.
Establish connecton with modem	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
Disconnect from modem	Terminate connection (for modem drivers)
Driver in counting simulation mode	Driver is set into counting simulation mode. All values are initialized with *0* and incremented in the set update time by *1* each time up to the maximum value and then start at *0* again.
Driver in static simulation mode	No communication to the controller is established. All values are initialized with *0*.
Driver in programmed simulation mode	The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime.

DRIVER COMMAND FUNCTION IN THE NETWORK

If the computer on which the **Driver commands** function is executed is part of the zenon network, further actions are also carried out:

- A special network command is sent from the computer to the project server.
  It then executes the desired action on its driver.
- In addition, the Server sends the same driver command to the project standby.
  The standby also carries out the action on its driver.

This makes sure that Server and Standby are synchronized. This only works if the Server and the Standby both have a working and independent connection to the hardware.

10 Error analysis

Should there be communication problems, this chapter will assist you in finding out the error.

10.1 TCP API error numbers

ERROR CODES IN THE API

The following is a list of possible error codes returned by the WSAGetLastError call, along with their ex-tended explanations. Errors are listed in alphabetical order by error macro. Some error codes defined in Winsock2.h are not returned from any function-these are not included in this topic.

Error (Code)	Meaning	Description
WSAEACCES **(10013)**	Permission denied.	An attempt was made to access a socket in a way forbidden by its access permissions. An example is using a broadcast address for sendto without broadcast permission being set using set-sockopt(SO_BROADCAST). Another possible reason for the WSAEACCES error is that when the bind function is called (on Windows NT 4 SP4 or later), another application, service, or kernel mode driver is bound to the same address with exclusive access. Such exclusive access is a new feature of Windows NT 4 SP4 and later, and is imple-mented by using the SO_EXCLUSIVEADDRUSE option.
WSAEADDRINUSE **(10048)**	Address already in use.	Typically, only one usage of each socket address (protocol/IP address/port) is permitted. This error oc-curs if an application attempts to bind a socket to an

Error (Code)	Meaning	Description
		IP address/port that has already been used for an existing socket, or a socket that was not closed properly, or one that is still in the process of closing. For server applications that need to bind multiple sockets to the same port number, consider using set-sockopt(SO_REUSEADDR). Client applications usually need not call bind at all-connect chooses an unused port automatically. When bind is called with a wildcard address (involving ADDR_ANY), a WSAEADDRINUSE error could be delayed until the specific address is committed. This could happen with a call to another function later, including connect, listen, WSAConnect, or WSAJoinLeaf.
WSAEADDRNOTAVAIL **(10049)**	Cannot assign requested address.	The requested address is not valid in its context. This normally results from an attempt to bind to an address that is not valid for the local machine. This can also result from connect, sendto, WSAConnect, WSAJoinLeaf, or WSASendTo when the remote address or port is not valid for a remote machine (for example, address or port 0).
WSAEAFNOSUPPORT **(10047)**	Address family not supported by protocol family.	An address incompatible with the requested protocol was used. All sockets are created with an associ-ated address family (that is, AF_INET for Internet Protocols) and a generic protocol type (that is, SOCK_STREAM). This error is returned if an incorrect protocol is explicitly requested in the socket call, or if an address of the wrong family is used for a socket, for example, in sendto.
WSAEALREADY **(10037)**	Operation already in progress.	An operation was attempted on a nonblocking socket with an operation already in progress-that is, calling connect a second time on a nonblocking socket that is already connecting, or canceling an asynchronous request (WSAAsyncGetXbyY) that has already been canceled or completed.
WSAECONNABORTED **(10053)**	Software caused connection abort.	An established connection was aborted by the software in your host machine, possibly due to a data transmission time-out or protocol error.
WSAECONNREFUSE	Connection	No connection could be made because the target

Error (Code)	Meaning	Description
D **(10061)**	refused.	machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host-that is, one with no server application running.
WSAECONNRESET **(10054)**	Connection reset by peer.	An existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, or the remote host uses a hard close (see setsockopt for more information on the SO_LINGER option on the remote socket.) This error may also result if a connection was broken due to keep-alive activity detecting a failure while one or more operations are in progress. Operations that were in progress fail with WSAENETRESET. Subsequent operations fail with WSAECONNRESET.
WSAEDESTADDRREQ **(10039)**	Destination address required.	A required address was omitted from an operation on a socket. For example, this error is returned if sendto is called with the remote address of ADDR_ANY.
WSAEFAULT **(10014)**	Bad address.	The system detected an invalid pointer address in attempting to use a pointer argument of a call. This error occurs if an application passes an invalid pointer value, or if the length of the buffer is too small. For instance, if the length of an argument, which is a SOCKADDR structure, is smaller than the sizeof(SOCKADDR).
WSAEHOSTDOWN **(10064)**	Host is down.	A socket operation failed because the destination host is down. A socket operation encountered a dead host. Networking activity on the local host has not been initiated. These conditions are more likely to be indicated by the error WSAETIMEDOUT.
WSAEHOSTUNREAC H **(10065)**	No route to host.	A socket operation was attempted to an unreachable host. See WSAENETUNREACH.
WSAEINPROGRESS **(10036)**	Operation now in progress.	A blocking operation is currently executing. Windows Sockets only allows a single blocking operation-per-task or thread-to be outstanding, and if any other function call is made (whether or not it references that or any other socket) the function fails with the

Error (Code)	Meaning	Description
		WSAEINPROGRESS error.
WSAEINTR **(10004)**	Interrupted function call.	A blocking operation was interrupted by a call to WSACancelBlockingCall.
WSAEINVAL **(10022)**	Invalid argument.	Some invalid argument was supplied (for example, specifying an invalid level to the setsockopt function). In some instances, it also refers to the current state of the socket-for instance, calling accept on a socket that is not listening.
WSAEISCONN **(10056)**	Socket is already connected.	A connect request was made on an already-connected socket. Some implementations also return this error if sendto is called on a connected SOCK_DGRAM socket (for SOCK_STREAM sockets, the to pa-rameter in sendto is ignored) although other implementations treat this as a legal occurrence.
WSAEMFILE **(10024)**	Too many open files.	Too many open sockets. Each implementation may have a maximum number of socket handles avail-able, either globally, per process, or per thread.
WSAEMSGSIZE **(10040)**	Message too long.	A message sent on a datagram socket was larger than the internal message buffer or some other network limit, or the buffer used to receive a datagram was smaller than the datagram itself.
WSAENETDOWN **(10050)**	Network is down.	A socket operation encountered a dead network. This could indicate a serious failure of the network sys-tem (that is, the protocol stack that the Windows Sockets DLL runs over), the network interface, or the local network itself.
WSAENETRESET **(10052)**	Network dropped connection on reset.	The connection has been broken due to keep-alive activity detecting a failure while the operation was in progress. It can also be returned by setsockopt if an attempt is made to set SO_KEEPALIVE on a con-nection that has already failed.
WSAENETUNREACH **(10051)**	Network is unreachable.	A socket operation was attempted to an unreachable network. This usually means the local software knows no route to reach the remote host.
WSAENOBUFS **(10055)**	No buffer space	An operation on a socket could not be performed because the system lacked sufficient buffer space or

Error (Code)	Meaning	Description
	available.	because a queue was full.
WSAENOPROTOOPT **(10042)**	Bad protocol option.	An unknown, invalid or unsupported option or level was specified in a getsockopt or setsockopt call.
WSAENOTCONN **(10057)**	Socket is not connected.	A request to send or receive data was disallowed because the socket is not connected and (when send-ing on a datagram socket using sendto) no address was supplied. Any other type of operation might also return this error-for example, setsockopt setting SO_KEEPALIVE if the connection has been reset.
WSAENOTSOCK **(10038)**	Socket operation on nonsocket.	An operation was attempted on something that is not a socket. Either the socket handle parameter did not reference a valid socket, or for select, a member of an fd_set was not valid.
WSAEOPNOTSUPP **(10045)**	Operation not supported.	The attempted operation is not supported for the type of object referenced. Usually this occurs when a socket descriptor to a socket that cannot support this operation is trying to accept a connection on a datagram socket.
WSAEPFNOSUPPORT **(10046)**	Protocol family not supported.	The protocol family has not been configured into the system or no implementation for it exists. This mes-sage has a slightly different meaning from WSAEAFNOSUPPORT. However, it is interchangeable in most cases, and all Windows Sockets functions that return one of these messages also specify WSAEAFNOSUPPORT.
WSAEPROCLIM **(10067)**	Too many processes.	A Windows Sockets implementation may have a limit on the number of applications that can use it simul-taneously. WSAStartup may fail with this error if the limit has been reached.
WSAEPROTONOSUP PORT **(10043)**	Protocol not supported.	The requested protocol has not been configured into the system, or no implementation for it exists. For example, a socket call requests a SOCK_DGRAM socket, but specifies a stream protocol.
WSAEPROTOTYPE **(10041)**	Protocol wrong type for socket.	A protocol was specified in the socket function call that does not support the semantics of the socket type requested. For example, the ARPA Internet UDP

Error (Code)	Meaning	Description
		protocol cannot be specified with a socket type of SOCK_STREAM.
WSAESHUTDOWN **(10058)**	Cannot send after socket shutdown.	A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous shutdown call. By calling shutdown a partial close of a socket is requested, which is a signal that sending or receiving, or both have been discontinued.
WSAESOCKTNOSUPPORT **(10044)**	Socket type not supported.	The support for the specified socket type does not exist in this address family. For example, the optional type SOCK_RAW might be selected in a socket call, and the implementation does not support SOCK_RAW sockets at all.
WSAETIMEDOUT **(10060)**	Connection timed out.	A connection attempt failed because the connected party did not properly respond after a period of time, or the established connection failed because the connected host has failed to respond.
WSATYPE_NOT_FOUND **(10109)**	Class type not found.	The specified class was not found.
WSAEWOULDBLOCK **(10035)**	Resource temporarily unavailable.	This error is returned from operations on nonblocking sockets that cannot be completed immediately, for example recv when no data is queued to be read from the socket. It is a nonfatal error, and the operation should be retried later. It is normal for WSAEWOULDBLOCK to be reported as the result from calling connect on a nonblocking SOCK_STREAM socket, since some time must elapse for the connection to be established.
WSAHOST_NOT_FOUND **(11001)**	Host not found.	No such host is known. The name is not an official host name or alias, or it cannot be found in the data-base(s) being queried. This error may also be returned for protocol and service queries, and means that the specified name could not be found in the relevant database.
WSA_INVALID_HANDLE	Specified event object handle is invalid.	An application attempts to use an event object, but the specified handle is not valid.

Error (Code)	Meaning	Description
(OS dependent)		
WSA_INVALID_PARAMETER **(OS dependent)**	One or more parameters are invalid.	An application used a Windows Sockets function which directly maps to a Win32 function. The Win32 function is indicating a problem with one or more parameters.
WSAINVALIDPROCTABLE **(OS dependent)**	Invalid procedure table from service provider.	A service provider returned a bogus procedure table to Ws2_32.dll. (Usually caused by one or more of the function pointers being null.)
WSAINVALIDPROVIDER **(OS dependent)**	Invalid service provider version number.	A service provider returned a version number other than 2.0.
WSA_IO_INCOMPLETE **(OS dependent)**	Overlapped I/O event object not in signaled state.	The application has tried to determine the status of an overlapped operation which is not yet completed. Applications that use WSAGetOverlappedResult (with the fWait flag set to FALSE) in a polling mode to determine when an overlapped operation has completed, get this error code until the operation is com-plete.
WSA_IO_PENDING **(OS dependent)**	Overlapped operations will complete later.	The application has initiated an overlapped operation that cannot be completed immediately. A comple-tion indication will be given later when the operation has been completed.
WSA_NOT_ENOUGH_MEMORY **(OS dependent)**	Insufficient memory available.	An application used a Windows Sockets function that directly maps to a Win32 function. The Win32 func-tion is indicating a lack of required memory resources.
WSANOTINITIALISED **(10093)**	Successful WSAStartup not yet performed.	Either the application has not called WSAStartup or WSAStartup failed. The application may be access-ing a socket that the current active task does not own (that is, trying to share a socket between tasks), or WSACleanup has been called too many times.
WSANO_DATA **(11004)**	Valid name, no data record of requested type.	The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for. The usual example for this is a host name-to-address translation attempt (using gethostbyname or WSAAsyncGetHostByName) which

Error (Code)	Meaning	Description
		uses the DNS (Domain Name Server). An MX record is returned but no A record-indicating the host itself exists, but is not directly reachable.
WSANO_RECOVERY **(11003)**	This is a nonrecoverabl e error.	This indicates some sort of nonrecoverable error occurred during a database lookup. This may be be-cause the database files (for example, BSD-compatible HOSTS, SERVICES, or PROTOCOLS files) could not be found, or a DNS request was returned by the server with a severe error.
WSAPROVIDERFAILE DINIT **(OS dependent)**	Unable to initialize a service provider.	Either a service provider's DLL could not be loaded (LoadLibrary failed) or the provider's WSPStartup/NSPStartup function failed.
WSASYSCALLFAILUR E **(OS dependent)**	System call failure.	Returned when a system call that should never fail does. For example, if a call to WaitForMultipleObjects fails or one of the registry functions fails trying to manipulate the protocol/name space catalogs.
WSASYSNOTREADY **(10091)**	Network subsystem is unavailable.	This error is returned by WSAStartup if the Windows Sockets implementation cannot function at this time because the underlying system it uses to provide network services is currently unavailable. Users should check: That the appropriate Windows Sockets DLL file is in the current path. That they are not trying to use more than one Windows Sockets implementation simultaneously. If there is more than one Winsock DLL on your system, be sure the first one in the path is appropriate for the network subsystem currently loaded. The Windows Sockets implementation documentation to be sure all necessary components are currently installed and configured correctly.
WSATRY_AGAIN **(11002)**	Nonauthoritati ve host not found.	This is usually a temporary error during host name resolution and means that the local server did not receive a response from an authoritative server. A retry at some time later may be successful.

Error (Code)	Meaning	Description
WSAVERNOTSUPPORTED **(10092)**	Winsock.dll version out of range.	The current Windows Sockets implementation does not support the Windows Sockets specification ver-sion requested by the application. Check that no old Windows Sockets DLL files are being accessed.
WSAEDISCON **(10101)**	Graceful shutdown in progress.	Returned by WSARecv and WSARecvFrom to indicate that the remote party has initiated a graceful shut-down sequence.
WSA_OPERATION_ABORTED **(OS dependent)**	Overlapped operation aborted.	An overlapped operation was canceled due to the closure of the socket, or the execution of the SIO_FLUSH command in WSAIoctl.

10.2 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer program that was also installed with zenon. You can find it under **Start/All programs/zenon/Tools 8.20 -> Diagviewer.**

zenon driver log all errors in the LOG files.LOG files are text files with a special structure. The default folder for the LOG files is subfolder **LOG** in the folder **ProgramData**. For example:

%ProgramData%\COPA-DATA\LOG.

Attention: With the default settings, a driver only logs error information. With the Diagnosis Viewer you can enhance the diagnosis level for most of the drivers to "Debug" and "Deep Debug". With this the driver also logs all other important tasks and events.

In the Diagnosis Viewer you can also:

- Follow newly-created entries in real time
- customize the logging settings
- change the folder in which the LOG files are saved

Note:

1. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
2. The Diagnosis Viewer does not display all columns of a LOG file per default. To display more columns activate property **Add all columns with entry** in the context menu of the column header.

3. If you only use **Error-Logging**, the problem description is in the column **Error text**. For other diagnosis level the description is in the column **General text**.
4. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** or **Error code** or **Driver error parameter** (**1** and **2**). Hints on the meaning of error codes can be found in the driver documentation and the protocol/PLC description.
5. At the end of your test set back the diagnosis level from **Debug** or **Deep Debug**. At **Debug** and **Deep Debug** there are a great deal of data for logging which are saved to the hard drive and which can influence your system performance. They are still logged even after you close the Diagnosis Viewer.

> **Attention**
>
> In Windows CE errors are not logged per default due to performance reasons.

You can find further information on the Diagnosis Viewer in the Diagnose Viewer manual.

10.3 Driver monitoring

Runtime monitors the availability of the driver by means of a watchdog. If a driver is no longer available, the *INVALID* status bit is also set for all checked-in variables.

Possible causes for a triggering of the watchdog:

- The driver process is no longer running.

  Check whether the driver EXE file is still running in the Task Manager.
- Operating system is busy with processes that have a higher priority.

  Check the configuration of your system to see whether there is sufficient memory and CPU power. In this case, the driver only resets the *INVALID* status bit if there is a value change on the connected party. Static values retain the *INVALID* status bit until the next time the Runtime or the driver is started.

CONFIGURATION OF WATCHDOG

For the monitoring of communication in the Runtime, the connection to the driver is checked in a fixed, prescribed time period of 60 seconds. This process is repeated several times. If, within 5 attempts (= within 5 minutes), no valid connection to the driver is detected, the *INVALID* bit is set for the checked-in (*advised*) variables. In addition, the *INVALID* bit is also set when new variables are advised. The *INVALID* bit will no longer be reset.

Corresponding LOG entries are created for this.

LOG ENTRY

An error message is logged in the LOG when the watchdog is triggered:

Parameter	Description
Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.	No communication with driver within the given time. • *<time>*: Time (in milliseconds) • *<drvDesc>*: Driver name • *<drvExe>*: Driver EXE name • *<drvId>*: Driver ID in the zenon project
Communication with %s timed out. Invalid-Bit will be set.	Communication to the *%s* driver could not be established after 5 attempts within 60 seconds. The *INVALID* bit is set for the variable.
Communication with %s timed out. Timeout happened %d times	Communication to the *%s* driver could not be established after *%d* times within 60 seconds.

10.4 Check list

Questions and hints for fault isolation:

GENERAL TROUBLESHOOTING

- Analysis with the help of the Diagnosis Viewer **(on page 62) (the important errors are logged).**
- Can the Kaba server be reached using the *Ping* command?

  Ping: **Open command line -> ping <IP address > (e.g.: ping 192.168.0.100) -> Press the Enter key**.

  Do you receive an answer with a time or a timeout?
- Can the Kaba server be reached on the corresponding port via *Telnet*?

  Telnet: **Command line: enter: telent <IP address port number> (for example for Modbus: telnet 192.168.0.100 502) -> Press the Enter key**.

  If the monitor display turns black, a connection could be established.
- Does the access data correspond? (e.g. user and password)

  has the symbolic address of the variable been provided correctly?

SOME VARIABLES REPORT INVALID.

- Can the KABA server be contacted?

VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY

Driver is not working. Check the:

- Installation of zenon
- the driver installation
- Incorrect symbolic address of the "empty" variable

VARIABLES ARE DISPLAYED WITH A BLUE DOT

The communication in the network is faulty:

- With a network project:
  Is the network project also running on the server?
- With a stand-alone project or a network project which is also running on the server:
  Deactivate the property **Read from Standby Server only** in node **Driver connection**/**Addressing**.

VALUES ARE DISPLAYED INCORRECTLY

Check the information for the calculation in node **Value calculation** of the variable properties.

DRIVER FAILS OCCASIONALLY

Analysis with the Diagnosis Viewer (on page 62):
-> Which messages are displayed?