zenon

by COPA-DATA

zenon driver manual

SNMP32

v.8.20

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 SNMP32

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.
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

Term	Description
	name for the service.

3 Snmp32 - data sheet

General:	
Driver file name	Snmp32.exe
Driver name	SNMP driver
PLC types	SNMP-compatible devices
PLC manufacturer	SNMP

Driver supports:	
Protocol	SNMPv1; SNMPv2c
Addressing: Address-based	Address based
Addressing: Name-based	--
Spontaneous communication	X
Polling communication	X
Online browsing	X
Offline browsing	--
Real-time capable	--
Blockwrite	--
Modem capable	--
RDA numerical	--
RDA String	--
Hysteresis	X
extended API	--

Driver supports:	
Supports status bit **WR-SUC**	X
alternative IP address	--

Requirements:	
Hardware PC	Standard network card
Software PC	- The Windows "SNMP Trap" service must be installed and started - UDP traffic for port 161 and port 162 should be enabled on any firewall (or snmptrap.exe application)
Hardware PLC	--
Software PLC	SNMP agent (server)
Requires v-dll	--

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

4 Driver history

Date	Driver version	Change
07.07.08	1400	Created driver documentation

DRIVER VERSIONING

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

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

Example

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

5 Requirements

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

5.1 PC

HARDWARE

- Network card.

SOFTWARE

- TCP/IP protocol
- Installed and running SNMP trap service.
  Note: The Windows service is not started by default.
- 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.

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

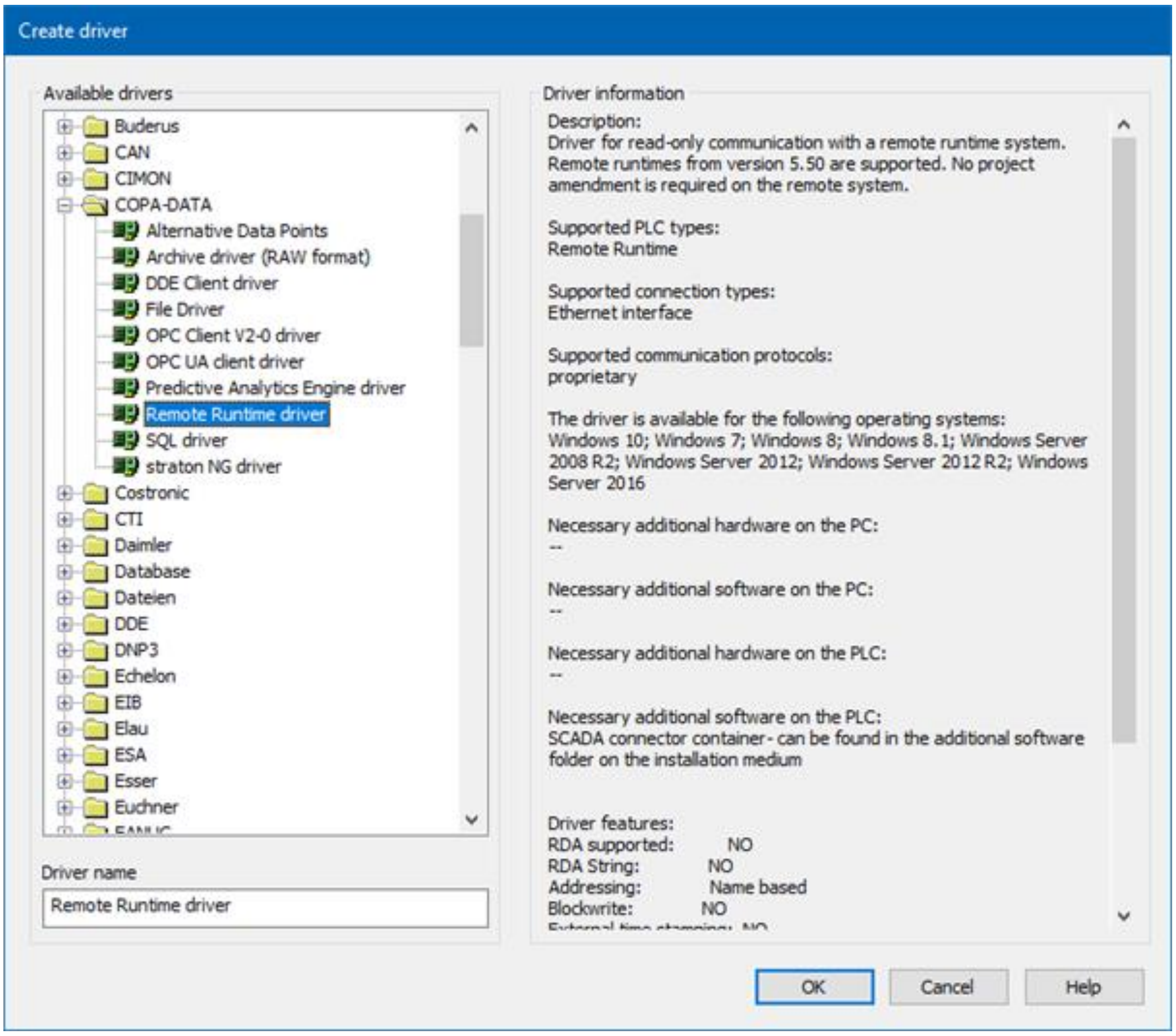
In this chapter you will learn how to use the driver in a project and which settings you can change.

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.



Available drivers

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

Driver name: Remote Runtime driver

Driver information

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

Supported PLC types:
Remote Runtime

Supported connection types:
Ethernet interface

Supported communication protocols:
proprietary

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

Necessary additional hardware on the PC:
--

Necessary additional software on the PC:
--

Necessary additional hardware on the PLC:
--

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

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

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

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

CLOSE DIALOG

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

Information

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

CREATE NEW DRIVER

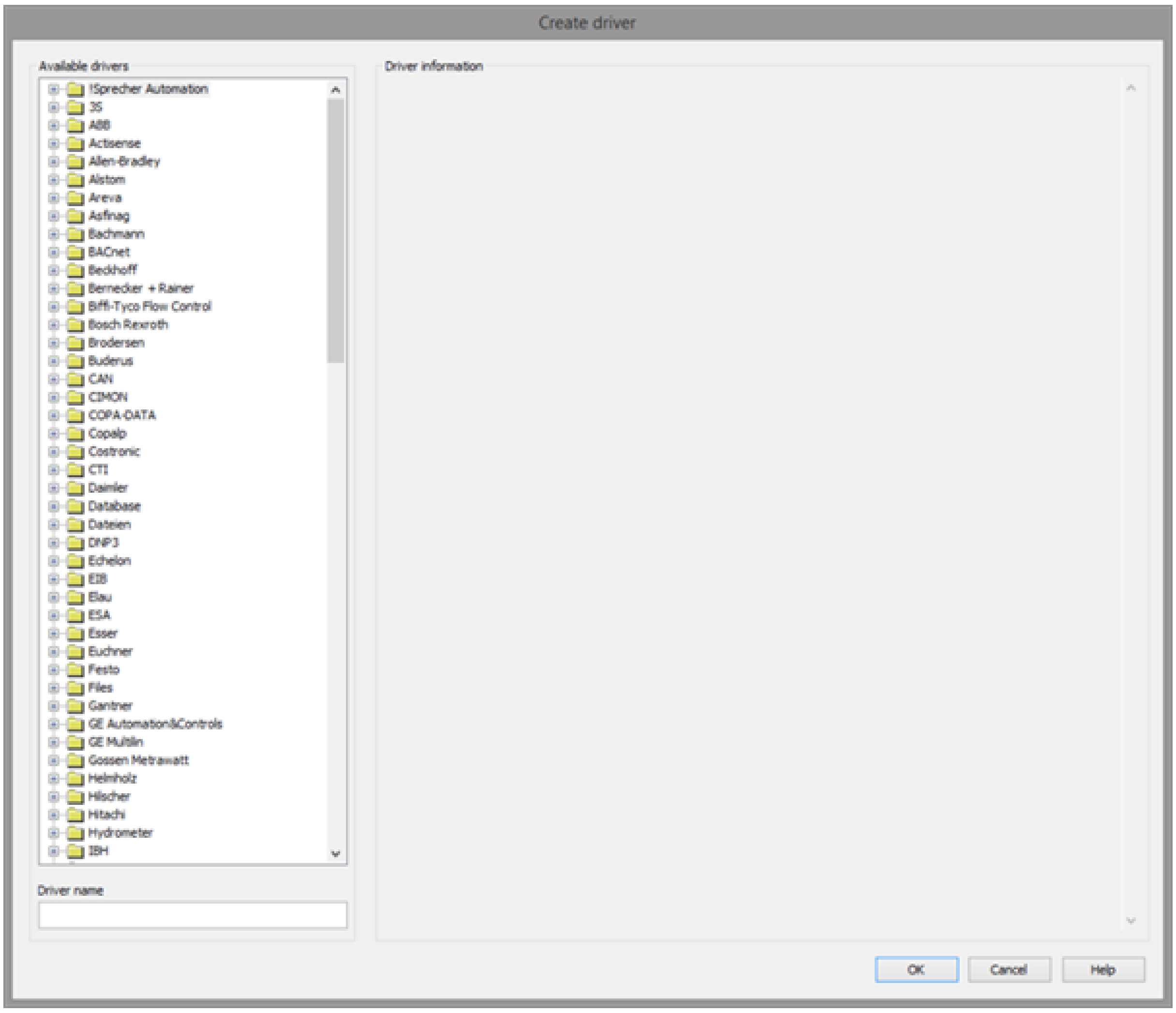
In order to create a new driver:

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

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

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

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


Create driver

Available drivers

- !Sprecher Automation
- 3S
- ABB
- Actisense
- Allen-Bradley
- Alstom
- Areva
- Asfinag
- Bachmann
- BACnet
- Beckhoff
- Bernecker + Rainer
- Biffi-Tyco Flow Control
- Bosch Rexroth
- Brodersen
- Buderus
- CAN
- CIMON
- COPA-DATA
- Copalp
- Costronic
- CTI
- Daimler
- Database
- Dateien
- DNP3
- Echelon
- EIB
- Elau
- ESA
- Esser
- Euchner
- Festo
- Files
- Gantner
- GE Automation&Controls
- GE Multilin
- Gossen Metrawatt
- Helmholz
- Hilscher
- Hitachi
- Hydrometer
- IBH

Driver name

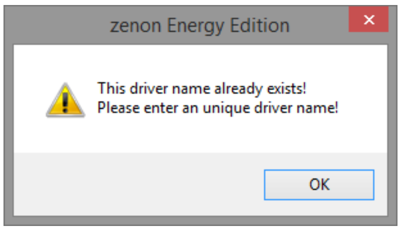
Driver information

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 Energy Edition

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

ZENON PROJECT

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

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

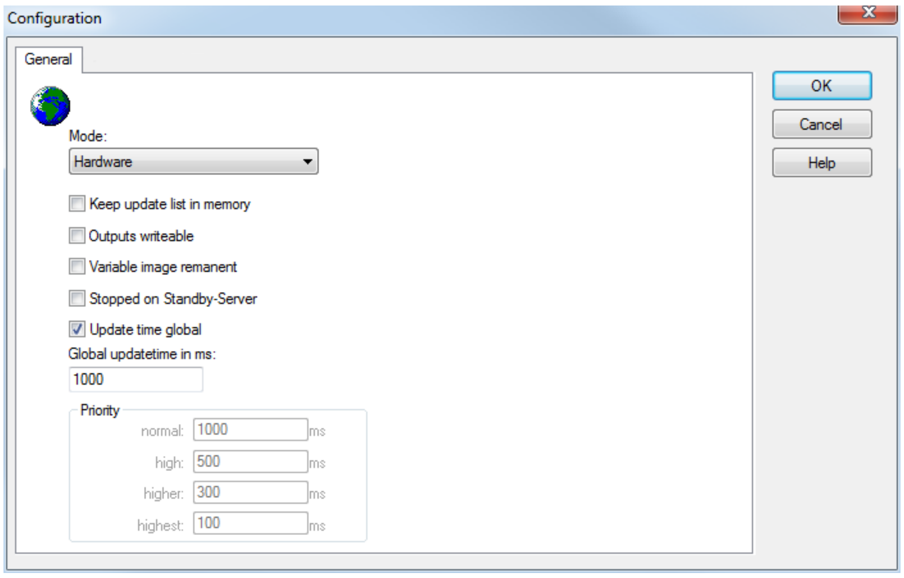
> **Information**
>
> Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

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


Configuration

General

Mode: Hardware

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

Global updatetime in ms: 1000

Priority

normal: 1000 ms

high: 500 ms

higher: 300 ms

highest: 100 ms

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

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

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

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

CLOSE DIALOG

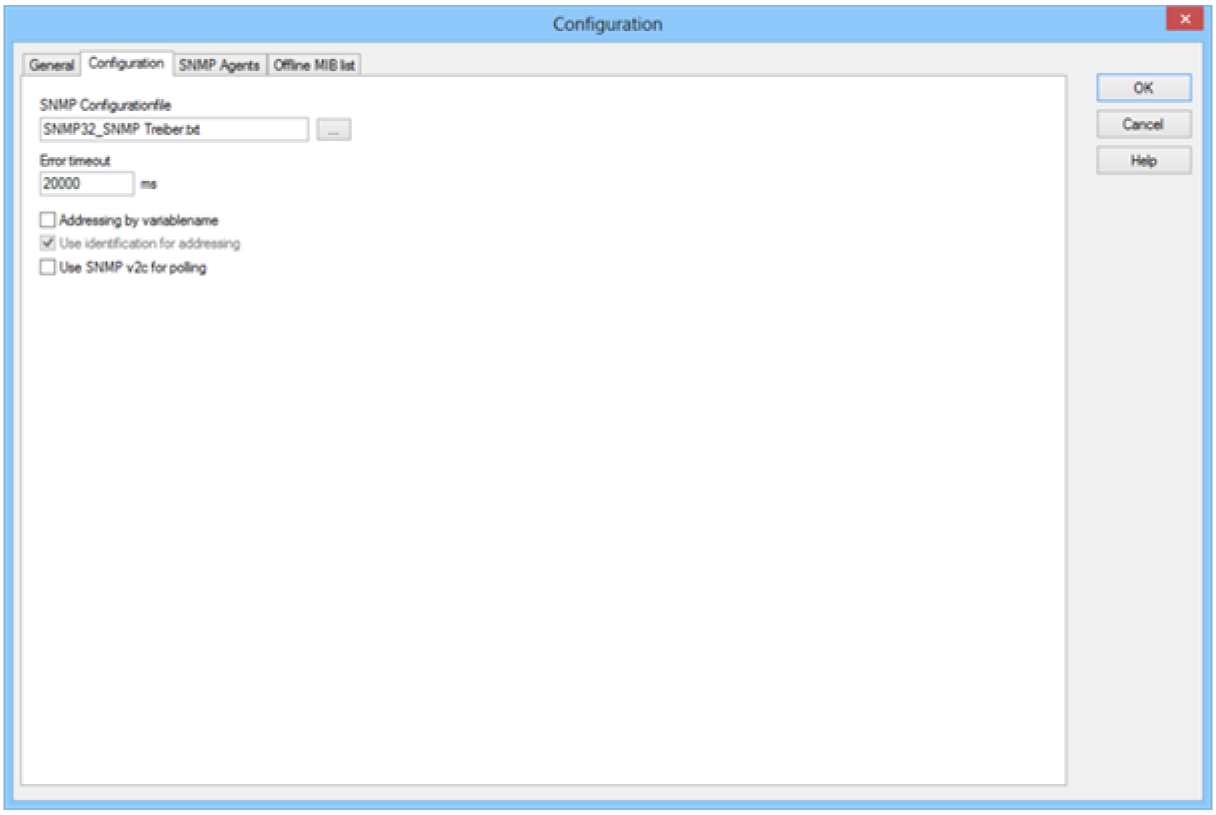
Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR SPONTANEOUS DRIVERS

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

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

6.2.2 Configuration

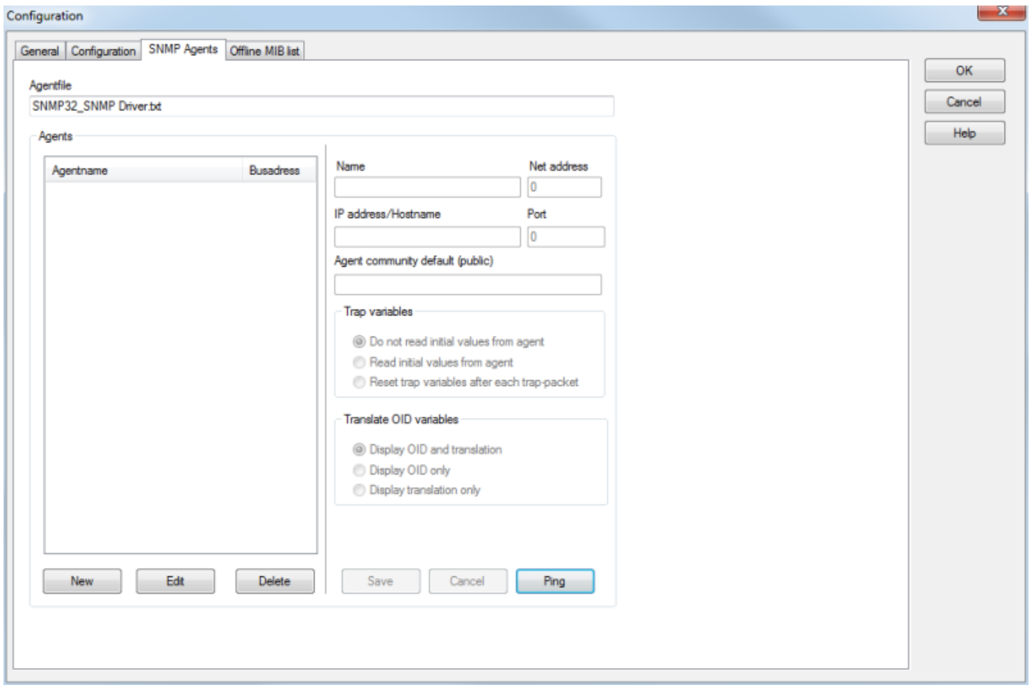


SNMP Configurationfile
SNMP32_SNMP Treiber.txt
Error timeout
20000 ms
☐ Addressing by variablename
☑ Use identification for addressing
☐ Use SNMP v2c for polling

Parameter	Description
SNMP configuration file	File in which the settings are saved. It is in the zenon project directory.
Error response time (ms)	The time in milliseconds, for which the driver waits, before it outputs an error message. I.e.: If the SNMP agents cannot be accessed, due to a short-term network overload for example, the driver waits this period of time before it marks the variables as invalid. If the driver reaches the agents within that time, the short-term fault is ignored without a message. **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.
Addressing using variable name or identification	*Active:* Addressing is carried out using the names of the variables or using identification. The **browse agent** and **receive online traps** properties are then

Parameter	Description
	not available. *Inactive:* Addressing is carried out by means of offset in assignment file. **Attention:** The type of addressing is important for variable import (on page 44). Subsequent amendments of addressing from **variable name** or**identification** to **offset** can lead to communication problems.
Using identification for addressing	Identification is used. Is not available if "**Addressing using variable name or identification**" was activated.
Use SNMP v2c for polling	*Active*: For all SNMP queries (GET, GETNEXT, SET) of this driver instance, **SNMP v2c** is used: ▸ For browsing the agents in the Offline Mib list ▸ In the Runtime *Inactive*: **SNMP v1** is used. **Note:** The driver instance can always receive **v1** and **v2c** TRAPs regardless of this setting. **Hint:** A single driver instance can be created for v2c and v1. However if you give each SNMP agent its own driver instance (**v1** or **v2c**), the waiting for the timeout of a non-functional agent does not have an effect on the communication with the other agents.
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



Agentfile: SNMP32_SNMP Driver.txt

Net address: 0

Port: 0

Trap variables: [✓ Do not read initial values from agent]

Translate OID variables: [✓ Display OID and translation]

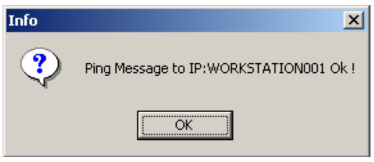
Parameter	Description
Agent file	The file where the configurations of the single SNMP agents are saved. This file complies with the SNMP configuration file and thus cannot be changed in this dialog.
Net address	The zenon internal Net address of the agents. Complies with the net address in the variable definition. You can use this net address e.g. for logical grouping of your variables.
Agent name	The freely definable name of a single SNMP agent.
IP address/host name	The IP address of the single network participants/agents. Here either the IP address or the computer name (requiring a properly working name resolution) can be used. If traps are used, no agents with the same IP address can be configured.
IP port	Port address Default: • SNMP: *161* • Traps: *162*
Agent community default (public)	The access identification of the agent. Is always "public" (read only) by default.

Parameter	Description
	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.
Trap Variables	Settings for traps.
Do not read any initial values from the agent	No initial value is read for traps. 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. 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*. The trap value can be recognized by the reaction matrix and written in the AML and CEL 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.

Parameter	Description
Translate OID variables	Settings for OID variables. Allows separate configuration of the translation for each agent. Note: Only available from version 6.50. All other versions are automatically handled with the standard **Only display OID**. **Requirements:** The following properties define how the OID translation is applied to ingoing values for variables from the SNMP driver. To do this, the following 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**. - Individually-installed MIBS must be in the following path: *%ProgramData%\COPA-DATA\zenon820\CommunicationProfiles\SNMP-MIBS* Incoming values that do not meet these conditions are forwarded without treatment.
Display OID and translation	*Active*: OID and translation are transferred as a value. The OID is translated in a written 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	*Active*: Only OID is transferred as a value. Note: Standard for all versions prior to zenon 6.50.
Only display translation	*Active*: Only the translation is transferred as a value. The OID is translated in a described 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.
New	Defining new agents
Delete	Delete agent.

Parameter	Description
Edit	Edit agent.
Save	Saves the agent settings that have been set.
Undo	Discards the agent settings that have been set.
Ping	Tests a configured **agent IP address**.

Result of a successful ping attempt with name resolution:



Info

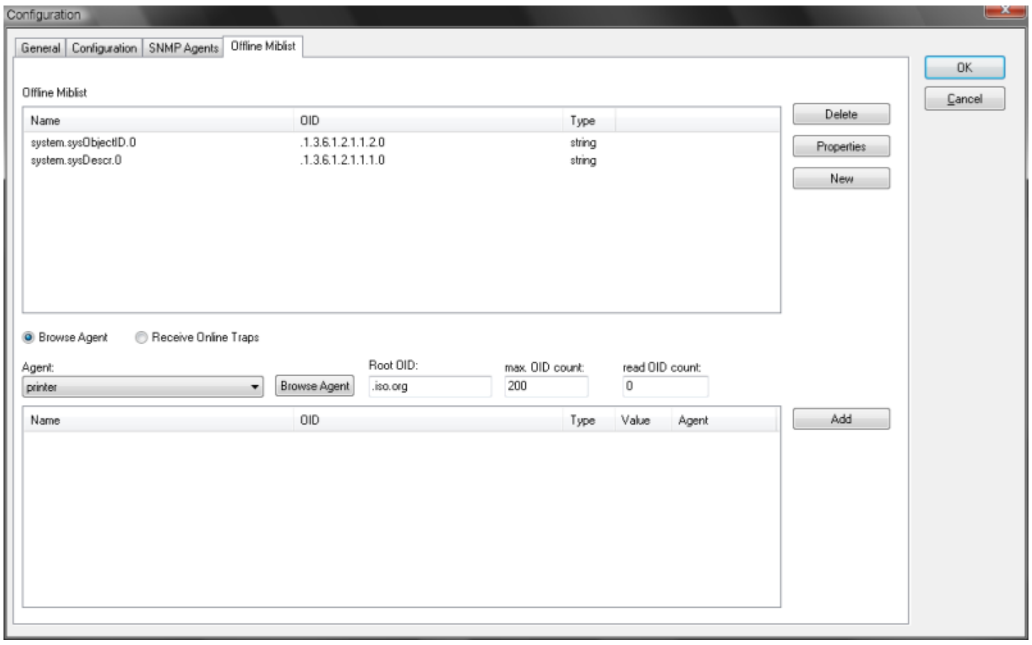
Ping Message to IP:WORKSTATION001 Ok !

If you get no positive response to a ping, it may be, that the name could not be resolved correctly. In this case, try pinging the IP address. If this fails again, this means the network participant / agent is not accessible.

Information

Maximum number of connections: 256 (0-255).

6.2.4 Offline MIB list



Configuration

General | Configuration | SNMP Agents | Offline Miblist

Offline Miblist

Name	OID	Type
system.sysObjectID.0	.1.3.6.1.2.1.1.2.0	string
system.sysDescr.0	.1.3.6.1.2.1.1.1.0	string

[✓ Browse Agent] Receive Online Traps

Agent: printer

Root OID: .iso.org

max. OID count: 200

read OID count: 0

Name	OID	Type	Value	Agent

Parameters	Description
Offline MIB list	Contains all properties (=SNMP objects) of an SNMP agent that should be available/selectable in the variable definition. SNMP objects that are not transferred to this list cannot subsequently be defined as variables.
Delete	Deletes selected OID from the list.
Properties	Displays the properties of a selected entry.
New	Allows manual creation of a new entry (on page 27).
Browse agent	***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 seconds are answered. A number that is too low can have an effect on the reading of the OIDs using the browse agent.**
Receive online traps	*Active:* The SNMP trap messages from different agents are received and displayed using the **SNMP Trap** service in Windows. To do this, the agent does not necesarily need to be configured; traps from non-configured agents are also listed. If the trap message comes from an agent that is already configured, the name of the agent is displayed in the "**Agent**" column. The following must be the case for the computer to receive traps: ‣ The firewall must allow UDP Traffic to port *161* and *162* ‣ The firewall must allow the **snmptrap.exe** process ‣ The configuration computer must be defined as the target for SNMP traps **Note:** If you receive an error mssage in relation to **0x2a** or **0x64**, it is possible that the **SNMP Trap** Windows service does not run.
Agent	Selecetion of the agent that is to be searched for supported information from the drop-down list with the command "GETNEXT" and the MIB objects of which should be displayed.

Parameters	Description
	This list contains all agents that were created in the **SNMP Agents** (on page 21) tab.
Browse agent	Searches through selected agent.
Root OID	The definition "**.iso.org**" or "**.1.3**" is to be used here.
Max. OID number	The number of OIDs that should be read by the browser.
Read OID number	The status display of the OIDs read until this point in time.

Add area	Adds selected OIDs to the offline MIB list. From here, OIDs of different agents can be created using driver online import.

Attention

The SNMP driver first looks for the agent and then the variable that belongs to the OID on the basis of the network address. That means:

- Only one variable can be defined for an OID per agent. If there are already several variables with the same OID for an agent, the variable that is updated when a trap arrives is not defined.
- Several variables with the same OID but different agents are permitted.

A similar restriction applies for the agents. For trap variables, the agent is assigned via the sender of the IP address. Several agents with the same IP address, or several drivers with agents with the same IP address are not supported. The agent or driver that receives the trap value is not defined.

DIFFERENCES BETWEEN MIB EDITOR AND ONLINE VARIABLE IMPORT

If variables are created offline using the MIB editor (on page 24) or using online variable import (on page 44), there are some differences:

VARIABLES AND AGENT

- If traps are received for the **offline MIB list**, traps of all IP addresses are displayed. When adding received traps to the **offline MIB list**, information to the agent is not saved.

- If variables are imported by the driver, ensure that the correct variables are created for the correct agent as a trap.

CONFIGURED AND UNCONFIGURED AGENTS IN ONLINE IMPORT

If traps are received via the import dialog, SMNP traps of all other IP addresses are displayed.

- If a trap comes from an agent that is laready configured, the agent is amended accordingly when the trap is added: The Net address is set correctly.
- If the trap comes from an unconfigured agent (source IP address is displayed in the list of traps received) and the variale is added, the net address *0* is used. However, this net addres could also be in use by another agent, which would lead to problems.

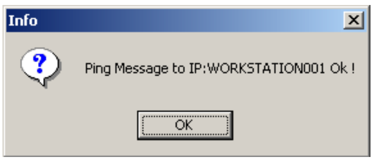
ADDRESSING

When browsing an agent or when receiving traps in the driver online import, variables are not added to the **Open MIB list** and do not receive an offset.

If the type of addressing in the driver configuration (on page 19)is changed from **variable name** or **Identification** to **offset**, these variables will no longer communicate with each other. You should therefore select the type of communiction before you create variables.

6.2.4.1 Creating the OID manually

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


Info

Ping Message to IP:WORKSTATION001 Ok !

Parameter	Description
Name	Here a freely definable name can be used.
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 otherwise.
Datatype	Integer or String, depending on the accessed SNMP datatype.

6.2.4.2 Translating OID variables

OID translation is available for variables of the driver object types:

- SNMP trap
- SNMP variable

The function of OID translation is only available for drivers for zenon 6.50 and later versions. The list of all contents of a trap is supported in each driver version

Information

Notes on copyright

The function of OID translation was taken from the Net SNMP Open Source Library, version 5.6.1.1.

Copyright 1989, 1991, 1992 by Carnegie Mellon University. Derivative Work - 1996, 1998-2000.

Copyright 1996, 1998-2000 The Regents of the University of California. All Rights Reserved.

CONFIGURATION

Overview of necessary configurations and the results of this:

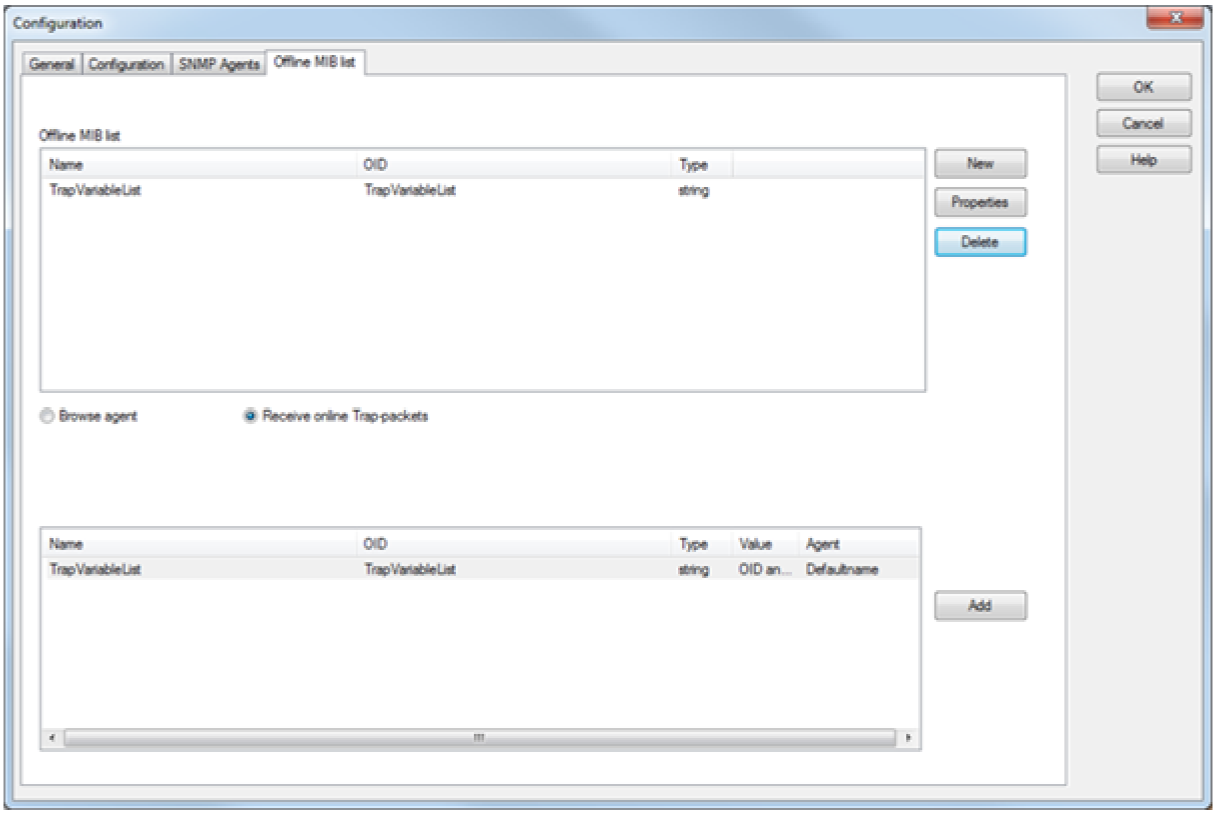
1. **Driver configuration:**
   a) An **SNMP driver** is created in the Editor.
   a) The required agents are configured (on page 21) on the driver.
   b) The desired OID translation is set for the agents.
   c) In the Offline MIB list (on page 24), the agents are browsed and the desired OIDs are added to the offline MIB list.
   d) After this, **Receive online traps** is switched, to insert the list of all contents of the traps into the offline MIB list. After traps have been received by the agents, the necessary OIDs from the traps are added to the list. The driver configuration is thus concluded.
2. **Configuration in the Editor:**
   a) The variables from the driver are imported in the Editor, whereby the required variables from the offline MIB list are added for each agent.
   b) The variables are linked to screens with elements so that they can be looked at when the program is running.
   c) The Runtime files are created and Runtime is started.

3. **Behavior in the Runtime:**
   a) Runtime starts the SNMP driver.
   b) The SNMP driver instance initializes the OID translation.
   c) The first values are read after the variables are registered on the driver:
      For all all variables of the driver-object type **Ping status**, **SNMP Counter**, **SNMP Variable** and **SNMP Traps** if **Read initial values for traps** is active.
   d) For variables of the driver object types **SNMP Variable** and **SNMP Trap**, received values of SNMP data type **OID** are handled in accordance with the configured OID translation for the respective agents.
   e) For each further polling and each trap received, the OID translation for values of SNMP data type **OID** are used in accordance with the respective agent configuration.
      Furthermore, for traps, the list of all content is created.

6.2.4.3 String variable with all trap contents

The list of all content of a trap is displayed in the dialog of the offline MIB list if this is set to receive traps.

A click on **Receive online traps** creates an example entry for each configured agent, which is displayed in the list of received traps. For each trap received, a list of all content like this is created and inserted as the last received entry for this trap.



Offline MIB list

Name	OID	Type
TrapVariableList	TrapVariableList	string

( ) Browse agent [✓ Receive online Trap-packets]

Name	OID	Type	Value	Agent
TrapVariableList	TrapVariableList	string	OID an...	Defaultname

Received variables of SNMP data type OID are always displayed as **OID and translation**. The descriptive text translated from the OID is displayed as a name.

The object can be inserted in the **Offline MIB list** in the list for **Receive online trap**. When importing driver variables from the **Offline MIB list**, a string variable with all trap contents can be stored for each agent. This must be created for the SNMP driver with the following parameters:

- Driver object type: **SNMP trap**
- Net address: Index of the agent that it concerns
- Identification: **TrapVariableList**
- Offset

Hint: Add the example entry of the agent into the offline   MIB list and then import the variable from the driver.

FORMAT OF THE STRING VARIABLES

The string is in the same format as an INI file format:

The first node **[DEFAULT]** contains an entry **COUNT**, which provides the number of variable bindings in a trap. After this, there is an **ENTRY_[Nr]** node for each variable binding with the contents **OID**, **OID_TRANSLATED**, **TYPE** and **VALUE**.

SCHEMA

Parameter	Description
[DEFAULT]	
COUNT=x	Number of entries.
[ENTRY_x]	Index
OID=*x*	OID in numerical format.
OID_TRANSLATED=*x*	Descriptive text for the OID, stating if the OID translation is available.
[TYPE]=*x*	zenon data type of the value
[VALUE]=*x*	Received value. Received OID values are entered here according to the availability and configuration of the OID translation.

EXAMPLE

[DEFAULT]

COUNT=*7*

[ENTRY_000]

OID=*.1.3.6.1.2.1.1.3.0*

OID_TRANSLATED=*DISMAN-EVENT-MIB::sysUpTimeInstance*

TYPE=*u32*

VALUE=*59769454*

[ENTRY_001]

OID=*.1.3.6.10.60.30.10.10.40.10.0*

OID_TRANSLATED=*SNMPv2-MIB::snmpTrapOID.0*

TYPE=*string*

VALUE=*.1.3.6.1.6.3.1.1.5.3 (IF-MIB::linkDown)*

[ENTRY_002]

OID=*.1.3.6.1.2.1.2.2.10.1.6*

OID_TRANSLATED=*IF-MIB::ifIndex.6*

TYPE=*i32*

VALUE=*6*

[ENTRY_003]

OID=*.1.3.6.1.2.1.2.2.10.7.6*

OID_TRANSLATED=*IF-MIB::ifAdminStatus.6*

TYPE=*i32*

VALUE=*1*

[ENTRY_004]

OID=*.1.3.6.1.2.1.2.2.1.8.6*

OID_TRANSLATED=*IF-MIB::ifOperStatus.6*

TYPE=*i32*

VALUE=*2*

[ENTRY_005]

OID=*.1.3.6.1.3.1057.1*

OID_TRANSLATED=*SNMPv2-SMI::experimental.1057.1*

TYPE=*string*

VALUE=*192.168.0.120*

[ENTRY_006]

OID=*.1.3.6.10.60.30.10.10.40.30.0*

OID_TRANSLATED=*SNMPv2-MIB::snmpTrapEnterprise.0*

TYPE=*string*

VALUE=*.1.3.6.1.6.3.1.1.5 (SNMPv2-MIB::snmpTraps)*

6.2.5 Example configuration

To receive and display translated OID variables and lists of content of traps from an agent:

1. Configure the SNMP agent

2. Create an SNMP driver
3. Configure the SNMP agents in the driver
4. Add the required OIDs to the offline MIB list
5. Import variables from the driver
6. Display the variables in a screen

CONFIGURING THE SNMP AGENT

The agent must grant the **public** standard SNMP community reading rights at least.

The IP address of the subsequent computer must be entered as the destination fro traps. If the engineering station and Runtime computer are different computers, the IP address of the engineering station can also be entered here. If the agent supports this, you can also receive traps from agents at the time of design. If the agent does not support multiple trap destinations, the IP address of the engineering station can be entered as an individual trap destination. However this entry must be replaced with the IP address of the Runtime computer before Runtime is started, so that it receives the traps.

CREATING AN SNMP DRIVER

In order to create a new driver:

1. Open the dialog to create a new driver using the **New driver** command:
   - in the context menu in the **Driver** project node
     or
   - Context menu in the driver list
     or
   - Toolbar in the driver view
2. Select the **SNMP** driver in the dialog

CONFIGURING THE SNMP AGENTS IN THE DRIVER

In the driver configuration, you configure the settings for the agents in the SNMP Agents (on page 21) tab. In doing so, please note:

- Change the port accordingly: The port for SNMP requests is 161 and 162 for traps.
- The standard community **public** must be permitted.
- The IP address of the SNMP agent must be entered as the IP address.
- You control the reading of initial values for trap variables and whether these are treated as wipers using the options in the **Trap variables** configuration area.

- You control the OID translation using the options in the OID variables configuration area. Here, one of the variants to be translated is selected (everything except "Only display OID").

ADD REQUIRED OIDS TO OFFLINE MIB LIST

Because only traps are of interest in this article, activate the **Receive online traps:** option in the Offline MIB list (on page 24) tab

- The **TrapVariableList** automatically created for the agent is added to the offline MIB list.
- The required OIDs are added to the list after the trap has been received.

Close the driver configuration dialog.

IMPORT VARIABLES FROM THE DRIVER

Open the context menu of the SNMP driver in the driver list and select the **Import variables from driver** entry.

In the dialog (on page 44) that opens:

- Select the entries in the offline MIB list
- Set the agents to be used
- Import the variables by clicking on OK in the variable list

DISPLAYING VARIABLES IN A SCREEN

You display the selected variables in a dynamic numerical value or in a dynamic text, regardless of their data type. A large element should be configured for the list of all contents of a trap, because these can become very long.

A ping status variable for the agent can be created as an option and linked via a switch, in order to display the availability of the agent.

BEHAVIOR IN THE RUNTIME

Runtime starts the SNMP driver after it has been started. The following steps are carried out by Runtime once the driver instance has been created:

1. The driver runs through its initialization.
   In doing so, the OID translation is also initialized and the index for the MIBs is created.
2. Runtime registers the required variables on the driver.
   Based on the network address, the driver decides which variables are assigned to which agents and reads the configuration of the agent from the driver configuration file.

3. For polling variables, the first values are read and the polling cycle is started.
   Received **OID** SNMP data values are translated according to the agent configuration.
4. Record traps.
   The following happens for incoming traps:
   - First the individual variable bindings are processed
   - OIDs translated
   - Values sent to the Runtime
   - and recorded in a list.

   Based on this list, the string for the variable with the list of all content of a trap is generated and sent to Runtime.
5. When Runtime is ended, the driver is sent a command to end.
6. The driver runs through its deinitialization and deletes the index that was created in the process.

7 Creating variables

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

7.1 Creating variables in the Editor

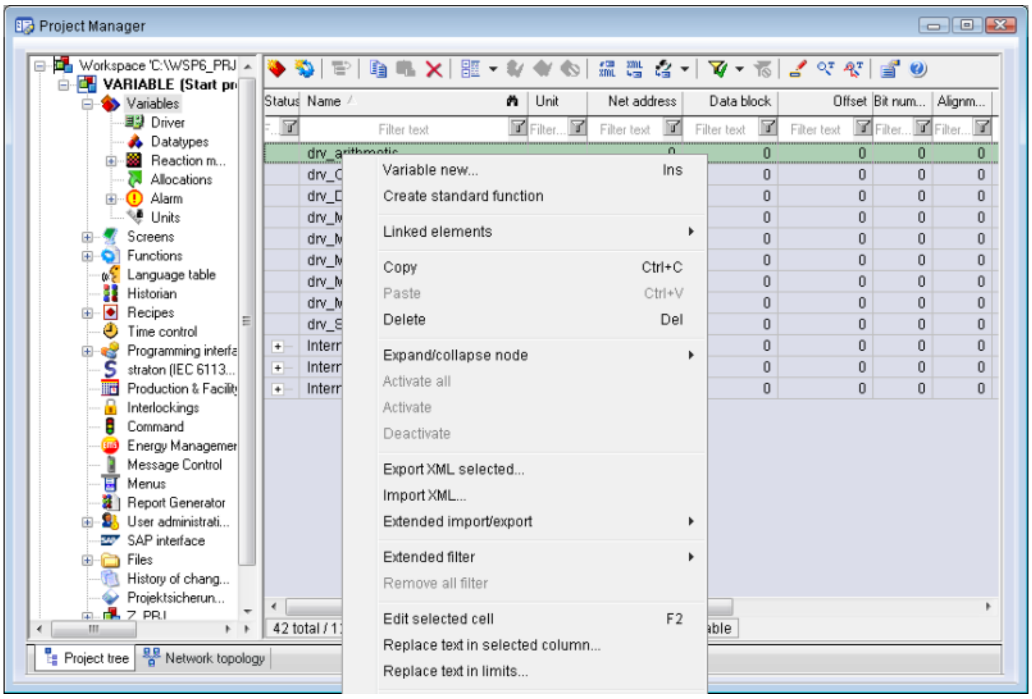
Variables can be created:

- as simple variables
- in arrays
- as structure variables

VARIABLE DIALOG

To create a new variable, regardless of which type:

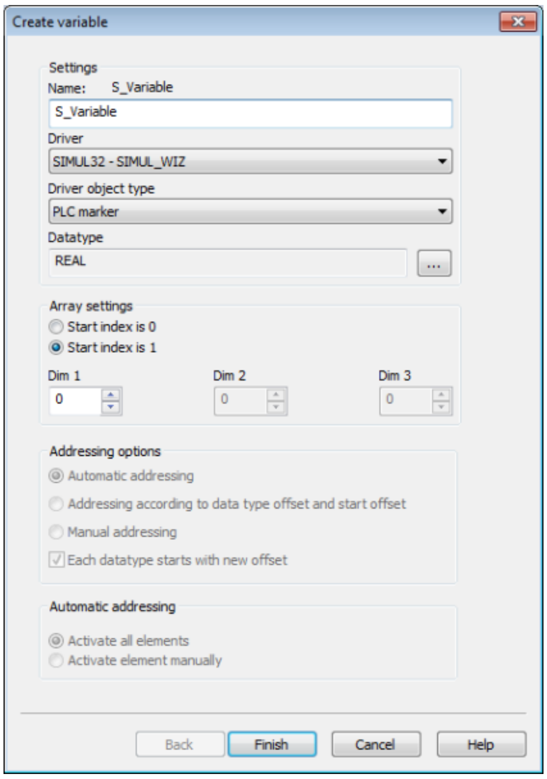
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

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

CREATE VARIABLE DIALOG



Create variable

Settings

Name: S_Variable

S_Variable

Driver

SIMUL32 - SIMUL_WIZ

Driver object type

PLC marker

Datatype

REAL

Array settings

Start index is 0 [✓ Start index is 1]

Dim 1	Dim 2	Dim 3
0	0	0

Addressing options

[✓ Automatic addressing]

Addressing according to data type offset and start offset

Manual addressing

[✓ Each datatype starts with new offset]

Automatic addressing

[✓ Activate all elements]

Activate element manually

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

Property	Description
Name	Freely definable name. **Attention:** For every zenon project the name must be unambiguous.
Identification	Freely definable identification. E.g. for Resources label, comments, ... If the variable identification is used for addressing, the OID must be entered here.
Net address	Network address of variables. This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.
Data block	not used for this driver
Offset	Offset of variables. Equal to the memory address of the variable in the PLC. Adjustable from *0* to *4294967295*.
Alignment	not used for this driver
Bit number	Number of the bit within the configured offset. Possible entries: *0* to *65535*.
String length	Only available for String variables. Maximum number of characters that the variable can take.
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.
Driver connection/Priorit	Setting the priority class. The variable of the priority class is thus assigned as it was configured in the driver dialog in the **General** tab. The priority

Property	Description
y	classes are only used if the **global update time** is deactivated. If the **global update time** option is activated and the priority classes are used, there is an error entry in the log file of the system. The driver uses the highest possible priority.

When importing driver variables from the **Offline MIB list**, a string variable with all trap contents can be stored for each agent. This must be created for the SNMP driver. For details, see the String variable with all trap contents (on page 29).

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:

OBJECTS FOR PROCESS VARIABLES IN ZENON

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
Ping status	64	X	X	*BOOL*	
SNMP traps	67	X	X	*DINT, UDINT, STRING*	V1 and V2 traps are supported.
SNMP variables	65	X	X	*DINT, UDINT, STRING*	
SNMP counter	66	X	X	*DINT, UDINT*	
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).

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
					Note: The addressing and the behavior is the same for most zenon drivers. You can find detailed information on this in the Communication details (Driver variables) (on page 55) chapter.

Object	Read	Write	Comment
Configuration			Opens the driver configuration menu
Ping status	X	--	Bit.
+i/u32Bit	X	--	
String	X	--	

Key:

X: supported

--: not supported

Attention

The driver resets the value for TrapVariableList non-automatically. The values of other variables are reset automatically.

HOW THE SNMP COUNTER WORKS

A counter calculates the average of the value change over time (in seconds) between two read cycles and sends this to Runtime. The value from the OID is not sent to Runtime however. If the new value from the OID is les than the previous value (overflow), this value is ignored and no new value for the counter is sent to Runtime until a new value can be calculated.

Network traffic can be measured using this counter: To do this, the agent must continuously count up the number of bytes, for example. With a read cycle of 1000 ms, you receive the number of bytes per second.

EXAMPLE

- Driver update cycle: *30 s*
- Value 1: *2500*
- Value 2: *7500*
- Result for counter variable: *166*

  Because: 5000 value difference over 30 seconds = *166*

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
	LINT	26

PLC	zenon	Data type
	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 Import variables from the driver

To import variables from the driver:

1. Select **Import variables from driver** command from the driver context menu
2. The configuration dialog is opened

   The dialog options depend on the settings in the configuration (on page 19)

   a) Addressing via offset

b) Addressing using variable name or identification

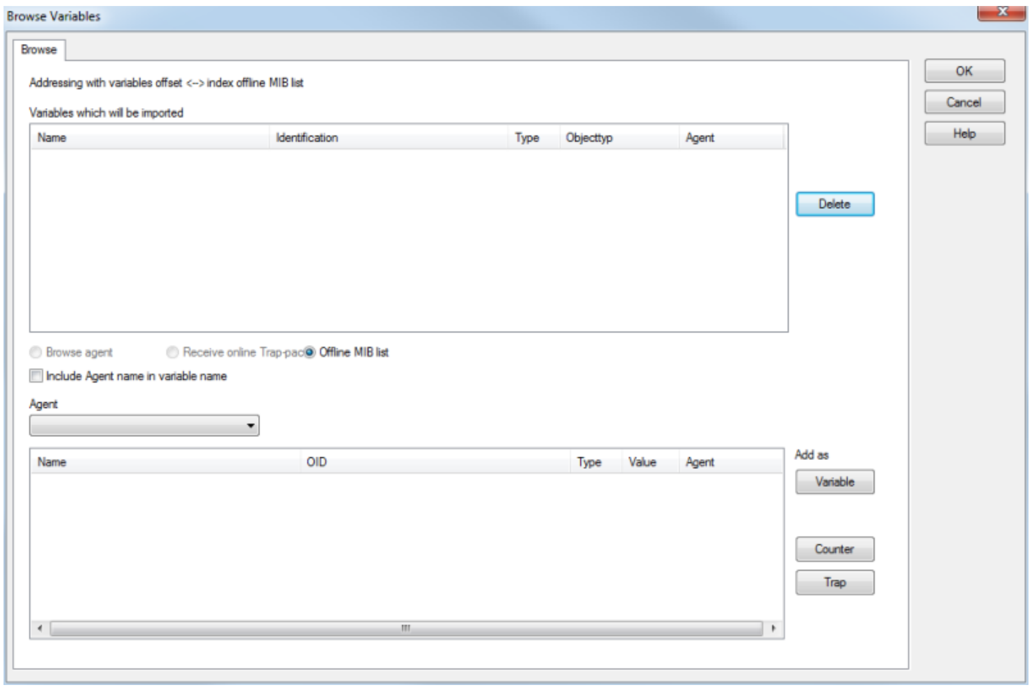
Attention

The SNMP driver first looks for the agent and then the variable that belongs to the OID on the basis of the network address. That means:

- Only one variable can be defined for an OID per agent. If there are already several variables with the same OID for an agent, the variable that is updated when a trap arrives is not defined.
- Several variables with the same OID but different agents are permitted.

A similar restriction applies for the agents. For trap variables, the agent is assigned via the sender of the IP address. Several agents with the same IP address, or several drivers with agents with the same IP address are not supported. The agent or driver that receives the trap value is not defined.

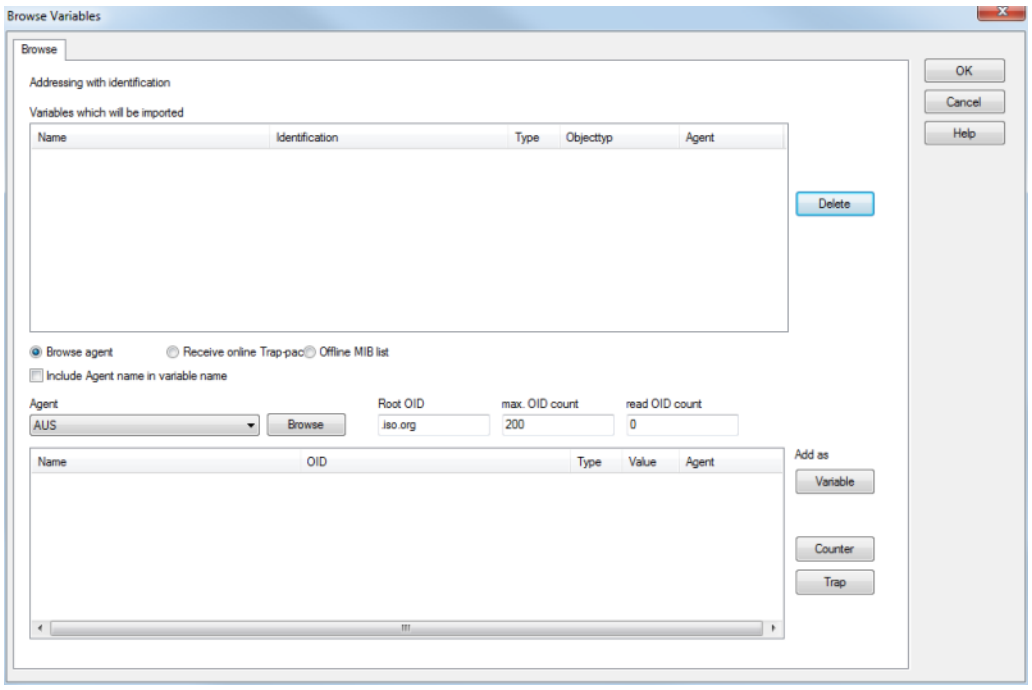
ADDRESSING VIA OFFSET



Parameters	Description
Addressing using variable offset <--> Offline MIB list index	Note the addressing defined under configuration (on page 19).
zenon Variable for creation	List of the variables to be created in zenon.

Parameters	Description
Delete	Removes entries from the list.
Browse Agent	Cannot be used when addressing via offset.
Receive online traps	Cannot be used when addressing via offset.
Offline MIB list	Active: Variables are imported from the offline MIB list (on page 24). List entries are displayed in the list field below.
Include agent name in the variable name.	*Active:* The name of the agent is placed in front of the variable name; both entries are separated by an underscore: **Agent_Variable Name**.
Agent	Drop-down list of the agents.
List	**List of variables that are contained in the offline MIB list.**
Add as:	Selection using buttons, of how variables to be imported are to be added to the "zenon variables to create" list:
▸ **Variable**	Add as variable.
▸ **Counter**	Add as counter.
▸ **Trap**	Add as trap.
OK	Accepts all settings and closes the dialog. The variables in the "zenon variables to create" list are added.
Cancel	All settings are discarded and the dialog is closed.
Help	Opens online help.

ADDRESSING USING VARIABLE NAME OR IDENTIFICATION



(●) Browse agent

Agent: AUS

Root OID: .iso.org

max. OID count: 200

read OID count: 0

TAGs	Description
Addressing using variable offset <--> Offline MIB list index	Note the addressing defined under configuration (on page 19).
zenon Variable for creation	List of the variables to be created in zenon.
Delete	Removes entries from the list.
Browse agent	***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 seconds are answered. A number that is too low can have an effect on the reading of the OIDs using the browse agent.**
Receive online traps	*Active:* The SNMP trap messages from different agents are received and displayed using the **SNMP Trap** service in Windows. To do this, the agent does not necesarily need to be configured; traps from non-configured agents are also listed. If the trap message comes from an agent that is already configured, the name of the agent is displayed in the "**Agent**" column.

TAGs	Description
	The following must be the case for the computer to receive traps: ▸ The firewall must allow UDP Traffic to port *161* and *162* ▸ The firewall must allow the **snmptrap.exe** process ▸ The configuration computer must be defined as the target for SNMP traps **Note:** If you receive an error mssage in relation to **0x2a** or **0x64**, it is possible that the **SNMP Trap** Windows service does not run.
Agent	Selecetion of the agent that is to be searched for supported information from the drop-down list with the command "GETNEXT" and the MIB objects of which should be displayed. This list contains all agents that were created in the **SNMP Agents** (on page 21) tab.
Browse agent	Searches through selected agent.
Root OID	The definition "**.iso.org**" or "**.1.3**" is to be used here.
Max. OID number	The number of OIDs that should be read by the browser.
Read OID number	The status display of the OIDs read until this point in time.
List	**List of variables that are contained in the offline MIB list.**
Add as:	Selection using buttons, of how variables to be imported are to be added to the "zenon variables to create" list:
▸ **Variable**	Add as variable.
▸ **Counter**	Add as counter.
▸ **Trap**	Add as trap.
OK	Accepts all settings and closes the dialog. The variables in the "zenon variables to create" list are added.
Cancel	All settings are discarded and the dialog is closed.
Help	Opens online help.

DIFFERENCES BETWEEN MIB EDITOR AND ONLINE VARIABLE IMPORT

If variables are created offline using the MIB editor (on page 24) or using online variable import (on page 44), there are some differences:

VARIABLES AND AGENT

- If traps are received for the **offline MIB list**, traps of all IP addresses are displayed. When adding received traps to the **offline MIB list**, information to the agent is not saved.
- If variables are imported by the driver, ensure that the correct variables are created for the correct agent as a trap.

CONFIGURED AND UNCONFIGURED AGENTS IN ONLINE IMPORT

If traps are received via the import dialog, SMNP traps of all other IP addresses are displayed.

- If a trap comes from an agent that is laready configured, the agent is amended accordingly when the trap is added: The Net address is set correctly.
- If the trap comes from an unconfigured agent (source IP address is displayed in the list of traps received) and the variale is added, the net address *0* is used. However, this net addres could also be in use by another agent, which would lead to problems.

ADDRESSING

When browsing an agent or when receiving traps in the driver online import, variables are not added to the **Open MIB list** and do not receive an offset.

If the type of addressing in the driver configuration (on page 19)is changed from **variable name** or **Identification** to **offset**, these variables will no longer communicate with each other. You should therefore select the type of communiction before you create variables.

7.4.3 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 data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Nu	3	Net address

Identification	Typ e	Field size	Comment
	m		
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	Floa t	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)

Identification	Typ e	Field size	Comment
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.
MAXGRAD	N	16	Gradient overflow for counter REMA.

Attention

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

LIMIT VALUE DEFINITION

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

Identification	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	technical value or ID number of a linked variable for a dynamic limit value (see VARIABLEx) (if VARIABLEx is *1* and here it is *-1*, the existing variable linkage is not overwritten)
SCHWWERT1	F	16	Threshold value for limit value
HYSTERESE1	F	14	Is not used
BLINKEN1	R	1	Set blink attribute
BTB1	R	1	Logging in CEL
ALARM1	R	1	Alarm
DRUCKEN1	R	1	Printer output (for CEL or Alarm)
QUITTIER1	R	1	Must be acknowledged
LOESCHE1	R	1	Must be deleted
VARIABLE1	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	R	1	Functions linking
ASK_FUNC1	R	1	Execution via Alarm Message List
FUNC_NR1	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/Event Group
A_KLASSE1	N	10	Alarm/Event Class

Identification	Type	Field size	Comment
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.

7.5 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
LineStateConferenced	*BOOL*	24.11	Connection in conference mode.
LineStateOnHoldPendConf	*BOOL*	24.12	Connection in hold for conference
LineStateOnHoldPendTransfe r	*BOOL*	24.13	Connection in hold for transfer
LineStateDisconnected	*BOOL*	24.14	Connection terminated.
LineStateUnknow	*BOOL*	24.15	Connection status unknown
ModemStatus	*UDINT*	24	Current modem status

Name from import	Type	Offset	Description
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).

Name from import	Type	Offset	Description
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.
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

Name from import	Type	Offset	Description
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 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).
MaxUpdateTimeHigh	*UDINT*	57	Time since the last update of the priority group

Name from import	Type	Offset	Description
er			**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.
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.

For this status bit INVALID is requested via the combined element of via reaction matrices.

Value *1*: Device can be reached.

Value with status INVALID: Communication is disturbed.

SNMP TRAP

The driver supports the receipt of SNMP traps with an SNMPV1 and SNMPv2c header.

INFORMS (traps with confirmation of receipt) are not supported.

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

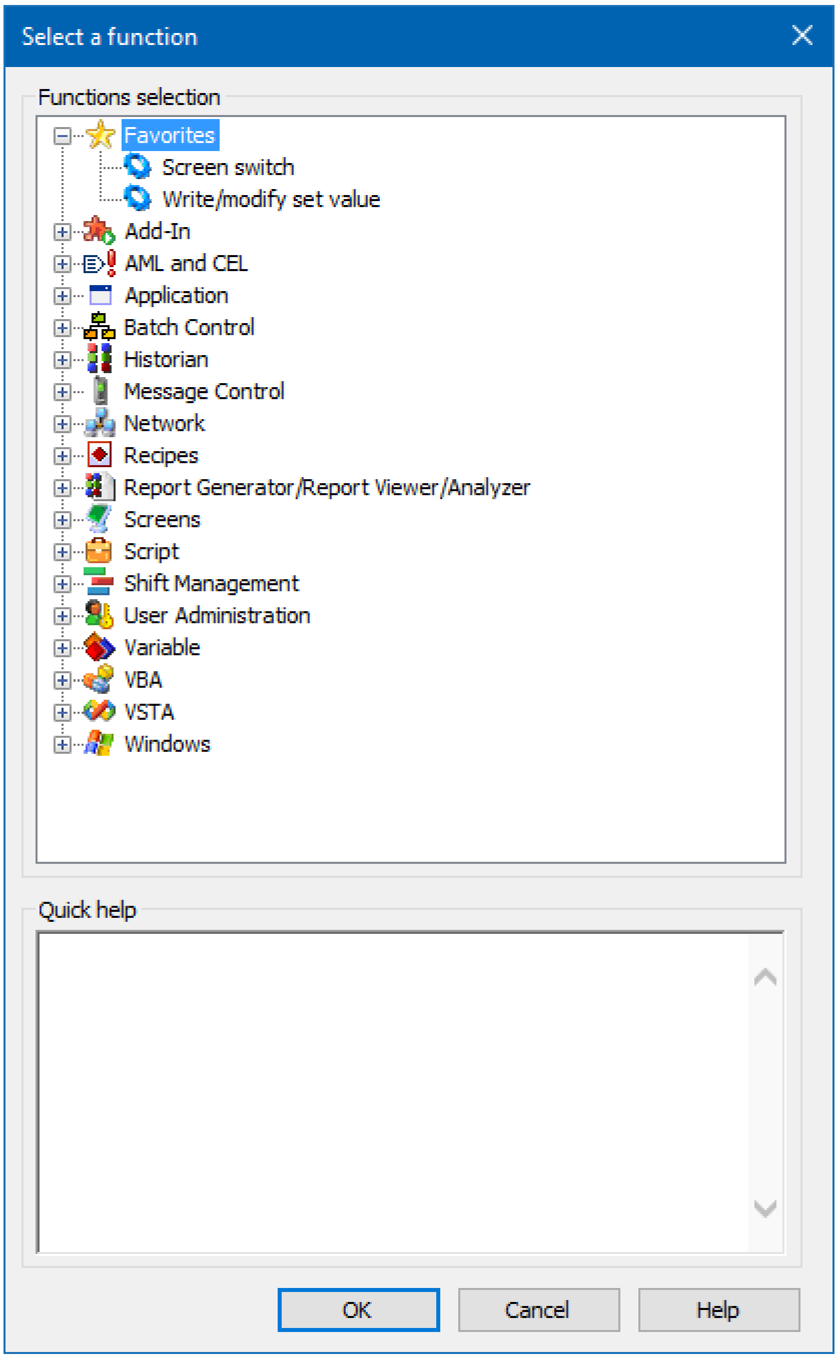
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

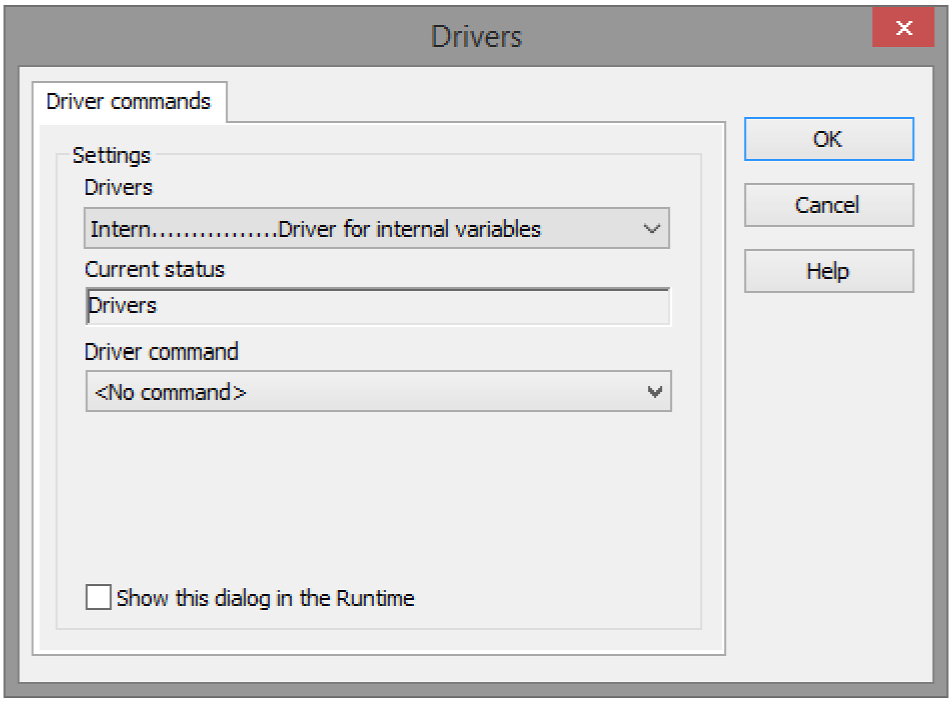
1. Create a new function in the zenon Editor.

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



Settings

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

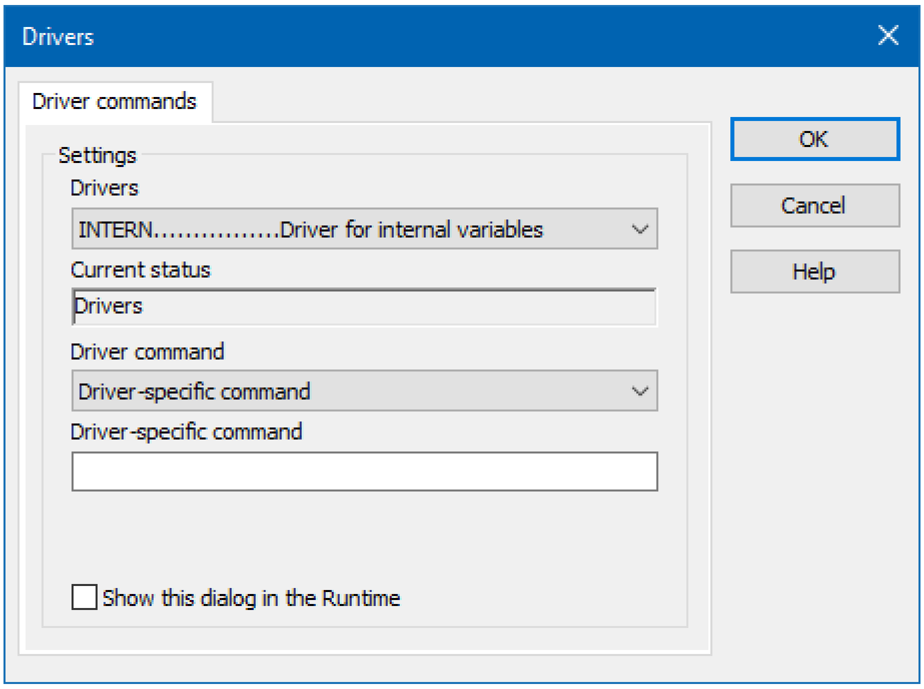
Current status: Drivers

Driver command: <No command>

☐ Show this dialog in the Runtime

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

DRIVER COMMAND DIALOG



Settings

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

Current status: Drivers

Driver command: Driver-specific command

Driver-specific command:

☐ Show this dialog in the Runtime

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

CLOSE DIALOG

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

AVAILABLE DRIVER COMMANDS

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

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

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

DRIVER COMMAND FUNCTION IN THE NETWORK

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

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

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

10 Error analysis

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

10.1 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 Error treatment

ERROR MESSAGE

Error text	Description
Error on SnmpMgrRequest GET oid=33.1.3.1.0, specified. Busadress=0 Error=0x28 Error=0x	Error code as hexadecimal number à 0x28 = 40 Dec
GET Error: errorStatus=2, errorIndex=1 oid=.1.3.6.1.4.1.171.20.1.2.1.1.1.7.0, specified. Busadress=0 Error=0x0	Variable name or identification in zenon is not valid. (there is no OID in the MIB on the agent) Other possible causes: Initial values for traps should be read off by the agent, but the agent does not make any OIDs available on the GET, but only sends these OIDs spontaneously as traps. **errorStatus=2**: = *no such item*
error Trap Start 0x64!	Error message in the Editor. Windows trap service

Error text	Description
	was not started. **Note:** occurs after **Receive online traps** was activated in the driver configuraiton (on page 24) or in the import assistant.
Init TRAP Error 0x64	Message from the Diagnosis Server. No traps are received. Cause: the "**SNMP Trap**" Windows service was not started. **Note:** The "**SNMP Trap**" service (**snmptrap.exe**) requires the UDP ports *161* and *162*. Theses must be enabled in the firewall. If the service has been started, but another application is occupyin port *162*, then: ▶ The driver cannot receive any traps ▶ No error message is displayed.

ENTRIES IN LOG FILE

Unexpected Error during OID Translation INIT. Tanslation of OIDs will not be available.	**An error occured when initializing the OID translation. The OID translaton is deactivated for this driver instance.**
Unexpected Error during translation of OID [numerische OID]	An unexpected error occured when translating the given OID.
Could not translate OID [numerische OID]	The given OID could not be translated. It is possible that it cannot be found in the standard MIB tree.
OID Translation successfully initialized.	The OID translation was initiazed successfully.
Translation of OID [numerische OID] successfull	The given OID was translated successully.

ERROR NUMBERS

Error numbers for **SnmpMgrRequest**:

ID	Description	Description
0	SNMPAPI_FAILURE	*/* Generic error code */*
1	SNMPAPI_SUCCESS	*/* Generic success code */*

ID	Description	Description
2	SNMPAPI_ALLOC_ERROR	*/* Error allocating memory */*
3	SNMPAPI_CONTEXT_INVALID	*/* Invalid context parameter */*
4	SNMPAPI_CONTEXT_UNKNOWN	*/* Unknown context parameter */*
5	SNMPAPI_ENTITY_INVALID	*/* Invalid entity parameter */*
6	SNMPAPI_ENTITY_UNKNOWN	*/* Unknown entity parameter */*
7	SNMPAPI_INDEX_INVALID	*/* Invalid VBL index parameter */*
8	SNMPAPI_NOOP	*/* No operation performed */*
9	SNMPAPI_OID_INVALID	*/* Invalid OID parameter */*
10	SNMPAPI_OPERATION_INVALID	*/* Invalid/unsupported operation */*
11	SNMPAPI_OUTPUT_TRUNCATED	*/* Insufficient output buf len */*
12	SNMPAPI_PDU_INVALID	*/* Invalid PDU parameter */*
13	SNMPAPI_SESSION_INVALID	*/* Invalid session parameter */*
14	SNMPAPI_SYNTAX_INVALID	*/* Invalid syntax in smiVALUE */*
15	SNMPAPI_VBL_INVALID	*/* Invalid VBL parameter */*
16	SNMPAPI_MODE_INVALID	*/* Invalid mode parameter */*
17	SNMPAPI_SIZE_INVALID	*/* Invalid size/length parameter */*
18	SNMPAPI_NOT_INITIALIZED	*/* SnmpStartup failed/not called */*
19	SNMPAPI_MESSAGE_INVALID	*/* Invalid SNMP message format */*
20	SNMPAPI_HWND_INVALID	*/* Invalid Window handle */*
40	SNMP_MGMTAPI_TIMEOUT	
41	SNMP_MGMTAPI_SELECT_FDERRORS	
42	SNMP_MGMTAPI_TRAP_ERRORS	
43	SNMP_MGMTAPI_TRAP_DUPINIT	
44	SNMP_MGMTAPI_NOTRAPS	
45	SNMP_MGMTAPI_AGAIN	
46	SNMP_MGMTAPI_INVALID_CTL	

ID	Description	Description
47	SNMP_MGMTAPI_INVALID_SESSION	
48	SNMP_MGMTAPI_INVALID_BUFFER	
99	SNMPAPI_OTHER_ERROR	*/* For internal/undefined errors */*
100	SNMPAPI_TL_NOT_INITIALIZED	*/* TL not initialized */*
101	SNMPAPI_TL_NOT_SUPPORTED	*/* TL does not support protocol */*
102	SNMPAPI_TL_NOT_AVAILABLE	*/* Network subsystem has failed */*
103	SNMPAPI_TL_RESOURCE_ERROR	*/* TL resource error */*
104	SNMPAPI_TL_UNDELIVERABLE	*/* Destination unreachable */*
105	SNMPAPI_TL_SRC_INVALID	*/* Source endpoint invalid */*
106	SNMPAPI_TL_INVALID_PARAM	*/* Input parameter invalid */*
107	SNMPAPI_TL_IN_USE	*/* Source endpoint in use */*
108	SNMPAPI_TL_TIMEOUT	*/* No response before timeout */*
109	SNMPAPI_TL_PDU_TOO_BIG	*/* PDU too big for send/receive */*
199	SNMPAPI_TL_OTHER	*/* Undefined TL error */*

10.4 Check list

- 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?

TRAPS AND OIB TRANSLATION

For diagnosis, the network traffic can be recorded and analyzed with a tool such as Wireshark. Wireshark translates OIDs for the same standard MIBs as the SNMP driver. For this reason, the OID translation can be checked by analyzing the Wireshark captures.

Problem	Diagnostics	Possible cause
The agent is not sending	Events that cause traps are triggered. However	Trap dispatch is not configured correctly. The target address set at the agent must be the IP

Problem	Diagnostics	Possible cause
any traps.	Wireshark does not show any incoming traps.	address of the computer with the SNMP driver.
Traps are not displayed in Runtime.	Wireshark displays incoming traps, but registered variables are not updated.	▸ The Windows SNMP trap service is not running. This service must run in order for traps to be able to be received. ▸ The SNMP trap service does run, but has no access to the network. The Windows firewall (or other firewall that is used) must allow the service to receive UDP packets from port 162. ▸ The agent is not configured correctly. The IP address of the agent in the SNMP driver configuraiton must correspond to the actual IP address of the SNMP agent. The incoming trap is assigned to a configured agent in the driver on the basis of the IP address. ▸ The variables have false OIDs. Within an agent, the variable bindings contained in the trap are assinged the trap variables currently registered for the agent on the basis of their respective OIDs. A trap variable is only updated if a trap with a variable binding with the saem OID arrives. ▸ The traps are set as wipers. If, for an agent, the option **Reset trap variables after each trap** is active, the variables of the values *0* and/or *empty string* are reset for all variables updated by a trap straight after the values have been sent. In a text element or numerical element, this can look as though the variables have never been updated. Solutions: - Limit values to value changes of the variables affected - Log the variables affected in archives - Deactivation of the reset
The OID	Although the OID translation	The MIB files are not there. The OID translation

Problem	Diagnostics	Possible cause
translation does not work.	is activated adn variable values arrive with translatable OIDs (see Wireshark), no translations are displayed in Runtime.	only works if there is an *SNMP-MIBS* folder with the MIB files as content. Path: *%ProgramData%\COPA-DATA\zenon8.20\CommunicationProfiles\SNMP-MIBS* No OIDs can be used if this is not the case. The only translation that is carried out is replacement of the first figure (always 1) by "iso", the root element of all OIDs.