zenon driver manual
SNMPNG32

v.8.20

COPADATA

Contents

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 SNMPNG32

The Simple Network Management Protocol, abbreviated to SNMP, is a network protocol designed to enable the monitoring and control of network elements (such as routers, servers, switches, printers, computers etc.) from a central station. In doing so, the protocol regulates the communication between the devices monitored and the monitoring station. SNMP describes the structure of the data packets that can be sent and the communication sequence. The protocol is designed in such a way that each network-compatible device can be incorporated into the monitoring. The network management tasks that are possible with SNMP include:

▶ Monitoring of network components,

▶   Remote control and remote configuration of network components,

▶   Error detection and error messaging.

As a result of its simplicity, modularity and diversity, SNMP has developed into a standard that is supported by both most management programs and end devices. SNMP is thus, for example, not just dependent on the IP network protocol as transport.

DEFINITION OF TERMS

Term	Description
SNMP	Simple Network Management Protocol This protocol is used for remote maintenance, diagnosis and the protection of networks and hosts. SNMP can be used to manage devices that execute an SNMP agent.
SNMP agent	Serves as a so-called provider, i.e.: it provides information about one device to other SNMP management workstations (in our case to zenon with the SNMP driver). These cyclically poll the SNMP agents for information about the corresponding device properties.
SNMP object	Parts of a device accessed by the SNMP agent or modified by an SNMP agent are called SNMP objects.
MIB	Management Information Bases Is a logical database, which contains a group/collection of SNMP objects. As different network management services can be used for different device types and protocols, each service has its own MIB.
OID	Object Identifier Is a specific detail information of a SNMP object.
TCP/IP	Transmission Control Protocol / Internet Protocol This is a four layer set of manufacturer-independent, frequently used application and transport protocols. Is used by the zenon SNMP driver to read network information via SNMP.
ICMP	Internet Control Message Protocol This protocol sends error and control messages to the participating computers during the transmission process.
Ping	Checks the accessibility of another computer.

Term	Description
TRAP	In SNMP this is a message, which an agent sends to a management system. Therefore the occurrence of an event is displayed on the host, where the agent is executed. The SNMP service can e.g. be configured in the way, that it sends a trap when receiving an information request that neither contains the correct community name nor an accepted host name for the service.

3 Driver history

Date	Build number	Change
8/1/2014	12101	Created driver documentation
12/1/2019	62706	Display of data of SNMP agents in STRING variables with a driver object type in a table.

4 SNMPNG32 - data sheet

General:	
Driver file name	SNMPNG32.exe
Driver name	SNMP Driver New Generation
PLC types	SNMP-compatible devices
PLC manufacturer	SNMP

Driver supports:	
Protocol	SNMPv1; SNMPv2c; SNMPv3; ICMP Ping
Addressing: Address-based	Address based
Addressing: Name-based	--
Spontaneous	X

Driver supports:	
communication	
Polling communication	X
Online browsing	X
Offline browsing	X
Real-time capable	--
Blockwrite	--
Modem capable	--
RDA numerical	--
RDA String	--
Hysteresis	--
extended API	X
Supports status bit **WR-SUC**	X
alternative IP address	--

Requirements:	
Hardware PC	--
Software PC	- The Windows "SNMP Trap" service must not be started. - The SNMPNG32 driver uses its own service zenSnmpTrapSrv.exe to receive traps. The service is installed by the Setup. - UDP traffic for port 161 and port 162 should be enabled at any firewall (or zenSnmpTrapSrv.exe application)
Hardware PLC	--
Software PLC	--
Requires v-dll	--

Platforms:	
Operating systems	Windows 10; Windows 7; Windows 8; Windows 8.1; Windows

Platforms:	
	Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

5 Requirements

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

5.1 PC

HARDWARE

▶   Network card.

SOFTWARE

▶   TCP/IP protocol

▶   Installed and running zenon SNMP trap service.
Note: The service is registered in the appropriate version when zenon is registered and started by the driver if required.

▶   UDP port 161 open for sending and receiving (for other variables). This also applies for the control unit and all devices in the network through which data flows.

▶   UDP port 162 open for receiving (for traps). This also applies to all devices in the network through which data flows. The port for sending must be open on the control unit.

> **⚠Attention**
>
> If the old SNMP32 driver and the new SNMPNG32 driver are operated at the same time, trap receipt only works with one of the two drivers. This depends on whether either the Windows trap service (SNMP32 driver) or the SNMPNG32 trap service was started.

5.2 PLC

HARDWARE

▶ SNMP-compatible network participant.

SOFTWARE

▶ running SNMP service and according SNMP agent

▶ TCP/IP protocol

▶ UDP port 161 open for sending and receiving.

▶ UDP port 162 open for sending (for traps).

6 Configuration

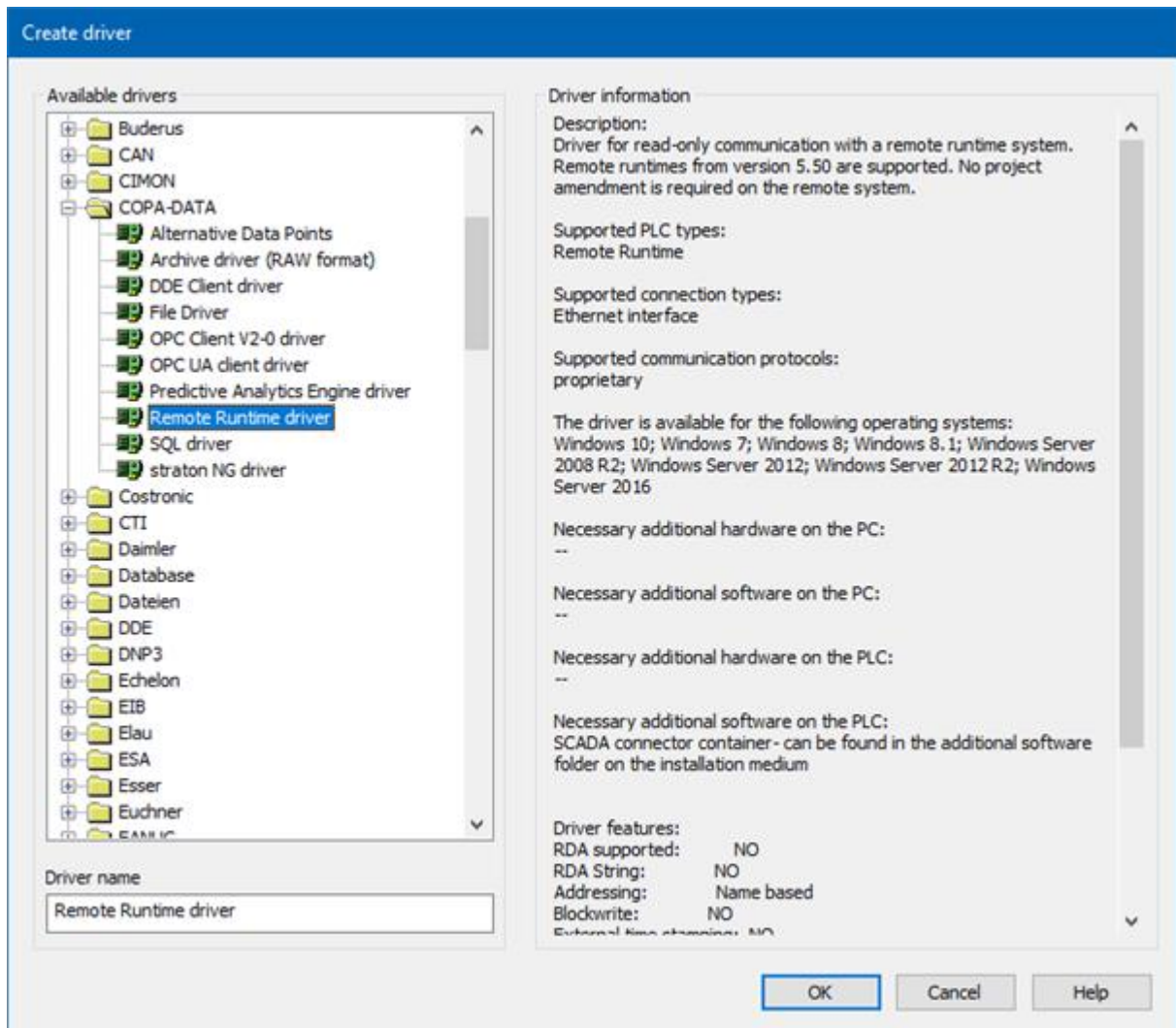In this chapter you will learn how to use the driver in a project and which settings you can change.

> 💡 **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.

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].
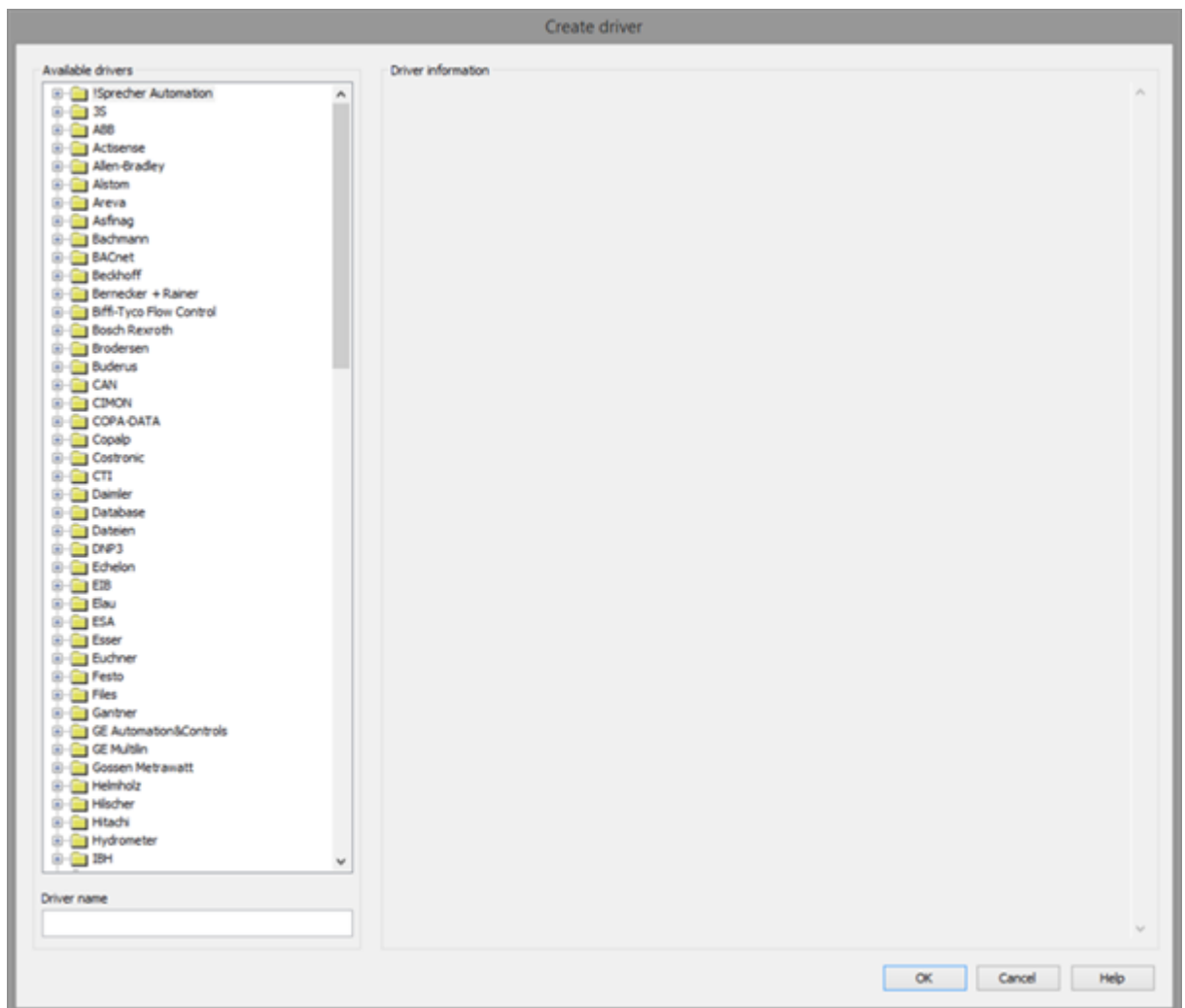
CREATE NEW DRIVER

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.
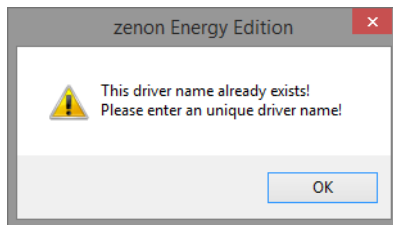
ZENON PROJECT

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

▸ **Intern**

▸ **MathDr32**
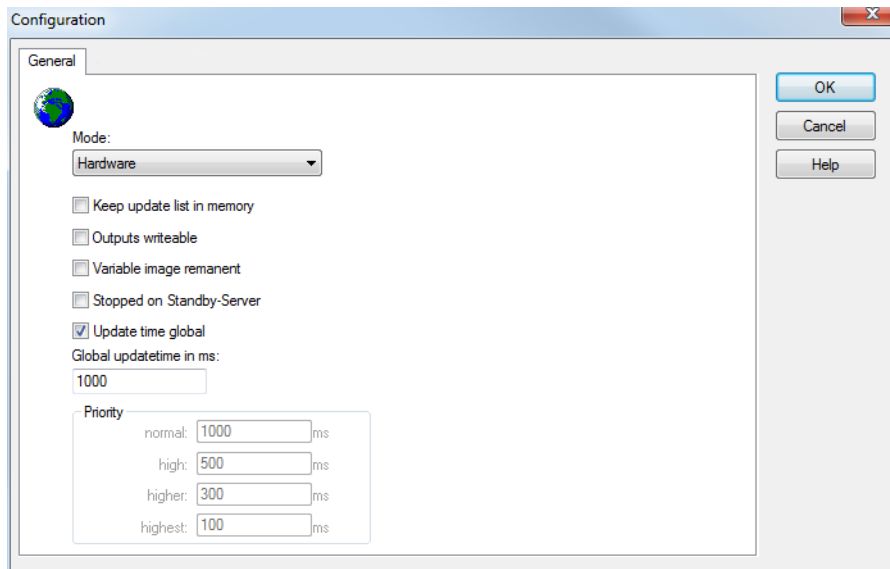
▸ **SysDrv**

> 💡 **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.

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 Configuration

General settings for the SNMP_NG driver

| Parameter	Description

SNMP configuration file	Name of the configuration file (on page 67) in which the driver settings are saved.
	This file contains the configuration file for:
	
	  ▸ SNMP agents (on page 20)
	
	  ▸ Offline MIB list (on page 29)
	
	Value: *must not be empty and must be unique*
	Non-permitted characters: *$%:/*\"?'<>!|*
	
	Default:: *snmpng32_[driver description].txt*

...	Path to the folder that contains the file. Click on the **...** button to open the dialog to select the folder.
Wiederholungen	Number of retries if establishing a query is not successful. Value: *0 to 4.294.967.294* Default: *5*
Timeout before retry	Timeout until next query in milliseconds (timeout

Parameter	Description
	between the retries). Value: *0 to 4.294.967.294* Default: = *1000 milliseconds*
Error response time (ms)	The time in milliseconds that is waited until the next poll is started after a poll (including the configured repetitions) has failed. All read requests up to that point return I-Bit. **Note:** Does not apply to traps if no initial value is read during agent configuration. These variables then remain empty without status, even if there is no connection to the agent. Value: *0 to 4.294.967.294* Default: = *30000 milliseconds*
Addressing mode	Addressing mode of the driver: ▸ Symbolic address is OID ▸ Variable offset is index in the offline MIB list **Note:** This is the only mode in which OIDs with more than 255 characters can be used. ▸ Variable name is OID ▸ Identification of the variables is OID Default: *Symbolic address is OID*
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.

6.2.3 SNMP agents

Agents are programs that run directly on the monitored devices and are used to monitor the network. This task can also be taken on by the hardware directly.

The agents or devices are in a position to record the status of the network device, to change settings themselves or to trigger actions. In doing so, the SNMP protocol enables communication between SCADA and the agents via the network.

CONFIGURATION DIALOG OF THE SNMP AGENTS.
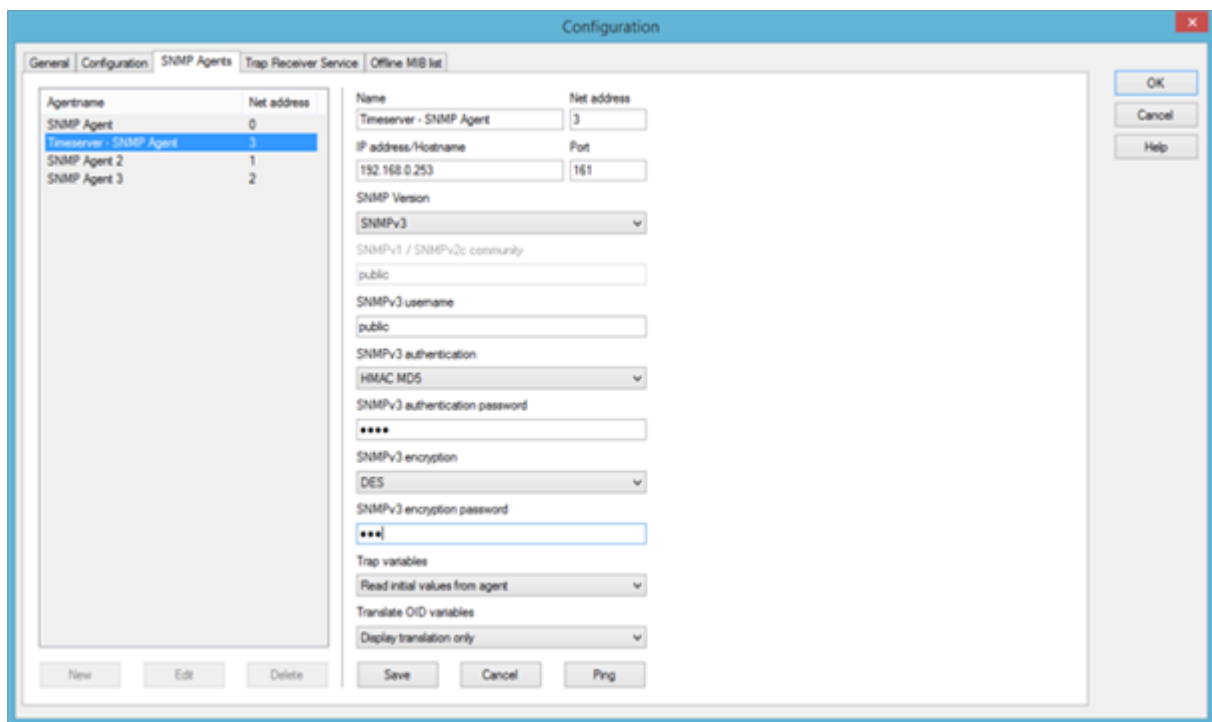
AGENT LIST

In the agent list, the configured agents are displayed, selected for changes and can also be deleted in this list. New agents are added with the **New** button. A maximum of 256 agents (network address *0* to *255*) can be administered.

> **⚠Attention**
>
> If the driver configuration dialog is closed whilst this tab is in editing mode, an error message is displayed and the changes are not saved.

Parameter	Description
Agent name	List of all names of the configured agents. Selection for editing by clicking.
Net address	Configured network address (serial number) of the agent.
New	Create new agents.

Parameter	Description
	Unlocks agent configuration.
Edit	Edit selected agents. Allows changes to the agent configuration. *Inactive* if no agent name has been selected.
Delete	Deletes the selected agents. *Inactive* if no agent name has been selected.

AGENT CONFIGURATION

Details for the agent configurations selected from the agent list.

Note: Only one agent per IP address is permitted!

| Parameter	Description
Name	Name of the agent. Maximum length = *1024 characters*. Default: *SNMP Agent*
Net address	Configured network address (serial number) of the agent. **Note:** This is used for zenon variables in the "Network address" property in order to address the agents at the driver. Value: *0 to 255* Default: *Lowest network addressed that has not yet been issued.*
IP address/host name	IPv4 address or host name of the agent. Maximum length = *1024 characters*. Default: *127.0.0.1*
Port	Protocol port of the agent that is used for polling. Value: *0 to 65.535* Default: *161*
SNMP version	The SNMP version to be used is selected with "SNMP version". ▸ SNMPv1 ▸ SNMPv2c

Parameter	Description
	‣ SNMPv3 Default: *SNMPv1*
SNMPv1 / SNMPv2c community	Community name for SNMP versions SNMPv1 or SNMPv2c. Not active if the **SNMP version** property is *SNMPv3* . **Note:** Community names are the access identification of the agents. This is always "public" (read only) by default. A community name serves as a password that is defined for one or more SNMP hosts. Accepted community names are only used for the authentication of incoming messages. This can be configured in the SNMP service settings of the agent. Maximum length = *1024 characters*. Default: *public*
SNMPv3 user name	User name for authentication via SNMPv3. Only active if the **SNMP Version** property is *SNMPv3*. Maximum length = *1024 characters*. Mandatory field for SNMPv3 authentication. Default: *SNMP User*
SNMPv3 authentication	Type of authentication for the SNMPv3 authentication mechanism. Only active if the **SNMP Version** property is *SNMPv3*. ‣ No authentication ‣ HMAC MD5 ‣ HMAC SHA1 Default: *o authentication*.
SNMPv3 authentication password	Input field for the password for authentication via SNMPv3. Only available if **SNMPv3 authentication** is *HMAC MD5* or *HMAC SHA1*. **Note:** The actual password length and text are protected with periods. The password must be at least 8 characters long.

Parameter	Description
	Mandatory field for SNMPv3 authentication. Value: *1024* characters.
SNMPv3 encryption	Drop-down list to select the type of "SNMPv3 encryption". ▸ No encryption ▸ DES ▸ AES Only available if **SNMPv3 authentication** is *HMAC MD5* or *HMAC SHA1*. Default:*No encryption*
SNMPv3 encryption password	Input field for the password for encrypted communication via SNMPv3. Only available if **SNMPv3** encryption is *AES* or *DES*. **Note:** The actual password length and text are protected with dots. The password must be at least 8 characters long. Mandatory field for SNMPv3 encryption. Value: *1024* characters.
Trap Variables	Drop-down list to select the handling of trap variables: ▸ *Do not read any initial values from agents* No initial values are read for traps. Variables are not reset. ▸ *Read initial values from agents* Start values are read for traps. Variables are not reset. ▸ *Reset trap variables after each trap* No initial values are read for traps. Variables are reset to *0* (*empty* with string data type). Default:*Do not read any initial values from the agent*
Translate OID variables	With "Translate OID variables", there is the selection of how string variables are set if their SNMP data type is OID. Default = display OID and translation. Available values: ▸ *Display OID and translation* OID and translation are transferred as a value

Parameter	Description
	Format = "[OID] ([translation])"
	‣ *Display OID only* Only the OID is transferred as a value.
	‣ *Display translation only* Only the translation is transferred as a value.
	Note: Individually-installed MIBS must be in the following path: *%ProgramData%\COPA-DATA\zenon8.20\CommunicationProfiles\SNMP-MIBS*
Save	Saves the agent settings and leaves editing mode. **Note:** Before saving, there is a check of the configuration entered. A warning dialog appears in the event of erroneous or incomplete entries. Clicking on the **OK** button in this dialog returns to the agent configuration.
Cancel	Discards the agent settings and leaves editing mode.
Ping	Tests a configured **agent IP address**. The "**Ping**" button is always active if the tab is in editing mode **or** if an agent in the agent list is selected. Clicking on this button pings the IP address (the host name) of the agent.

CHECK DIALOG FOR THE "PING" BUTTON

Result of a successful ping attempt with name resolution:
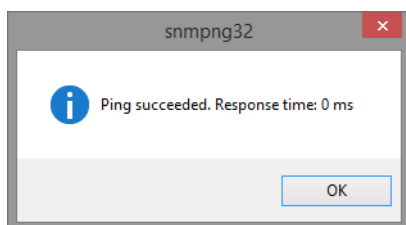
The user is informed of the result in a message box

> ‣ *Ping worked with response time*
> ‣ *Ping unsuccessful with error message*

Hint: If you do not receive a positive response to a ping, it may be the case under certain circumstances that the name has not been triggered correctly. In this case, try pinging the IP address. If this also fails, this means the network participant / agent is probably not accessible.

6.2.3.1 SNMP agents

TRAP VARIABLES

Traps are unrequested messages from an agent to the manager, which are sent when an event happens. These traps are assigned a variable.

Parameter	Description
Do not read any initial values from the agent	No initial value is read for traps. Trap variables are not reset. The value and status of the variable remain empty until a trap is received. **Recommendation:** Select this option if the agent sends traps that cannot be read using GET and are thus not included in the MIB list.
Read initial values from the agent	Start values for traps are read by the agent. Trap variables are not reset. Start value for the trap variables is read by the agent using GET. You receive the last valid value from the last trap or the initial value hen Runtime starts. If the start value was read successfully, the trap variable is no longer read cyclically, but only spontaneously. **Recommendation:** Select this option if traps from the agent are read using GET in order to then expect spontaneous messages (traps).
Reset trap variables after each trap	No initial value is read for traps. The value and status of the variable remain empty until a trap is received. If a trap is received, the value is sent to Runtime and then immediately overwritten with an empty string or the value *0*. A trap value can be recognized by the reaction matrix and written to the AML and CEL

Parameter	Description
	If, when the option is active, the value of the trap variables in Runtime is *0* or an empty string with the status **Spontaneous**, a trap was already received.
	Recommendation: Select this option to react to traps in Runtime that always have the same value to create a CEL entry or alarm.

TRANSLATING OID VARIABLES

Settings for object intensifier variables (OI variables). These allow separate configuration of the translation for each agent.

Requirements:

The following properties define how the OID translation is applied to ingoing values for variables from the SNMPNG driver.

To do this, two conditions must be met:

▸ The variable must be of the driver object type **SNMP variable** or **SNMP trap**.

▸ The incoming value must be of the SNMP data type **OID**.

Incoming values that do not meet these conditions are forwarded without treatment.

Parameter	Description
Display OID and translation	OID and translation are transferred as a value.
	The OID is translated into a descriptive text and the value is forwarded to Runtime in the following format: **[OID] ([Text])**
	Example: **.1.3.6.1.6.3.1.1.5.3 (IF-MIB::linkDown)**.
	If the translation is not successful, only the OID is transferred to Runtime as a value.
Only display OID	Only the OID is transferred as a value.
Only display translation	Only the translation is transferred as a value.
	The OID is translated into a descriptive text and only the translated text is forwarded to Runtime. If the translation is not successful, the OID is transferred to Runtime as a value.

6.2.4 Trap recipient service

The polling behavior of the trap recipient service is configured in this tab.

The trap recipient service reads the configuration that has been set here when the service is started and before starting a polling query cycle (i.e. before it runs through the list of registered agents and checks which still need to be polled).
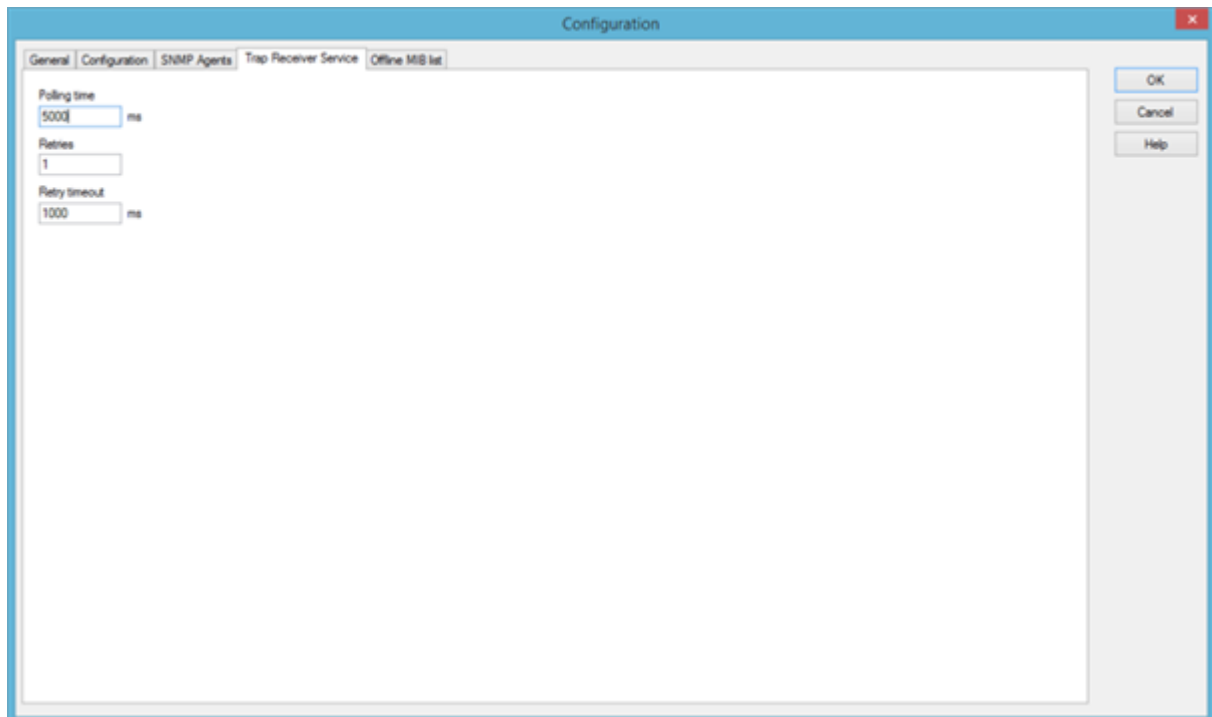
Parameter	Description
Query time	Time in milliseconds before a new query (polling) of the agent. Value: *0 to 4.294.967.294* Default: *= 5000 milliseconds*
Repetitions	Number of retries if establishing a query is not successful. Value: *0 to 4.294.967.294* Default: *5*
Timeout before retry	Timeout until next query in milliseconds (timeout between the retries). Value: *0 to 4.294.967.294* Default: *= 1000 milliseconds*

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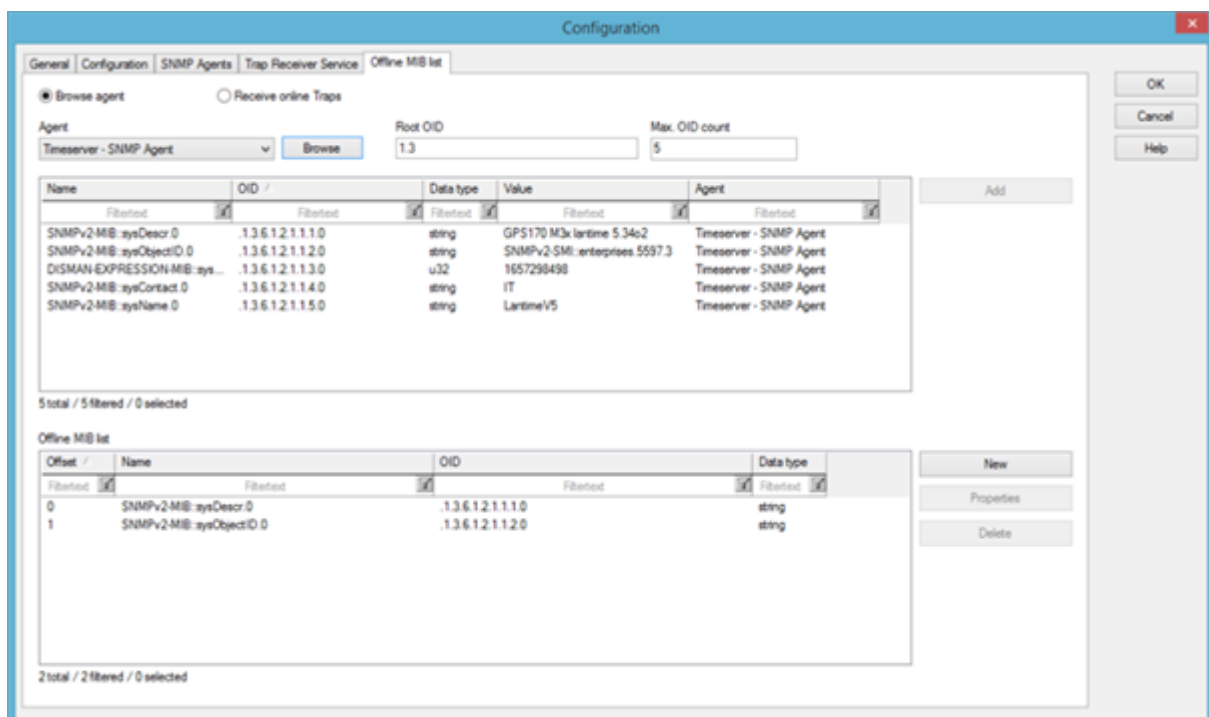
Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

🔆 Information

With SNMPv3, the receipt of one or more traps is only possible up to a maximum of 150 seconds after the last poll.

6.2.5 Offline MIB list

🔆 Information

The control elements to search agents and receive online traps are inactive if no agent is configured in the SNMP Agents (on page 20) tab.

Parameter	Description
Browse agent	Activates "Browsing" mode *If active:* Agent is searched through according to OIDs. The selected agent is searched through using **GETNEXT** for OIDS until either the defined maximum number of OIDs is received or the agent responds to GET with **endofMIBview** as an identifier that there are no further OIDs. **Note:** For some SNMP agents, it is possible to configure how many SNMP packets per second are answered. A number that is too low can have an effect on the reading of the OIDs using **browse agent**.
Receiving online traps	Switches to trap receipt mode. *Active:* The SNMP trap messages from all configured agents are received and displayed by the driver using the **SNMP trap** service implemented in zenon. The following must be the case for the computer to receive traps: ▸ The firewall must allow UDP Traffic to port *162* ▸ The configuration computer must be defined as the target for SNMP traps

AGENT

Parameter	Description
Agent	Selection of the agent from a drop-down list. Clicking on **Browse** browses through the selected agent according to supported information and its MIB objects. The result is then displayed in a receipt list. **Note:** This list contains all agents that were created in the SNMP Agents tab.

Parameter	Description
Browse	Searches through the selected agent.
	Clicking on **Browse** browses through the selected agent according to supported information and its MIB objects.
	The result is displayed in the receipt window.
	Note: Only active if **Browse agent** has been selected.
Start OID	Initial OID for online browsing. Only OIDs that start with the entered value are listed in the receipt list.
	Default:*1.3* Must not be empty.
	Example: Input 1.3.6.1.3 lists all OIDs that start with 1.3.6.1.3. OIDs that start with 1.3.6.1.2 are not listed in this example. **Note:** Only active if **Browse agent** has been selected.
Max. OID number	Maximum number of the OIDs that can be read for the selected agents.
	Default:*200* Must not be empty.
	Value: *1 to 4,294,967,294*
	Note: Only active if **Browse agent** has been selected.

RECEIPT LIST - LIST VIEW

The list view (receipt list) displays the OIDs received via **Browse agent** or **Receive online traps**. The list can be sorted and filtered.

Parameter	Description
Name	Name of the OID to be transferred.
OID	OID number.
Data type	Data type of the transferred OIDs.

Parameter	Description
	Supported data types:
	▸ *i32*
	▸ *u32*
	▸ *string*
Value	Current value of the OID received.
Agent	Name of the agent that was browsed or that sent the received trap.
Add	Adds selected OIDs from the **receipt list** of the **offline MIB list**. If the selected OID is already in the **offline MIB list** or is invalid, you are informed of this with a dialog. The "correct" OIDs are accepted into the **offline MIB list**. **Note:** Only active if at least one OID from the receipt list has been selected.

OFFLINE MIB LIST

In this list, the OIDs received by the receipt list can be edited. In addition, separate OIDs can be created and configured. The list can contain a maximum of 1,073,741,824 entries.

If there was no connection to the desired network participant at the time of project configuration, the OID can also be created without a browser. To do this, click on the **New** button in the **offline MIB list**.

Multiple selection is only supported for the **Remove** button.

Parameter	Description
Offset	Index of the entry. Is issued automatically. Serial number.
Name	(translator) Name of the OID of the management information base (MIB).
OID	OID of the management information base (MIB).
Data type	Data type of the transferred OIDs. Supported data types: ▸ *i32*

Parameter	Description
	▸ *u32*
	▸ *string*
New	Creates a new OID for the **offline MIB list** and opens the dialog for its settings (on page 33).
Properties	Opens the dialog to edit the properties (on page 33) of the selected OID. **Note:** Only individual selection from the **offline MIB list** is possible.
Delete	Removes selected OID(s) from the **offline MIB list**. **Note:** Multiple selection for Remove is possible.
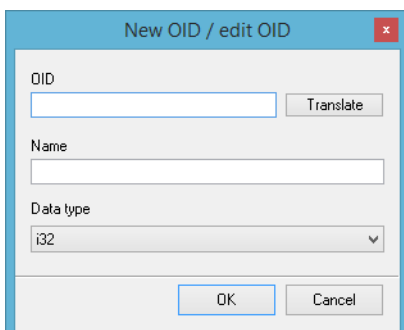
CALCULATION OF THE OFFSET

The offset is calculated automatically and cannot be changed manually. If OIDs are deleted from the offline MIB list, the index is recreated automatically. The gaps in the offset that have arisen as a result of this deletion are automatically closed

6.2.5.1 New OID / Edit OID

Dialog for new creation and editing an element in the offline MIB list. The offset of the element cannot be edited in the process.

If there was no connection to the desired network participant at the time of project configuration, the OID can also be created without a browser.

Parameter	Description
OID	The accurate identification of the OID. In doing so, ensure that the complete and correct identification is entered here; otherwise, the zenon SNMP driver will not be able to read the data.

Parameter	Description
	Must not be empty
	Must contain an interpretable (= standards-compliant) OID.
Translation	Button to create the name for the element. Here, the given OID is interpreted, translated into text display and entered in **Name**.
Name	Translated name of the OID.
	Must not be empty.
	Note: If possible, accept from OID with Translate button.
Data type	Drop-down list to select the data type of the OID. Supported data types:
	▸ i32
	▸ u32
	▸ string
	Default: *i32*

CLOSE DIALOG

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

6.2.5.2 Dialogs for trap receipt mode (receive online traps)

OIDS RECEIPT DIALOG

After clicking on the **Browse** button, the result is visualized in a dialog.
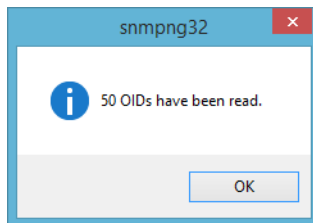
The status and the number of read OIDs is displayed. Clicking on the **OK** button closes the dialog and returns to the **offline MIB list**.

TRAP RECEIPT MODE (RECEIVE ONLINE TRAPS)

This dialog is always displayed if the configuration dialog for the offline MIB list (on page 29) or the variable import dialog (on page 50) switch to trap receipt mode. The dialog accepts the messages from the trap recipient service.
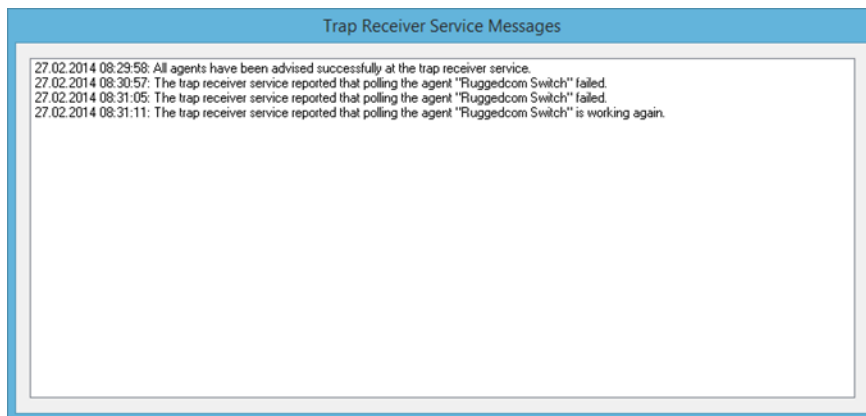
POSSIBLE MESSAGES:

▶ Agent registration successful or failed

▶ Polling of an agent by the trap recipient service has failed

▶ Polling of an agent by the trap recipient service is successful again after a previous failure.

To close this dialog again, switch from **Receive online traps** to **Browse agent** in the **offline MIB list** tab.
The dialog is also closed if, in the driver configuration dialog, a switch to a different tab is made or a button is clicked.

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:
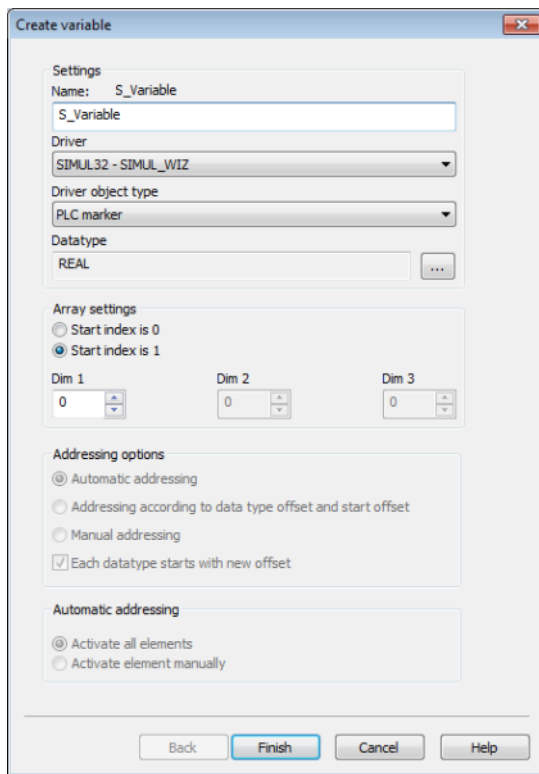
1. Select the **New variable** command in the **Variables** node in the context menu
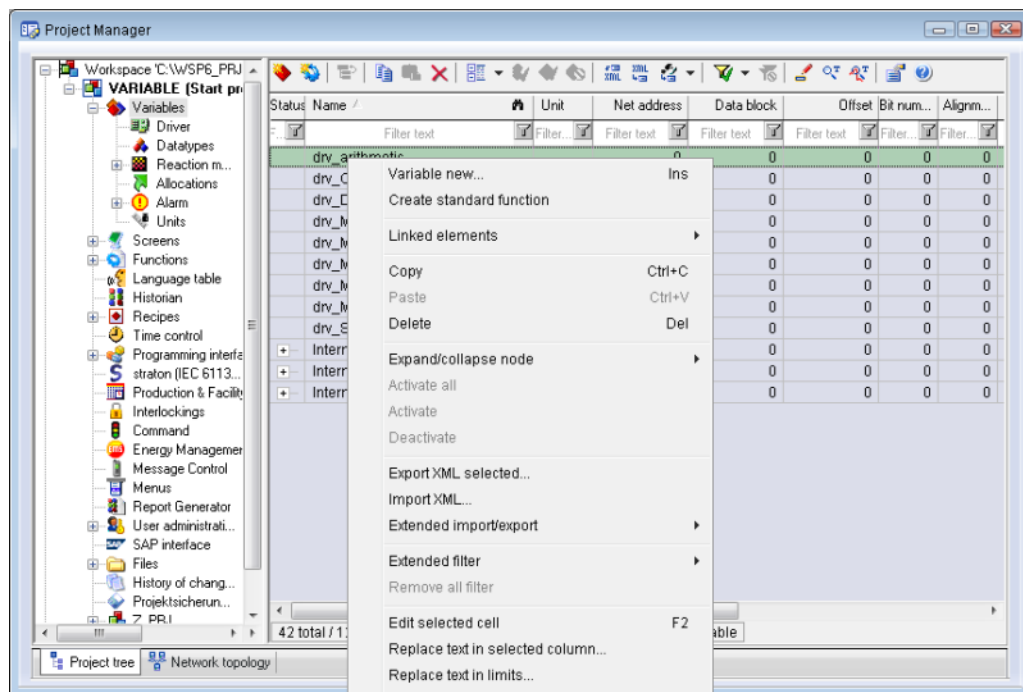
The dialog for configuring variables is opened

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

CREATE VARIABLE DIALOG

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.
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

Property	Description
	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. If the driver is addressed by means of identification, the OID must be entered here.
Identification	If the driver is addressed by means of identification, the OID must be entered here. Freely definable identification. E.g. for Resources label, comments, ...
Addressing	
Net address	Network address of the agent from which the variable comes.
Data block	not used for this driver
Offset	If the driver addresses by offset = Index in the offline MIB list, the index of the OID must be entered into the offline MIB list. Adjustable from *0* to *4294967295*.
Alignment	not used for this driver
Bit number	not used for this driver
String length	Only available for String variables. Maximum number of characters that the variable can take.
Symbolic address	Option to choose 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.

Group/Property	Description
	Maximum length: *1024* characters.
Driver connection/**Driver Object Type**	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Driver connection/**Data Type**	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. **Attention:** If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

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
Ping status	64	X	--	*BOOL (Pingstatus),* *UDINT (Round Trip Time bei Erfolg)* *STRING (Fehlermeldung bei Fehlschlag)*	Is for the connectivity check of an agent using *ICMP ping*. **Note:** A ping status variable never goes to I-bit (with the exception of when the network address does not exist).
SNMP tables	68	X	--	*STRING*	You can find further information on this in the **Tables** (on page 58) chapter in this manual.
SNMP variable	65	X	X	*DINT, UDINT, STRING*	Variable that is read with *SNMP GET* and can be

Driver Object Type	Channel type	Read	Write	Supported data types	Description
					written with *SNMP SET*
SNMP counter	66	X	X	*DINT, UDINT*	Variable whose value is calculated: The current absolute value is read from the agent with *SNMP GET*. The variable value is calculated as "[current value] – [previous value]" and sent up (relative counter). The value at the agent can be set with *SNMP SET*.
SNMP trap	67	X	X	*DINT, UDINT, STRING*	Values for these variables are written up if an SNMP TRAP has been received (with the exception of initial values). The value at the agent can be set with *SNMP SET*.
SNMP-Table	68	X	--	*STRING*	SNMP variables under the specified address (=OID) are passed to the zenon Runtime as CSV string in order to be able to display them as table.
Communication details	35	X	X	*BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING*	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.

Driver Object Type	Channel type	Read	Write	Supported data types	Description
					You can find detailed information on this in the Communication details (Driver variables) (on page 61) 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
-	UINT	2
-	INT	1
	UDINT	4
	DINT	3
-	ULINT	27

PLC	zenon	Data type
-	LINT	26
-	REAL	5
-	LREAL	6
	STRING	12
-	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.

7.4   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 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

Identification	Type	Field size	Comment
			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
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

Identification	Type	Field size	Comment 48
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. 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.

Identification	Type	Field size	Comment	
MAXGRAD	N	16	Gradient overflow for counter REMA.	49

⚠**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
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

Identification	Type	Field size	Comment
			(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.
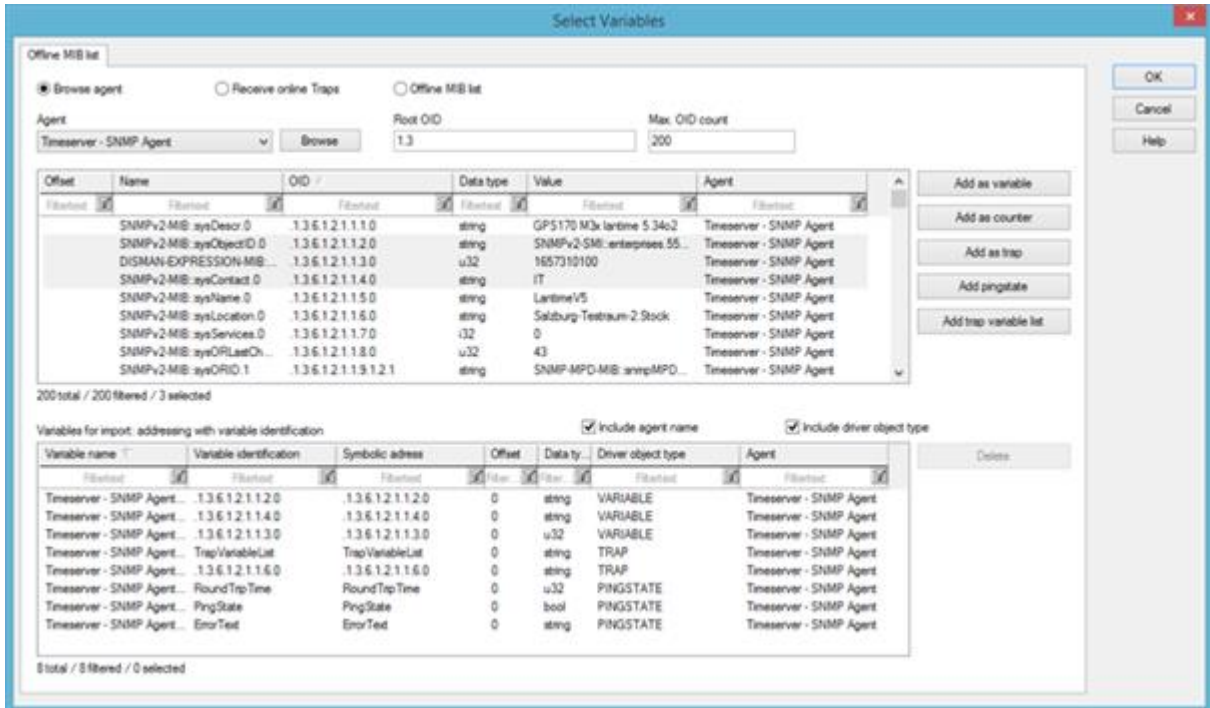
7.4.3 Online import

The dialog for the online import of variables consists of two tabs:

▶

▶

In this dialog, the driver reads data from the SNMP agent. The content is stored in a STRING variable of the **SNMP table** driver object type.

7.4.3.1 Online import

Parameter	Description
Browse agent	Activates the "Browsing" mode. OIDs are read off from the agent directly. **Note:** Only active if the addressing mode of the driver is not "*Offset = Index in the OFFLINE MIB list*".
Receiving online traps	Switches to trap receipt mode. OIDs are read off by the agent via trap. **Note:** Only active if the addressing mode of the driver is not "*Offset = Index in the OFFLINE MIB list*".
Offline MIB list	Uses OIDs from the offline MIB list of the driver configuration

AGENT

Parameter	Description
Agent	Selection of a configured agent from drop-down

Parameter	Description
	list.
	Note: This list contains all agents that were created in the **SNMP Agents** tab of the driver configuration.
Browse	Searches through the selected agent.
	Clicking on **Browse** browses with the selected settings.
	The result is displayed in the source list. A dialog informs you of the result.
	Note: Only active if dialog is in browsing mode (= **Browse agent** active).
Start OID	Start OID for online browsing. Only OIDs that start with the value entered are displayed in the source list.
	Default:*1.3* Must not be empty.
	Example: Input 1.3.6.1.3 lists all OIDs that start with 1.3.6.1.3. OIDs that start with 1.3.6.1.2 are not listed in this example. **Note:** Only active if dialog is in browsing mode (= **Browse agent** active).
Max. OID number	Maximum number of the OIDs that can be read for the selected agents.
	Default:*200* Must not be empty.
	Value: *1 to 4,294,967,294*
	Note: Only active if dialog is in browsing mode (= **Browse agent** active).

SOURCE LIST - LIST VIEW

The source list lists the available OIDs.

Depending on the dialog mode, the data in the list comes from:

▶ The agents that have been browsed in browsing modes:
Offset remains empty

▶ The received traps of all agents in trap recipient mode:
Offset remains empty

▶ The offline MIB list in the offline MIB list mode:
Value and agent remain empty

Parameter	Description
Offset	Offset of an OID. **Note:** is only filled with a value if OIDs are listed by means of the offline MIB list. This offset cannot be changed.
Name	Name of the OID to be transferred.
OID	OID number
Data type	Data type of the transferred OIDs. Supported data types: ▶ i32 ▶ u32 ▶ string
Value	Current value of the received OID. With **Browse agent**, this is always the same as **Agent**.
Agent	Name of the agent with which an OID was read. **Note:** This list corresponds to the selected agent from the drop-down list **Agent**.
Add as variable	Accepts the selected OIDs in the target list of the variables to be imported.
Add as counter	Accepts the selected OIDs in the target list of the variables to be imported. **Note:** Only numerical data types can be transferred (data type *u32* or data type *i32*).
Add as trap	Accepts the selected OIDs in the target list of the variables to be imported.
Add ping status	Accepts the selected OIDs in the target list of the variables to be imported.

Parameter	Description
	Note: Is always active - no OID needs to be selected for this. Instead of a number of entries in the source list, the following "OIDs" and data type are always used, whereby the OID is also the starting point for the constructed name and the offset is always 0: ▸ OID = "*PingState*", data type: *bool*. ▸ OID = "*RoundTripTime*", data type: *u32*. ▸ OID = "*ErrorText*", data type: *string*.
Add trap variable list	Accepts the selected OIDs in the target list of the variables to be imported. **Note:** Only active if the addressing mode of the driver is not "*Offset = Index in the OFFLINE MIB list*".

TARGET LIST - PREVIEW OF THE NEW VARIABLES TO BE CREATED

The target list lists the OIDs transferred from the source list and contains a preview of the variables to be created. The list can be sorted and filtered. The list entries cannot be edited in this listing.

These OIDs are transferred to the target list of the variables to be newly created by clicking on one of the buttons.

Note:

Option	Description
Add name of the agent	If active, the agent name is placed in front of the variable name. **Attention:** This setting is only used for transfer by means of clicking on a button. Subsequent activation does not activate the variable name of the OIDs that have already been transferred.
Insert driver object type	If active, the driver object type is placed in front of the variable name. **Attention:** This setting is only used for transfer by means of clicking on a button. Subsequent

Option	Description
	activation does not activate the variable name of the OIDs that have already been transferred.
Variable name	Name of the variable to be created
Identification	Identification of the variables to be created.
Symbolic address	Symbolic address of the variable to be created.
Offset	Offset of the variable to be created This offset is issued automatically.
Data type	Data type of the variable to be created. Supported data types: ‣ i32 ‣ u32 ‣ string
Driver Object Type	Driver object type of the new variable to be created. This depends on the button to accept source list entries in the target list entries.
Agent	Name of the agent. Accepts the agent selected at the time of transfer from the drop-down list **Agent**.
Remove	Removes selected elements from the target list of the new variables to be created. **Note:** Multiple selection is possible.

CLOSE DIALOG

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

Note: Clicking on the **OK** button closes the dialog and creates the variables in the zenon Editor.

7.4.3.1.1 Variable online import - buttons

When transferring OIDs from the source list into the target list with the new variables to be created, the following different rules are applicable for the buttons.

GENERAL:

The buttons are only active if at least one entry in the source list is selected. The **Add ping status** and **Add trap variable list** buttons are always active.

When clicking on the button:

▶ The length of the OID is checked:

    ▶ If the addressing mode in the driver is: "Offset = Index in the offline MIB list": OID longer than 128 characters is shortened

    ▶ Error message if OID is longer than 128 characters

▶ The complete OID name is generated.
The complete OID name comprises:

    ▶ Validated OID name
The validated OID name depends on the selected button. Please note the information provided in the detailed description of the buttons.

    ▶ + name of the agent if **Insert name of the agent** has been activated.
The name of the agent is inserted before the validated OID name, followed by an underscore.

    ▶ + driver object type if **Insert driver object type** has been activated.
The name of the driver object type is inserted before the validated OID name, but after the agent name, followed by an underscore.

▶ If the length of the name is checked for the new variable to be created:
If the (compiled) name is longer than 128 characters, the name is shortened to 128 characters.

▶ If the name of the variable to be created is generated:

    ▶ If the addressing mode in the driver is: "Variable name is OID":
Validated OID

    ▶ Otherwise:
Complete OID name

▶ If the identification is generated:

    ▶ If the addressing mode in the driver is: "Variable name is OID":
Complete OID name

    ▶ Otherwise:
Validated OID

▶ If the symbolic address is generated:
Symbolic address = validated OID.

▶ If the offset is generated.
If there is not offset: Offset = *0*.

▶ The driver object type is generated according to the selected button.
Please find details in the following description of the buttons.

▶ The agent selected in the drop-down list is transferred.

▶ The selected entries from the source list are checked and any errors are output in a dialog.
in doing so, all possible OIDs are transferred to the target list. Non-permitted OIDs are not
taken into account during the transfer.

▶ The selected entries in the target list or the new variables to be created are transferred.

ADD AS VARIABLE

▶ Driver object type is always *VARIABLE*

ADD AS COUNTER

▶ Driver object type is always *COUNTER*

▶ Entries from the source list with the *string* data type are not permitted

ADD AS TRAP

▶ Driver object type is always *TRAP*

ADD PING STATUS

▶ Button is always active

▶ Driver object type is always *PINGSTATE*

▶ Instead of the entries selected in the source list, the following entries are always written:

▶ OID = "*PingState*", data type: *bool*.

▶ OID = "*RoundTripTime*", data type: *u32*.

▶ OID = "*ErrorText*", data type: *string*.

▶ The offset is always *0*.

ADD TRAP VARIABLE LIST

The button is only active if the driver addressing mode is not "*Offset = Index in the offline MIB list*".

▶ Driver object type is always *Trap*.

- ▸ Instead of the entry selected in the source list, the following entry is always written: OID = "TrapVariableList", data type = *string*.

- ▸ The offset is always *0*.

7.4.3.2 Tables
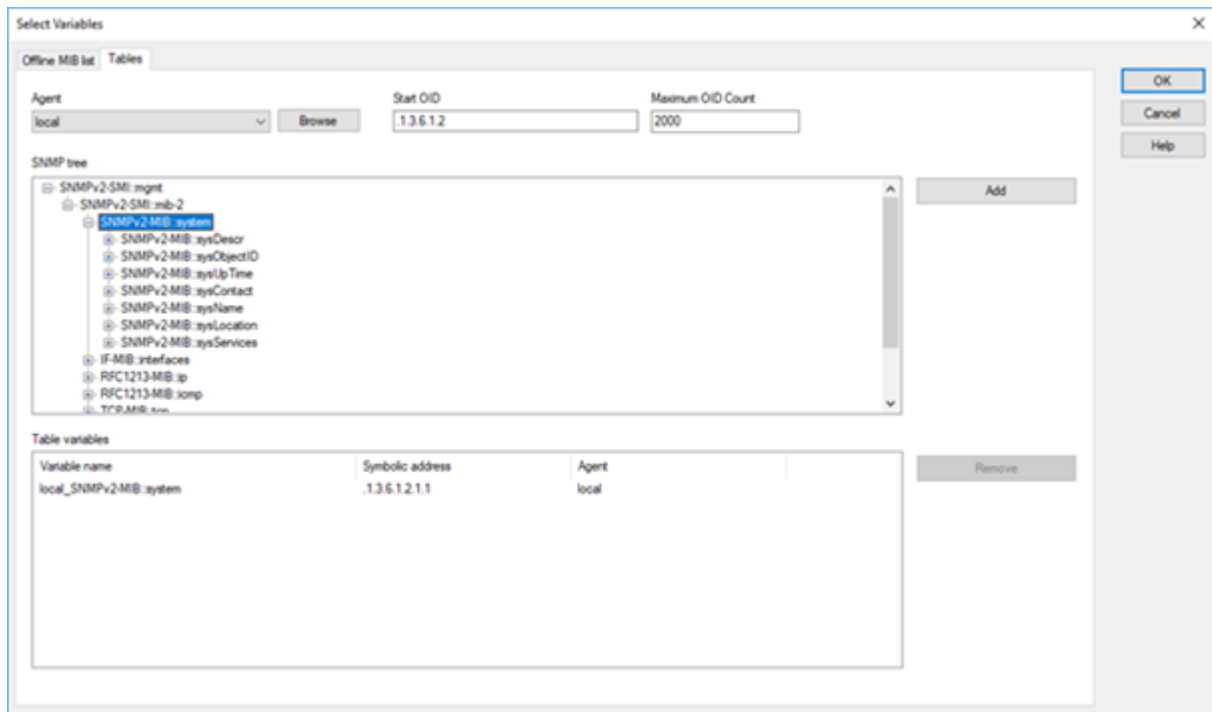
In this dialog, the driver reads data from the SNMP agent. The content is stored in a STRING variable of the **SNMP table** driver object type.

Parameter	Description
Agent	Name of the agent whose content is read by clicking button **Browse**. Select from drop-down list. **Note:** This list contains all agents that were created in the **SNMP Agents** (on page 20) tab of the driver configuration. Default:*local* Results are displayed in a tree structure at **SNMP tree**.
Browse	Starts the readout of the contents of the selected agent.

Parameter	Description
	Results are displayed in a tree structure at **SNMP tree**.
Start OID	Start OID for online browsing. Only OIDs that start with the value entered are displayed in the source list. Must not be empty. **Example:** Input 1.3.6.1.3 lists all OIDs that start with 1.3.6.1.3. OIDs that start with 1.3.6.1.2 are not listed in this example. Results are displayed in a tree structure at **SNMP tree**.
Maximum OID Count	Maximum number of the OIDs that can be read for the selected agents. Default:*200* Must not be empty. Value: *1 to 4,294,967,294* Results are displayed in a tree structure at **SNMP tree**.

SNMP tree	Tree structure of the OID of the read **SNMP Agents**.
	
	If you select a node which displays a table, button **Add** is enabled.
	
	▸ *[+]*
	  Expand the tree view.
	
	▸ *[-]*
	  Collapse the tree view
	
	Note: the configured MIB files are also used to display the tree nodes. If these are not available or the path is not configured correctly, they will be displayed in numerical form using the OID.

Parameter	Description
Add	Adds a selected variable to the list of **Table variables** for the import in the zenon Editor. If a node is selected in the **SNMP tree**, all OIDs contained therein are added **Tables variables**. In doing so, the OIDs in the contained subnodes are also taken into account. Only active if, in the **SNMP tree**, at least one node or entry has been selected.
Table variables	List of variables which are applied from **SNMP tree** to the zenon Editor via import. ▸ *Variable name* ▸ *Symbolic address* Symbolic address of the variable. ▸ *Agent* Name of the agent.
Remove	Deletes the selected variable from the list **Table variables** without requesting confirmation. Only active if an entry has been selected in **Table variables**.

CLOSE DIALOG

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

GENERAL

For a variable of **Driver object type** *SNMP-Table* the driver reads all related OIDs and writes the result in a string variable in the following form. In doing so, the driver attempts to interpret SNMP variables that are below the OID of the zenon table variables in the SNMP tree structure as a table.

C…Spalte

R…Reihe

Titel C1, Zelle C1/R1, Zelle C1/R2, Zelle C1/…, Zelle C1/Rn;

Titel C2, Zelle C2/R1, Zelle C2/R2, Zelle C2/…, Zelle C2/Rn;

Titel C…, Zelle C…/R1, Zelle C…/R2, Zelle C…/…, Zelle C…/Rn;

Titel Cm, Zelle Cm/R1, Zelle Cm/R2, Zelle Cm/…, Zelle Cm/Rn;

To do this, the tables (related OIDs) must meet the following conditions:

- ▶ at least two columns
- ▶ at least one row
- ▶ all columns have the same number of cell entries (=rows)

If one of these conditions is not meet, the content of the variable remains empty. This is also protocoled in the LOG file.

If the content of the table does not fit the string variable, it is truncated after the last complete column.

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
LineStateProceeding	*BOOL*	24.9	Dialing completed
LineStateOnHold	*BOOL*	24.10	Connection in hold

Name from import	Type	Offset	Description
LineStateConferenced	*BOOL*	24.11	Connection in conference mode.
LineStateOnHoldPendConf	*BOOL*	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	*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 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 interface

Value = _0_ if the driver cannot establish any serial connection. |

Name from import	Type	Offset	Description
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
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

Name from import	Type	Offset	Description
			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.
RdErrDatablockNo	UDINT	46	Block number when the last reading error occurred.
RdErrMarkerNo	UDINT	47	Marker number when the last reading error occurred.

Name from import	Type	Offset	Description
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:

GET, SET, GETNEXT

Data is read from a devices that support SNMP with the *GET*, *SET* and *GETNEXT* functions.

PING STATUS

Via the ping status you define whether the end device can be reached via ICMP protocol.

SNMP TRAP

The driver supports the receipt of SNMP traps with an SNMPV1, SNMPv2c or SNMPv3 header.

NOT SUPPORTED:

▸ IPv6 as transport protocol

▸ The "SNMP over TLS" connection type

▸ The "INFORM" packet type

8.1 SNMP configuration file

The SNMBNG32 driver writes its configuration (agents and offline MIB list) in INI format to a text file in the project directory "FILES\zenonzenon\custom\drivers".

> 👍 **Hint**
>
> Mark the project in the project manager -> press keys **Ctrl+Alt+E** -> the Windows Explorer opens folder *Project_SQL_directory/FILES/*

When compiling the project in zenon, this file is copied to the Runtime file directory.

FILE STRUCTURE OF A DRIVER CONFIGURATION FILE

The agents and open MIB items are saved in a file with the INI structure. There are the network addresses 0-255 for the agents. In the table, the entries for the agent are documented with the network address 0.

AGENT

INI entry	Description
[AGENT_HW_0]	INI entries of the agent configuration. In the configuration file, 255 agents are entered automatically. Configured agents follow this documented file structure. Agents that have not yet been configured have an empty entry: **Example:** [AGENT_HW_*{serial number}*]
AgentName	Visual name of the agent. Default: *empty*
TrapMode	Trap mode of the agent: ▸ 0 = do not read any initial values ▸ 1 = read initial values ▸ 2 = do not read any initial values and treat trap variables as wipers (after setting a value to 0 / resetting empty string) Default: *0*
TranslationMode	OID translation mode of the agent: ▸ 0 = display OID and translation ▸ 1 = display OID only ▸ 2 = display translation only Default: *0*

INI entry	Description
ItemCount	Number of the items to be read when browsing. Default: *200*
RootOID	Root OID for the browsing of an agent. Default: *.1.3*
AgentAddress	IPv4 address of the agent OR host name. Default: *127.0.0.1*
AgentPort	Port for SNMP polling at the agent. Default: *161*
SnmpVersion	SNMP version of the agent: ‣ 0 = SNMPv1 ‣ 1 = SNMPv2c ‣ 2 = SNMPv3 Default: *0*
SnmpCommunity	Community string for the agent (SNMPv1 and SNMPv2c). Default: *public*
SnmpUser	User name for the agent. Default:: *SNMP User*

INI entry	Description
AuthMethod	Authentication method for the agent (SNMPv3): ▸ 0 = no authentication ▸ 1 = authentication with HMAC MD5 ▸ Authentication with HMAC SHA-1 Default: *0*
AuthKey	Authentication password for the agent (SNMPv3); is stored in encrypted form.
PrivMethod	Encryption method for the agent (SNMPv3): ▸ 0 = no encryption ▸ 1 = encryption with DES ▸ 2 = encryption with AES Default: *0*
PrivKey	Encryption password for the agent (SNMPv3); is stored in encrypted form. Default: *empty*

MIBS

INI entry	Description
[MIB_LIST]	INI entries for MIBs. The INI entries of the MIBs start after the entries for the agents. This means that the first MIB entry always starts after the *[AGENT_HW_255]* entry.

INI entry	Description
MibItemCount	Number of items in the offline MIB list. Default: *0*
Name_0	Visual name of the MIB item Default: *empty*

DataType_0	Data type of the MIB item:
	
	   ▸ 0 = signed 32-bit integer
	
	   ▸ 1 = unsigned 32-bit integer
	
	   ▸ 2 = string
	
	Default: *0*

OID_0	Text OID of the MIB item Default: *empty*
NumericalOID_0	Numerical OID of the MIB item. This entry is placed in front of each OID Default: *empty*
Length	Number of fields of the numeric OID of the MIB item. (for example 1.3.6 has 3 fields) Default: *0*

INI entry	Description
	72
NumericalOID_0_Field_0	Value for a field of the numerical OID of the MIB item. The last figure of the section is the field index. Default: *0*

EXAMPLE

INI entry for two MIBs, from string data type with 8 items each:

```
[MIB_LIST]

MibItemCount=2


Name_0=SNMPv2-MIB::sysDescr.0

DataType_0=2

OID_0=.1.3.6.1.2.1.1.1.0

NumericalOID_0=.1.3.6.1.2.1.1.1.0

NumericalOID_0_Length=9

NumericalOID_0_Field_0=1

NumericalOID_0_Field_1=3

NumericalOID_0_Field_2=6

NumericalOID_0_Field_3=1

NumericalOID_0_Field_4=2

NumericalOID_0_Field_5=1

NumericalOID_0_Field_6=1

NumericalOID_0_Field_7=1

NumericalOID_0_Field_8=0


Name_1=SNMPv2-MIB::sysObjectID.0

DataType_1=2
```

```
OID_1=.1.3.6.1.2.1.1.2.0

NumericalOID_1=.1.3.6.1.2.1.1.2.0

NumericalOID_1_Length=9

NumericalOID_1_Field_0=1

NumericalOID_1_Field_1=3

NumericalOID_1_Field_2=6

NumericalOID_1_Field_3=1

NumericalOID_1_Field_4=2

NumericalOID_1_Field_5=1

NumericalOID_1_Field_6=1

NumericalOID_1_Field_7=2

NumericalOID_1_Field_8=0
```

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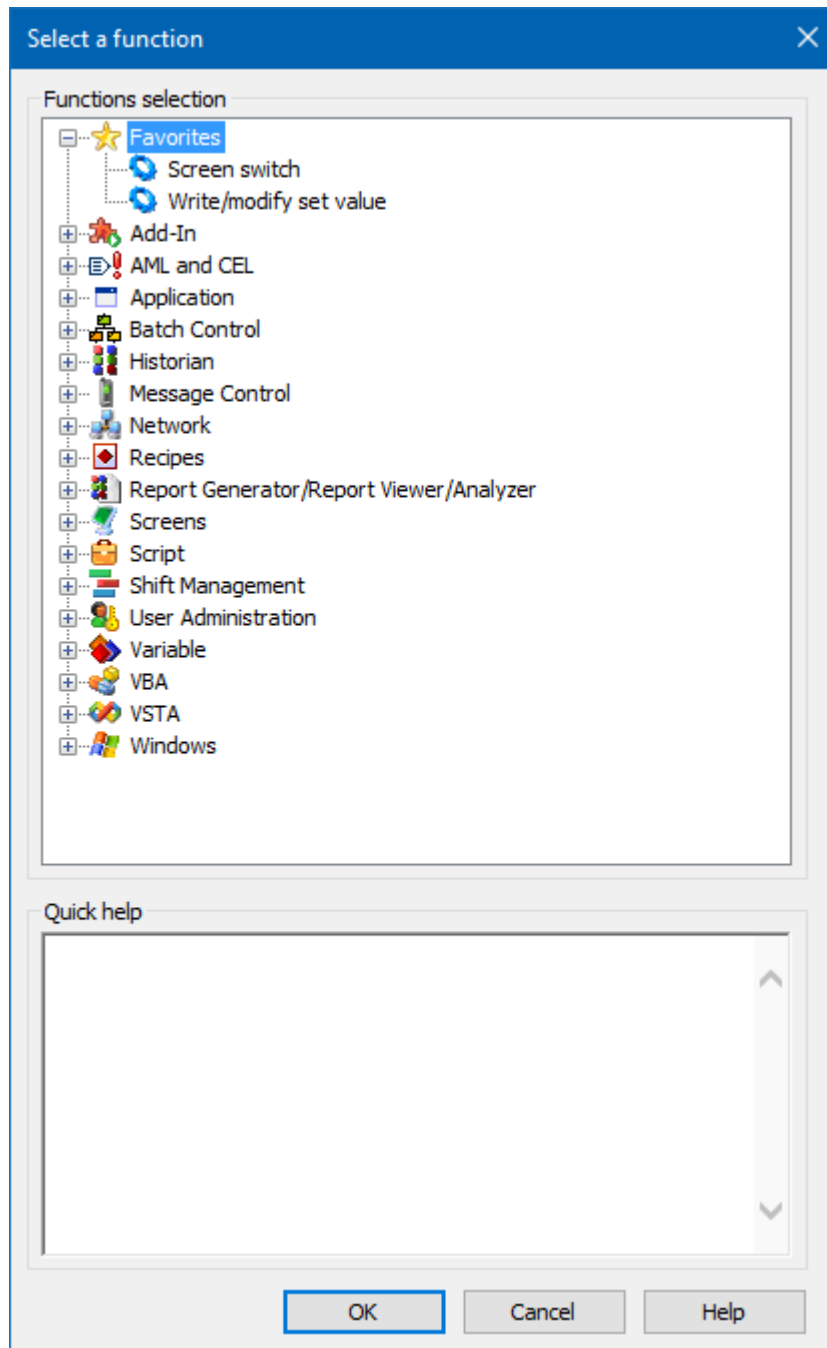
> ⚠**Attention**
>
> The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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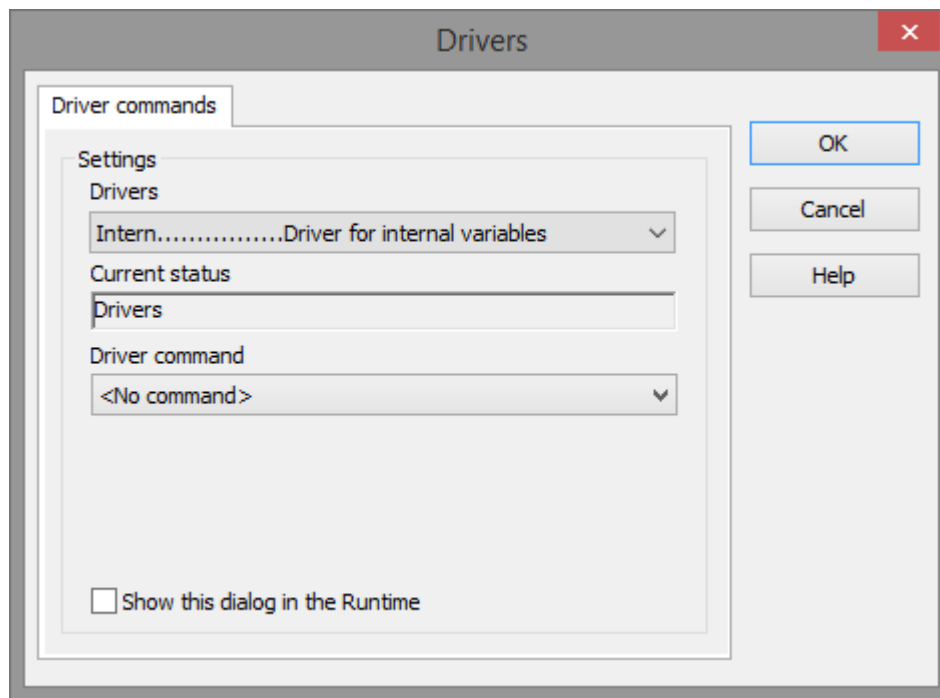To configure the function:

1. Create a new function in the zenon Editor.

    The dialog for selecting a function is opened
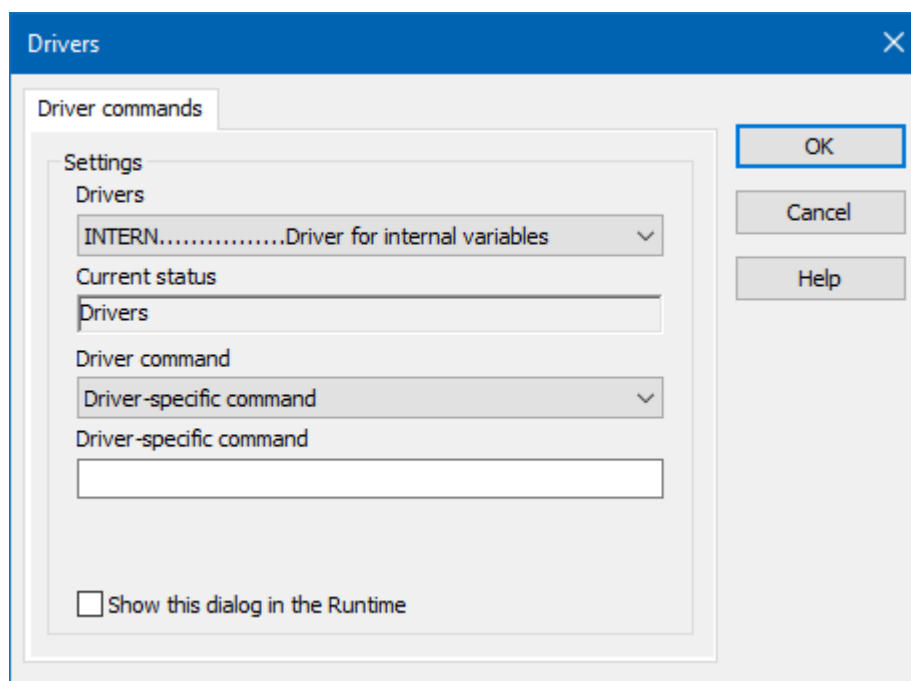

2. Navigate to the node **Variable.**
3. Select the **Driver commands** entry.

The dialog for configuration is opened

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

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 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.

10.2 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.

10.3 Check list

Questions and hints for fault isolation:

▸ Is the SNMP driver correctly installed? (It has to be installed on all devices that should be read – except for the ping status request)

▸ Is the correct key installed and selected? (default=public)?

▸ Is the TCP/IP protocol installed?

▸ Are the properties for authentication and encryption correctly configured for the agents?

GENERAL TROUBLESHOOTING

▸ Is the PLC connected to the power supply?

▸ Analysis with the Diagnosis Viewer (on page 78):
 -> Which messages are displayed?

▸ Are the participants available in the **TCP/IP** network?

▸ Can the PLC be reached via 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 PLC be reached at the respective 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.

▶ Are you using the correct cable which is recommended by the manufacturer for the connection between the PLC and the PC?

▶ Did you select the right COM port?

▶ Do the communication parameters match (Baud rate, parity, start/stop bits,...)?

▶ Is the COM port blocked by another application?

▶ Is the end device correctly configured for the use of SNMT?

▶ Did you configure the Net address in the address properties of the variable correctly?

> ▶ Does the addressing match with the configuration in the driver dialog?

> ▶ Does the net address match the address of the target station?

▶ Did you use the right object type for the variable?
Example: Driver variables based on driver object type **Communication details** are purely statistics variables. They do not communicate with the PLC.
You can find detailed information on this in the Communication details (Driver variables) (on page 61) chapter.

▶ Does the offset addressing of the variable match the one in the PLC?

SOME VARIABLES REPORT INVALID.

▶ At least one variable of the net address is faulty.

VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY

Driver is not working. Check the:

> ▶ Installation of zenon

> ▶ the driver installation

> ▶ The installation of all components
> -> Pay attention to error messages during the start of the Runtime.

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 78):
-> Which messages are displayed?

10.4 Error treatment

COMPATIBILITY OF SNMP32 AND SNMPNG32

The old (**SNMP32**) and the new (**SNMPNG32**) SNMP drivers should never be operated at the same time. The **SNMP32** driver needs the Windows SNMP trap service for trap receipt and starts this if necessary. The **SNMPNG32** driver needs the zenon SNMP trap service to receive traps and starts this if necessary. Both services attempt to open the UDP port 162 for monitoring. However because this can only be opened once per computer, "first come first served" is applicable in this case and one of the two traps cannot receive traps.

DRIVER INSTANCES

In general, it is recommended that each SNMP agent gets its own **SNMP NG** driver instance, so that a slow reacting or failed agent does not impair the polling communication of the other agents. In the trap recipient service, all registered SNMP agents of all driver instances are indeed polled, which leads to a mutual impairment on performance, but the standard retry settings are to be kept lower there in order to keep the effects as low as possible.

OFFSET: INDEX IN THE OFFLINE MIB LIST"

If the driver is operated with the "**Offset = Index in the offline MIB list**" addressing mode, it is recommended that, after the offline MIB list of all variables has been changed, all variables of this driver (with the exception of ping status and trap variable list) are to be checked and it is ensured that they point to the intended OID. This mode should in general only be used if an OID to be used has

more than 128 characters (maximum length of variable name, identification and symbolic address), because only this addressing mode can process this length.