



**COPADATA**  
do it your way

# zenon driver manual

## DNP332

v.8.00





**COPA-DATA**

©2018 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. Technical data is only used for product description and are not guaranteed qualities in the legal sense. Subject to change, technical or otherwise.

# Contents

<b>1. Welcome to COPA-DATA help .....</b>	<b>5</b>
<b>2. DNP332 .....</b>	<b>5</b>
<b>3. DNP332 - Data sheet.....</b>	<b>7</b>
<b>4. Driver history .....</b>	<b>9</b>
<b>5. Requirements.....</b>	<b>9</b>
5.1 PC .....	10
<b>6. Configuration .....</b>	<b>10</b>
6.1 Creating a driver.....	10
6.2 Settings in the driver dialog .....	14
6.2.1 General .....	15
6.2.2 Com .....	19
6.2.3 Connections.....	21
<b>7. Creating variables.....</b>	<b>25</b>
7.1 Creating variables in the Editor.....	25
7.2 Addressing.....	29
7.3 Driver objects and datatypes .....	30
7.3.1 Driver objects .....	30
7.3.2 Mapping of the data types .....	32
7.4 Creating variables by importing.....	33
7.4.1 XML import.....	34
7.4.2 DBF Import/Export .....	35
7.5 Communication details (Driver variables).....	41
<b>8. Driver-specific functions .....</b>	<b>46</b>
<b>9. Driver command function .....</b>	<b>49</b>
<b>10. Device Profile .....</b>	<b>52</b>
<b>11. Error analysis.....</b>	<b>52</b>

11.1	Analysis tool .....	52
11.2	Check list .....	53

# 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](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](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. DNP332

## TESTED WITH THE FOLLOWING HARDWARE AND SOFTWARE

zenon ProcessGateway with AccessDNP.DLL

## TESTING ENVIRONMENT

Tested with simulation driver variables, which serve as DNP3 slave via the zenon Process Gateway. zenon accesses these variables via the driver.

## INFORMATION ON THE DNP332, DNP3\_NG AND DNP\_NG DRIVER

The DNP332 driver and the DNP3\_NG will be replaced by the DNP3\_TG from version 7.20. For reasons of compatibility, the DNP3\_NG and the older DNP332 drivers are still included in the setup, but are no longer displayed in the driver selection list by default. Existing projects that are converted use the driver that was originally configured as before. The old driver can continue to be used normally in converted projects. However a switch to the new DNP3\_TG driver is also possible.

## DISPLAYING THE DNP332/DNP3\_NG IN THE DRIVER LIST

If you want to use the DNP332/DNP3\_NG driver in a new project in 7.20, the driver must be added to the driver list again:

1. Start the program **Driverinfo.exe** from the zenon installation medium; subfolder `\AdditionalSoftware\COPA-DATA DriverXML Editor`.
2. Open the driver XML file using the program called **Driverinfo**.  
Example: **TREIBER\_EN.XML** from the folder `C:\ProgramData\COPA-DATA\zenon7200`.  
In doing so, **EN** is the code for the language in the Editor and **7200** for the installed version, version 7.20.
3. Go to the `DNP3` folder and select **New Driver** in the context menu.
4. Enter **DNP332** or **DNP3\_NG** in all three fields of the dialog and confirm by clicking on **OK**.
5. Save the changes by clicking on the **Save** symbol in the toolbar.  
The driver can now be selected again in the Editor.
6. Repeat this step for each language that you use in the Editor.

## SWITCH TO THE NEW DNP3\_TG DRIVER

The DNP3\_TG driver is compatible with the old DNP3\_NG driver and DNP332 driver in principle. The **Replace driver** function in the Editor can also be used to switch from the DNP332 driver or from the DNP3\_NG driver to the DNP3\_TG driver. After the driver replacement, some settings need to be made again manually. Some functions in the DNP3\_TG driver are implemented differently and require a change to the project configuration.

When planning to replace a driver, please note the following:

General:

- ▶ Back up your project first

- ▶ Note the driver configuration of the old driver. These must be entered again once the driver has been replaced.
- ▶ Writing to a Frozen Counter variable no longer leads to an Immediate Freeze No Ack. This function can be achieved using a Commando variable.

#### DNP332 to DNP3\_TG:

- ▶ If you use Select Before Operate with the DNP332 driver for the Command Processing, deactivate the **Select Before Operate** property for the variable. Instead, select the `Auto-SBO` entry for `Binary Output` and `Analog Output` variables in the **Command Mode** property.
- ▶ If, you have used driver data types for **Analog Inputs** or **Counter** with DNP332 and these no longer exist in the DNP3\_NG driver, then you must amend the data types of the variables before the change. Background: With the DNP332 driver, it was possible to select data types that were not envisaged by the DNP3 standard. These data types can no longer be used with DNP3\_NG.

#### DNP3\_NG to DNP3\_TG:

- ▶ The **Command Mode** property for `Analog Output` and `Binary Output` variables must be set manually. Alternatively, it is possible to export the variables in XML format before the driver switch and to import them again after the switch. The DNP3\_TG driver also supports, in addition to Direct Operate and Auto SBO , Direct Operate No Ack.
- ▶ The property variation for the variable has been removed. Instead of configuring the variation for the variable, the variable can be excluded from the class poll with the **Classless Read** check box and explicitly read with the desired version with a user-defined command.
- ▶ The "class scan" variable scan no longer exists for the DNP3\_TG. It is best to delete variables of this type before the driver switch. To trigger a class poll, explicit reading or also a Cold Restart, create Command variables from the DNP3\_TG driver variables.
- ▶

## 3. DNP332 - Data sheet

General:	
Driver file name	DNP332.exe
Driver name	DNP3 driver
PLC types	All DNP3 compatible PLCs
PLC manufacturer	DNP3; GE Harris;

<b>Driver supports:</b>	
Protocol	DNP3; GE Harris(DNP3);
Addressing: Address-based	X
Addressing: Name-based	--
Spontaneous communication	--
Polling communication	X
Online browsing	--
Offline browsing	--
Real-time capable	X
Blockwrite	--
Modem capable	--
Serial logging	X
RDA numerical	--
RDA String	--
Hysteresis	--
extended API	--
Supports status bit <b>WR-SUC</b>	--
alternative IP address	--

<b>Requirements:</b>	
Hardware PC	Serial interface; Standard LAN Adapter
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	--

Platforms:	
Operating systems	Windows CE 6.0, Embedded Compact 7; Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2, Server 2016;
CE platforms	x86; ARM;

## 4. Driver history

Date	Driver version	Change
6/18/2008	1500	SBO timeout can be set in the driver configuration
7/7/2008	1600	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

Serial interface, Ethernet TCP/IP.

### SOFTWARE

Unless it is already there, copy the driver DNP332.exe to the zenon program directory.

### CE

Copy the driver DNP332.dll to the zenon CE program directory.

## 6. Configuration

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



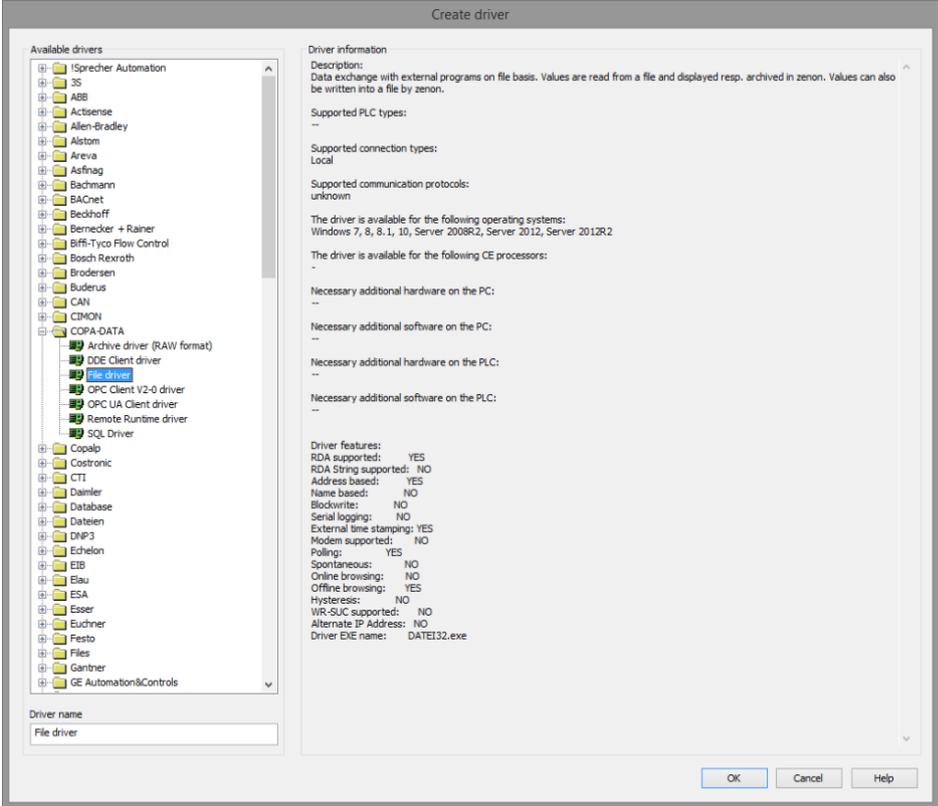
### Information

*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

Note that the **ADP driver** can only be created once per project. The driver can be found in the driver group COPA-DATA.

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



Parameter	Description
<b>Available drivers</b>	<p>List of all available drivers.</p> <p>The display is in a tree structure:            [+] expands the folder structure and shows the drivers contained therein.            [-] reduces the folder structure</p> <p>Default: no selection</p>
<b>Driver name</b>	<p>Unique <b>Identification</b> of the driver.</p> <p>Default: Empty</p> <p>The input field is pre-filled with the pre-defined <b>Identification</b> after selecting a driver from the list of available drivers.</p>
<b>Driver information</b>	<p>Further information on the selected driver.</p> <p>Default: Empty</p> <p>The information on the selected driver is shown in this area after selecting a driver.</p>

## CLOSE DIALOG

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



### Information

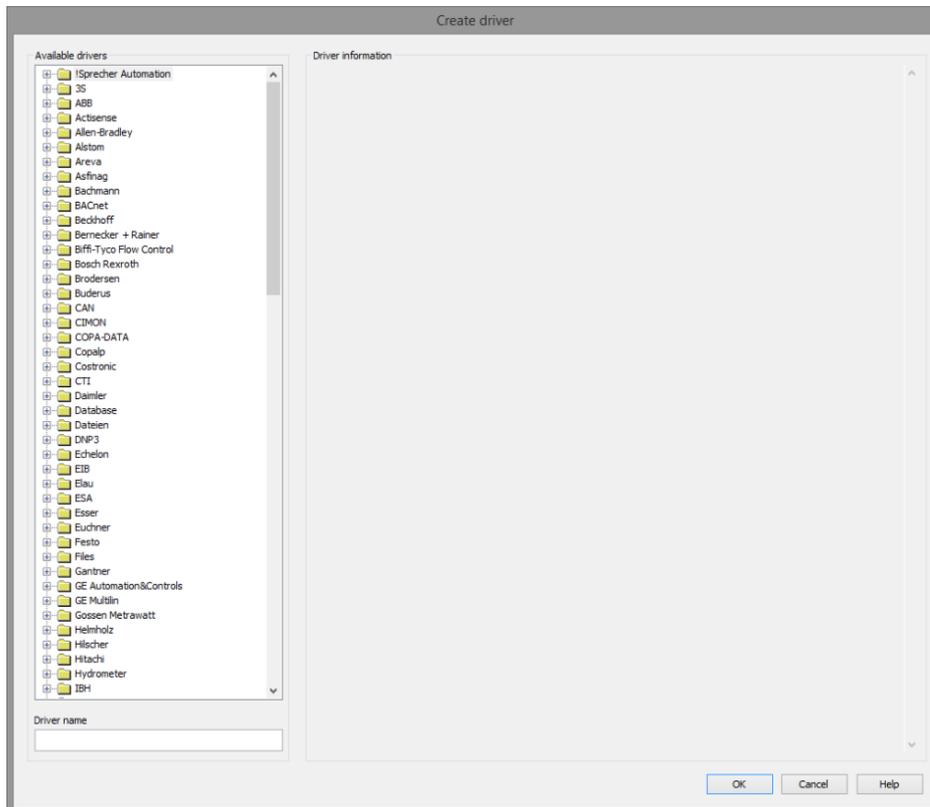
*The content of this dialog is saved in the file called `Treiber_[Language].xml`. You can find this file in the following folder: `C:\ProgramData\COPA-DATA\zenon[version number]`.*

## 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 dialog offers a list of all available drivers.



- 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**:

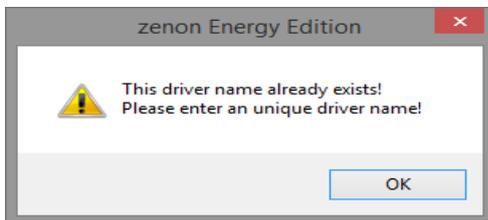
- The **Driver name** must be unique.  
If a driver is used more than once in a project, a new name has to be given each time. This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
- The **Driver name** is part of the 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.

- Confirm the dialog by clicking on the **OK** button.  
The configuration dialog for the selected driver is opened.

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

## DRIVER NAME DIALOG ALREADY EXISTS

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



## ZENON PROJECT

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

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



### Information

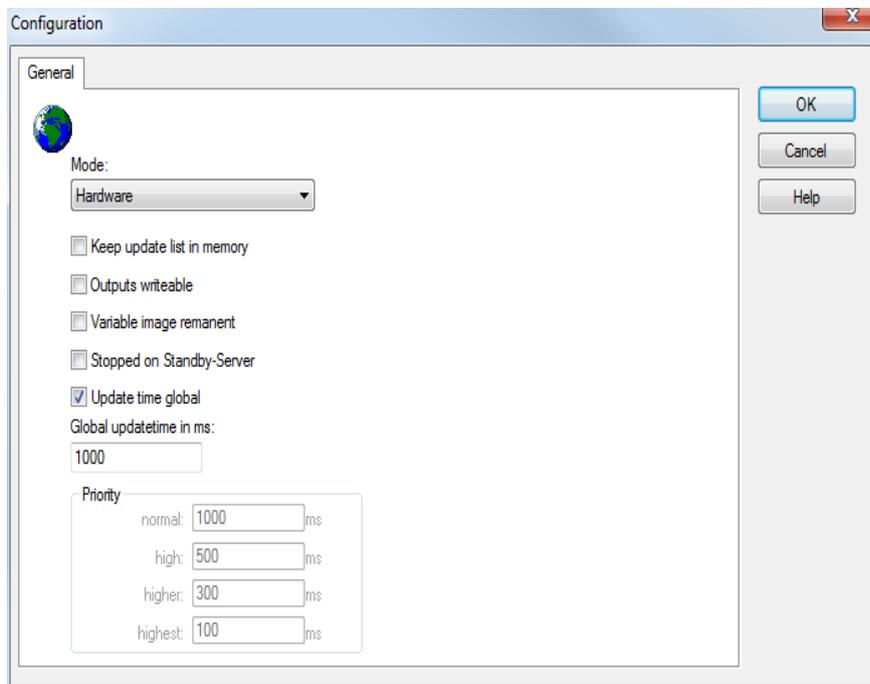
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

## 6.2 Settings in the driver dialog

You can change the following settings of the driver:

## 6.2.1 General

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



Option	Description
<b>Mode</b>	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> <li>▶ <b>Hardware:</b> A connection to the control is established.</li> <li>▶ <b>Simulation - static:</b> 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.</li> <li>▶ <b>Simulation - counting:</b> 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.</li> <li>▶ <b>Simulation - programmed:</b> No 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 zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).</li> </ul>
<b>Keep update list in the memory</b>	<p>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.</p>
<b>Output can be written</b>	<ul style="list-style-type: none"> <li>▶ <b>Active:</b> Outputs can be written.</li> <li>▶ <b>Inactive:</b> Writing of outputs is prevented.</li> </ul> <p><b>Note:</b> Not available for every driver.</p>
<b>Variable image remanent</b>	<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"><li>▶ one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active</li></ul> <p>The variable image is always saved if:</p> <ul style="list-style-type: none"><li>▶ the variable is of the object type <b>Driver variable</b></li><li>▶ the driver runs in simulation mode. (not programmed simulation)</li></ul> <p>The following states are not restored at the start of the Runtime:</p> <ul style="list-style-type: none"><li>▶ SELECT (8)</li><li>▶ WR-ACK (40)</li><li>▶ WR-SUC (41)</li></ul> <p>The mode <b>Simulation - programmed</b> at the driver start is not a criterion in order to restore the remanent variable image.</p>
--	--

<p><b>Stop on Standby Server</b></p>	<p>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.</p> <p>Attention: If this option is active, the gapless archiving is no longer guaranteed.</p> <ul style="list-style-type: none"> <li>▶ <i>Active:</i> 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 <b>switched off (statusverarbeitung.chm::/24150.htm)</b> but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.</li> </ul> <p>Default: <i>Inactive</i></p> <p>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.</p>
<p><b>Global Update time</b></p>	<p>Setting for the global update times in milliseconds:</p> <ul style="list-style-type: none"> <li>▶ <i>Active:</i> The set <b>Global update time</b> is used for all variables in the project. The priority set at the variables is not used.</li> <li>▶ <i>Inactive:</i> The set priorities are used for the individual variables.</li> </ul> <p>Exceptions: Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the <b>Spontaneous driver update time</b> section.</p>
<p><b>Priority</b></p>	<p>The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.</p> <p>The variables are allocated separately in the settings of the variable properties.</p> <p>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.</p> <p>Attention: Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.</p>

#### CLOSE DIALOG

Option	Description
<p><b>OK</b></p>	<p>Applies all changes in all tabs and closes the dialog.</p>

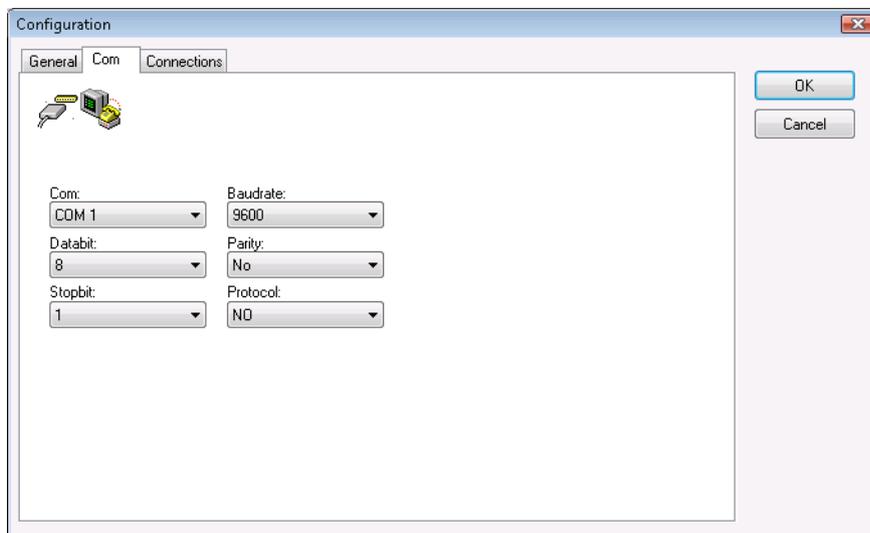
<b>Cancel</b>	Discards all changes in all tabs and closes the dialog.
<b>Help</b>	Opens online help.

## 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 Com



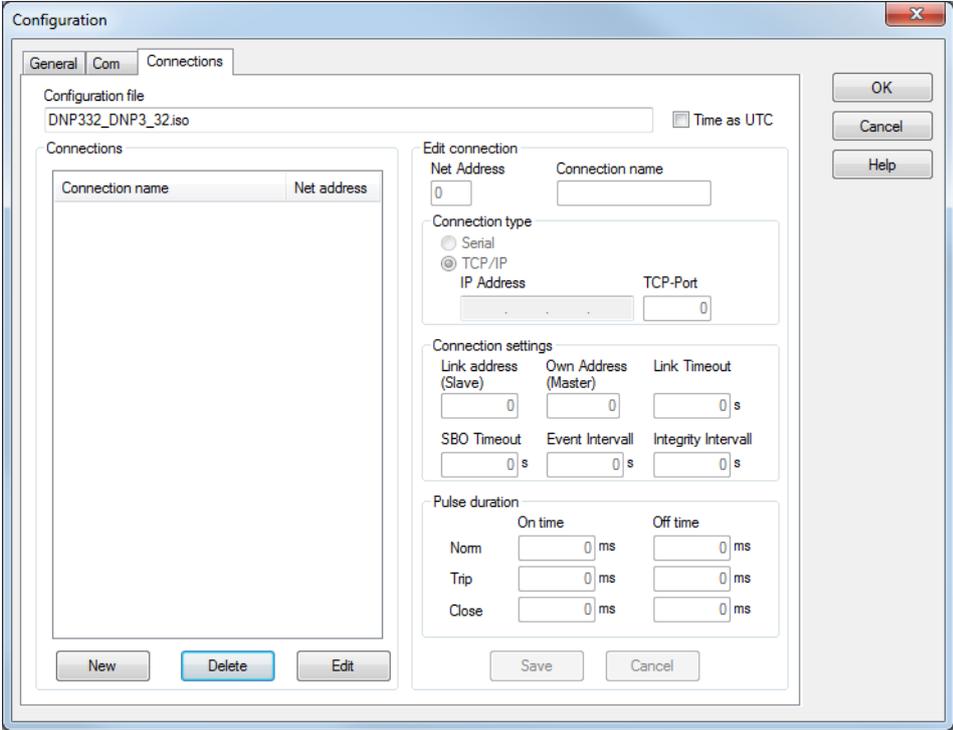
Parameters	Description
<b>Com</b>	Selection Com port. Default: 1
<b>Baud rate</b>	Selection baud rate. Adapting to PLC. Default: 9600
<b>Data bits</b>	Number of data bits. Adapting to PLC. Default: 8
<b>Stop bit</b>	Selection stop bit. Adapting to PLC. Default: 1
<b>Parity</b>	Selection parity. Adapting to PLC. Default: No
<b>Protocol</b>	Selection protocol. Adapting to PLC. Default: No



### Information

*The exact settings depend on the used PLCs. Take the valid values from the manual of your PLC.*

### 6.2.3 Connections



### CONFIGURATION FILE

Parameters	Description
<b>Configuration file</b>	Name of the file, in which the connection information is stored. For your information only. Cannot be changed.

## CONNECTIONS

Parameters	Description
<b>Connections</b>	Connection settings
<b>List of connections</b>	Displays the configured connections.
<b>New</b>	Creates new connection
<b>Delete</b>	Deletes selected connection.
<b>Edit</b>	Edits selected connection.

## EDIT CONNECTION

Parameters	Description
<b>Edit connection</b>	Properties for configuration of selected/new connection.
<b>Net address</b>	Net address of the connection. Value between 0 and 255.
<b>Connection name</b>	Name of connection. Freely selectable.

## CONNECTION TYPE

Parameters	Description
<b>Connection type</b>	Configuration of the connection type and the connection parameters.
<b>Serial</b>	Active: serial connection
<b>TCP/IP</b>	Active: Connection via TCP/IP
<b>IP address</b>	IP address of PLC.
<b>TCP-port</b>	Port that is used for communication. Default: 20000

## PULSE DURATION

Parameters	Description
<b>Connection settings</b>	
<b>Link address (slave)</b>	Link address of the PLC (of the DNP3 slave).
<b>Own address (master)</b>	Own address (of the DNP3 master driver itself).
<b>Link Timeout</b>	<p>Only available for previously-saved serial connection.</p> <p>Individual time-out for current connection if several slaves share a serial connection. Allows the polling of several slaves if the first slave queried does not respond.</p> <p>Recommendation: Value less than standard timeout for the connection (30 s).</p> <p>Default: 30 s</p> <p>See also driver-specific functions (on page 46)/<b>Link Timeout</b>.</p>
<b>SBO time-out</b>	<p>Time in seconds which is waited for an answer from the PLC after the <code>Select</code>. If the time expires, the <code>Select</code> is considered as denied.</p> <p>Default: 30</p>
<b>Event interval</b>	<p>Prompting the driver for new events in seconds.</p> <p>Default: 1</p>
<b>Integrity interval</b>	<p>Integrity poll of the driver in seconds.</p> <p>Default: 3600</p>

## PULSE DURATION

Parameters	Description
<b>Pulse duration</b>	Defines pulse duration for <b>Norm</b> , <b>Trip</b> and <b>Close</b> for each connection. <b>Norm</b> , <b>Trip</b> and <b>Close</b> define which relays are switched.
<b>Norm</b>	<p>Equals NUL. Pulse duration norm:</p> <ul style="list-style-type: none"> <li>▶ <b>On time:</b> Period of time in milliseconds in which the output is ON Default: 1000</li> <li>▶ <b>Time off</b> Period of time in milliseconds in which the output is OFF Default: 1000</li> </ul>
<b>Trip</b>	<p>Pulse duration trip:</p> <ul style="list-style-type: none"> <li>▶ <b>On time:</b> Period of time in milliseconds in which the output</li> </ul>

	<p>is ON Default: 1000</p> <p>▶ <b>Time off</b> Period of time in milliseconds in which the output is OFF Default: 1000</p>
<b>Close</b>	<p>Pulse duration close:</p> <p>▶ <b>On time:</b> Period of time in milliseconds in which the output is ON Default: 1000</p> <p>▶ <b>Time off</b> Period of time in milliseconds in which the output is OFF Default: 1000</p>
<b>Save</b>	Saves parameters for connection and deactivates editing mode.
<b>Undo</b>	Cancels changes and deactivates editing mode without saving.
<b>OK</b>	<p>Accept changes in the dialog and close dialog.</p> <p>Only available if no connection is in the "edit" state.</p>
<b>Cancel</b>	Discards changes and closes dialog.
<b>Help</b>	Opens online help.



### Information

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

### CREATE NEW CONNECTION

1. click on the button **New**
2. Enter the connection details.
3. Click on **Save**

### EDIT CONNECTION

1. select the connection in the connection list
2. click on the button **Edit**
3. change the connection parameters

4. finish with **Save**

#### **DELETE CONNECTION**

1. select the connection in the connection list
2. click on the button **Delete**
3. the connection will be removed from the list

## **7. Creating variables**

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

### **7.1 Creating variables in the Editor**

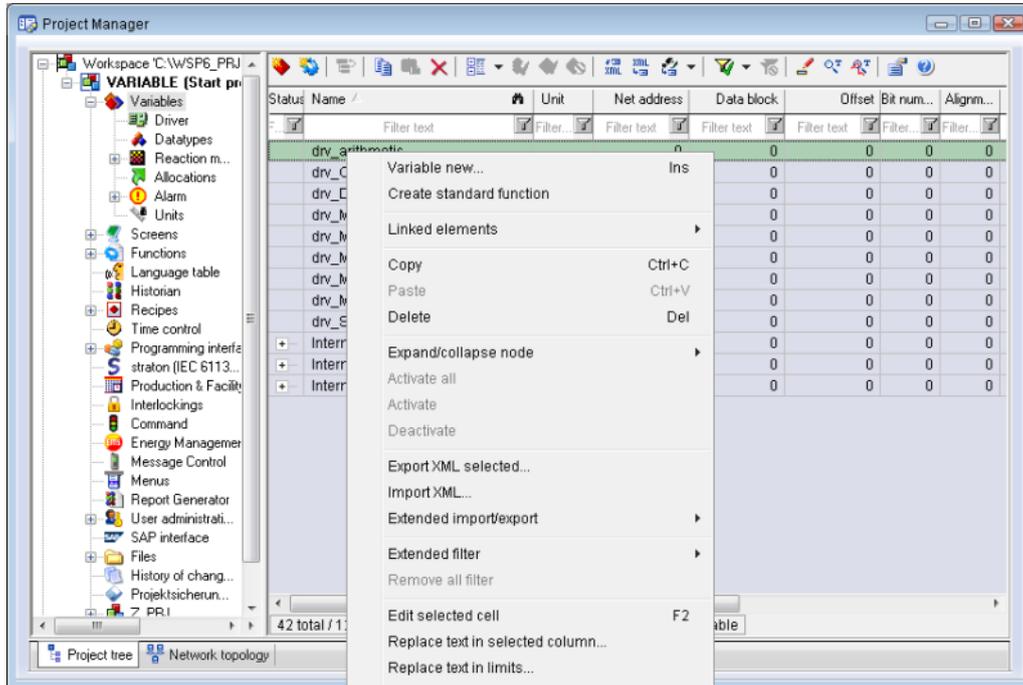
Variables can be created:

- ▶ 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:

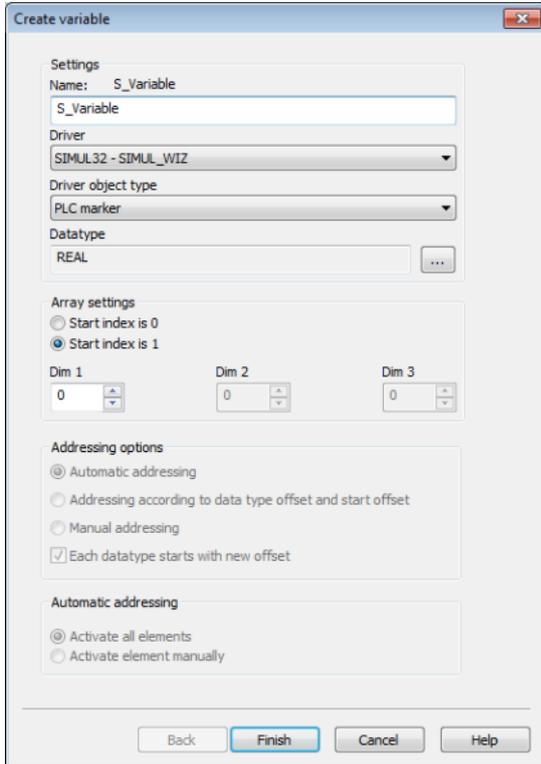
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

2. Configure the variable

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



Property	Description
<p><b>Name</b></p>	<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 <b>Finish</b> button remains inactive.</p> <p>Note: For some drivers, the addressing is possible over the property <b>Symbolic address</b>, as well.</p>
<p><b>Drivers</b></p>	<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 (<b>Intern.exe (Main.chm://Intern.chm://Intern.htm)</b>) is automatically loaded.</p>
<p><b>Driver Object Type (cti.chm://28685.htm)</b></p>	<p>Select the appropriate driver object type from the drop-down list.</p>

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

## INHERITANCE FROM DATA TYPE

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

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

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

## 7.2 Addressing

Group/Property	Description
<b>General</b>	Property group for general settings.
<b>Name</b>	Freely definable name. Attention: For every zenon project the name must be unambiguous.
<b>Identification</b>	Freely definable identification. E.g. for Resources label, comments, ...
<b>Addressing</b>	
<b>Net address</b>	Network address of variables. This address refers to the bus address in the connection configuration of the driver. This defines the Outstation, on which the variable resides.
<b>Data block</b>	For variables of object type <code>Extended data block</code> , enter the datablock number here. Adjustable from 0 to 4294967295. You can take the exact maximum area for data blocks from the manual of the PLC.
<b>Offset</b>	Offset of variables. Equal to the memory address of the variable in the PLC. Adjustable from 0 to 4294967295.
<b>Alignment</b>	not used for this driver
<b>Bit number</b>	Number of the bit within the configured offset. Possible entries: 0 to 65535.
<b>String length</b>	Only available for String variables. Maximum number of characters that the variable can take.
<b>Driver connection/Data Type</b>	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.
<b>Driver connection/Driver Object Type</b>	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
<b>Driver connection/Priority</b>	Assigns a variable a priority for the update time.

## VARIABLE ADDRESSING

The variables are allocated to the memory area of the PLC with the help of property **Offset**.

## WINDOWS CE

This driver is available for Windows CE (x86 platforms).

## 7.3 Driver objects and datatypes

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

### 7.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
<b>Analog Inputs</b>	68	X	--	REAL	
<b>Analog Output Statuses</b>	69	X	X	UDINT, DINT, REAL, UINT, INT	
<b>Binary Inputs</b>	64	X	--	BOOL	
<b>Binary Inputs Double</b>	71	X	--	USINT	
<b>Binary Output Statuses</b>	65	X	X	BOOL, USINT	
<b>Frozen Counters</b>	67	X	X	BOOL, UDINT, DINT, USINT, UINT, INT, SINT	
<b>Running Counters</b>	66	X	X	BOOL, UDINT, DINT, USINT, UINT, INT, SINT	
<b>String Data</b>	70	X	X	STRING	
<b>Communication details</b>	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	<p>Variables for the static analysis of the communication; is transferred between driver and Runtime (not to the PLC).</p> <p><b>Note:</b> The addressing and the behavior is the same for most zenon drivers.</p> <p>You can find detailed information on this in the Communication details (Driver variables) (on page 41) chapter.</p>

Key:

**X**: supported

--: not supported

## COMMANDS

You can use a **command processing** to first send a `Select` and then an `Operate` to data points. The SBO timeout can be configured in the driver configuration. A counter starts running if a `Select` is sent. After the configured SBO timeout has passed, a `COT_actterm` with a set PN bit will be sent back. If the `Select` action is cancelled, a `COT_deact` will be sent back to notify you of this. After the SBO timeout has passed, a `COT_deactcon` will be sent. If an `Operate` is sent within the timeout, the procedure will stay as usual.

### 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.

## DRIVER OBJECT TYPES AND SUPPORTED IEC DATA TYPES FOR PROCESS VARIABLES IN ZENON

Driver object types	Channel type	Supported datatypes (DataType)	Read	Write	Comment
Binary Input	64	BOOL	X	--	
Binary Output	65	BOOL	--	X	Value 1: LATCH_ON Value 0: LATCH_OFF  1 if writing was successful, 0 if an error occurred
Binary Output	65	USINT	--	X	Value 1: PULSE_ON Value 2: PULSE_OFF Value 3: LATCH_ON Value 4: LATCH_OFF  1 for successful writing, 0 for error
Running Counter	66	BOOL,SINT,INT,DINT	X	--	
Frozen Counter	67	BOOL,SINT,INT,DINT	X	X	Writing freezes the counter
Analog Input	68	REAL	X	--	
Analog Output	69	REAL	--	X	1 for successful writing, 0 for error
String Data	70	STRING	X	X	

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

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.

#### 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 cannot 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 (main.chm::/13046.htm)** chapter.

## 7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.



### Information

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

### IMPORT DBF FILE

To start the import:

1. right-click on the variable list
2. in the drop-down list of **Extended export/import...** select the **Import dBase** command
3. follow the 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 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
<b>KANALNAME</b>	Char	128	Variable name. The length can be limited using the <b>MAX_LAENGE</b> entry in the <b>project.ini</b> file.
<b>KANAL_R</b>	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 <b>MAX_LAENGE</b> entry in the <b>project.ini</b> file.
<b>KANAL_D</b>	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
<b>TAGNR</b>	C	128	Identification. The length can be limited using the <b>MAX_LAENGE</b> entry in the <b>project.ini</b> file.
<b>EINHEIT</b>	C	11	Technical unit
<b>DATENART</b>	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
<b>KANALTYP</b>	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
<b>HWKANAL</b>	Num	3	Net address
<b>BAUSTEIN</b>	N	3	Datablock address (only for variables from the data area of the PLC)
<b>ADRESSE</b>	N	5	Offset
<b>BITADR</b>	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
<b>ARRAYSIZE</b>	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

<b>LES_SCHR</b>	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
<b>MIT_ZEIT</b>	R	1	time stamp in zenon (only if supported by the driver)
<b>OBJEKT</b>	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTTYP and DATENTYP
<b>SIGMIN</b>	Float	16	Non-linearized signal - minimum (signal resolution)
<b>SIGMAX</b>	F	16	Non-linearized signal - maximum (signal resolution)
<b>ANZMIN</b>	F	16	Technical value - minimum (measuring range)
<b>ANZMAX</b>	F	16	Technical value - maximum (measuring range)
<b>ANZKOMMA</b>	N	1	Number of decimal places for the display of the values (measuring range)
<b>UPDATERATE</b>	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
<b>MEMTIEFE</b>	N	7	Only for compatibility reasons
<b>HDRATE</b>	F	19	HD update rate for historical values (in sec, one decimal possible)
<b>HDTIEFE</b>	N	7	HD entry depth for historical values (number)
<b>NACHSORT</b>	R	1	HD data as postsorted values
<b>DRRATE</b>	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
<b>HYST_PLUS</b>	F	16	Positive hysteresis, from measuring range
<b>HYST_MINUS</b>	F	16	Negative hysteresis, from measuring range
<b>PRIOR</b>	N	16	Priority of the variable
<b>REAMATRIZE</b>	C	32	Allocated reaction matrix
<b>ERSATZWERT</b>	F	16	Substitute value, from measuring range
<b>SOLLMIN</b>	F	16	Minimum for set value actions, from measuring range
<b>SOLLMAX</b>	F	16	Maximum for set value actions, from measuring range
<b>VOMSTANDBY</b>	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
<b>RESOURCE</b>	C	128	Resources label. Free string for export and display in lists.  The length can be limited using the MAX_LAENGE entry in <b>project.ini</b> .
<b>ADJWVBA</b>	R	1	Non-linear value adaption: 0: Non-linear value adaption is used

			1: Non-linear value adaption is not used
<b>ADJZENON</b>	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
<b>ADJWVBA</b>	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
<b>ZWREMA</b>	N	16	Linked counter REMA.
<b>MAXGRAD</b>	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
<b>AKTIV1</b>	R	1	Limit value active (per limit value available)
<b>GRENZWERT1</b>	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)
<b>SCHWWERT1</b>	F	16	Threshold value for limit value
<b>HYSTERESE1</b>	F	14	Is not used
<b>BLINKEN1</b>	R	1	Set blink attribute
<b>BTB1</b>	R	1	Logging in CEL
<b>ALARM1</b>	R	1	Alarm
<b>DRUCKEN1</b>	R	1	Printer output (for CEL or Alarm)
<b>QUITTIER1</b>	R	1	Must be acknowledged
<b>LOESCHE1</b>	R	1	Must be deleted
<b>VARIABLE1</b>	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
<b>FUNC1</b>	R	1	Functions linking
<b>ASK_FUNC1</b>	R	1	Execution via Alarm Message List
<b>FUNC_NR1</b>	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
<b>A_GRUPPE1</b>	N	10	Alarm/Event Group
<b>A_KLASSE1</b>	N	10	Alarm/Event Class
<b>MIN_MAX1</b>	C	3	Minimum, Maximum
<b>FARBE1</b>	N	10	Color as Windows coding
<b>GRENZTXT1</b>	C	66	Limit value text
<b>A_DELAY1</b>	N	10	Time delay
<b>INVISIBLE1</b>	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. These 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** (on the installation medium in the `\Predefined\Variables` folder) and can be imported from there.

**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 only for pure polling drivers
- ▶ Connection-related information such as ErrorMSG 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	24.10	Connection in hold
LineStateConferenced	BOOL	24.11	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	BOOL	24.13	Connection in hold for transfer
LineStateDisconnected	BOOL	24.14	Connection terminated.
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped  For <code>driver stop</code> , the variable has the value <code>TRUE</code> and an <b>OFF</b> bit. After the driver has started, the variable has the value <code>FALSE</code> and no <b>OFF</b> bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

ConnectionStates	STRING	61	<p>Internal connection status of the driver to the PLC.</p> <p>Connection statuses:</p> <p>0 : Connection OK</p> <p>1 : Connection failure</p> <p>2 : Connection simulated</p> <p>Formating:</p> <p>&lt;Netzadresse&gt;:&lt;Verbindungszustand&gt;;...;;</p> <p>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.</p> <p>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.</p>
------------------	--------	----	--

## CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	BOOL	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	BOOL	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings <b>PhoneNumberSet</b> and <b>ModemHwAdrSet</b> .
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
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 <b>Normal</b> in milliseconds (ms).
MaxUpdateTimeHigher	UDINT	57	Time since the last update of the priority group <b>Higher</b> in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group <b>High</b> in milliseconds (ms).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group <b>Highest</b> 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:

### COMMANDS WITH SELECT BEFORE OPERATE (SBO)

With the help of command processing, you can first send a `Select` and then an `Operate` to variables.

As the control (Remsdaq) does not support `Cancel` or does not have a `Select` time-out, you must set this at the driver. Therefore the **SBO** time-out can be configured in the driver configuration (on page 21).

The driver behaves as follows:

- ▶ If a `Select` is written, it is acknowledged and a timer starts to run.

- ▶ After the set `SBO Timeouts` has passed, a `COT_actterm` with a set `N_CONF` (PN-Bit) will be sent back.
- ▶ If the `Select` action is cancelled, a `COT_deact` will be sent back to notify you of this. After the **SBO Timeouts** has passed, a `COT_deactcon` will be sent.
- ▶ If an `Operate` is sent within the timeout, the procedure will stay as usual.

## LINK TIMEOUT

Serial connections have a standard timeout of 30 seconds. If two or more slaves with a polling time of less than 30 seconds share a serial connection and a slave does not respond, the connection remains blocked and the other slaves also do not receive any values. To prevent this, an individual timeout can be set for each slave using the **Link Timeout** option in the configuration of the connections (on page 21).

Recommendation: Set this value as less than the standard timeout for the connection (30 s).

If a slave runs into a timeout, it is put offline and waits 30 seconds until the next query. If this query is also not possible, another wait of 30 seconds is carried out. Other slaves can therefore poll normally during the waiting time.

## ERROR FILE

The driver supports central logging in the Log Server.

## EXTENDED ERROR FILE

The driver does not support extended logging.

## SERIAL LOGGING

To activate this option, you must add section `[RS232LOG]` to the **project.ini**:

```
[RS232LOG]
```

```
LOGCOMx=y
```

Parameters	Description
y	Value 0 or 1: 0 = turns log off 1 = turns log on
x	Number of the selected interface.

**EXAMPLE:**

[RS232LOG]

LOGCOM1=1

Activates logging for COM1.

**Information**

*We recommend activating logging only for a short time, when problems occur. Active logging needs considerable computer performance. Additionally the log file needs a lot of memory within short time.*

Parameters	Description
<b>Blockwrite</b>	Not supported.
<b>Redundancy</b>	Yes
<b>RDA</b>	Not supported.
<b>Real time stamping</b>	Yes
<b>Browsing</b>	Not supported.
<b>Polling</b>	The driver requests the variables cyclically every 60 minutes.
<b>Spontaneous</b>	Yes, only value changes are transferred. Exception: Cyclic total update every 60 minutes.
<b>Number of PLCs</b>	One driver can connect to any number of PLCs.

**HYSTERESIS**

If an identical value, however with a more recent time stamp is received, then this value is sent by the driver as a new value in Runtime.

## 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

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



### Information

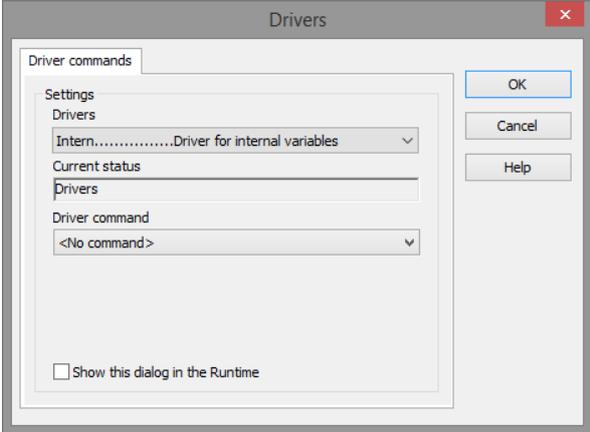
*This chapter describes standard functions that are valid for most zenon drivers. However, not all functions described here are available for every driver. For example, a driver that does not, according to the data sheet, support a modem connection also does not have any modem functions.*

### CONFIGURATION OF THE FUNCTION

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

1. Create a new function in the zenon Editor.
2. Navigate to the node **Variable**.
3. Select the **Driver commands** entry.  
The dialog for configuration is opened
4. Select the desired driver and the required command.
5. Close the dialog by clicking on **OK** and ensure that the function is executed in Runtime. Heed the notices in the **Driver command function in the network** section.

**DRIVER COMMAND DIALOG**



Option	Description
<b>Driver</b>	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
<b>Current status</b>	Fixed entry which has no function in the current version.
Driver command	Drop-down list for the selection of the command:
<No command>	No command is sent. A command that already exists can thus be removed from a configured function.
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 <code>switched off (OFF; Bit 20)</code> .
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	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	Driver is set into counting simulation mode. All values are initialized with 0.
Driver in programmed simulation mode	Driver is set into counting 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.
<b>Show this dialog in the Runtime</b>	The dialog is shown in Runtime so that changes can be made.

## 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, 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. Device Profile

The device profile for the driver is installed by the setup. You can find the file `DNP332.xml` in the following folder:

- `%ProgramData%\COPA-DATA\zenon[Version]\CommunicationProfiles\Dnp3\Driver\`

## 11. Error analysis

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

### 11.1 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 8.00 -> 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 (main.chm::/12464.htm) manual.

## 11.2 Check list

- ▶ Is the configuration of the PLC and the connection correct?
- ▶ Is the IP address of the PLC configured correctly (with connection via TCP/IP)
- ▶ Did you analyze the error file (up to zenon version 6.01) or the central log file (version zenon 6.20 and higher)?