



COPADATA
do it your way

zenon driver manual

SNMP32

v.7.11





©2014 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

Distribution and/or reproduction of this document or parts thereof in any form are permitted solely with the written permission of the company COPA-DATA. The technical data contained herein has been provided solely for informational purposes and is not legally binding. Subject to change, technical or otherwise.

Contents

1. Welcome to COPA-DATA help	5
2. SNMP32	6
3. SNMP32 - Data sheet	7
4. Driver history	8
5. Requirements.....	9
5.1 PC	9
5.2 Control	10
6. Configuration	10
6.1 Creating a driver.....	10
6.2 Settings in the driver dialog	12
6.2.1 General	13
6.2.2 Configuration.....	16
6.2.3 SNMP Agents	18
6.2.4 Offline MIB list.....	22
6.2.5 Example configuration.....	31
7. Creating variables.....	34
7.1 Creating variables in the Editor.....	34
7.2 Addressing.....	37
7.3 Driver objects and datatypes	38
7.3.1 Driver objects	38
7.3.2 Mapping of the data types	39
7.4 Creating variables by importing	40
7.4.1 XML import.....	41
7.4.2 Import variables from the driver	41
7.4.3 DBF Import/Export	47
7.5 Driver variables	53
8. Driver-specific functions	58

9. Driver commands	59
10. Error analysis.....	61
10.1 Analysis tool	61
10.2 Troubleshooting	64
10.3 Check list	67

1. Welcome to COPA-DATA help

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 (<mailto:documentation@copadata.com>).

PROJECT SUPPORT

You can receive support for any real project you may have from our Support Team, who you can contact via email at support@copadata.com (<mailto: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 (<mailto:sales@copadata.com>).

2. SNMP32

DEFINITION OF TERMS

SNMP :	<p>Simple Network Management Protocol</p> <p>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.</p>
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	<p>Management Information Bases</p> <p>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.</p>
OID	<p>Object Identifier</p> <p>Is a specific detail information of a SNMP object.</p>
TCP/IP	<p>Transmission Control Protocol / Internet Protocol</p> <p>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.</p>
ICMP	<p>Internet Control Message Protocol</p> <p>This protocol sends error and control messages to the participating computers during the transmission process.</p>
Ping	Checks the accessibility of another computer.
TRAP	<p>In SNMP this is a message, which an agent sends to a management system. Therefore the occurrence of an event is displayed on the host, where the agent is executed. The SNMP service can e.g. be configured in the way, that it sends a trap when receiving an information request that neither contains the correct community name nor an accepted host name for the service.</p>

3. SNMP32 - Data sheet

General:	
Driver file name	SNMP32.exe
Driver name	SNMP Driver
PLC types	Devices supporting SNMP
PLC manufacturer	SNMP;

Driver supports:	
Protocol	SNMP; SNMPv1; SNMPv2c;
Addressing: Address-based	x
Addressing: Name-based	x
Spontaneous communication	x
Polling communication	x
Online browsing	x
Offline browsing	-
Real-time capable	-
Blockwrite	-
Modem capable	-
Serial logging	-
RDA numerical	-
RDA String	-

Requirements:	
Hardware PC	Standard network card
Software PC	- the Windows "SNMP Trap" service must be installed and started- firewalls must be configured to allow UDP traffic on port 161 and port 162 (or allow snmptrap.exe)
Hardware PLC	-
Software PLC	SNMP-Agent (Server)
Requires v-dll	-

Platforms:	
Operating systems	Windows Vista, 7, 8, 8.1 Server 2008/R2, Server 2012/R2;
CE platforms	-;

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 Control

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

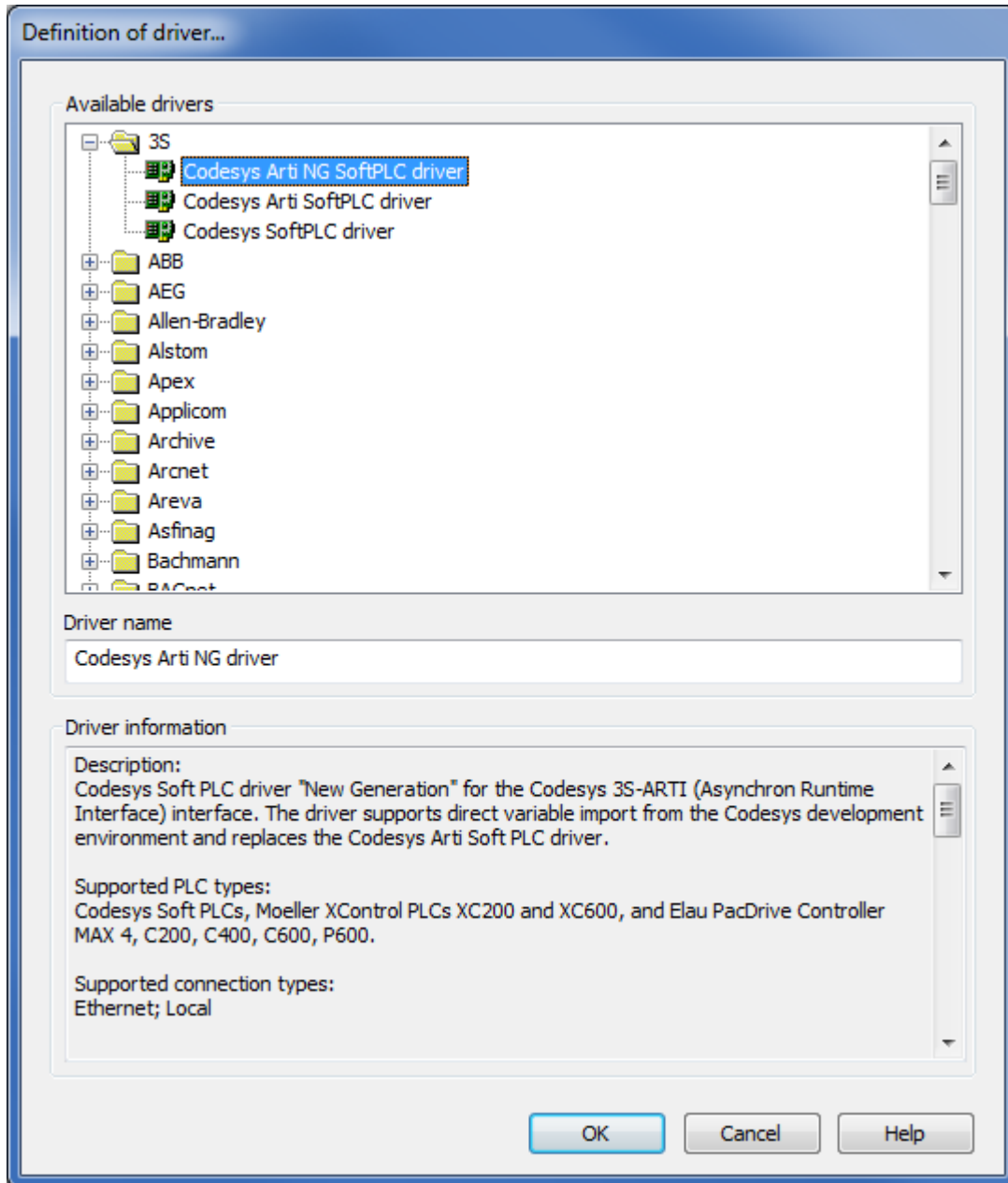
Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.

6.1 Creating a driver

In order to create a new driver:

1. Right-click on **Driver** in the Project Manage and select **Driver new** in the context menu.

2. In the following dialog the control system offers a list of all available drivers.



3. Select the desired driver and give it a name:
- The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.
 - 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 with **OK**. In the following dialog the single configurations of the drivers are defined.

Only the respective required drivers need to be loaded for a project. Later loading of an additional driver is possible without problems.



Information

For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:

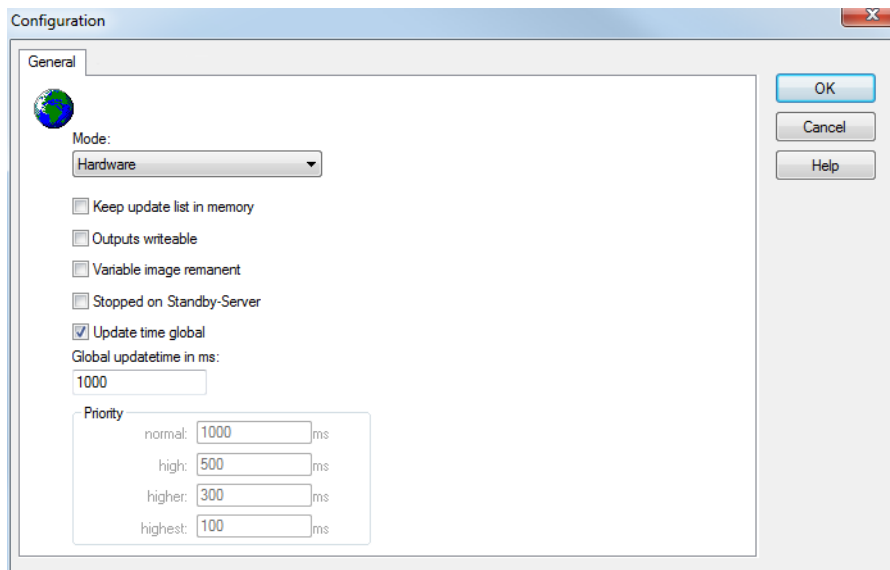
- ▶ Internal
- ▶ MathDr32
- ▶ SysDrv.

▶

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General



Parameters	Description
Mode	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> ▶ Hardware: <p>A connection to the control is established.</p> ▶ Simulation static <p>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 straton. 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.</p> ▶ Simulation - counting <p>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.</p> ▶ Simulation - programmed <p>N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the straton Workbench and runs in a straton Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).</p>
Keep update list in the memory	<p>Variables which were requested once are still requested from the control even if they are currently not needed.</p> <p>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.</p>
Outputs writeable	<p>Active: Outputs can be written.</p> <p>Inactive: Writing of outputs is prevented.</p> <p>Note: Not available for every driver.</p>

Variable image remanent	<p>This option saves and restores the current value, time stamp and the states of a data point.</p> <p>Fundamental requirement: The variable must have a valid value and time stamp.</p> <p>The variable image is saved in mode hardware if:</p> <ul style="list-style-type: none"> ▶ one of the states <code>S_MERKER_1(0)</code> up to <code>S_MERKER8(7)</code>, <code>REVISION(9)</code>, <code>AUS(20)</code> or <code>ERSATZWERT(27)</code> is active <p>The variable image is always saved if:</p> <ul style="list-style-type: none"> ▶ the variable is of the object type <code>Driver variable</code> ▶ the driver runs in simulation mode. (not programmed simulation) <p>The following states are not restored at the start of the Runtime:</p> <ul style="list-style-type: none"> ▶ <code>SELECT(8)</code> ▶ <code>WR-ACK(40)</code> ▶ <code>WR-SUC(41)</code> <p>The mode Simulation - programmed at the driver start is not a criterion in order to restore the remanent variable image.</p>
Stopped on Standby Server	<p>Setting for redundancy at drivers which allow only on communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade.</p> <p>Attention: If this option is active, the gapless archiving is no longer guaranteed.</p> <p>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 (<code>statusverarbeitung.chm: :/24150.htm</code>) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.</p>
Update time global	<p>Active: The set <code>Update time global</code> in ms is used for all variables in the project. The priority set at the variables is not used.</p> <p>Inactive: The set priorities are used for the individual variables.</p>
Priority	<p>Here you set the polling times for the individual priorities. All variables with the according priority are polled in the set time. The allocation is taken</p>

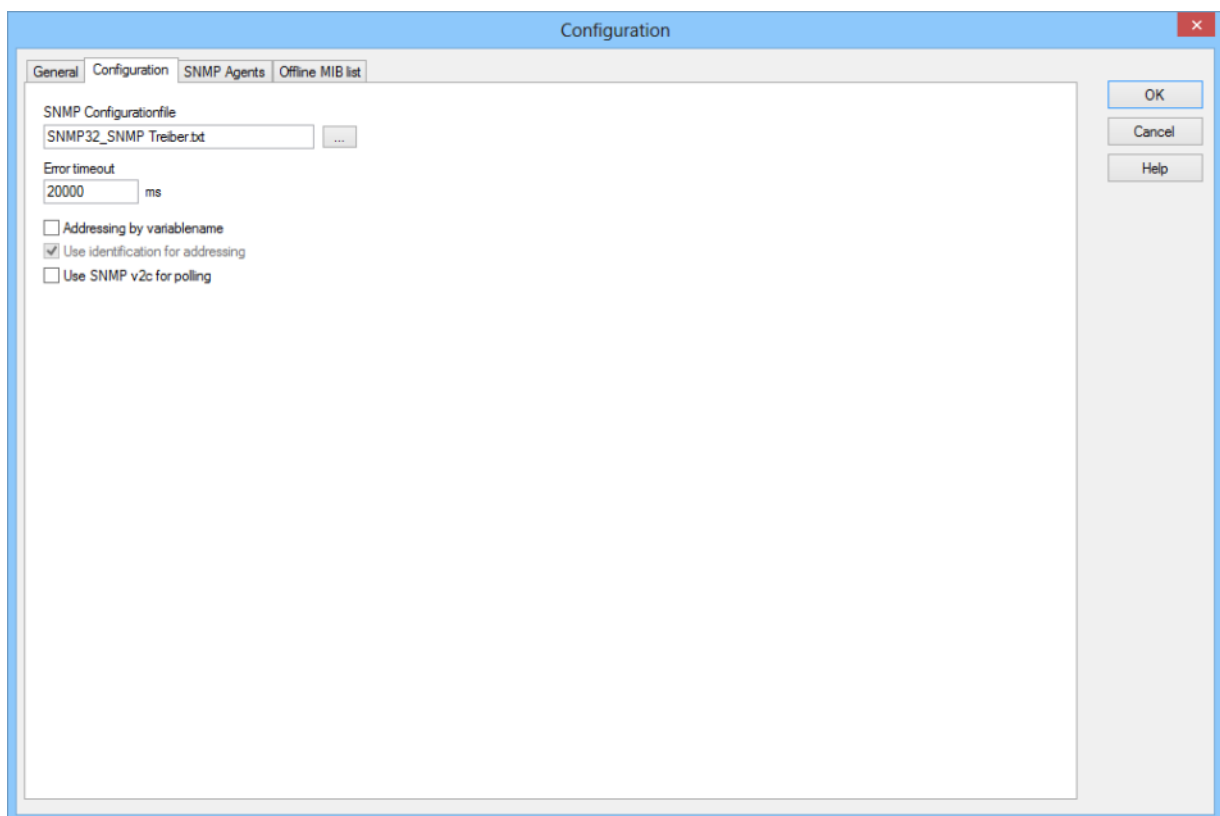
	place for each variable separately in the settings of the variable properties. The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities. Thus the communication load is distributed better.
OK	Accepts settings in all tabs and closes dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR CYCLICAL DRIVER

The following applies for cyclical drivers:

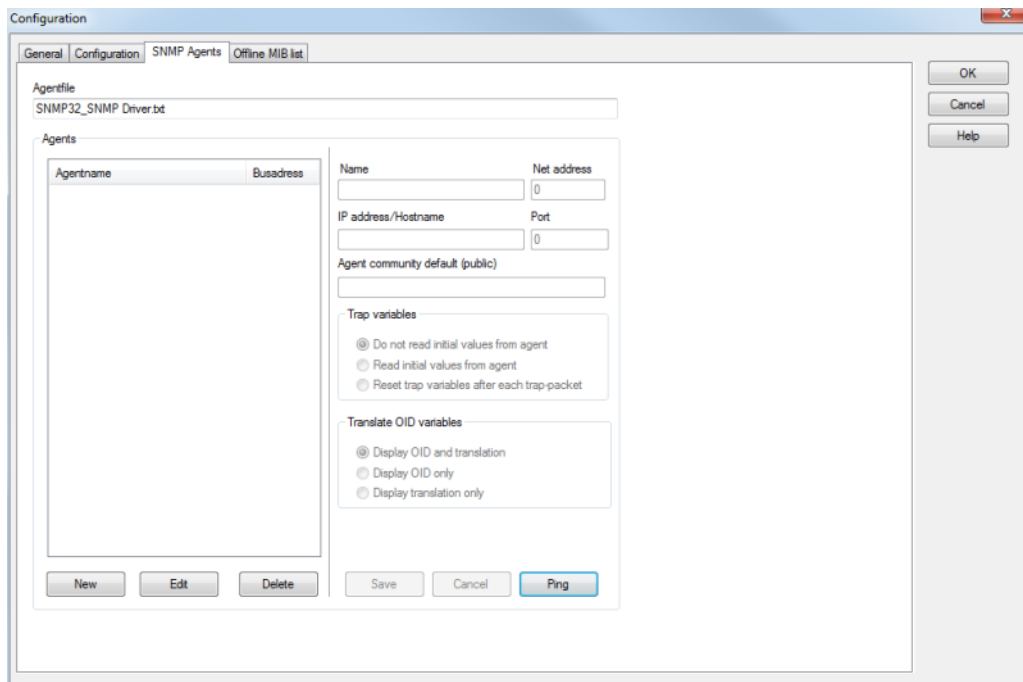
For **Set value**, **Advising** of variables and **Requests**, a read cycle is immediately triggered for all drivers - regardless of the set update time. This ensures that the value is immediately available for visualization after writing. Update times can therefore be shorter than pre-set for cyclical drivers.

6.2.2 Configuration



Parameters	Description
SNMP configuration file	File in which the settings are saved. It is in the zenon project directory.
Error response time (ms)	<p>The time in milliseconds, for which the driver waits, before it outputs an error message.</p> <p>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.</p> <p>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.</p>
Addressing using variable name or identification	<p>Active: Addressing is carried out using the names of the variables or using identification. The browse agent and receive online traps properties are then not available.</p> <p>Inactive: Addressing is carried out by means of offset in assignment file.</p> <p>Attention: The type of addressing is important for variable import (on page 41). Subsequent amendments of addressing from variable name or identification to offset can lead to communication problems.</p>
Using identification for addressing	<p>Identification is used.</p> <p>Is not available if "Addressing using variable name or identification" was activated.</p>

6.2.3 SNMP Agents



The image shows a 'Configuration' window with tabs for 'General', 'Configuration', 'SNMP Agents', and 'Offline MIB list'. The 'SNMP Agents' tab is active. At the top, there is a text field for 'Agentfile' containing 'SNMP32_SNMP.Driver.txt'. Below this is a table titled 'Agents' with columns 'Agentname' and 'Busaddress'. To the right of the table are input fields for 'Name', 'Net address', 'IP address/Hostname', and 'Port'. Below these are fields for 'Agent community default (public)' and 'Trap variables'. The 'Trap variables' section has three radio buttons: 'Do not read initial values from agent' (selected), 'Read initial values from agent', and 'Reset trap variables after each trap-packet'. The 'Translate OID variables' section has three radio buttons: 'Display OID and translation' (selected), 'Display OID only', and 'Display translation only'. At the bottom left are buttons for 'New', 'Edit', and 'Delete'. At the bottom right are buttons for 'Save', 'Cancel', and 'Ping'. On the far right of the window are buttons for 'OK', 'Cancel', and 'Help'.

Agentname	Busaddress
-----------	------------

Name: Net address:

IP address/Hostname: Port:

Agent community default (public):

Trap variables

- ☒ Do not read initial values from agent
- ☐ Read initial values from agent
- ☐ Reset trap variables after each trap-packet

Translate OID variables

- ☒ Display OID and translation
- ☐ Display OID only
- ☐ Display translation only

New Edit Delete Save Cancel Ping OK Cancel Help

Parameters	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: <ul style="list-style-type: none"> ▶ SNMP: 161 ▶ Traps: 162
Agent community default (public)	The access identification of the agent. Is always "public" (read only) by default. A community name serves as a password that is defined for one or more SNMP hosts. Accepted community names are only used for the authentication of incoming messages. This can be configured in the SNMP service settings of the agent.
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

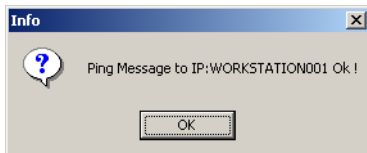
	<p>hen Runtime starts. If the start value was read successfully, the trap variable is no longer read cyclically, but only spontaneously.</p> <p>Recommendation: Select this option if traps from the agent are read using GET, in order to then expect spontaneous messages (traps).</p>
<p>► Reset trap variables after each trap</p>	<p>No initial value is read for traps.</p> <p>The value and status of the variable remain empty until a trap is received.</p> <p>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</p> <p>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.</p> <p>Recommendation: Select this option to react to traps in Runtime that always have the same value to create a CEL entry or alarm.</p>

Translating OID variables	<p>Settings for OID variables. Allows separate configuration of the translation for each agent.</p> <p>Note: Only available from version 6.50. All other versions are automatically handled with the standard <code>Only display OID</code>.</p> <p>Requirements:</p> <p>The following properties define how the OID translation is applied to ingoing values for variables from the SNMP driver. To do this, two conditions must be met:</p> <ul style="list-style-type: none"> ▶ 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. <p>Incoming values that do not meet these conditions are forwarded without treatment.</p>
<code>Display OID and translation</code>	<p>Active: OID and translation are transferred as a value.</p> <p>The OID is translated in a written text and the value is forwarded to Runtime in the following format. <code>[OID] ([Text])</code></p> <p>Example: <code>.1.3.6.1.6.3.1.1.5.3 (IF-MIB::linkDown)</code>.</p> <p>If the translation is not successful, only the OID is transferred to Runtime as a value.</p>
<code>Only display OID</code>	<p>Active: Only OID is transferred as a value.</p> <p>Note: Standard for all versions prior to zenon 6.50.</p>
<code>Only display translation</code>	<p>Active: Only the translation is transferred as a value.</p> <p>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.</p>
New	Defining new agents
Delete	Delete agent.
Edit	Edit agent.
Save	Saves the agent settings that have been set.
Discard	Discards the agent settings that have been set.

Ping

Tests a configured agent IP address.

Result of a successful ping attempt with name resolution:

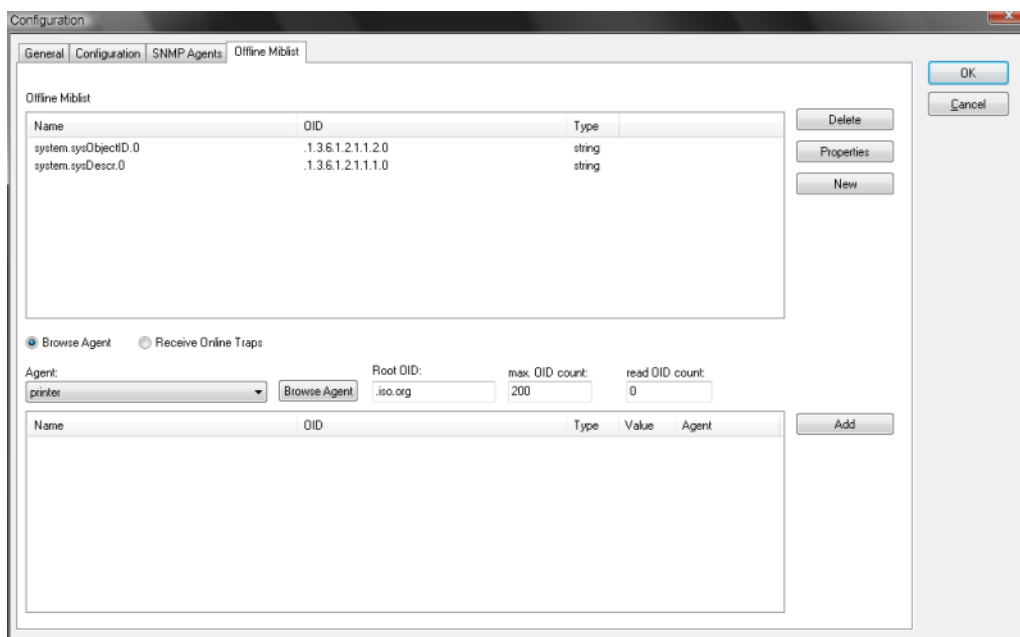


If you get no positive response to a ping, it may be, that the name could not be resolved correctly. In this case, try pinging the IP address. If this fails again, this means the network participant / agent is not accessible.

**Information**

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

6.2.4 Offline MIB list



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 25).
Browse Agent	<p>Active: Agent is searched through according to OIDs.</p> <p>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.</p> <p>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.</p>
Receive online traps	<p>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 necessarily 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.</p> <p>The following must be the case for the computer to receive traps:</p> <ul style="list-style-type: none"> ▶ 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 <p>Note: If you receive an error message in relation to 0x2a or 0x64, it is possible that the SNMP Trap Windows service does not run.</p>
Agent	Selection 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 18) 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 <code>offline MIB list</code> . 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 the MIB editor (on page 22) or using online variable import (on page 41), 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 already 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 variable is added, the net address 0 is used. However, this net address could also be in use by another agent, which would lead to problems.

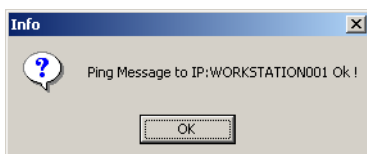
ADDRESSING

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 16) 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 communication before you create variables.

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.



Parameters	Description
Name	Here a freely definable name can be used.
OID	The accurate identification of the OID. Please make sure that the complete and correct identification is entered here; otherwise, the zenon SNMP driver will not be able to read the data.
Datatype	Integer or String, depending on the accessed SNMP datatype.

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:

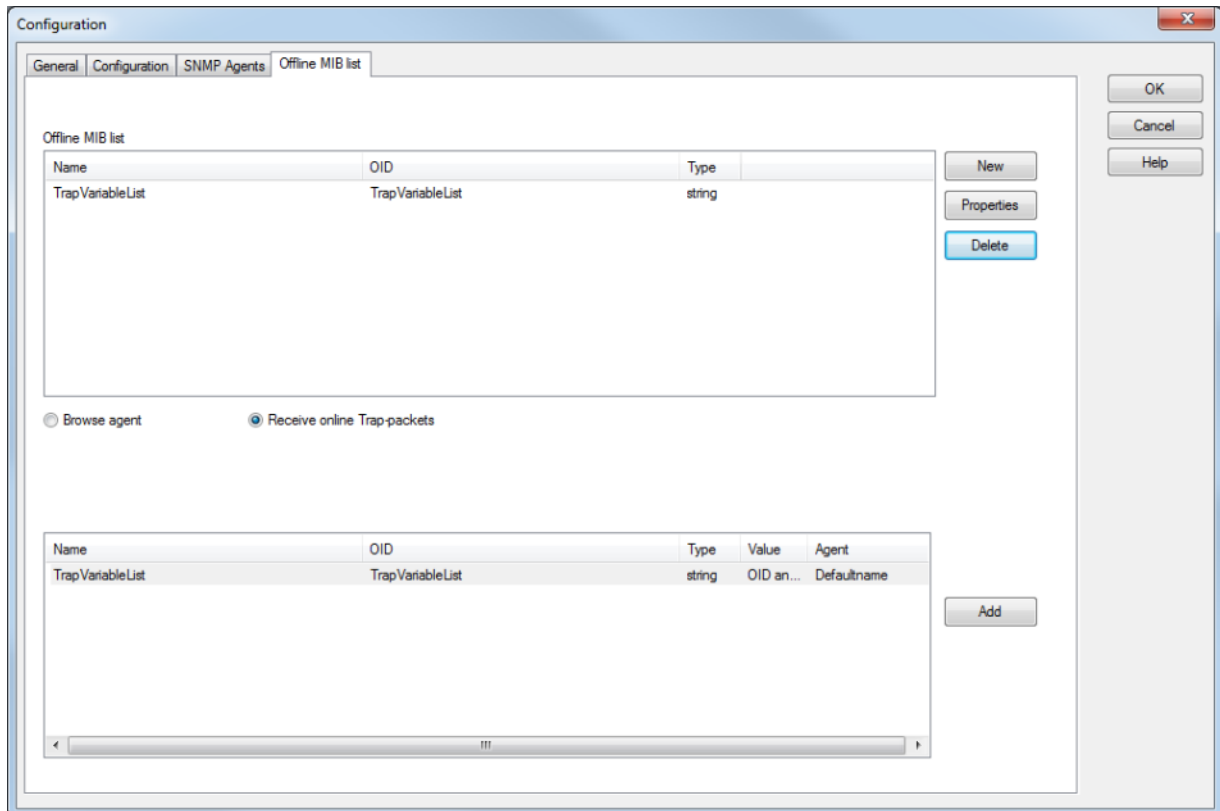
1. Driver configuration:
 - a) An `SNMP driver` is created in the Editor.
 - b) The required agents are configured (on page 18) on the driver.
 - c) The desired OID translation is set for the agents.
 - d) In the Offline MIB list (on page 22), the agents are browsed and the desired OIDs are added to the offline MIB list.
 - e) 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 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.

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.



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_[N]** node for each variable binding with the contents **OID**, **OID_TRANSLATED**, **TYPE** and **VALUE**.

SCHEMA

Parameters	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 for 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 18) tab. In doing so, 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 22) 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 41) 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 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. Recording traps.
With traps that are received, the individual variable bindings are processed first - translate OIDs, send values to 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

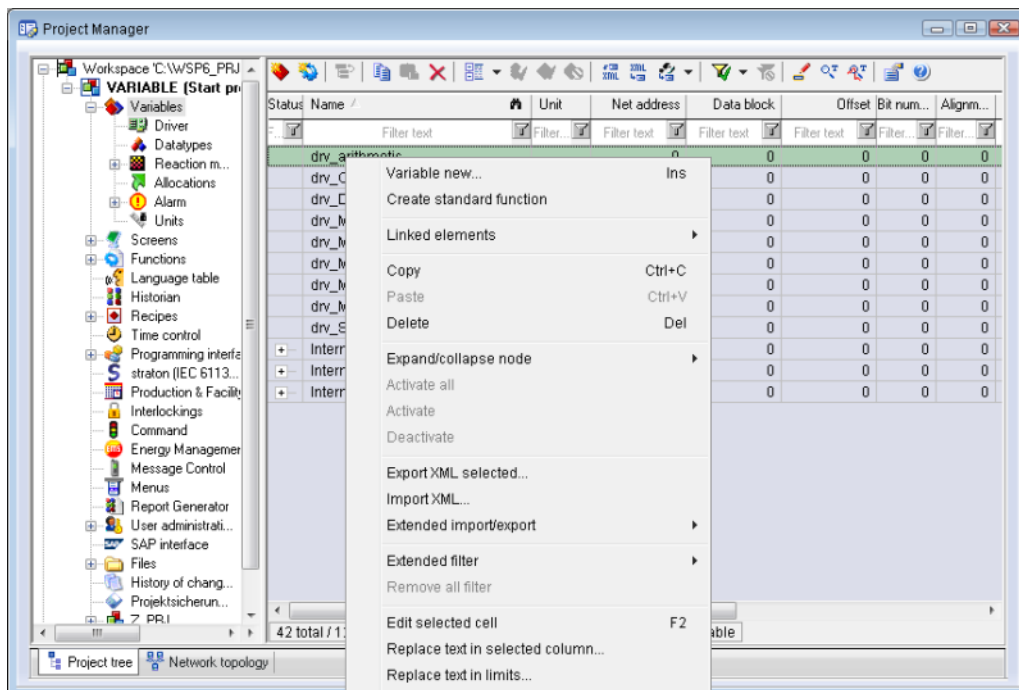
Variables can be created:

- ▶ as simple variables
- ▶ in arrays (main.chm::/15262.htm)
- ▶ as structure variables (main.chm::/15278.htm)

VARIABLE DIALOG

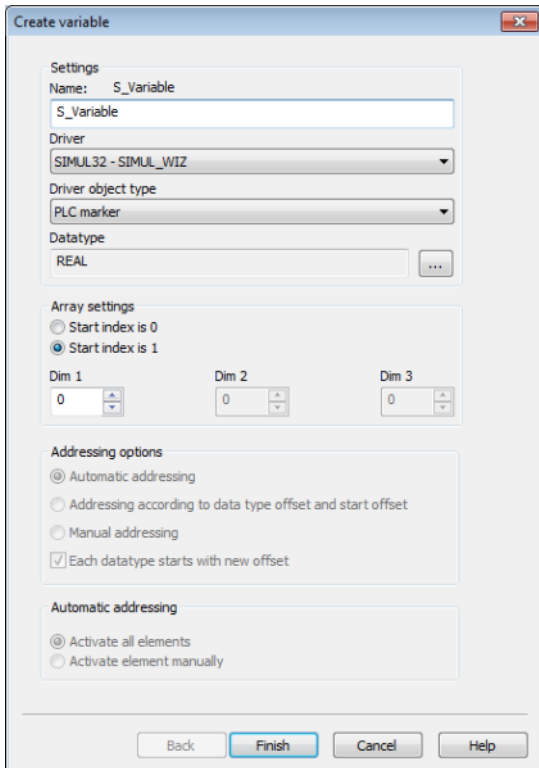
To create a new variable, regardless of which type:

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



2. The dialog for configuring variables is opened

3. configure the variable
4. The settings that are possible depends on the type of variables



Parameters	Description
Name	<p>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.</p> <p>Maximum length: 128 characters</p> <p>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.</p>
Drivers	<p>Select the desired driver from the drop-down list.</p> <p>Note: If no driver has been opened in the project, the driver for internal variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is automatically loaded.</p>
Driver object type (cti.chm::/28685.htm)	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 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.

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	<p>Freely definable name.</p> <p>Attention: the name must be unique within each control system project.</p> <p>If the variable name is used for addressing, you must enter the OID here.</p>
Identification	<p>Any text can be entered here, e.g. for resource labels, comments ...</p> <p>If the variable identification is used for addressing, you must enter the OID here.</p>
Net address	<p>Bus address or net address of the variable.</p> <p>This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.</p>
Data block	not used for this driver
Offset	Offset of the variable; the memory address of the variable in the PLC. Configurable [0.. 4294967295]
Alignment	not used for this driver
Bit number	<p>Number of the bit within the configured offset.</p> <p>Valid input [0.. 65535].</p>
String length	Only available for String variables: Maximum number of characters that the variable can take.
Driver object type	Depending on the employed driver, an object type is selected during the creation of the variable; the type can be changed here later.
Data type	<p>Data type of the variable, which is selected during the creation of the variable; the type can be changed here later.</p> <p>ATTENTION: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.</p>

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 27).

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	R / W	BOOL	
SNMP traps	67	R / W	DINT, UDINT, STRING	V1 and V2 traps are supported.
SNMP variables	65	R / W	DINT, UDINT, STRING	
SNMP counter	66	R / W	DINT, UDINT	
Driver variable	35	R / W	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. Find out more in the chapter about the Driver variables (on page 53)

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



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 less 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

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.

Control	zenon	Data type
	BOOL	8
	USINT	9

	SINT	10
	UINT	2
	INT	1
	UDINT	4
	DINT	3
	ULINT	27
	LINT	26
	REAL	5
	LREAL	6
	STRING	12
	WSTRING	21
	DATE	18
	TIME	17
	DATE_AND_TIME	20
	TOD (Time of Day)	19

Data type: The property `Data type` is the internal numerical name 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 (main.chm::/13028.htm) manual in the Variables (main.chm::/13045.htm) section.

7.4.1 XML import

For the import/export of variables the following is true:

- ▶ The import/export must not be started from the global project.
- ▶ The start takes place via:
 - Context menu of variables or data typ in the project tree
 - or context menu of a variable or a data type
 - or symbol in the symbol bar variables



Attention

When importing/overwriting an existing data type, all variables based on the existing data type are changed.

Example:

There is a data type XYZ derived from the type `INT` with variables based on this data type. The XML file to be imported also contains a data type with the name XYZ but derived from type `STRING`. If this data type is imported, the existing data type is overwritten and the type of all variables based on it is adjusted. I.e. the variables are now no longer `INT` variables, but `STRING` variables.

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 16)

- a) Addressing via offset
- b) Addressing using variable name or identification



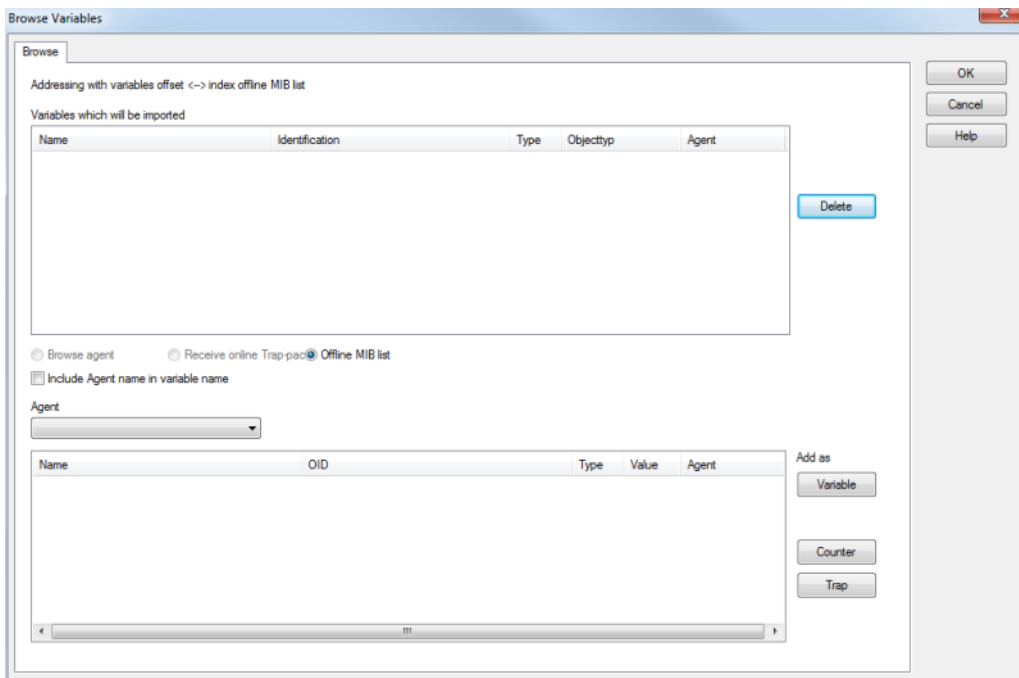
Attention

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



Browse Variables

Browse

Addressing with variables offset <-> Index offline MIB list

Variables which will be imported

Name	Identification	Type	Objecttyp	Agent

Delete

☐ Browse agent
 ☐ Receive online Trap-pac
 ☒ Offline MIB list

☐ Include Agent name in variable name

Agent: ▼

Name	OID	Type	Value	Agent

Add as

Variable

Counter

Trap

OK

Cancel

Help

Parameters	Description
Addressing using variable offset <--> Offline MIB list index	Note the addressing defined under configuration (on page 16).
zenon Variable for creation	List of the variables to be created in zenon.
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 22). 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.
▶ Counters	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 Variables

Browse

Addressing with identification

Variables which will be imported

Name	Identification	Type	Objecttyp	Agent
------	----------------	------	-----------	-------

Delete

☒ Browse agent

☐ Receive online Trap-pac

☐ Offline MIB list

☐ Include Agent name in variable name

Agent

AUS

Browse

Root OID

.iso.org

max. OID count

200

read OID count

0

Name	OID	Type	Value	Agent
------	-----	------	-------	-------

Add as

Variable

Counter

Trap

OK

Cancel

Help

Parameters	Description
Addressing using variable offset <--> Offline MIB list index	Note the addressing defined under configuration (on page 16).
zenon Variable for creation	List of the variables to be created in zenon.
Delete	Removes entries from the list.
Browse Agent	<p>Active : Agent is searched through according to OIDs.</p> <p>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.</p> <p>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.</p>
Receive online traps	<p>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 necessarily 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.</p> <p>The following must be the case for the computer to receive traps:</p> <ul style="list-style-type: none"> ▶ 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 <p>Note: If you receive an error message in relation to 0x2a or 0x64, it is possible that the SNMP Trap Windows service does not run.</p>
Agent	Selection 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 18) 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.
▶ Counters	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 the MIB editor (on page 22) or using online variable import (on page 41), 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, SNMP traps of all other IP addresses are displayed.

- ▶ If a trap comes from an agent that is already 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 variable is added, the net address 0 is used. However, this net address 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 16) 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 communication 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 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 export 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 there 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

Description	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the MAX_LAENGE entry in project.ini .
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in project.ini .
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 project.ini .
Unit	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Bus address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADDRESS	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 Recipe Group Manager
LES_SCHR	R	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTYP and DATENTYP
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MENTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix

ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in project.ini .
ADJWVBA	R	1	Non-linear value adaption: 0: Non-linear value adaption is used 1: Non-linear value adaption is not used
ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
ZWREMA	N	16	Linked counter REMA.
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 DEFINITION

Limit definition for limit values 1 to 4, and status 1 bis 4:

Description	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	hnical value or ID number of a linked variable for a dynamic limit (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
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
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 Driver variables

The driver kit implements a number of driver variables. These are divided into:

- ▶ Information
- ▶ Configuration
- ▶ Statistics and
- ▶ Error messages

The definitions of the variables defined in the driver kit are available in the import file `drvvar.dbf` (on the CD in the directory: `CD_Drive:/Predefined/Variables`) and can be imported from there.

Note: Variable names must be unique in zenon. If driver variables are to be imported from `drvvar.dbf` again, the variables that were imported beforehand must be renamed.



Information

Not every driver supports all driver variants.

For example:

- ▶ Variables for modem information are only supported by modem-compatible drivers
- ▶ Driver variables for the polling cycle only for pure polling drivers
- ▶ Connection-related information such as ErrorMessage only for drivers that only edit one connection at 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	12:00 AM	Connection in hold
LineStateConferenced	BOOL	12:00 AM	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	12:00 AM	Connection in hold for conference
LineStateOnHoldPendTransfer	BOOL	24.13	Connection in hold for transfer
LineStateDisconnected	BOOL	24.14	Connection terminated.
LineStateUnknow	BOOL	24.15	Connection status unknown

ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped For <code>driver stop</code> , the variable has the value <code>TRUE</code> and an <code>OFF</code> bit. After the driver has started, the variable has the value <code>FALSE</code> and no <code>OFF</code> bit.
SimulRTState	UDINT	60	Informes the status of Runtime for driver simulation.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If <code>TRUE</code> , 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 <code>SrvDrvVarApplyCom</code> being called (which currently has no further function).
ApplyModem	BOOL	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method <code>SrvDrvVarApplyModem</code> . This closes the current connection and opens a new one according to the settings <code>PhoneNumberSet</code> and <code>ModemHwAdrSet</code> .
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	<code>TRUE</code> , if update time is global
TreiberSimul	BOOL	5	<code>TRUE</code> , if driver in sin simulation mode
TreiberProzab	BOOL	6	<code>TRUE</code> , 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.
Baud rate	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
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).
MaxUpdateTimeHigher	UDINT	57	Time since the last update of the priority group Higher in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group High in milliseconds (ms).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group Highest in milliseconds (ms).
PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

ERROR MESSAGES

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

With the functions GET, SET and GETNEXT data are read from a device supporting SNMP.

PING STATUS

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

For this status bit INVALID (main.chm::/24148.htm) 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.

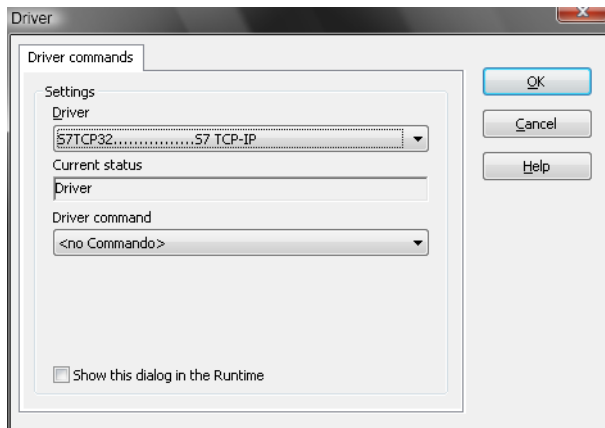
9. Driver commands

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.

Driver commands are used to influence drivers using zenon; start and stop for example. The engineering is implemented with the help of function **Driver commands**. To do this:

- ▶ create a new function
- ▶ select *Variables -> Driver commands*

- The dialog for configuration is opened



Parameters	Description
Drivers	Drop-down list with all drivers which are loaded in the project.
Current state	Fixed entry which has no function in the current version.
Driver commands	Drop-down list for the selection of the command.
► Start driver (online mode)	Driver is reinitialized 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	Enter driver-specific commands. Opens input field in order to enter a command.
► Activate driver write set value	Write set value to a driver is allowed.
► Deactivate driver	Write set value to a driver is prohibited.

write set value	
► Establish connection 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)
Show this dialog in the Runtime	The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

If the computer, on which the **driver command** function is executed, is part of the zenon network, additional actions are carried out. A special network command is sent from the computer to the project server, which 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 (main.chm::/12464.htm) program that was also installed with zenon. You can find it under *Start/All programs/zenon/Tools 7.11 -> Diagviewer*.

zenon driver log all errors in the log files. The default folder for the log files is subfolder **log** in directory **ProgramData**, example:
 C:\ProgramData\zenon\zenon7.11\LOG for zenon Version 7.11. Log files are text files with a special structure.

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 currently created entries live
- ▶ customize the logging settings
- ▶ change the folder in which the log files are saved

Hints:

1. In Windows CE even errors are not logged per default due to performance reasons.
2. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
3. 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.
4. If you only use **Error logging**, the problem description is in column **Error text**. For other diagnosis level the description is in column **General text**.
5. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** and/or **Error code** and/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.
6. 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**.



Information

You can find further information on the Diagnosis Viewer in the Diagnose Viewer (main.chm::/12464.htm) chapter.

10.2 Troubleshooting

ERROR MESSAGES

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 was not started. Note: occurs after Receive online traps was activated in the driver configuraiton (on page 22) 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: <ul style="list-style-type: none"> ▶ The driver cannot receive any traps ▶ No error message is displayed.

ENTRIES IN LOG FILE

Unexpected Error during OID Translation INIT. Translation 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 [numerical OID]	An unexpected error occured when translating the given OID.

Could not translate OID [numerical 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 initiated successfully.
Translation of OID [numerical OID] successful	The given OID was translated successfully.

ERROR NUMBERS

Error numbers for `SnmpMgrRequest`:

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

10.3 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 any traps.	Events that cause traps are triggered. However Wireshark does not show any incoming traps.	Trap dispatch is not configured correctly. The target address set at the agent must be the IP address of the computer with the SNMP driver.
Traps are not displayed in Runtime.	Wireshark displays incoming traps, but registered variables are not updated.	<ul style="list-style-type: none"> ▶ 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 <code>Reset trap variables after each trap</code> is active, the variables of the values <code>0</code> and/or <code>empty string</code> 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: <ul style="list-style-type: none"> - Limits to value changes of the variables affected - Log the variables affected in archives - Deactivation of the reset

<p>The OID translation does not work.</p>	<p>Although the OID translation is activated and variable values arrive with translatable OIDs (see Wireshark), no translations are displayed in Runtime.</p>	<p>The MIB files are not there. The OID translation only works if, in the zenon installation directory, there is an <code>SNMP-MIBS</code> folder with the MIB files as content. 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.</p>
---	---	--