



zenon
by COPA-DATA

zenon driver manual

3S_V3

v.8.10



COPADATA

© 2019 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

1	Welcome to COPA-DATA help.....	5
2	3S_V3.....	5
3	3S_V3 - data sheet.....	7
4	Driver history.....	9
5	Requirements	9
5.1	PC	10
5.2	PLC	11
6	Configuration	12
6.1	Creating a driver.....	13
6.2	Settings in the driver dialog.....	16
6.2.1	General.....	17
6.2.2	Settings.....	20
6.2.3	Information	30
7	Creating variables.....	30
7.1	Creating variables in the Editor.....	30
7.2	Addressing.....	34
7.3	Driver objects and datatypes.....	35
7.3.1	Driver objects	36
7.3.2	Mapping of the data types	37
7.4	Creating variables by importing.....	38
7.4.1	XML import	39
7.4.2	DBF Import/Export	40
7.4.3	Online- and Offline-Import.....	45
7.5	Communication details (Driver variables).....	51
8	Driver-specific functions.....	57
8.1	Redundancy.....	57
8.2	Response to the PLC.....	59
8.3	Byte sequence	60
8.4	Exception handling and alarming	61
8.5	RDA.....	65

8.6 Real-time capability.....	67
9 Driver command function	67
10 Error analysis	72
10.1 Analysis tool	72
10.2 Error numbers.....	73
10.3 Check list	75

1 Welcome to COPA-DATA help

ZENON VIDEO-TUTORIALS

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

GENERAL HELP

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

PROJECT SUPPORT

You can receive support for any real project you may have from our Support Team, who you can contact via email at support@copadata.com.

LICENSES AND MODULES

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

2 3S_V3

GENERAL

The **3S_V3** driver uses the PLC-Handler from CoDeSys to connect to the PLC. The PLC-Handler is an interface to controllers with software versions V2.3 and V3.



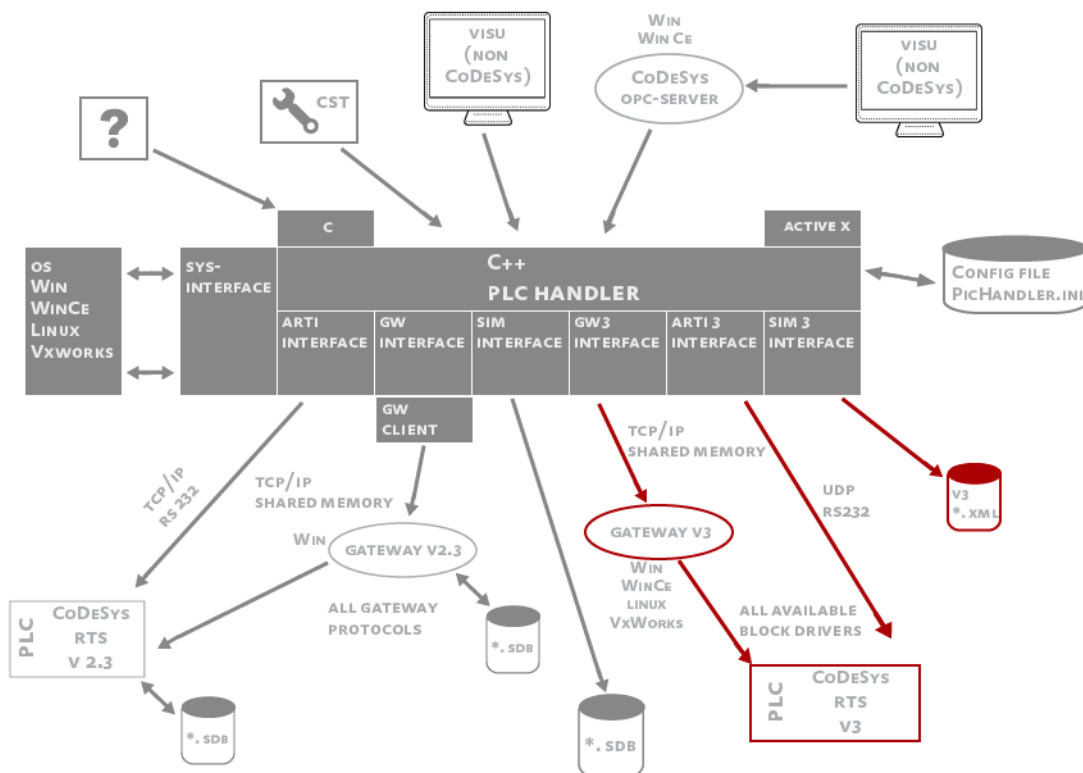
Information

From zenon 7.50, the driver supports both CoDeSys software version 2.3 as well as version 3.

PLC-HANDLER

Different versions of the PLC Handler are not compatible. The correct PLC handler for the respective driver version must always be used. PLC-Handlers of other versions must be replaced. For details see the **PC** (on page 10) chapter:

CONNECTION



Version V3 and version V2.3 controllers are addressed by means of the **node address**, which is unique in the network that can be contacted.

A connection to the control can be established.

- ▶ using ARTI (Asynchronous RunTime Interface) via the local network card
or
- ▶ a gateway on the local computer or a remote computer

The driver can address several controllers at the same time. In doing so, a separate PLC handler instance is created for each connection.

SYMBOLIC ADDRESSING

The PLC-Handler only addresses variables in the controller using its symbolic name. Addressing using the offset of the variables is not possible. However, for complex variables (structures, arrays) the whole (binary) data block of the variable can be addressed using the **base name**. This is used by the driver for RDA (on page 57).

In zenon versions up to 7.00, variables are addressed using variable identification. From version 7.10 onwards, it is possible to select whether the identification or the **symbolic address** option is used.

In order for the PLC handler to find the variables via the symbolic name in the controller, you must select the variables used in the zenon project in the CoDeSys project configuration software using a symbol configuration object and transfer this to the controller. In doing so, a symbol file in XML format is created at the same time. This can be used for offline importing of variables.

Note: The connection is broken if there are no symbols in the controller.

3 3S_V3 - data sheet

General:	
Driver file name	3S_V3.exe
Driver name	3S v3 Treiber für PLC Handler
PLC types	3S CoDeSys v3 based controllers; PAC Drive LMC from Schneider Electric. V2 controllers from version V2.3.9.33
PLC manufacturer	3S; Festo; Elau; Schneider; Wago; Bosch Rexroth

Driver supports:	
Protocol	3S-Arti; 3S-Gateway
Addressing: Address-based	Name based
Addressing: Name-based	--
Spontaneous communication	--
Polling communication	X
Online browsing	X

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

Requirements:	
Hardware PC	--
Software PC	PLC handler from 3S; the 'PLCHandlerDll.dll' file is automatically installed by the setup. The "GDrvABBTcpIpL2X.dll" file is needed if "TCP L2ABB" Gateway protocol is used for V2.3. This must be obtained from ABB.
Hardware PLC	--
Software PLC	--
Requires v-dll	X

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

4 Driver history

Date	Build number	Change
21.06.13	7.11.0.7498	Created driver documentation
17.08.15	7.50.0.21651	Driver supports: <ul style="list-style-type: none"> ▶ Selection of the origin of the time stamp ▶ Block arrays ▶ Amendable prefix for imported variable names ▶ Exception handling
30.09.15	7.20.0.21830	Driver supports and requires PLC Handler 3.5.7.0.
10.1.2016	7.50.0.25589	Additional support of CoDeSys version 2.3.

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

ADDITIONAL SOFTWARE

PC

The driver needs the **PLCHandlerDll.dll**, which is produced by 3S and contains **PLCHandler-SDK**. The required version depends on the version of zenon used:

ZENON 7.50

From version 7.50, zenon supports the communication of both V3 and V2 connections.

The following rules are applicable for a connection to the PLC with version 2:

- ▶ If the connection is established using ARTI (peer-to-peer), no additional software is required.
- ▶ If the connection is established by means of a gateway, the following files must be present in the same directory in which the driver .exe file is also located:
 - ▶ GClient.dll
 - ▶ GDrvBase.dll
 - ▶ GHandle.dll
 - ▶ GHandleStdcall.dll
 - ▶ GSymbol.dll
 - ▶ GUtil.dll

Note: The directory can also be stated with the *PATH* environment variable.

- ▶ If the gateway is to be started automatically on the driver (with the driver at the latest), the following files must be in the same directory in which the client DLLs are located for a connection to V 2 (see above):
 - ▶ Gateway.exe
 - ▶ GatewayDDE.exe
 - ▶ CommSym.dll
 - ▶ CommUsr.dll
- ▶ The version of the gateway software and the DLLs is 2.3.9.33 (or higher).

NOTE ON COMPATIBILITY

Communication to controllers with version 2 software is only possible from zenon 7.50. Note the following requirements for older versions:

From zenon version	Required PLC handler version	Notes
7.00	3.5.3.60	The PLCHandlerDll.dll must be present in the Windows folder.
7.11	3.5.4.0	The PLC Handler is installed in the Windows folder.
7.20 Build 21830	3.5.7.0	From zenon 7.20, the respective current version of the PLCHandlerDll.dll is installed automatically. The installation folder is now the same folder as that of the driver.



Information

Ensure that PLC firmware and PLC-Handler have the same version number. Only this way can correct communication be guaranteed.

Note: You can find an overview of the versions used on the computer in the driver dialog in the Information tab.

CE

No software needs to be installed for CE; **PLCHandler-SDK** has a fixed link to the driver.

Under Windows CE it is not possible to use several drivers of the same type.



Attention

CE is only supported up to version 7.20.
From version 7.50, Runtime files for 7.20 must be created.

5.2 PLC

CONTROLS

The 3S_V3 driver is to connect a PLC that uses 3S CoDeSys V3 or V2.3 software.

Examples: **3S CoDeSys**, **Control Win V3**, or **Schneider LMC PacDrive**.

The connection to the PLC is possible via

- ▶ The ARTI interface (*Asynchronous RunTime Interface*)
- ▶ A local or a remote gateway.



Information

From zenon 7.50, both CoDeSys software version 3 and version 2 are supported by the driver.

6 Configuration

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

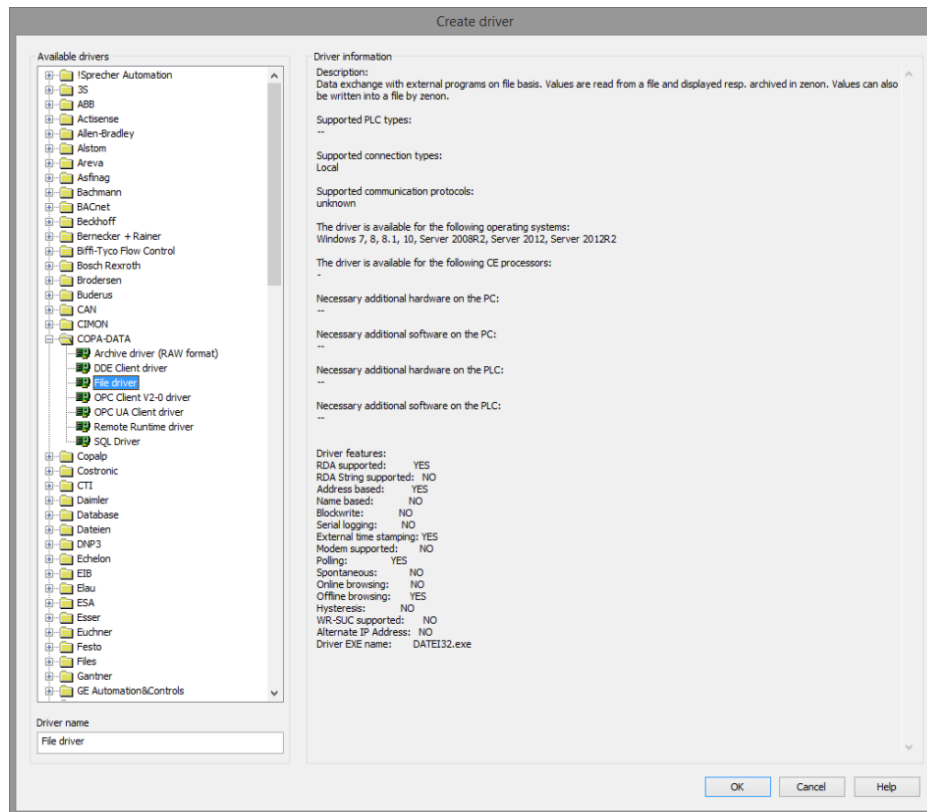


Information

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



Parameter	Description
Available drivers	<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: <i>no selection</i></p>
Driver name	<p>Unique Identification of the driver.</p> <p>Default: <i>Empty</i></p> <p>The input field is pre-filled with the pre-defined Identification after selecting a driver from the list of available drivers.</p>
Driver information	<p>Further information on the selected driver.</p> <p>Default: <i>Empty</i></p> <p>The information on the selected driver is shown in this area after selecting a driver.</p>

CLOSE DIALOG

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



Information

The content of this dialog is saved in the file called Treiber_[Language].xml. You can find this file in the following folder:

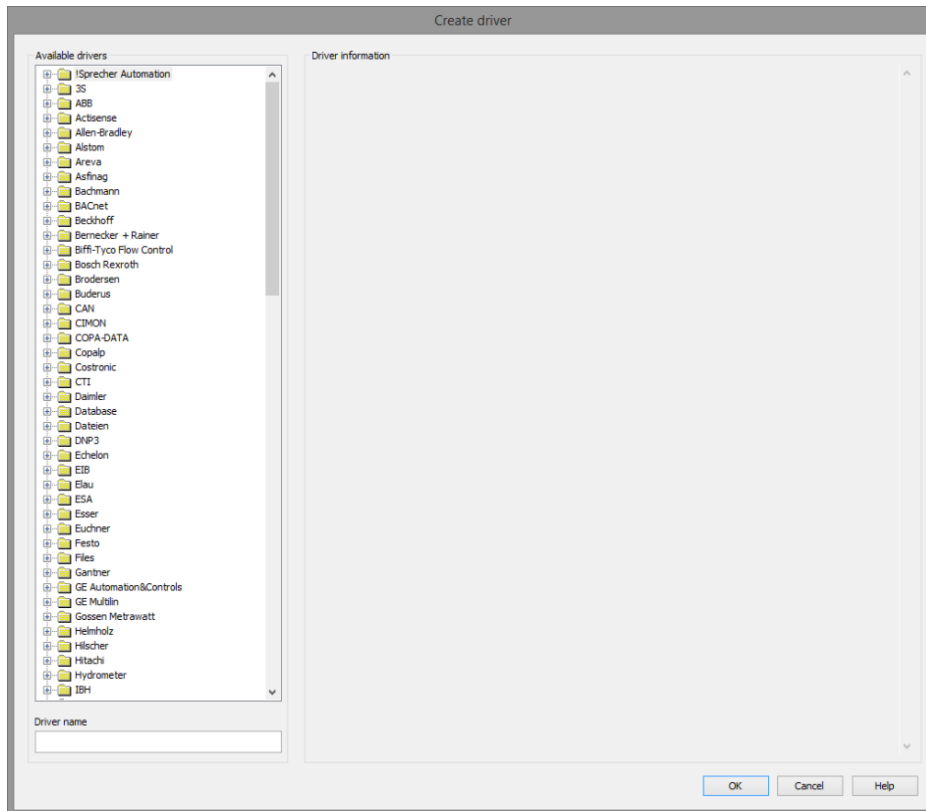
C:\ProgramData\COPA-DATA\zenon[version number].

CREATE NEW DRIVER

In order to create a new driver:

1. Right-click on **Driver** in the Project Manager and select **New driver** in the context menu.
Optional: Select the **New driver** button from the toolbar of the detail view of the **Variables**.
The **Create driver** dialog is opened.

- The dialog offers a list of all available drivers.

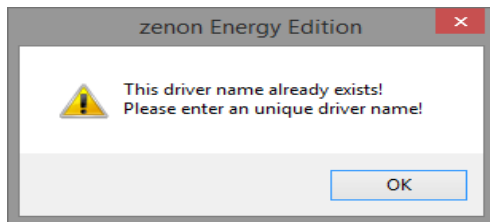


- Select the desired driver and name it in the **Driver name** input field.
This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.
The following is applicable for the **Driver name**:
 - ▶ The **Driver 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

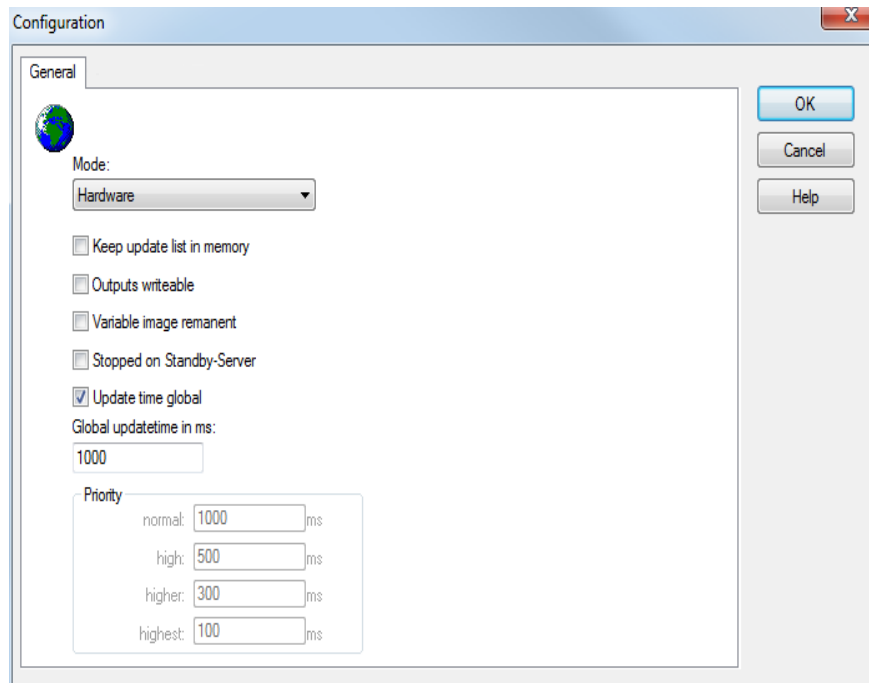
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

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
Mode	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> ▶ <i>Hardware:</i> A connection to the control is established. ▶ <i>Simulation - static:</i> 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. ▶ <i>Simulation - counting:</i> No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values

Option	Description
	<p>within a value range automatically.</p> <ul style="list-style-type: none"> ▶ <i>Simulation - programmed:</i> 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).
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. 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>
Output can be written	<ul style="list-style-type: none"> ▶ <i>Active:</i> Outputs can be written. ▶ <i>Inactive:</i> Writing of outputs is prevented. <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 hardware mode if one of these statuses is active:</p> <ul style="list-style-type: none"> ▶ User status <i>M1 (0)</i> to <i>M8 (7)</i> ▶ <i>REVISION(9)</i> ▶ <i>AUS(20)</i> ▶ <i>ERSATZWERT(27)</i> <p>The variable image is always saved if:</p> <ul style="list-style-type: none"> ▶ the variable is of the object type Driver variable ▶ the driver runs in simulation mode. (not

Option	Description
	<p>programmed simulation)</p> <p>The following states are not restored at the start of the Runtime:</p> <ul style="list-style-type: none"> ▶ <i>SELECT(8)</i> ▶ <i>WR-ACK(40)</i> ▶ <i>WR-SUC(41)</i> <p>The mode Simulation - programmed at the driver start is not a criterion in order to restore the remanent variable image.</p>
Stop on Standby Server	<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"> ▶ <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 switched off (statusverarbeitung.chm::/24150.htm) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian. <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>
Global Update time	<p>Setting for the global update times in milliseconds:</p> <ul style="list-style-type: none"> ▶ <i>Active:</i> The set Global update time is used for all variables in the project. The priority set at the variables is not used. ▶ <i>Inactive:</i> The set priorities are used for the individual variables.

Option	Description
	Exceptions: Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the Spontaneous driver update time section.
Priority	<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
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR SPONTANEOUS DRIVERS

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

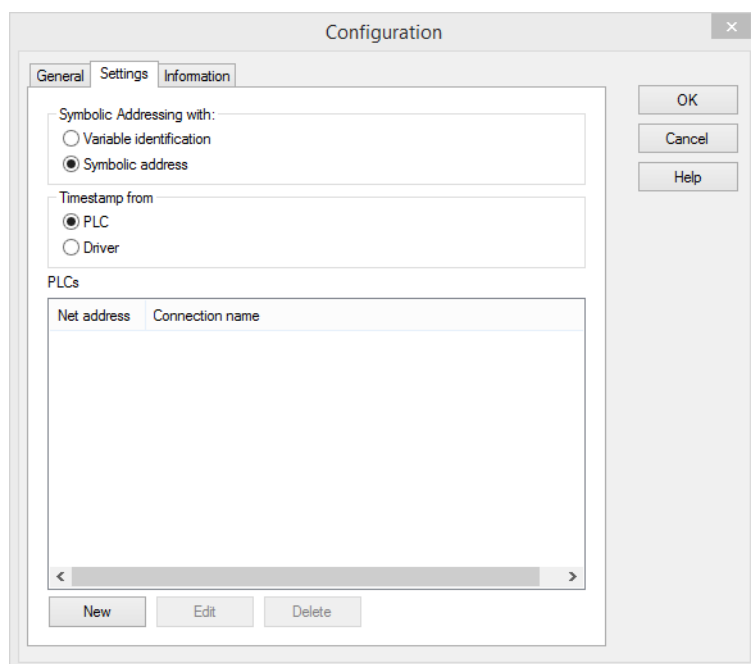
Spontaneous drivers are **ArchDrv**, **BiffiDCM**, **BrTcp32**, **DNP3**, **Esser32**, **FipDrv32**, **FpcDrv32**, **IEC850**, **IEC870**, **IEC870_103**, **Otis**, **RTK9000**, **S7DCOS**, **SAIA_Slave**, **STRATON32** and **Trend32**.

6.2.2 Settings

You configure the following in the settings tab:

- ▶ The type of symbolic addressing of the variables
- ▶ Origin of the time stamp

► Connection to the PLC



Parameter	Description
Symbolic addressing via	<p>Options field to select which source is used for the symbolic addressing of the variables:</p> <ul style="list-style-type: none"> ► Variable identification Symbolic addressing of variables is carried out using the variable identification. ► Symbolic addressing Symbolic addressing of the variables is carried out using the symbolic address. <p>Note: This selection can be made from zenon version 7.10 onwards.</p> <p>The variable identification is always used up to and including version 7.00. In this case, the options variable identification and symbolic address are not displayed.</p>
Time stamp of	<p>Options field to select how the time stamp of new variable values is determined:</p> <ul style="list-style-type: none"> ► PLC: Time stamp of the controller is used ► Driver: Time stamp of the driver is used

CONTROLS

The connections are configured by clicking on the **New** button in the dialog to configure a connection (on page 23). Existing connection configurations can be amended by selecting them and clicking on the **Edit** button.

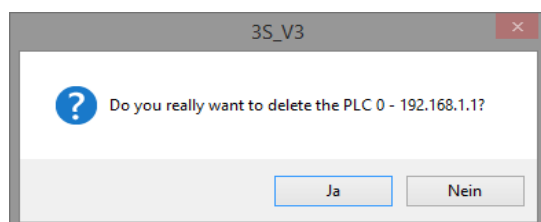
Parameter	Description
controls	<p>List of configured connections.</p> <ul style="list-style-type: none"> ▶ <i>Net address:</i> Net address of the connection, as configured for the connection. The network address must be unique. ▶ <i>Connection name:</i> Name of the connection, as configured for the connection. This connection name corresponds to the configured node address (on page 23) in the dialog (on page 23) to configure a connection. <p>The width of the columns in the list can be amended by holding down the mouse button and clicking on the column edge.</p>
New	Opens the dialog (on page 23) for configuring a connection.
Edit	Opens the dialog (on page 23) for the selected connection to configure a connection.
Delete	Deletes the selected connection after a confirmation message.

CLOSE DIALOG

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

CONFIRMATION REQUEST WHEN DELETING A CONNECTION

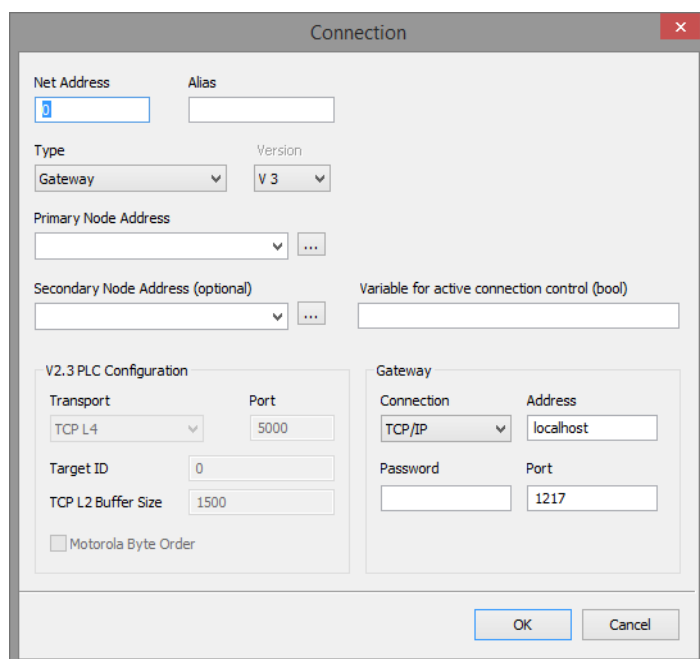
If a PLC is selected for deletion, a request for confirmation is shown before deletion.



Parameter	Description
Yes	The connection is removed from the list of configured connections (PLCs).
No	The connection is not deleted and remains in the list of configured connections. The dialog is closed.

6.2.2.1 Connection

Configuration of the connection to a PLC.



Parameter	Description
Net address	<p>Entry of the net address.</p> <p>The net address must be unique and must not be used twice. If a net address has already been issued, this is shown by a notice dialog.</p> <p>The entry is checked after clicking on the button OK. In case of failure, a warning dialog pops up.</p> <p>Attention: If a numerical figure outside the value range is entered, the net address is configured with the value -1.</p> <p>Default: The lowest net address that has not yet been issued is preset as a</p>

Parameter	Description
	<p>default value.</p> <p>This net address corresponds to the zenon Net address property.</p>
Alias	<p>An alias can be entered for each connection. This serves as a prefix for the names of the variables that are imported from the driver.</p> <p>If this alias is empty, the node address is used as a prefix.</p> <p>Maximum length: <i>8 characters</i></p> <p>Note: Entry of more than 8 characters is not possible.</p>
Connection type	<p>Selection of the connection type from drop-down list:</p> <ul style="list-style-type: none"> ▶ <i>Gateway:</i> Communication to the PLC is by means of a gateway to the local or a remote computer You can find further configuration possibilities for the gateway connection type in the Gateway configuration group. ▶ <i>ARTI</i> Direct communication to the PLC by means of the local network card (peer-to-peer connection) <p>Default: <i>Gateway</i></p>
Version	<p>Selection of the version to be used from a drop-down list:</p> <ul style="list-style-type: none"> ▶ V3 ▶ V2.3 You can find further configuration possibilities for the use of V2.3 in the V2.3 PLC configuration configuration group. <p>Default: <i>V3</i></p>
Primary node address	<p>Node address or node name of the controller.</p> <p>For redundant communication: Node address or node name of the primary controller.</p> <p>Entry of node address or node name in the field or select from a drop-down list. Clicking on the ... button opens the dialog (on page 29) to select a PLC.</p> <p>Attention: This search for a controller is not available for the V2.3 communication. The search through the network is not possible with V2.3.</p> <p>When opened, a search for controllers in the network that can be contacted using the selected connection type is carried out. If the search</p>

Parameter	Description
	<p>for a controller is not possible, because the project computer and controller are in different networks, the address must be entered manually.</p> <p>Redundancy:</p> <p>The driver connects to the primary connection by default. If this cannot be established or fails, an attempt to switch to the secondary connection is made. The primary connection always has higher priority in the process. If both are available, communication is always via the primary address.</p> <p>Note: In doing so, all variables are always only signed in for the active connection. Values are always only read by the active connection. Values are however written to both controllers if the connections are available.</p> <p>Node address and node name</p> <ul style="list-style-type: none"> ▶ The node address can be different for different connection types. However it is always unique for the selected connection type. Note: A controller may get a new node address after a restart, because the original address is issued. ▶ The node name is not unique There can be several controllers on one device for example. These can then only be identified via the node address.
Secondary node address	<p>Node address or node name of the redundant controller.</p> <p>Entry of a node address or a node name in the field or select from a drop-down list. Clicking on the ... button opens the dialog (on page 29) to select a PLC.</p> <p>Note: The secondary node address can also remain empty. In this case, in the event of a failure of the primary connection, no redundancy switching is carried out.</p>
Variable to control the active connection	<p>Name of the variable with which the communication connection is selected.</p> <p>This variable must be configured by the connection status in the zenon Editor.</p> <p>Note:</p> <ul style="list-style-type: none"> ▶ Enter the symbolic address of the variable. This corresponds to the Symbolic address variable property in the <CD_PRODUCDTNAME> Editor. In principle, it is sufficient if this variable is configured on both

Parameter	Description
	<p>controllers.</p> <ul style="list-style-type: none"> ▶ The variable must be of the data type <i>BOOL</i>. <p>If the secondary node address is empty, this option has no meaning.</p>

V2.3 PLC CONFIGURATION

Configurations for communication with the 3S V3 driver to a PLC with V 2.3 software.

If **Version** V3 is selected in the property, the properties of the V2.3 PLC configuration are grayed out.

Parameter	Description
Transport	<p>Drop-down list for the selection of the communication protocol.</p> <ul style="list-style-type: none"> ▶ <i>TCP L4</i> ▶ <i>TCP L2 Route</i> ▶ <i>TCP L2</i> ▶ <i>TCP L2ABB</i> <p>Note: Only supports ABB controllers. For the configuration, GDrvABBTcpIpL2X.dll must be present on the system and in the same folder as Gateway.exe. The DLL must be obtained from the manufacturer, ABB, directly. For further notes, see also the checklist (on page 75).</p> <p>Default: <i>TCP L4</i></p>
Port	<p>Communication port number.</p> <p>Default: <i>5000</i></p> <p>The entry is checked after clicking on the button OK. In case of failure, a warning dialog pops up.</p>
Target ID	<p>Unique ID of the target device.</p> <p>Default: <i>0</i></p> <p>The entry is checked after clicking on the button OK. In case of failure, a warning dialog pops up.</p>
TCP L2 Buffer Size	<p>Buffer size of the TCP L2 protocol.</p> <p>Is only used for the TCP L2 communication</p>

Parameter	Description
	<p>protocols. This property has no effect on TCP L4.</p> <p>Default: 1500</p> <p>The entry is checked after clicking on the button OK. In case of failure, a warning dialog pops up.</p>
Motorola Byte Order	<p>Byte sequence for the receiving or sending of data</p> <ul style="list-style-type: none"> ▶ <i>Activated:</i> Data is processed in Big-Endian (Motorola) format. ▶ <i>Not activated:</i> Data is processed in Little-Endian (Intel) format. <p>Note: The byte order is also automatically detected on the basis of the RDA type. You can find further information on this in the Byte sequence (on page 60) chapter.</p>

GATEWAY

Configuration of the connection to the Gateway.

If the **ARTI connection type** property has been selected, the properties of the gateway configuration have been grayed out.

Parameter	Description
Connection	<p>Selection of the connection type from the PLC handler to the gateway via a drop-down list:</p> <ul style="list-style-type: none"> ▶ <i>TCP/IP</i> TCP/IP connection ▶ <i>Shared Memory</i> Use of the shared memory on the local computer <p>Default: <i>TCP/IP</i></p>
Address	<p>Input field for the address of the gateway.</p> <p>The address can be configured as both numerical (192.168.1.1) and alphanumerical (e.g.: <i>localhost</i>).</p> <p>A check to see if the input is valid is not carried out.</p>

Parameter	Description
	Note: Only available if <i>TCP/IP</i> is configured as a connection .
Password	Password of the gateway (optional). Note: Only available if <i>TCP/IP</i> is configured as a connection .
Port	Port of the gateway. Note: Only available if <i>TCP/IP</i> is configured as a connection . Default: 1217

COMMUNICATION SETTINGS

Allows the configuration of the communication behavior of a connection.

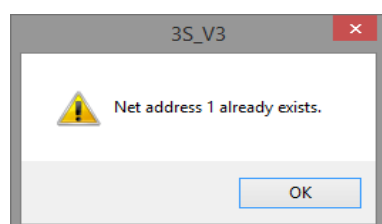
Parameter	Description
Timeout	Defines the timeout of the communication in ms. Permitted settings for this timeout are in the area from 1ms to 1000000ms. Default: 5000ms
Repeat attempts	Defines the number of repeat attempts when receiving incoming data before a COMM_FATAL error is signaled. Permitted settings for this value are in the range of 1 to 10. Default: 3

CLOSE DIALOG

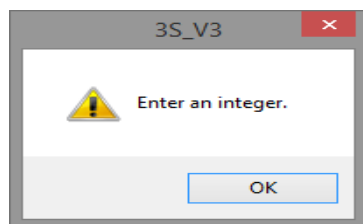
Parameter	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

WARNING DIALOG IN THE EVENT OF INCORRECT INPUT

Net address already present:

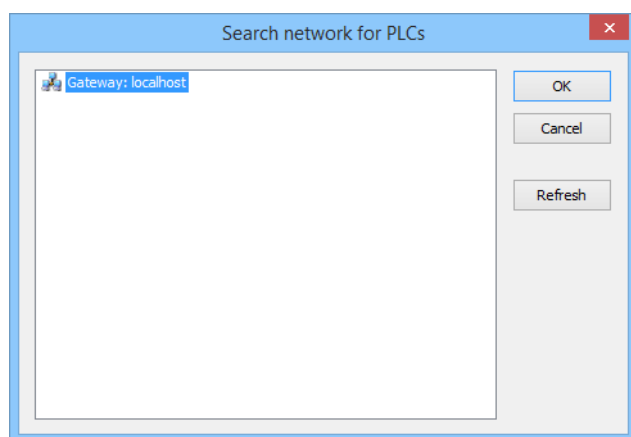


If an invalid entry is projected (such as letters), it will also be shown in a warning dialog.



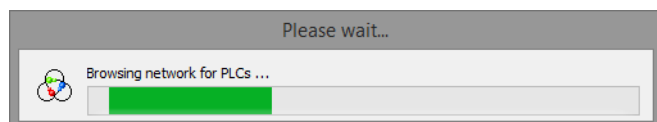
6.2.2.1.1 Search controller in the network

Clicking on the **Node address** button opens the dialog to select a PLC. A search for PLCs is carried out on opening. Controllers that have been found are offered for selection.



Parameter	Description
List of controllers	Lists all controllers that have been found. With a connection via gateway, the address configured in the connection is shown as a main node.
OK	Accepts the selected PLC and closes the dialog.
Cancel	Discards selection and closes the dialog.
Reload	Searches for controllers again.

SEARCHING THE NETWORK

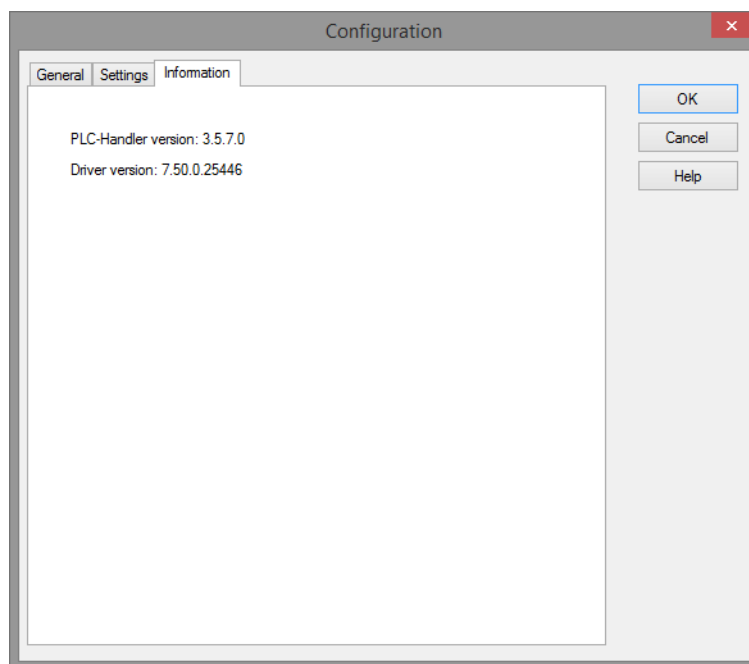


The searching of a network can - depending on the design and size of network - take some time. A dialog with a progress bar is visualized during this time.

6.2.3 Information

Information on the versions of:

- ▶ PLC-Handler
- ▶ Drivers



7 Creating variables

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

7.1 Creating variables in the Editor

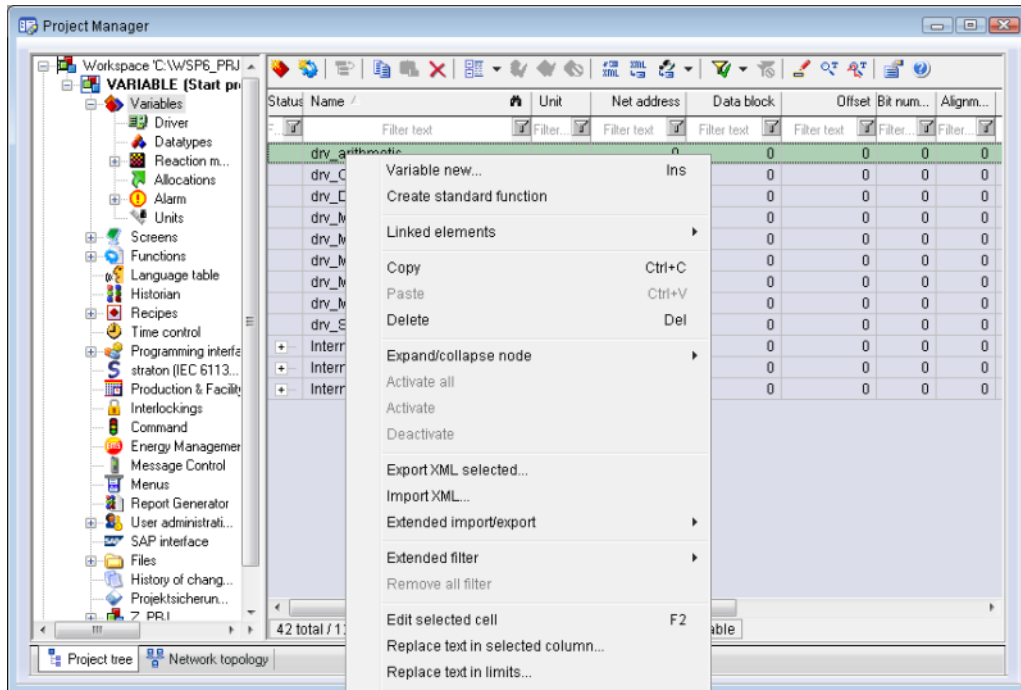
Variables can be created:

- ▶ 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:

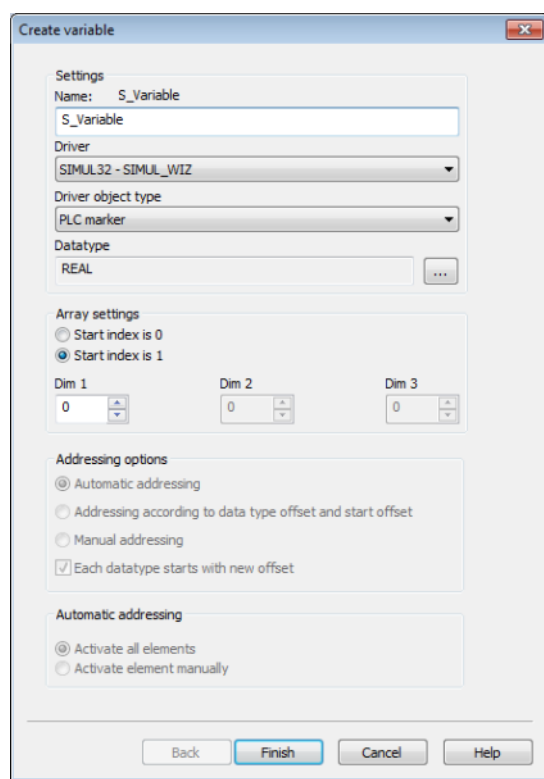
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

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

CREATE VARIABLE DIALOG



Property	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> <p>Note: For some drivers, the addressing is possible over the property Symbolic address, as well.</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

Property	Description
	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 addressing	Expanded settings for arrays and structure variables. You can find details in the respective section.

SYMBOLIC ADDRESS

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

Maximum length: 1024 characters.

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

- ▶ 3S_V3
- ▶ AzureDrv
- ▶ BACnetNG
- ▶ IEC850
- ▶ KabaDPsServer
- ▶ OPCUA32
- ▶ Phoenix32
- ▶ POZYTON
- ▶ RemoteRT
- ▶ S7TIA
- ▶ SEL
- ▶ SnmpNg32
- ▶ PA_Drv

INHERITANCE FROM DATA TYPE

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

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

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

7.2 Addressing

Group/Property	Description
General	Property group for general settings.
Name	<p>Freely definable name.</p> <p>Attention: For every zenon project the name must be unambiguous.</p>
Identification	<ul style="list-style-type: none"> Up to zenon Version 7.00: Symbolic address From version 7.10 on: Either the Symbolic address or a freely-selectable identification, for example for a resource label, comments...
Addressing	Property group for addressing
Net address	<p>Network address of variables.</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	not used for this driver
Alignment	not used for this driver
Bit number	not used for this driver
String length	<p>Only available for String variables.</p> <p>Maximum number of characters that the variable can take.</p>
Symbolic address	<p>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.</p> <p>Maximum length: 1024 characters.</p>

Group/Property	Description
Driver connection/Driver Object Type	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Driver connection/Data Type	<p>Data type of the variable. Is selected during the creation of the variable; the type can be changed here.</p> <p>Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.</p>
Driver connection/Priority	<p>Setting the priority class. The variable of the priority class is thus assigned as it was configured in the driver dialog in the General tab. The priority classes are only used if the global update time is deactivated.</p> <p>If the global update time option is activated and the priority classes are used, there is an error entry in the log file of the system. The driver uses the highest possible priority.</p>

BLOCK ARRAYS

Block arrays can be read, written and imported. The symbol name without indexes is used. The # character is not permitted.

Example: **MAIN.MyArray**.

The number of indexes must correspond to that in the PLC.

Example: **For MyArray: ARRAY [0..99] OF INT;**

Note: In the variable properties of zenon, the **Block array size** property in the **Additional settings** group must be set to the value *100*.



Attention

Note: In zenon, counting is started with 0, but with 1 in the PLC.

7.3 Driver objects and datatypes

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

7.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Description
Alarm Event	64	X	X	<i>INT, USINT</i>	Used for Exception Handling (on page 57)
Alarm Variable	65	X	--	<i>BOOL, DINT, STRING, UDINT</i>	Used for Exception Handling (on page 57)
Connection status	43	X	--	<i>BOOL, USINT</i>	<p>Configured variable (connection) denotes which connection is active if both connections are available.</p> <ul style="list-style-type: none"> ▶ 0: Primary connection ▶ 1: Secondary connection ▶ 3: Bit value 0 ▶ 4: Bit value 1 <p>Note that further information in relation to this in the Redundancy (on page 57) chapter.</p>
PLC marker	8	X	X	<i>BOOL, USINT, SINT, BYTE, UINT, WORD, INT, UDINT, DWORD, DINT, ULINT, LINT, LWORD, REAL, LREAL, STRING, WSTRING, DT, TOD, DATE, TIME</i>	

Driver Object Type	Channel type	Read	Write	Supported data types	Description
Status Feedback	66	X	--	<i>DINT</i>	Used for Exception Handling (on page 57)
<i>Communication details</i>	35	X	X	<i>BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING</i>	<p>Variables for the static analysis of the communication; is transferred between driver and Runtime (not to the PLC).</p> <p>Note: 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 51) chapter.</p>

Key:

X: supported

--: not supported

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
BIT, BOOL	BOOL	8
USINT	USINT	9
BYTE	BYTE	22
SINT	SINT	10
WORD	WORD	23
UINT, UINT16	UINT	2

Control	zenon	Data type
INT, INT16, ENUM	INT	1
UDINT	UDINT	4
DINT	DINT	3
ULINT	ULINT	27
DWORD	DWORD	24
LINT	LINT	26
REAL	REAL	5
LREAL	LREAL	6
STRING	STRING	12
WSTRING	WSTRING	21
DATE	DATE	18
TIME	TIME	17
DT, DATE_AND_TIME	DT	20
TOD, TIME_OF_DAY	TOD (Time of Day)	19

DATA TYPE

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

7.4 Creating variables by importing

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



Information

You can find details on the import and export of variables in the Import-Export (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. The import of variables is described in the following documentation. Data types are imported along the same lines.

REQUIREMENTS

The following conditions are applicable during import:

- ▶ **Backward compatibility**
At the XML import/export there is no backward compatibility. Data from older zenon versions can be taken over. The handover of data from newer to older versions is not supported.
- ▶ **Consistency**
The XML file to be imported has to be consistent. There is no plausibility check on importing the file. If there are errors in the import file, this can lead to undesirable effects in the project.
Particular attention must be paid to this, primarily if not all properties exist in the XML file and these are then filled with default values. E.g.: A binary variable has a limit value of 300.
- ▶ **Structure data types**
Structure data types must have the same number of structure elements.
Example: A structure data type in the project has 3 structure elements. A data type with the same name in the XML file has 4 structure elements. Then none of the variables based on this data type in the file are imported into the project.

Hint

You can find further information on XML import in the **Import - Export** manual, in the **XML import (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
KANALNAME	Character	128	Variable name. The length can be limited using the MAX_LAENGE entry in the project.ini file.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered

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

Identification	Type	Field size	Comment
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
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.

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



Attention

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

LIMIT VALUE DEFINITION

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

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

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

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

7.4.3 Online- and Offline-Import

The import of variables from symbolic variable names can be selected as either online from the controller or offline via a symbol file.

REQUIREMENTS FOR THE IMPORT

In order for variable import to work, the variables used in the zenon project must be selected in the **CoDeSys** project configuration software using a symbol configuration object and transferred to the controller. During this process, a symbol file is created in XML format at the same time, which can be used for offline variable import.

Depending on the settings for whether the **VariableIdentification** or the **Symbolic address** is to be used for addressing, the **symbolic name** is entered in the corresponding property.

VARIABLE NAME

The imported **symbolic name** with a prefix is entered as a variable name.

This prefix is:

- ▶ An alias of the configured connection
- ▶ The node address of the configured connection if no alias has been configured.

Note: You create this configuration in the driver dialog to create a new connection (on page 23).

It is thus guaranteed that the variable name also remains unique if there are the same symbol names in different controllers. The variable name can therefore not be used for symbolic addressing.

ENUM DATA TYPE - IMPORT FROM XML SYMBOL FILE

The following is applicable for the import of ENUM data types from an XML symbol file:

- ▶ If *typeclass=ENUM* is defined for a symbol type and a *basetype* is given for this, the variable is created in zenon with this *basetype*.
- ▶ If no *basetype* is given, the INT data type is used for the variable in zenon.

7.4.3.1 Importing variables with the wizard

To import variables from the PLC directly:

1. select **Import variables from driver** from the context menu of the driver
The import wizard is opened.
2. In the dialog, select the import source:
 - ▶ PLC:
The variables are imported from the PLC directly.
 - ▶ XML symbol file
The variables are imported from an XML file.
3. click on **Next**.
4. Select the controller and, for offline import, the XML file.
The list of controllers corresponds to the configuration of the driver configuration (as configured in connection (on page 23)).
5. click on **Next**.

6. configure the filter settings.
You can find further information on this in the description of the filter dialog.
7. Click on **Finish**.
The selection list is displayed
8. Select the variables to be imported.
9. Click on **OK**.

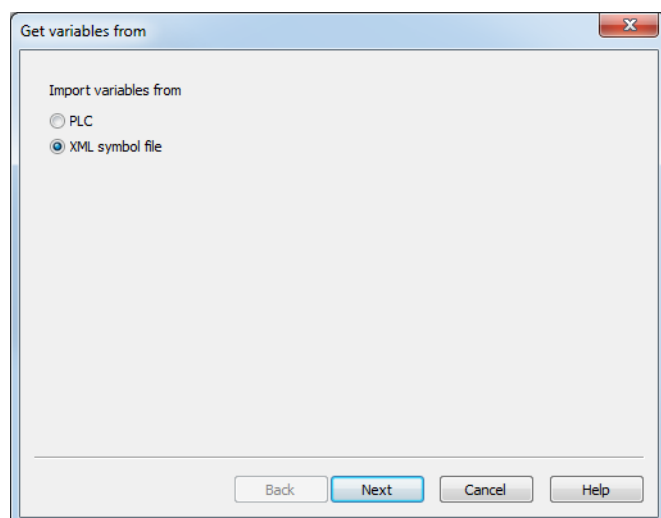
The variables are imported If an error occurs, a corresponding message is displayed.

7.4.3.2 Online- and Offline-Import

You select the desired options during the import process with the import wizard.

SELECTING THE IMPORT SOURCE

You select the import source for the variable import in this dialog.

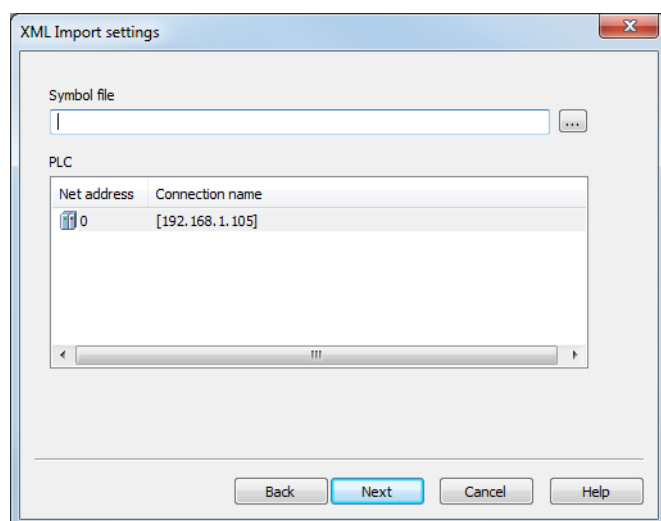


Parameter	Description
Import variables from	<p>Option field to select the import source:</p> <ul style="list-style-type: none"> ▶ <i>PLC</i>: Variables are always imported from a controller. ▶ <i>XML symbol file</i>: Variables are imported from an XML file created with the CoDeSys software.
Next	Accepts setting and switches to the next window.

Parameter	Description
Cancel	Cancels the import. No variables are imported.
Help	Opens online help.

SOURCE OF XML FILE: SELECTION OF THE XML FILE AND THE CONTROLLER

In this dialog, you select the source of the variables to be imported. The file is also given when importing from an XML symbol file.

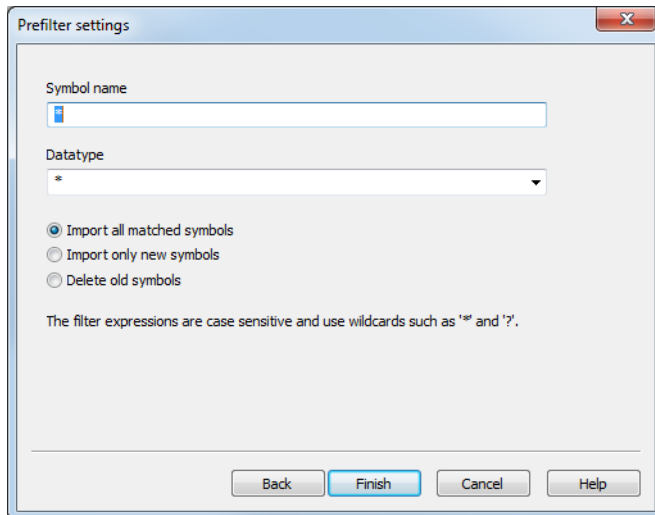


Parameter	Description
Symbol file	Selection of the symbol file. Only present if XML symbol file has been selected as a source when selecting the import source.
PLC	Selection of the controller, if variables are imported, from a list. The list of controllers corresponds to the configuration of the driver configuration (as configured in connection (on page 23)).
Back	Switches to the previous window.
Next	Accepts setting and switches to the next window. Note: If no connection is configured or selected, it is not possible to switch to the next window.
Cancel	Cancels the import. No variables are imported.
Help	Opens online help.

CONFIGURATION OF THE FILTER

When importing a variable, the variables to be imported can be filtered.

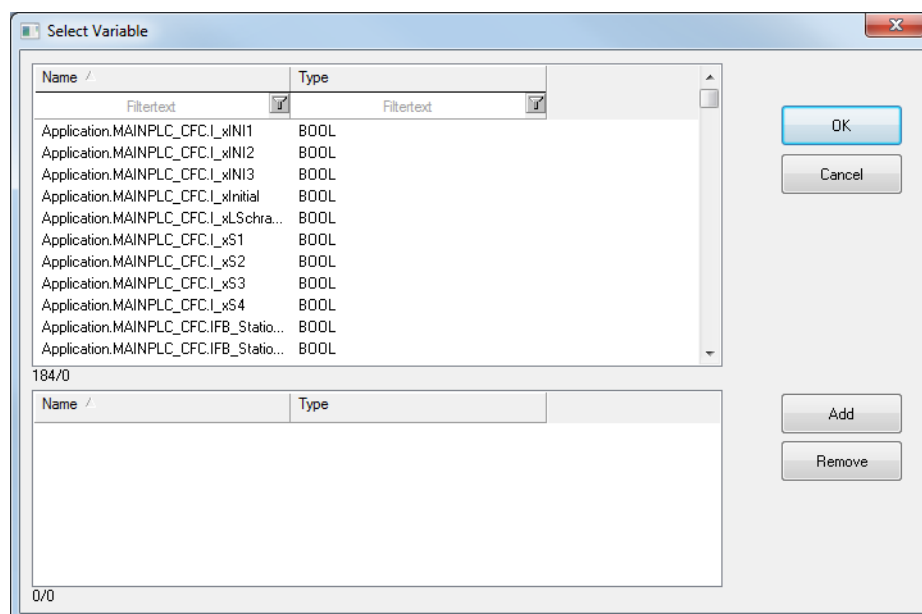
Either all existing or only new variables are offered for selection. In addition, variables that are no longer needed can be searched for and deleted.



Parameter	Description
Symbol name	<p>Entry of the symbol name as a filter criteria.</p> <p>The name can be entered with the following parameters:</p> <ul style="list-style-type: none"> ▶ * ▶ ? <p>Default: *</p> <p>Selects all symbol names, no prefiltering.</p>
Data type	<p>Entry of a data type in the combobox or selection from a drop-down list.</p> <p>The following placeholders can be used for input:</p> <ul style="list-style-type: none"> ▶ * ▶ ? <p>Default: *</p> <p>Selects all data types, no prefiltering.</p>
<i>Prefilter</i>	<p>Option field to prefilter the displayed variables in the subsequent dialog (variable selection dialog):</p> <ul style="list-style-type: none"> ▶ <i>Import all matched symbols</i> Offers all symbols that correspond to the filter for import.

Parameter	Description
	<ul style="list-style-type: none"> ▶ <i>Import only new symbols</i> Offers only symbols that are not yet present in the zenon project for import. ▶ <i>Delete old symbols</i> Offers all symbols that exist in the zenon project that do not exist in the controller for deletion.
Back	Switches to the previous window.
Finish	Applies setting and opens the variable selection dialog.
Cancel	Cancels the import. No variables are imported.
Help	Opens online help.

VARIABLE SELECTION DIALOG



Parameter	Description
List of variables that can be imported	<p>Displays all variables that can be imported or deleted depending on the settings in the wizard.</p> <ul style="list-style-type: none"> ▶ <i>Name</i> Name of the variable as defined in the controller or in the XML symbol file. ▶ <i>Type</i> Type of variable as defined in the controller or in the XML symbol file.

Parameter	Description
	<p>► Access</p> <p>The list can be sorted and filtered.</p>
List of variables to be imported	Displays all variables that are selected for import or deletion.
Add	Adds the variables highlighted in the Importable variables list to the Variables to be imported list. Multiple selection from the list is possible.
Remove	Deletes highlighted variables from the Variables to be imported list. Multiple selection from the list is possible.
OK	Accepts selection and imports or deletes variables.
Cancel	Cancels the import and closes the wizard. No variables are imported.

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** and can be imported from there.

Path to file: %ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables

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



Information

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

For example:

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

Name from import	Type	Offset	Description
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 <i>driver stop</i> , the variable has the value <i>TRUE</i> and an OFF bit. After the driver has started, the variable has the value <i>FALSE</i> and no OFF bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.
ConnectionStates	STRING	61	Internal connection status of the driver to the PLC. Connection statuses: 0: Connection OK 1: Connection failure 2: Connection simulated Formating: <Netzadresse>:<Verbindungszustand>;...;; A connection is only known after a variable has first signed in. In order for a connection to be contained in a string, a variable of this connection must be signed in once. The status of a connection is only updated if a variable of the connection is signed in. Otherwise there is no communication with the corresponding controller.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	<i>BOOL</i>	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	<i>BOOL</i>	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	<i>BOOL</i>	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings PhoneNumberSet and ModemHwAdrSet .
PhoneNumberSet	<i>STRING</i>	38	Telephone number, that should be used
ModemHwAdrSet	<i>DINT</i>	39	Hardware address for the telephone number
GlobalUpdate	<i>UDINT</i>	3	Update time in milliseconds (ms).
BGlobalUpdaten	<i>BOOL</i>	4	TRUE, if update time is global
TreiberSimul	<i>BOOL</i>	5	TRUE, if driver in sin simulation mode
TreiberProzab	<i>BOOL</i>	6	TRUE, if the variables update list should be kept in the memory
ModemActive	<i>BOOL</i>	7	TRUE, if the modem is active for the driver
Device	<i>STRING</i>	8	Name of the serial interface or name of the modem
ComPort	<i>UINT</i>	9	Number of the serial interface.
Baudrate	<i>UDINT</i>	10	Baud rate of the serial interface.
Parity	<i>SINT</i>	11	Parity of the serial interface
ByteSize	<i>USINT</i>	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.

Name from import	Type	Offset	Description
StopBit	<i>USINT</i>	13	Number of stop bits of the serial interface.
Autoconnect	<i>BOOL</i>	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	<i>STRING</i>	17	Current telephone number
ModemHwAdr	<i>DINT</i>	21	Hardware address of current telephone number
RxIdleTime	<i>UINT</i>	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	<i>UDINT</i>	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	<i>UDINT</i>	20	Number of ringing tones before a call is accepted
ReCallIdleTime	<i>UINT</i>	53	Waiting time between calls in seconds (s).
ConnectTimeout	<i>UINT</i>	54	Time in seconds (s) to establish a connection.

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	<i>UDINT</i>	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	<i>UDINT</i>	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	<i>UDINT</i>	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	<i>UDINT</i>	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	<i>UDINT</i>	33	Number of writing errors
ReadSucceedCount	<i>UDINT</i>	35	Number of successful reading attempts
MaxCycleTime	<i>UDINT</i>	22	Longest time in milliseconds (ms) required to

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

ERROR MESSAGE

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

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

8 Driver-specific functions

The driver supports the following functions:

8.1 Redundancy

The following types of communication are possible for the 3S_V3 driver:

- ▶ Only primary connection configured
If only the primary connection is configured, communication with the controller is via one individual connection. No communication is possible if this fails.
Configuration in the driver dialog: Primary node address option
- ▶ Primary and secondary address configured:
The driver connects to the primary connection by default. If this connection cannot be established, the connection is established with the secondary connection. If the primary connection to a PLC fails, communication is with the secondary connection.
The primary connection always has priority. If both configured connections are available, the primary connection is always used.
Configuration in the driver dialog: Primary node address and Secondary node address options
In doing so, all variables are always signed in for the active connection (*advised*). This means that values are always only read by the active connection. Values are always written to both connections.

- ▶ In doing so, it should be noted that that value of the variable is always read by the active connection. Both controllers must therefore be externally synchronized and have the corresponding logic implemented.
- ▶ Variable and primary and secondary connection are configured:
If both connections have been configured and are available, all connection statuses with a **connection status** driver object variable can be read:

CONNECTION STATUS DRIVER OBJECT TYPE

Composition of the *USINT* data type:

Bit position	Bit value 0	Bit value 1
0	No primary connection	Primary connection established
1	No secondary connection	Secondary connection established
2	Primary connection inactive	Primary connection active
3	Secondary connection inactive	Secondary connection active

With the *BOOL* data type, the respective status bit can be read directly by stating the corresponding bit number



Attention

When redundancy and exception handling are used at the same time, the **status feedback** driver object variables are always written to both controllers at the same time to respond to the start-up status.

The alarms are always only read by the active connection however.

You should therefore ensure that the memory blocks for the exception handling are synchronized in the controller in such a way that no alarms are lost or reported twice when the connection is switched.

The establishment of a connection to a redundant controller is possible due to the (optional) provision of an alternative (secondary) node address. In addition, the active connection can be selected through a Boolean variable (also optional)

CONNECTION STATUS DRIVER OBJECT TYPE

To display the status of the respective connection, a **connection status** driver object type variable is used.

For the USINT data type, it is composed as follows:

Bit	Bit value 0	Bit value 1
0	No primary connection	Primary connection established
1	No secondary connection	Secondary connection established
2	Primary connection inactive	Primary connection active
3	Secondary connection inactive	Secondary connection active

With the BOOL data type, the respective status bit can be read directly by stating the corresponding bit number

8.2 Response to the PLC

Whilst the connection is being established, the PLC should prevent values changing in the event array.

To do this, create, for each connection to a PLC, a **status feedback** object type variable in zenon. This reports the current run-up status to the PLC. These variables can take the following values:

- ▶ *-1: Not Running*
Connection terminated.
Before closing the connection properly (when closing Runtime), the status feedback variable is set to Not Running.
- ▶ *0: Initial*
Connection reestablished, start of restart.
The value only occurs when reestablishing the connection, not when starting Runtime. At this

point in time, there are not yet any variables requested by Runtime, so there value can therefore not be set

- ▶ *10: Starting*
During the request of variables at the PLC.
- ▶ *100: Running*
Once the initial values have been received by the PLC.

The PLC should prevent new events whilst the *status feedback* variable has the value *Starting*.

8.3 Byte sequence

The **3S V3** driver processes data with the Intel *Byte Order*. Data that is received from the PLC in Motorola byte order is amended if the **Motorola Byte Order** is activated in the configuration dialog of the connection (on page 23).

In addition, the byte order can be determined automatically using the RDA type.

The driver automatically uses the RDA type to detect the byte order in which the data is received from the PLC and converts this internally for processing the data in zenon. There is also a corresponding amendment of the byte sequence when communicating with the PLC.

DETECTION OF THE BYTE SEQUENCE ON THE BASIS OF THE RDA TYPE

The following is applicable for the automatic detection of the byte order by the **3S V3** driver:

- ▶ Intel byte order
RDA type: 1, 2, 3 or 4
Header and data are in Intel format. Data is processed by the **3S V3** driver directly. In doing so, no additional adaptation by the driver is necessary.
- ▶ Motorola byte order
Header and data are in Motorola format. In addition, the **Motorola Byte Order** is activated in the configuration dialog for the connection.
RDA type: 0x1000000, 0x2000000, 0x3000000 or 0x4000000
Header and data are in Motorola format and are amended internally by the driver.
- ▶ Motorola byte order – byte order amended by PLC
The **Motorola Byte Order** option is activated in the configuration dialog for the connection. In addition, the byte sequence is already amended to the Intel byte sequence by the PLC.
RDA type: 1, 2, 3 or 4
Header and data are communicated by the PLC in Intel format. Data is processed by the **3S V3** driver directly. In doing so, no additional adaptation by the driver is necessary.

BYTE ORDER - OVERVIEW

Intel	Motorola
1	<i>0x1000000</i>
2	<i>0x2000000</i>
3	<i>0x3000000</i>
4	<i>0x4000000</i>



Attention

For the 3S V3 RDA mechanism, the size of the trigger in the PLC also determines the size of the data array. Both sizes must be the same for flawless communication. The four header fields are always 4 bytes large. However, some PLCs do not allow mixed addressing. As a result, there can be problems if the trigger variable is less than 4 bytes. In this case, the header and data are read incorrectly.

8.4 Exception handling and alarming

Error events can be reported to zenon by the controller with the help of exception handling and assigned to an alarm variable using an ID.

For the replacement of error events, several blocks in the following form are created - in an array for example:

```
TYPE ST_ISA_ALARM :
```

```
STRUCT
```

```

    ID : DINT;                // identification for error location and type
    Value : DINT;             // error value, code, ...
    Message : STRING(80);     // error message
    TimeEvent.AlmDate : DATE; // date received (UTC)
    TimeEvent.AlmTime : TIME_OF_DAY; // time received (UTC)
    TimeAck.AlmDate : DATE;   // date cleared (UTC)
    TimeAck.AlmTime : TIME_OF_DAY; // time cleared (UTC)
    AlarmState : USINT;       // alarm status (received/cleared)
    ComState : INT;           // Handshake

```

```
END_STRUCT
```

```
END_TYPE
```

For this, the following applies:

- ▶ The names of the structure fields can be issued freely.
- ▶ The form of the structure (data types) must be retained.
Otherwise the data of an event can only be read with errors or incompletely.

Procedure:

- ▶ When an error occurs, the controller assigns any desired free block (handshake variable = 2) with the information on the event and then sets its handshake variable to 1. This signals to the driver that the block in question contains new data.
- ▶ When switching the handshake variable to 1, the driver reads the complete block and assigns the content using the alarm variables established via the ID in zenon.
- ▶ The symbol address of the block is determined from the symbol address of the handshake variables by cutting off the part after the last item.
For example: Symbol address of the handshake variable is **Alarm[8].ComState**. This results in the symbol address of the block being **Alarm[8]**.
- ▶ The driver then sets the handshake variable to 2. This signals to the controller that the block is free again and can be used for new events.
- ▶ If an error occurs during evaluation, the driver sets the handshake variable to the value 4 and thus signals the error status to the controller.
Errors are, for example: no alarm variable linked to the ID, error when reading, etc.

ALARM STATUS

The content of the **AlarmState** field determines whether the alarm is received or cleared. Regardless of this, the value and the time stamp (in UTC) of the attendant alarm variables are determined:

- ▶ 0: Empty.
No alarm, all other entries of the block are invalid.
- ▶ 1: Alarm received.
The alarm variable gets the value 1 and the time stamp from **TimeEvent.AmlDate** and **TimeEvent.AmlTime**.
- ▶ 2: Alarm cleared:
The alarm variable gets the value 0 and the time stamp from **TimeAck.AmlDate** and **TimeAck.AmlTime**.

HANDSHAKE

For the signaling of new events, a zenon variable with the object type **Alarm Event** must be created in the zenon project for the handshake variable of each event block. Handshake variables are distinguished from other variables in zenon using this object type.

In order for the mechanism to work, the handshake variables must be requested for the complete duration of zenon Runtime. This can be effected using the **Permanently read variable** property, for example.

The handshake variables can have the following values:

- ▶ 0: Locked.
The controller must not use the block for new events.
- ▶ 1: Read requested.
The controller has allocated the block with new values and is making a request to read the block.
- ▶ 2: Read successful.
The driver has read and evaluated the block successfully. The controller can use the block for new events again.
- ▶ 4: Read error.
If an error occurs hen evaluating the block, the driver sets the handshake variable to 4.

ALARM VARIABLES

Alarm variables are for displaying the events from the controller zenon as alarms. They are distinguished from other variables of the driver in zenon using the *alarm variable*. They are linked to an event in the controller using the ID. The ID denotes the location and type of error.

Up to four variables can be linked to each ID: A variable that is always requested with a limit value defined as an alarm as well as three further optional variables for further information about the alarm. These variables can then be used, for example, in **dynamic limit value texts** of the alarm variables.

For linking, the symbol addresses of the variables must follow a fixed format:

Address	Type	Mandatory	Description
Exception.<ID>.Alarm	BOOL or numerical	X	Signals the alarm using a limit value of 1 defined as an alarm. If this variable is not present or not requested, this is considered an error and the driver sets the handshake variable to the value 4.
Exception.<ID>.Value	Numerical	--	Gets the value of the Value field of the event block and can, for example, be used to denote an error code
Exception.<ID>.Message	STRING	--	Gets the value of the Message field of the event block and can

Address	Type	Mandatory	Description
			take an error text
Exception.<ID>.Info	STRING	--	<p>Gets a summary of the event in the following format:</p> <p><ID>;<Value>;<Message>;<AlarmState>;<ComState>;<TimeEvent>;<TimeAck></p> <p>Example:</p> <p>300010011;300010011;exception message1;2;1;2015-07-15T06:08:22,329+2.00;2015-07-15T06:08:38,829+2.00</p>

Attention: Note upper-case/lower-case letters.

Examples for the symbol addresses of alarm variables (ID = **300010011**):

- ▶ **Exception.300010011.Alarm**
- ▶ **Exception.300010011.Value**
- ▶ **Exception.300010011.Message**
- ▶ **Exception.300010011.Info**

The names of the alarm variables can be issued freely.

ACTION WHEN ESTABLISHING A CONNECTION TO THE PLC

If the driver establishes a connection to the PLC, it attempts to determine the current status of the alarm events. To do this, it is not just alarm events whose Handshake variables have the value *1 (read requested)* that are evaluated, but also those the with value *2 (read required)*. The alarm variables can thus be set to the status before closing the connection again. This action is applicable for the start of zenon Runtime as well as for the restoring after a connection has been lost.

Alarm variables for which there are no entries in the event array are set to an initial value (**0** or empty string). As a result, pending alarms get the status of *cleared*.

Attention: For a correct evaluation, it is necessary that the *received* event and the *cleared* events of an alarm are always in the same event entry in the array.

8.5 RDA

PROBLEM

The RDA mechanism actually needs a coherent block from zenon in the controller, which can be addressed via an offset. RDA is therefore generally not possible for drivers with symbolic addressing.

RDA header must always always be configured with *Intel* byte order in the PLC.

SOLUTION

The **3S_V3** driver uses a property of the PLC handler that makes it possible to read the whole binary data block of variables with complex types (structures, arrays) as a whole, using the base name.

CODESYS CONFIGURATION

In order for the basis names for complex types of 3S_V3 drivers to be read correctly, the **Display group entries (Export data entries)** entry must be activated in the Codesys symbol configuration.

To do this, carry out the following steps in Codesys:

1. Open your project in the Codesys program.
2. In the **Global variables** node, select the **Workspace** entry
The **Options** dialog is opened.
3. Select the **Symbol configuration** category in the dialog.
4. Click on the **Configure symbol file...** button
The **Set object attributes** dialog is opened.
5. In this dialog, activate the **Display group entries** checkbox for the desired structure that is to be used for RDA.
6. Confirm the configuration by clicking on the **OK** button.



Information

Note the block size when configuring RDA. This should be less than 95000 bytes.

Background: With synchronous reading of the data, communication with the PLC must not be interrupted. This is because the RDA trigger cannot be reset in the event of an interruption. zenon checks the size of the RDA block when requesting data. If it exceeds the transferable size in one go, the variable is not requested.

RDA PROGRAMMING

The driver determines the base name by separating the part after the dot from the symbol name of the trigger variable. This part is used as the name. The trigger variable must be the first element of a structure in order for this to work. The rest of the structure can be created as desired. When reading the overall structure, only the binary data that has been read in when creating the RDA data according to the zenon documentation (archivserver.chm::/28257.htm) needs to correspond.

EXAMPLES

RDA-TYP 3:

```

TYPE RDA_DATA_3 : (* Structure for RDA type 3 payload *)
STRUCT
Value : DINT; (* value
TimeStamp : ARRAY[0..7] OF BYTE; (* Time stamp (year, month, day, hour, minute, second, 1/100th second,
reserve) *)
END_STRUCT
END_TYPE

TYPE RDA_3 : (* Structure for RDA type 3 *)
STRUCT
Trigger : DINT; (* trigger variable *)
Count : UDINT; (* Number of data sets *)
Cycle : UDINT; (* Cycle time in [ms] (only relevant for type 1 and 4 *)
RDA_Type : UDINT; (* RDA type, 1 - 4 *)
Oldest : UDINT; (* Index of the oldest value (placeholder for compatibility reasons, only relevant
for type 1) *)
Data : ARRAY[0..19] OF RDA_DATA_3; (* payload *)
END_STRUCT
END_TYPE

```

RDA-TYP 4:

```

TYPE RDA_4 : (* Structure for RDA type 4 *)
STRUCT
Trigger : DINT; (* trigger variable *)
Count : UDINT; (* Number of data sets *)
Cycle : UDINT; (* Cycle time in [ms] (only relevant for type 1 and 4 *)
RDA_Type : UDINT; (* RDA type, 1 - 4 *)
Oldest : UDINT; (* Index of the oldest value (placeholder for compatibility reasons, only relevant
for type 1) *)

```

```
TimeStamp : ARRAY[0..7] OF BYTE; (* Time stamp of the first value (year, month, day, hour, minute,
second, 1/100th second, reserve)) *)

Data : ARRAY[0..19] OF DINT; (* Payload *)

END_STRUCT

END_TYPE
```

Example

If, for example, an *RDA_4* type **RDA_Test** variable is created in the CoDeSys project, then a variable with the symbolic address *RDA_Test.Trigger* is created in zenon, and **HD active** and **Updated values** are set. If the variable changes value from 0 to 1, the *.Trigger* part of the symbol name *RDA_Test.Trigger* is cut off and **RDA_Test** is read in as a binary data block. The RDA processing is then carried out with this data block.

8.6 Real-time capability

The **3S_V3** driver is real-time capable.

In doing so, note:

- ▶ The protocol only provides the time stamp with precision to the second.
- ▶ The time of the SCADA system and PLC must be synchronized manually.

9 Driver command function

The zenon **Driver commands** function is to influence drivers using zenon.

You can do the following with a driver command:

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

Note: This chapter describes standard functions that are valid for most zenon drivers.

Not all functions described here are available for every driver. For example, a driver that does not, according to the data sheet, support a modem connection also does not have any modem functions.



Attention

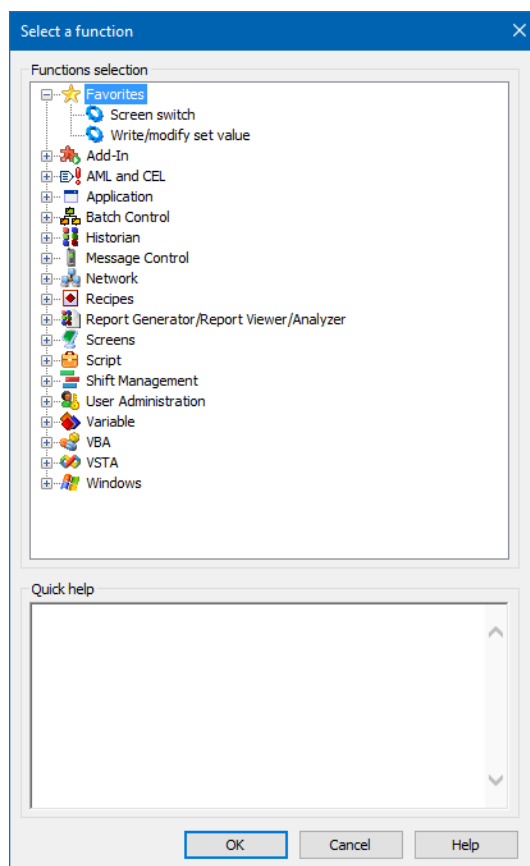
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

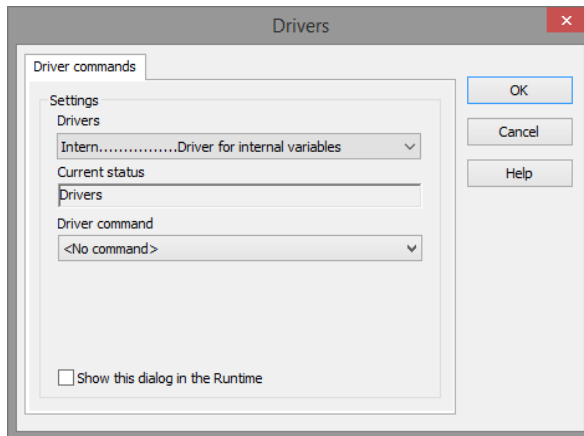
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



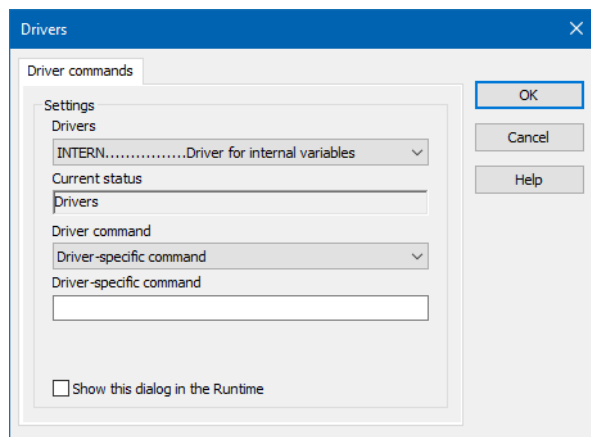
2. Navigate to the node **Variable**.
3. Select the **Driver commands** entry.

The dialog for configuration is opened



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

DRIVER COMMAND DIALOG



Option	Description
Driver	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
Current condition	Fixed entry that is set by the system. Has no function in the current version.
Driver command	Selection of the desired driver command from a drop-down list. For details on the configurable driver commands, see the available driver commands section.
Driver-specific command	Entry of a command specific to the selected driver.

Option	Description
	Note: Only available if, for the driver command option, the <i>driver-specific command</i> has been selected.
Show this dialog in the Runtime	<p>Configuration of whether the configuration can be changed in the Runtime:</p> <ul style="list-style-type: none"> ▶ <i>Active:</i> This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. ▶ <i>Inactive:</i> The Editor configuration is applied in the Runtime when executing the function. <p>Default: <i>inactive</i></p>

CLOSE DIALOG

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

AVAILABLE DRIVER COMMANDS

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

Driver command	Description
<No command>	<p>No command is sent.</p> <p>A command that already exists can thus be removed from a configured function.</p>
<i>Start driver (online mode)</i>	<p>Driver is reinitialized and started.</p> <p>Note: If the driver has already been started, it must be stopped. Only then can the driver be re-initialized and started.</p>
<i>Stop driver (offline mode)</i>	<p>Driver is stopped. No new data is accepted.</p> <p>Note: If the driver is in offline mode, all variables that were created for this driver receive the status <i>switched off</i> (OFF; Bit 20).</p>

Driver command	Description
<i>Driver in simulation mode</i>	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.
<i>Driver in hardware mode</i>	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.
<i>Driver-specific command</i>	Entry of a driver-specific command. Opens input field in order to enter a command.
<i>Activate driver write set value</i>	Write set value to a driver is possible.
<i>Deactivate driver write set value</i>	Write set value to a driver is prohibited.
<i>Establish connection with modem</i>	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
<i>Disconnect from modem</i>	Terminate connection (for modem drivers)
<i>Driver in counting simulation mode</i>	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.
<i>Driver in static simulation mode</i>	No communication to the controller is established. All values are initialized with 0.
<i>Driver in programmed simulation mode</i>	The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime.

DRIVER COMMAND FUNCTION IN THE NETWORK

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

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

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

10 Error analysis

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

10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under **Start/All programs/zenon/Tools 8.10 -> 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.

10.2 Error numbers

Note: The following description is only available in English.

Error text in the LOG	No.	Meaning
<i>RESULT_FAILED</i>	-1	Action erroneous.
<i>RESULT_OK</i>	0	Action successful
<i>RESULT_PLC_NOT_CONNECTED</i>	1	PLC not connected
<i>RESULT_PLC_LOGIN_FAILED</i>	2	Login to PLC has failed
<i>RESULT_PLC_NO_CYCLIC_LIST_DEFINED</i>	3	No cyclic list has been found
<i>RESULT_PLCHANDLER_INACTIVE</i>	4	PLCHandler instance was set inactive
<i>RESULT_LOADING_SYMBOLS_FAILED</i>	5	Loading of the symbols has failed
<i>RESULT_ITF_NOT_SUPPORTED</i>	6	The defined communication interface is not valid or not supported
<i>RESULT_COMM_FATAL</i>	7	Communication error occurred during action
<i>RESULT_NO_CONFIGURATION</i>	8	Wrong or erroneous configuration of the PLCHandler
<i>RESULT_INVALID_PARAMETER</i>	9	At least one parameter is invalid
<i>RESULT_ITF_FAILED</i>	10	Communication interface could not be initialized correctly (e. g. Gateway DLLs not available)
<i>RESULT_NOT_SUPPORTED</i>	11	Method not yet supported resp. implemented for this configuration (e. g. not supported for this interface)
<i>RESULT_EXCEPTION</i>	12	Handled exception in a low layer occurred

Error text in the LOG	No.	Meaning
		during action
<i>RESULT_TIMEOUT</i>	13	Timeout exceeded
<i>RESULT_STILL_CONNECTED</i>	14	PLC already connected (at a further ::Connect() call)
<i>RESULT_RECONNECTTHREAD_STILL_ACTIVE</i>	15	Reconnect Thread already active (started at a further ::Connect() call)
<i>RESULT_PLC_NOT_CONNECTED_SYMBOLS_LOADED</i>	16	No connection to the PLC, but symbols available offline
<i>RESULT_NO_UPDATE</i>	17	Asynchronous operation (e. g. cyclic read of variables) has not yet finished
<i>RESULT_OCX_CONVERSION_FAILED</i>	18	Error during conversion of values inside the PLCHandler's ActiveX interface occurred
<i>RESULT_TARGETID_MISMATCH</i>	19	PLC does not match to the passed target id etc.
<i>RESULT_NO_OBJECT</i>	20	No object found for the required action (e. g. tried to get an element beyond the end of the list)
<i>RESULT_COMPONENTS_NOT_LOADED</i>	21	PLCHandler instantiation has failed, because of missing components
<i>RESULT_BUSY</i>	22	Last action still in progress, cannot start the required one
<i>RESULT_DISABLED</i>	23	Feature is disabled by the configuration (e. g. Logging)
<i>RESULT_PLC_FAILED</i>	24	Communication to the PLC was successful, but the PLC has returned a bad result
<i>RESULT_INVALID_SYMBOL</i>	25	Specified symbol does not exist on PLC
<i>RESULT_BUFFER_TOO_SMALL</i>	26	User provided buffer is too small to store the required information
<i>RESULT_NO_PROJECT</i>	27	No project/application is loaded by the PLC, but the called method requires one to be executed
<i>RESULT_FILE_ERROR</i>	28	Operation failed because of an file access

Error text in the LOG	No.	Meaning
		error
<i>RESULT_RETAIN_MISMATCH</i>	29	Restored retain variables do not match to the project/application on the PLC
<i>RESULT_NO_ACCESS_RIGHTS</i>	30	Operation was denied by the PLC because of missing access rights
<i>RESULT_DUPLICATE_PLG_NAME</i>	31	Connect by PLC name failed, because there exist several PLCs with the specified name in the PLC network
<i>RESULT_SIZE_MISMATCH</i>	32	Service does not succeed because of a size mismatch



Information

To display these LOG messages, select the following configuration for the client configuration of the driver in the Diagnosis Viewer:

- ▶ Module **[Driver]**, Messagelevel **[Internal]**

10.3 Check list

Notes on establishing a connection and limiting errors:

CONNECTION BETWEEN ZENON AND AN ABB PLC VIA CODESYS GATEWAY

Note, when establishing a connection between zenon and an ABB PLC with the help of the CoDeSys gateway:

1. The specification of the L2 ABB protocol must be included in the registry.

This specification is normally entered during the installation of the ABB Automation Builders. If this installation cannot be carried out, the following entries can be made manually in the registry:

- ▶ **Key:** *[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\3S-Smart Software Solutions GmbH\Gateway Server\Drivers\ABB Tcp/Ip Level 2 AC]*
- ▶ **Path:** *GDrvABBTcpIpL2X.dll*
- ▶ **Address:** *192.168.0.10*
- ▶ **Motorola byteorder:** *Yes*

- ▶ **Port:** `dword:000004b0`
- 2. The DLL **GDrvABBTcpIpL2X.dll** must be in the same folder as **Gateway.exe**. The DLL must be obtained from the manufacturer ABB directly.

GENERAL TROUBLESHOOTING

- ▶ Is the PLC connected to the power supply?
- ▶ Analysis with the Diagnosis Viewers (on page 72):
-> Which messages are displayed?
- ▶ Are the participants available in the **TCP/IP** network?
- ▶ Can the PLC be reached via the *Ping* command?
Ping: **Open command line -> ping <IP address > (e.g.: ping 192.168.0.100) -> Press the Enter key.**
Do you receive an answer with a time or a timeout?
- ▶ Can the PLC be reached at the respective port via *TELNET*?
Telnet: **Command line: enter: telnet <IP address port number> (for example for Modbus: telnet 192.168.0.100 502) -> Press the Enter key.**
If the monitor display turns black, a connection could be established.
- ▶ Did you configure the Net address in the address properties of the variable correctly?
 - ▶ Does the addressing match with the configuration in the driver dialog?
 - ▶ Does the net address match the address of the target station?
- ▶ Did you use the right object type for the variable?
Example: Driver variables based on driver object type **Communication details** are purely statistics variables. They do not communicate with the PLC.
You can find detailed information on this in the Communication details (Driver variables) (on page 51) chapter.
- ▶ Does the offset addressing of the variable match the one in the PLC?

SOME VARIABLES REPORT INVALID.

- ▶ INVALID bits always refer to a net address.
- ▶ At least one variable of the net address is faulty.

VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY

Driver is not working. Check the:

- ▶ Installation of zenon

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

VARIABLES ARE DISPLAYED WITH A BLUE DOT

The communication in the network is faulty:

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

VALUES ARE DISPLAYED INCORRECTLY

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

DRIVER FAILS OCCASIONALLY

Analysis with the Diagnosis Viewer (on page 72):

-> Which messages are displayed?