COPADATA
do it your way

zenon driver manual

3S_Arti_NG

v.7.50
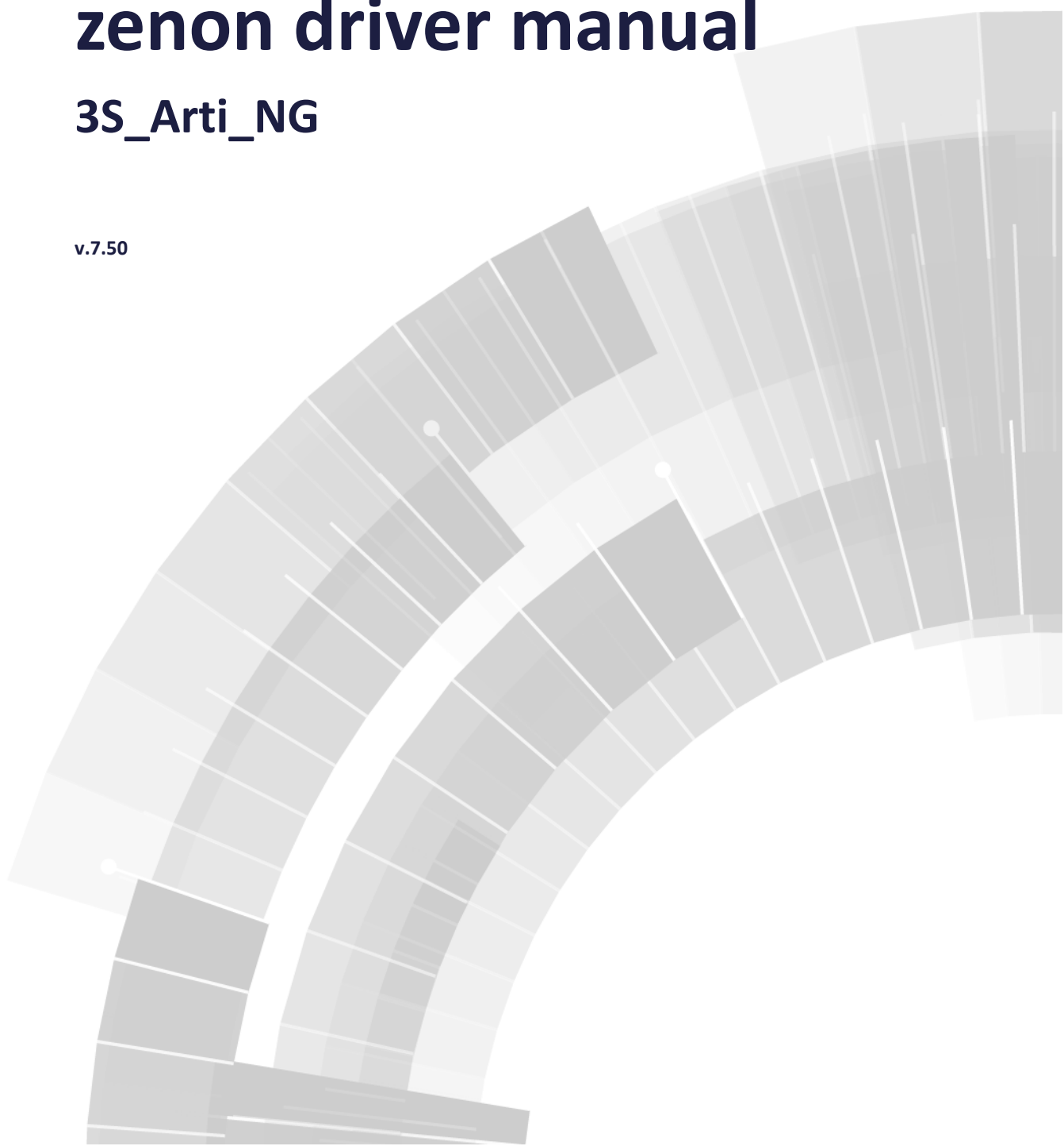
Contents

1. Welcome to COPA-DATA help

GENERAL HELP

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

PROJECT SUPPORT

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

LICENSES AND MODULES

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

2. 3S_Arti_NG

TESTED WITH THE FOLLOWING HARDWARE AND SOFTWARE

Firmware and software version

ELAU	MAX 4	
ELAU	EPAS-4	V 16
3S	CoDeSys	V 2.3

If both software tools (EPAS-4 and CoDeSys) are installed on a computer at the same time, there may be communication problems between zenon and each of the Soft PLCs.

TESTING ENVIRONMENT

- ▸ Communication with a CoDeSys SP RTE local
- ▸ Communication with a CoDeSys SP RTE via Ethernet
- ▸ Communication with an Elau MAX-4 SPS via Ethernet

SETTINGS IN CODESYS

The following settings must be made in order to enable connections between the control station and the CoDeSys Soft PLC:

Select the 3S CoDeSys SP RTE configuration in the target system settings.
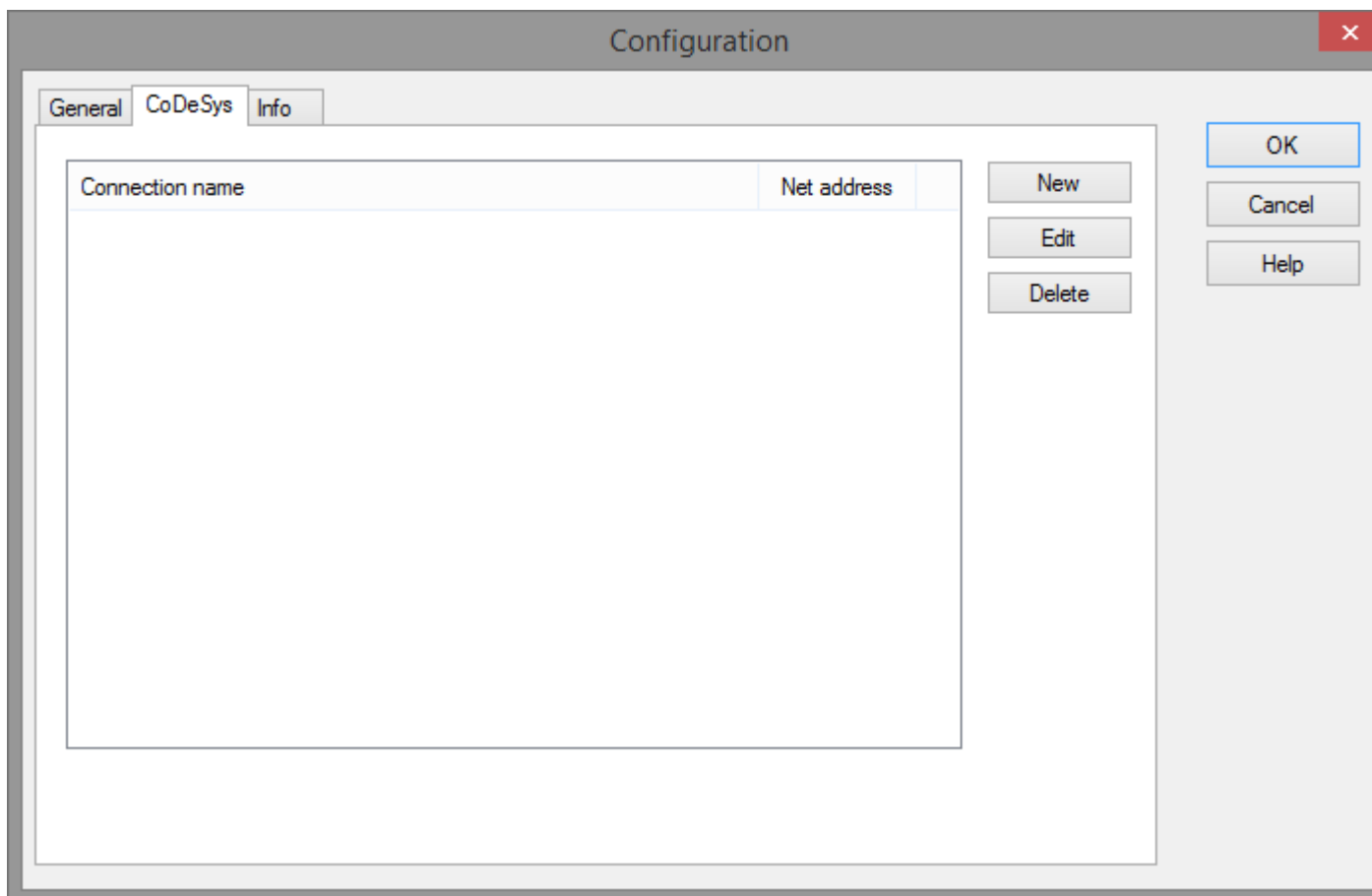
Activate the options "Send symbol file" and "Symbol configuration from INI file".
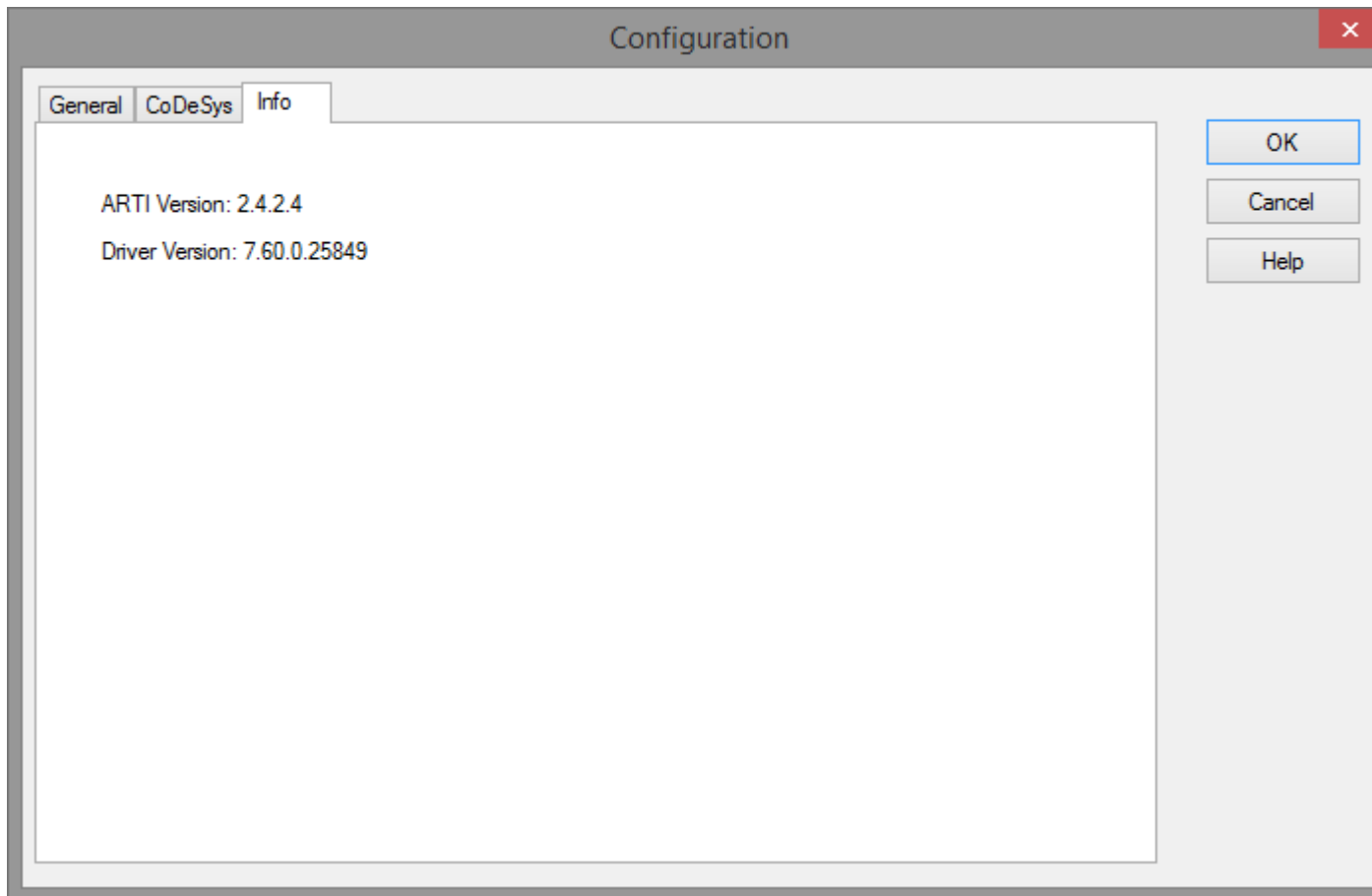
There must be a localhost with port 1200 in the communication parameters.
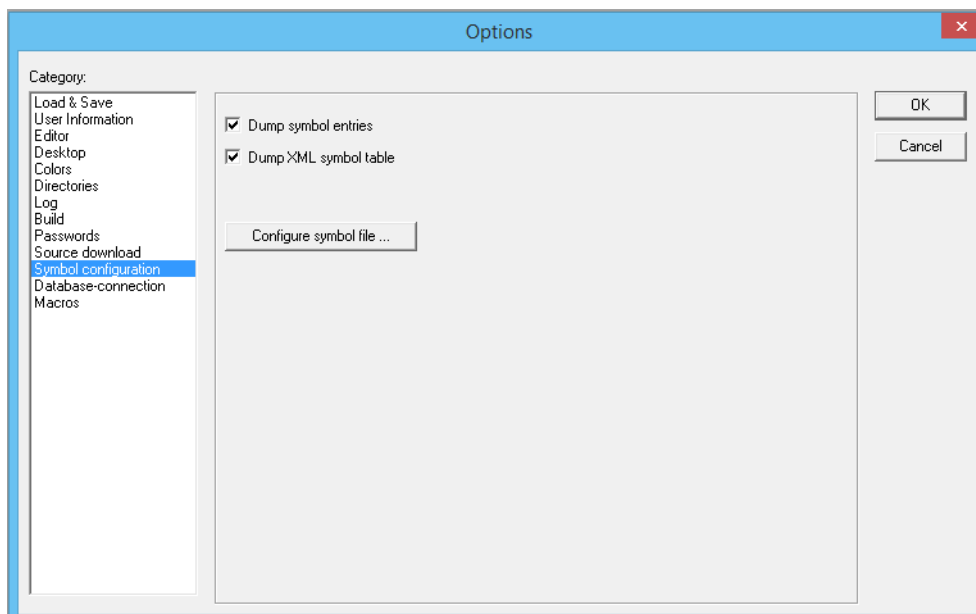
Select the parameters "Create symbol entries" and "Create XML symbol table" in the options under "Symbol configuration".

SETTINGS IN EPAS

The following settings must be made in order to enable connections between the control station and EPAS.
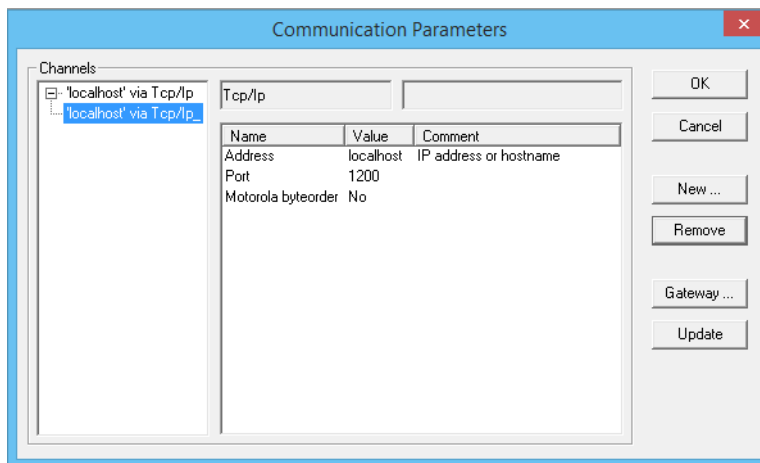
Create a new channel with IP address and port 5000 in the settings for the communication parameters.
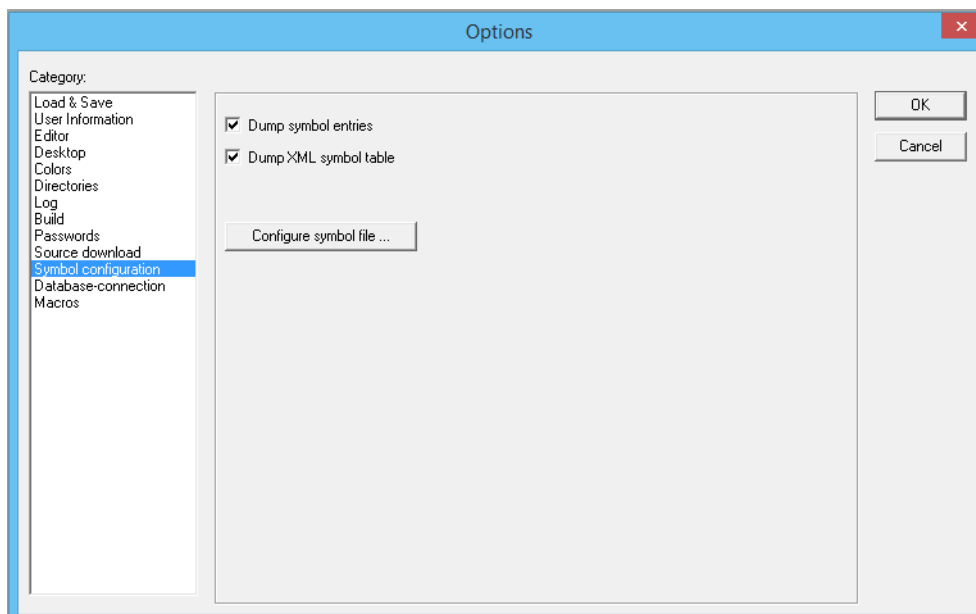
Activate the parameters "Create symbol entries" and "Create XML symbol table" in the options under "Symbol configuration".
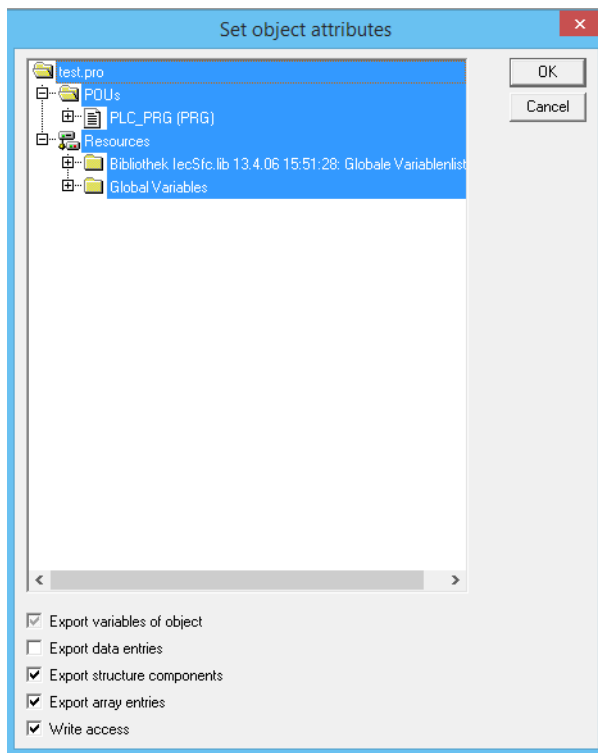
Deactivate the setting "Display group entries" via the button "Configure symbol file".

Activate the parameter "Update symbol file" in the options under "Login".

ERROR TIMEOUT

20 sec.

ACCESS METHODS

Polling

NUMBER OF PLCS

any

3. 3S_ARTI_NG - Data sheet

General:	
Driver file name	3S_ARTI_NG.exe
Driver name	Codesys Arti NG driver
PLC types	Codesys Soft PLCs, Moeller XControl PLCs XC200 and XC600, and Elau PacDrive Controller MAX 4, C200, C400, C600, P600.
PLC manufacturer	3S; Elau; Moeller;

Driver supports:	
Protocol	3S-Arti;
Addressing: Address-based	--
Addressing: Name-based	X
Spontaneous communication	--
Polling communication	X
Online browsing	X
Offline browsing	X
Real-time capable	--
Blockwrite	X
Modem capable	--
Serial logging	--
RDA numerical	--
RDA String	--

Requirements:	
Hardware PC	Standard network card
Software PC	Codesys software version 2.12 or higher incl. ARTI interface; Windows CE: SymArticlient.dll and Articlient.dll necessary, has to be ordered from 3S.
Hardware PLC	--
Software PLC	Codesys Software version 2.12 or higher incl. ARTI interface
Requires v-dll	X

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

4. Driver history

Date	Driver version	Comment
4/20/2006	100	Document created / BK
5/17/2006	300	Extension during the cause of the release /TS, MC
8/14/2008	1400	

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 form 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

- ▶ Installation of the current CoDeSys software (version 2.12 or higher)
- ▶ Install a Runtime system (with CoDeSys tool "InstallTarget")
- ▶ Set the communication parameters in CoDeSys

SUPPORTED CONNECTIONS.

At the moment, the connection types TCP/IP Level 2 and Level 4 Routing are supported (definition in the driver configuration).

SOFTWARE INSTALLATION - PC

Copy the driver file 3S_ARTI_NG.EXE to the current zenon directory (unless it is already there).

The driver needs the **ArtiClient.dll** und **SymArtiClient.dll** libraries. These dll files should be installed with the CoDeSys software. They are available on the COPA-DATA website in the customer area for download.

> 💡 **Information**
>
> *The* **ArtiClient.dll** *und* **SymArtiClient.dll** *must be version 2.4.2.3 or higher.*

SOFTWARE INSTALLATION - CE

Under CE, the driver 3S_ARTI_NG.DLL will automatically be copied to the CE device by the Editor via Remote Transport. The DLLs ArtiClient.dll and SymArtiClient.dll are also needed on the CE device. But there are own DLLs for Windows CE! At the moment, there are DLLs for X86 and StrongArm processors. Also these DLLs should be obtained from 3S. On the CE device these files have to be copied to the Runtime directory. (This is also where the files zenon6.ini and zenonrce.exe are stored).

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

5.2 Control

FOR WHICH PLCS

The driver 3S_ARTI_NG supports the connection to a 3S CoDeSys SoftPLC vie the ARTI interface (Asynchroneous RunTime Interface), which, as opposed to the normal 3S interface (3S_32), can also be used with Windows CE.

6. Configuration

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

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

6.1 Creating a driver

In order to create a new driver:

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

2. In the following dialog the control system offers a list of all available drivers.
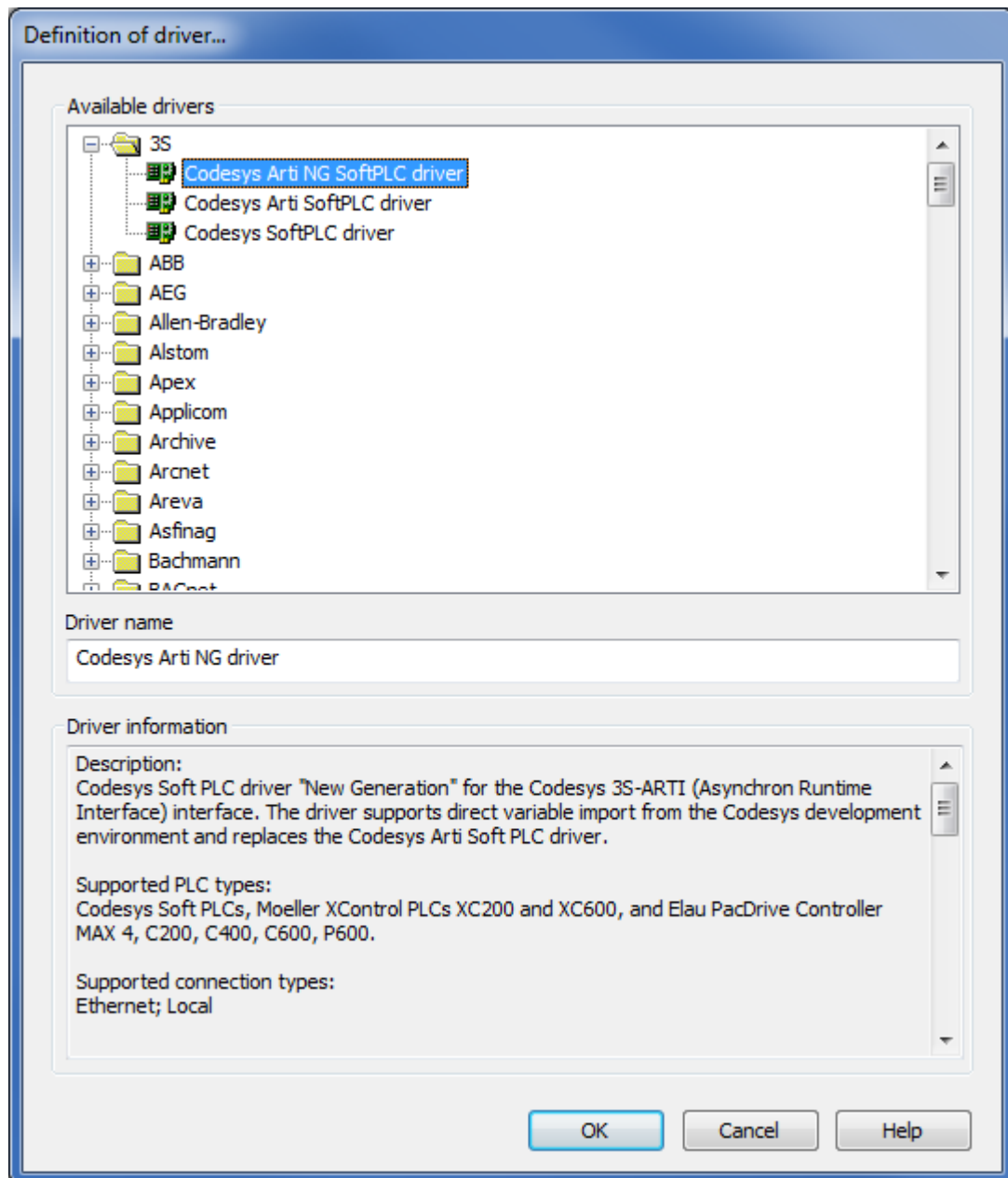
3. Select the desired driver and give it a name:

- The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.

- The driver name is part of the file name. Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (_).

- Attention: This name cannot be changed later on.

4. Confirm the dialog with **OK**. In the following dialog the single configurations of the drivers are defined.

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

> 💡 **Information**
>
> *For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:*
>
> ▸ Internal
>
> ▸ MathDr32
>
> ▸ SysDrv.
>
> ▸

6.2   Settings in the driver dialog

You can change the following settings of the driver:

> 💡 **Information**
>
> *The file <drivername>.csv contains all connection information. The connections can be defined and modified in the configuration dialog of the driver. See also the "driver specification".*
>
> *You can find it in the project directory in the folder \RT\FILES\zenon\custom\drivers*

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.

Parameters	Description
Mode	Allows to switch between hardware mode and simulation mode ▸ Hardware: A connection to the control is established. ▸ Simulation static No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▸ Simulation - counting No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically. ▸ Simulation - programmed N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the 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	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	`Active`: Outputs can be written. `Inactive`: Writing of outputs is prevented. Note: Not available for every driver.
Variable image remanent	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp.

	The variable image is saved in mode hardware if:
	▶ one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active
	The variable image is always saved if:
	▶ the variable is of the object type **Driver variable**
	▶ the driver runs in simulation mode. (not programmed simulation)
	The following states are not restored at the start of the Runtime:
	▶ SELECT(8)
	▶ WR-ACK(40)
	▶ WR-SUC(41)
	The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.
Stop on Standby Server	Setting for redundancy at drivers which allow only on communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade.
	Attention: If this option is active, the gapless archiving is no longer guaranteed.
	`Active`: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **switched off** (**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.
	Note: Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.
Global Update time	`Active`: The set **Global update time** in `ms` is used for all variables in the project. The priority set at the variables is not used. `Inactive`: The set priorities are used for the individual variables.
Priority	The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.
	The allocation to the variables takes place separately in the settings of the variable properties. The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities.

| | Thus the communication load is distributed better. |
| | Attention: Priority classes are not supported by each driver For example, drivers that communicate spontaneously do not support it. |

CLOSE DIALOG

Parameters	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 CYCLICAL DRIVERS

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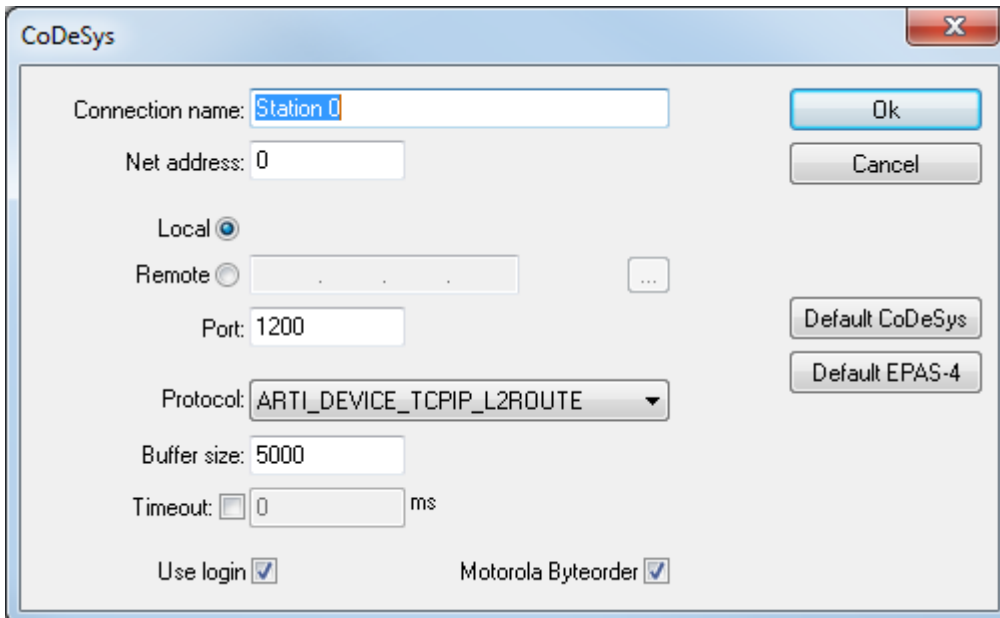
For **Set value**, **Advising** of variables and **Requests**, a read cycle is immediately triggered for all drivers - regardless of the set update time. This ensures that the value is immediately available for visualization after writing. Update times can therefore be shorter than pre-set for cyclical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

6.2.2   Driver dialog CoDeSys
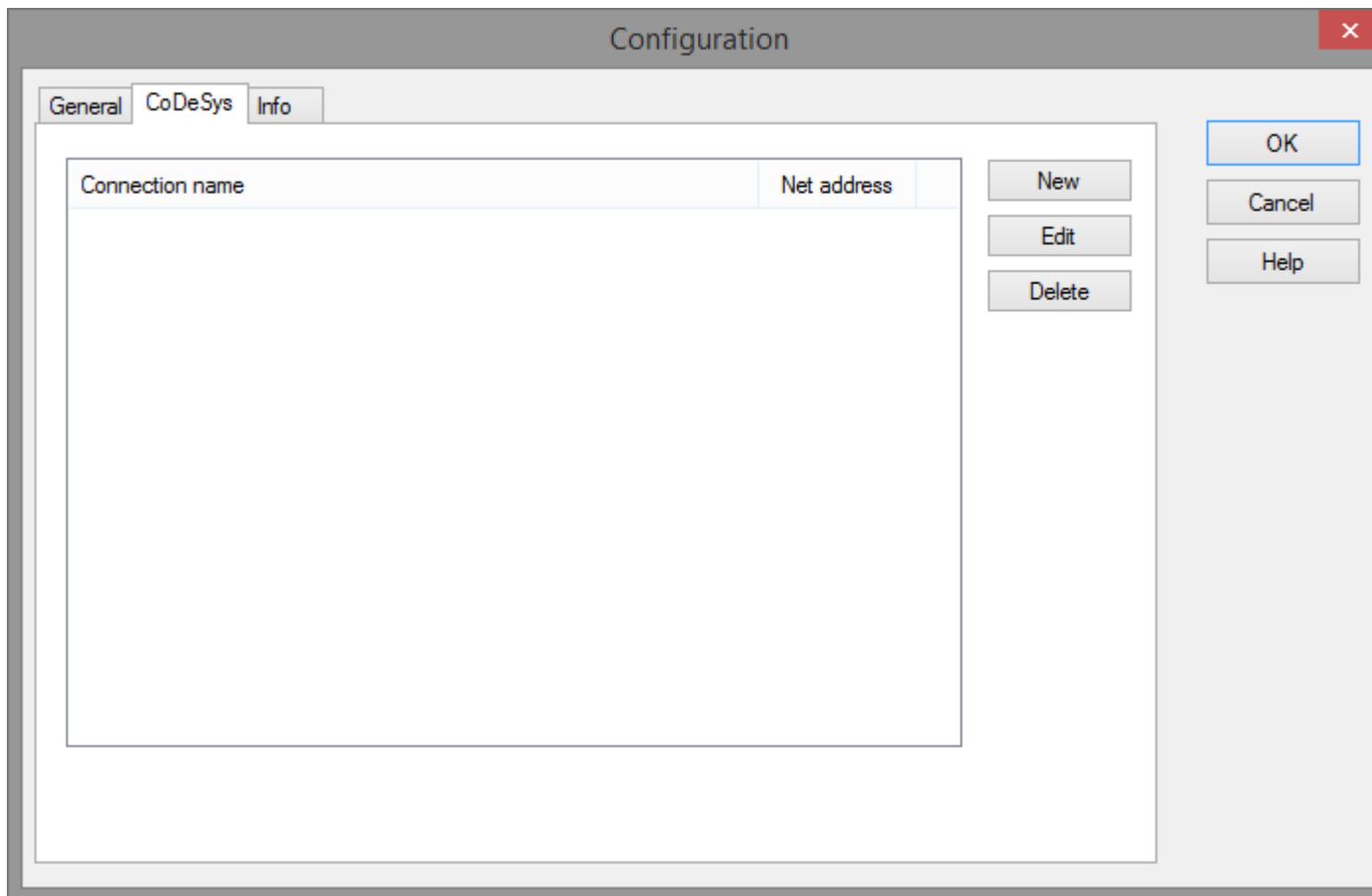
Setting	Description
New	This button creates a new connection and then opens the dialog with the connection settings.
Edit	Opens the dialog with the connections settings for the currently selected connection in the list.
Delete	Deletes the currently selected connection in the list.
Ok	Applies settings and closes the dialog.
Cancel	Closes the dialog without applying any settings.

DIALOG FOR CONNECTION SETTINGS

Setting	Description
Connection name	Freely definable name of the connection
Net address	Unique number of the connection. The **Net address** property in the properties of the variables must correspond.
Local	The PLC is running on the same computer.
Remote	The PLC is running on a remote computer in the network. Click on the **...** button to open the dialog for selecting a computer in the network.
Port	The port number of the PLC. ▸ Default CoDeSys: `1200` ▸ Default EPAS: `5000`
Protocol	Select a communication protocol. (TCP/IP Level 2 Route or Level 4)
Buffer size	Size of the communication buffer. ▸ Default CoDeSys: `5000` ▸ Default EPAS: `1500`
Timeout	`Active:` Enter a timeout for the **ARTIClient.dll** in milliseconds (ms). `Inactive:` The default timeout (ARTI_INFINITE ... infinite) is used.
Use login	`Active:` Opens a communication channel
Motorola byte folder	Enaples communication with Motorla CPU.
Default CoDeSys	Applies the default settings for CoDeSys PLCs. Attention: Existing settings will be overwritten!
Default EPAS	Applies the default settings for EPAS PLCs. Attention: Existing settings will be overwritten!
Ok	Applies settings and closes the dialog.
Cancel	Closes the dialog without applying any settings.

6.2.3   Driver dialog info
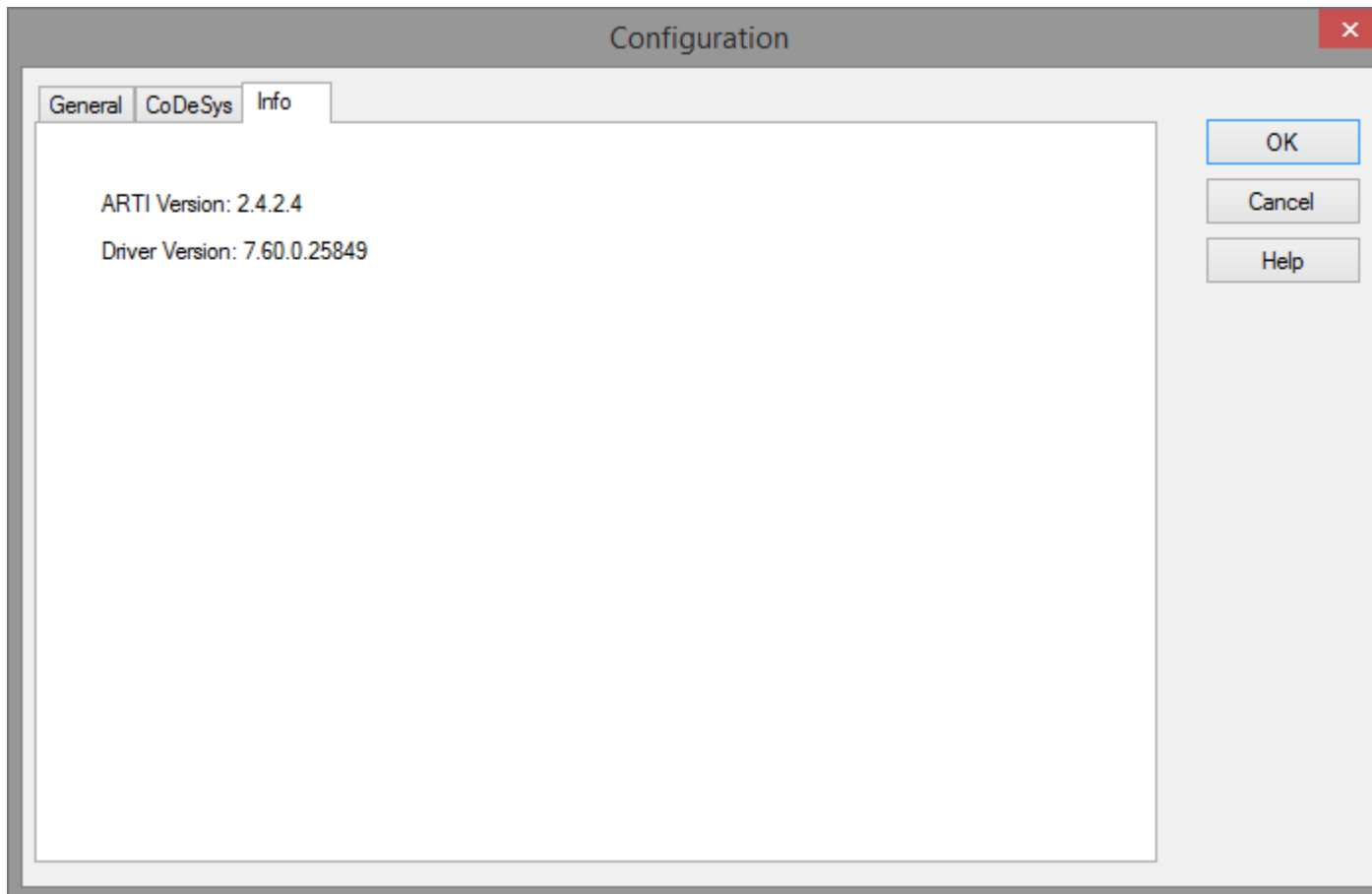
The version of the ARTILibrary and the driver version are displayed on the info page.

7. Creating variables

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

7.1   Creating variables in the Editor

Variables can be created:

▸   as simple variables

▸   in arrays (main.chm::/15262.htm)

▶ as structure variables (main.chm::/15278.htm)

VARIABLE DIALOG

To create a new variable, regardless of which type:

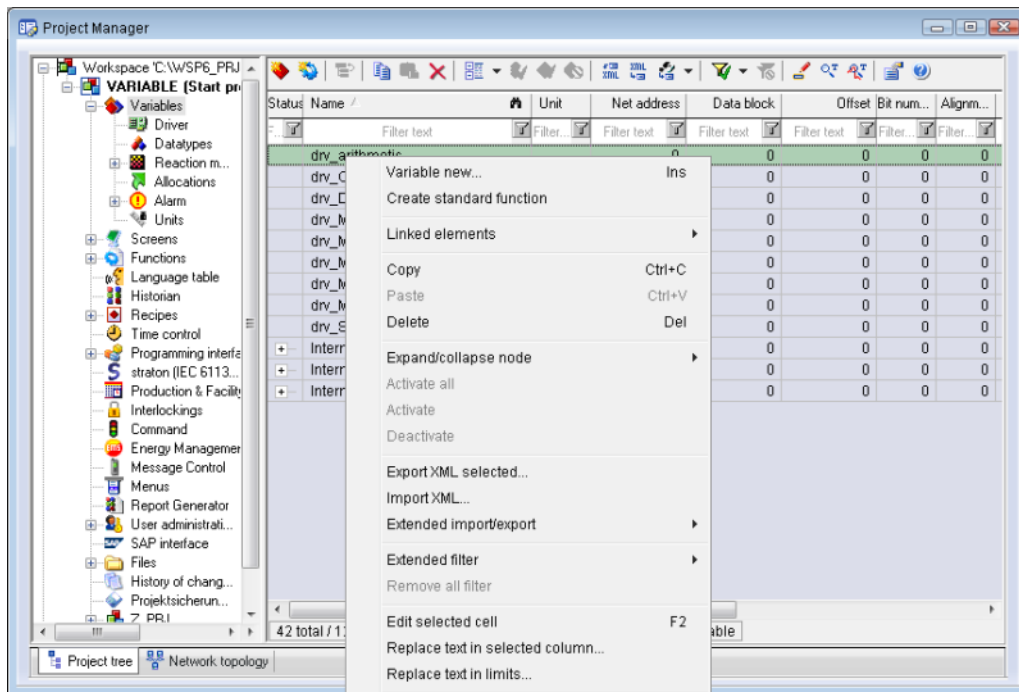1. Select the **New variable** command in the **Variables** node in the context menu

2. The dialog for configuring variables is opened
3. configure the variable

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

Property	Description
Name	Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name.
	Maximum length: 128 character
	Attention: The characters **#** and **@** are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive.
	Note: For some drivers, the addressing is possible over the property **Symbolic address**, as well.
Drivers	Select the desired driver from the drop-down list.
	Note: If no driver has been opened in the project, the driver for internal variables (**Intern.exe** (**Main.chm::/Intern.chm::/Intern.htm**)) is automatically loaded.
Driver object type (cti.chm::/28685.htm)	Select the appropriate driver object type from the drop-down list.
Data type	Select the desired data type. Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the Arrays chapter.
Addressing options	Expanded settings for arrays and structure variables. You can find details in the respective section.
Automatic element activation	Expanded settings for arrays and structure variables. You can find details in the respective section.

SYMBOLIC ADDRESS

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

Maximum length: `1024` characters.

INHERITANCE FROM DATA TYPE

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

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

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

7.2 Addressing

Group/Property	Description
General	
Name	Freely definable name.
	Attention: For every zenon project the name must be unambiguous.
Identification	Freely assignable identification, e.g. for resources label, comment ...
Addressing	
Net address	Bus address or net address of the variable.
	This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.
Data block	For variables of object type Extended data block, enter the datablock number here.
	Adjustable from 0 to 4294967295. You can take the exact maximum area for data blocks from the manual of the PLC.
Offset	Offset of the variable; the memory address of the variable in the PLC. Adjustable from 0 to 4294967295.
Alignment	not used for this driver
Bit number	Number of the bit within the configured offset.
	Possible entries: 0 ... 65535
String length	Only available for String variables: Maximum number of characters that the variable can take.
Driver connection/Data Type	Data type of the variable. Is selected during the creation of the variable; the type can be changed here.
	Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.
Driver connection/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/Priority	Assigns a variable a priority for the update time.
Symbol name	Is automatically set during online import.

VARIABLE MAPPING IN ZENON

The variable name is preceded with the prefix of **Net address** (station address) with a period [**.**] as a separator. The symbol name is the same name as the one used in the PLC program.

VARIABLE ADDRESSING VIA THE NAME

Variables can be addressed using a symbol name. Variables in zenon can thus be named regardless of their naming in the PLC. To create a symbol name for a variable:

1. Navigate to the **Addressing** node in the variable properties

2. Navigate to the **3S Arti specific** group

3. Enter a symbol name for the variable in the PLC in the input field of the **Symbol name** property

ADDRESSING OF BINARY VARIABLES

The 3S_Arti driver uses the following symbol names for communication with the PLC. To address individual bits in an INT:

1. An INT is imported from the PLC
   and then

2. 16 binary variables with the respective symbol names and a bit number between 0 and 15 are created manually

CHANGES TO THE PREFIX FOR A VARIABLE

The prefixes of the variables can also be changed later. Changes have different effects on the communication and import of variables.

COMMUNICATION

If the prefix is changed, communication no longer works. This is because the **Net address** of the variables and the symbol name is used for communication. The variable does not need a prefix for communication and does not need to correspond to the symbol name.

IMPORT

A change to the prefix influences online and offline import. A change has the following effects on the import:

▸ Searching for new variables no longer works.
All variables are marked as `new`, because there are no more variables with this **Net address** for import.

▸ Searching for deleted variables no longer works.
Variables that were deleted in the PLC program but are still present in zenon are not recognized. This is because there are no longer any variables for import with this **Net address** in zenon.

▸ Merging when changing data types no longer works.
When importing, the variable is not found in zenon.

7.3 Driver objects and datatypes

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

7.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
PLC marker	8	X	X	LREAL, WORD, DWORD, REAL, DATE_AND_TIME, DINT, UDINT, INT, UINT, TOD, STRING, BOOL, USINT, SINT	
Driver variable	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. Find out more in the chapter about the Driver variables (on page 42)

Driver object types	Channel type	Supported datatypes (DataType)	Read	Write	Comment
Marker	8	BOOL	X	X	
Marker	8	DATE_AND_TIME	X	X	
Marker	8	DINT	X	X	
Marker	8	INT	X	X	
Marker	8	LREAL	X	X	
Marker	8	REAL	X	X	
Marker	8	SINT	X	X	
Marker	8	STRING	X	X	
Marker	8	TOD	X	X	
Marker	8	UDINT	X	X	
Marker	8	UINT	X	X	
Marker	8	USINT	X	X	

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.

EXAMPLES FOR ALL POSSIBLE IEC DATA TYPES

PLC	zenon
BOOL	BOOL
DT or DATE_AND_TIME	DATE_AND_TIME
DINT	DINT
INT	INT
LREAL	LREAL
REAL	REAL
SINT	SINT
STRING	STRING
TOD or TIME_OF_DAY	TOD
UDINT	UDINT
UINT	UINT
USINT	USINT
WORD	WORD
DWORD	DWORD

Data type: The property **Data type** is the internal numerical name of the data type. It is also used for the extended DBF import/export of the variables.

7.4   Creating variables by importing

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

> 💡 **Information**
>
> *You can find details on the import and export of variables in the Import-Export (main.chm::/13028.htm) manual in the Variables (main.chm::/13045.htm) section.*

7.4.1   XML import

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

▶   The import/export must not be started from the global project.

▶ The start takes place via:

- Context menu of variables or data typ in the project tree
- or context menu of a variable or a data type
- or symbol in the symbol bar variables

⚠ **Attention**

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

Example:

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

7.4.2 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	Char	128	Variable name.
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (field/column must be entered manually).
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
KANAL_D	Log	1	The variable is deleted with the `1` entry (field/column has to be created by hand).
TAGNR	C	128	Identification.
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
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	Bus 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	L	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTTYP and DATENTYP
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	L	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	L	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in **project.ini**.
ADJWVBA	L	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	L	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	L	1	Set blink attribute
BTB1	L	1	Logging in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Printer output (for CEL or Alarm)
QUITTIER1	L	1	Must be acknowledged
LOESCHE1	L	1	Must be deleted
VARIABLE1	L	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	L	1	Functions linking
ASK_FUNC1	L	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	L	1	Invisible

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

7.4.3   Online import

Via the online import, you can read variables from a PLC or import them from an *.SYM_XML file. If a large number of variables is imported, the configured buffer size may be too small. In that case, you must increase the buffer size in the dialog window **CoDeSys** (on page 20).

To import variables online:

1. select the driver

2. click on **Import variables from driver** in the context menu

3. select either:

   - Symbol file (on page 39) and

   - browse using ARTI (on page 39)

4. follow the import assistant
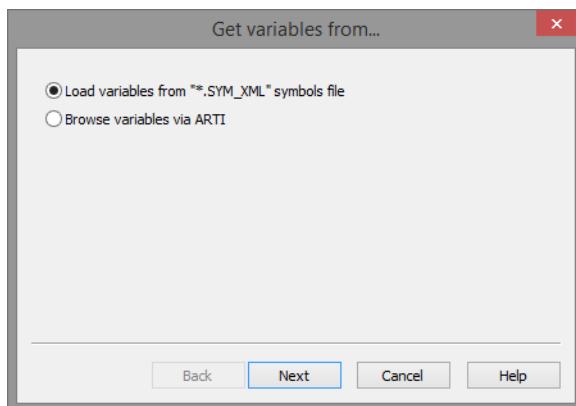
Parameters	Description
Load variables from "*.SYM_XML" symbols file	Read variables from an XML symbol file.
Browse variables via ARTI	Read variables from a PLC.

In the next stage, the files are imported from a symbol file (on page 39) or directly from ARTI (on page 39). After that, variables are pre-selected and assigned (on page 40).

Import from a symbol file

For XML importing, select a station (Net address) and the according symbol file. The stations that can be selected were created in the Driver configuration (on page 20) before.
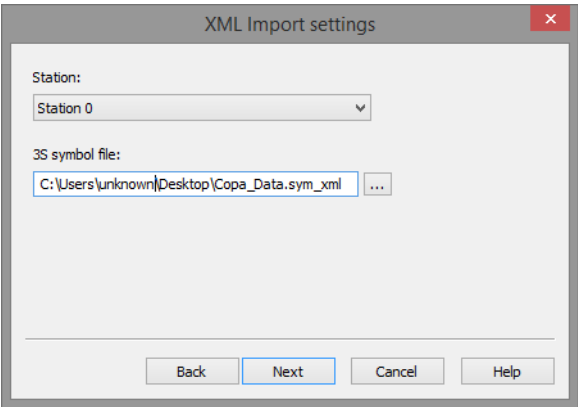
Parameters	Description
Station	The station that you want to create a variable for.
3S symbol file	Path of the XML symbol file

In the next stage, variables are pre-selected and assigned (on page 40).

Import via ARTI

When importing via the ARTI interface, select the station that you want to connect to. You must have configured the station in the Driver configuration (on page 20) before that.
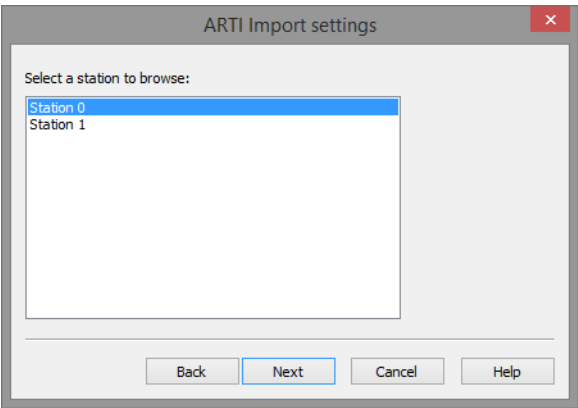
Parameters	Description
Select a station to browse	Select the station that you want to read a variable from.

In the next stage, variables are pre-selected and assigned (on page 40).

Pre-selection and assignment of the variables

It is possible to pre-filter the variables to be imported in order to limit the selection list:

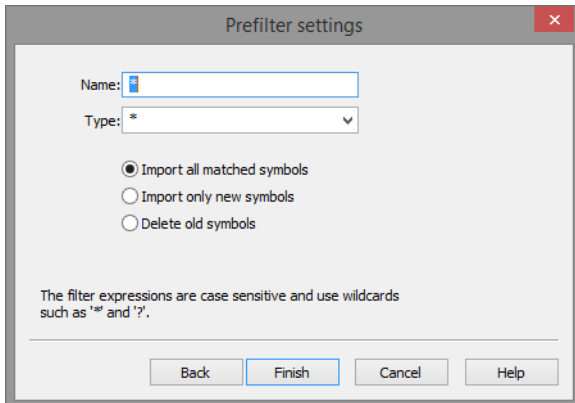| Parameters	Description
Name	Filter expression for the symbol name. You can use wild cards here. (Wildcards are only allowed as prefix or suffix; e.g. *xxx or xxx*.)
Type	Filter expression for the symbol type. You can use wild cards here. (Wildcards are only allowed as prefix or suffix; e.g. *xxx or xxx*.)
Import all matched symbols	All symbols from the import source that match the filter **Name** or **Type** will be offered.
Import only new symbols	All symbols from the import source that match the filter **Name** or **Type** and that do not exist in zenon will be offered. The symbol name is compared with the variable name created in zenon. Attention: This feature works only if the zenon variable name was not changed after importing!
Delete old symbols	All zenon that do not exist in the import source will be offered for deletion. For this, the symbol names will be compared with the address settings of the zenon. In this selection process, you cannot filter by **Name** or **Type**. Attention: After selecting and confirming the dialog, the variables will be deleted in zenon!

Take care of upper and lower cases when filtering by names and types.

SELECTION DIALOG

After reading the symbols, you can import a selection as zenon variables.
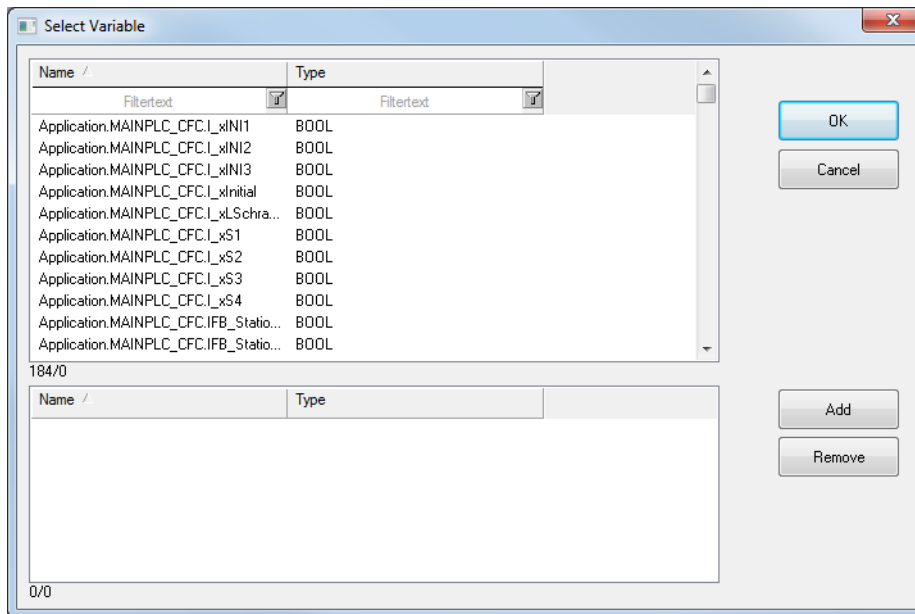
Parameters	Description
Ok	Create the selected symbols as variables in zenon.
Cancel	Cancel the import.
Add	Add the selected symbols to the selection list.
Remove	Remove the selected symbols from the selection list.

VARIABLE MAPPING IN ZENON

When importing to zenon, the net address (station address) and a separator (dot) will be put in front of the variable name. The symbol name is the same name as the one used in the PLC program. For example:

Variable name: `0.PLC_PRG.Timer1.IN`

Symbol name: `PLC_PRG.Timer1.IN`
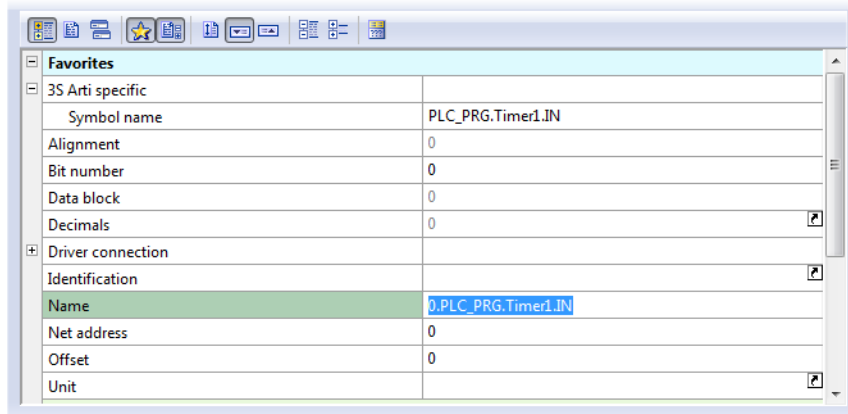
7.5   Driver variables

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

- ▶   Information
- ▶   Configuration
- ▶   Statistics and
- ▶   Error message

The definitions of the variables implemented in the driver kit are available in the import file **drvvar.dbf** (on the installation medium in the `\Predefined\Variables` folder) and can be imported from there.

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

> 💡 **Information**
>
> *Not every driver supports all driver variants.*
>
> *For example:*
>
> ▸ Variables for modem information are only supported by modem-compatible drivers
>
> ▸ Driver variables for the polling cycle only for pure polling drivers
>
> ▸ Connection-related information such as ErrorMSG only for drivers that only edit one
>   connection at a a time

INFORMATION

Name from import	Type	Offset	Description
MainVersion	UINT	0	Main version number of the driver.
SubVersion	UINT	1	Sub version number of the driver.
BuildVersion	UINT	29	Build version number of the driver.
RTMajor	UINT	49	zenon main version number
RTMinor	UINT	50	zenon sub version number
RTSp	UINT	51	zenon Service Pack number
RTBuild	UINT	52	zenon build number
LineStateIdle	BOOL	24.0	TRUE, if the modem connection is idle
LineStateOffering	BOOL	24.1	TRUE, if a call is received
LineStateAccepted	BOOL	24.2	The call is accepted
LineStateDialtone	BOOL	24.3	Dialtone recognized
LineStateDialing	BOOL	24.4	Dialing active
LineStateRingBack	BOOL	24.5	While establishing the connection
LineStateBusy	BOOL	24.6	Target station is busy

LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
LineStateProceeding	BOOL	24.9	Dialing completed
LineStateOnHold	BOOL	24.10	Connection in hold
LineStateConferenced	BOOL	24.11	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	BOOL	24.13	Connection in hold for transfer
LineStateDisconnected	BOOL	24.14	Connection terminated.
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped For `driver stop`, the variable has the value `TRUE` and an **OFF** bit. After the driver has started, the variable has the value `FALSE` and no **OFF** bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	BOOL	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	BOOL	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings **PhoneNumberSet** and **ModemHwAdrSet**.

PhoneNumberSet	STRING	38	Telephone number, that should be used
ModemHwAdrSet	DINT	39	Hardware address for the telephone number
GlobalUpdate	UDINT	3	Update time in milliseconds (ms).
BGlobalUpdaten	BOOL	4	TRUE, if update time is global
TreiberSimul	BOOL	5	TRUE, if driver in sin simulation mode
TreiberProzab	BOOL	6	TRUE, if the variables update list should be kept in the memory
ModemActive	BOOL	7	TRUE, if the modem is active for the driver
Device	STRING	8	Name of the serial interface or name of the modem
ComPort	UINT	9	Number of the serial interface.
Baudrate	UDINT	10	Baud rate of the serial interface.
Parity	SINT	11	Parity of the serial interface
ByteSize	USINT	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.
StopBit	USINT	13	Number of stop bits of the serial interface.
Autoconnect	BOOL	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	STRING	17	Current telephone number
ModemHwAdr	DINT	21	Hardware address of current telephone number
RxIdleTime	UINT	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)

WriteTimeout	UDINT	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	UDINT	20	Number of ringing tones before a call is accepted
ReCallIdleTime	UINT	53	Waiting time between calls in seconds (s).
ConnectTimeout	UINT	54	Time in seconds (s) to establish a connection.

STATISTICS

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

MaxCycleTime	UDINT	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	UDINT	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	UDINT	26	Number of writing attempts
ReadErrorCount	UDINT	34	Number of reading errors
MaxUpdateTimeNormal	UDINT	56	Time since the last update of the priority group **Normal** in milliseconds (ms).
MaxUpdateTimeHigher	UDINT	57	Time since the last update of the priority group **Higher** in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group **High** in milliseconds (ms).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group **Highest** in milliseconds (ms).
PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

ERROR MESSAGE

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

RdErrHwAdresse	DINT	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	UDINT	46	Block number when the last reading error occurred.
RdErrMarkerNo	UDINT	47	Marker number when the last reading error occurred.
RdErrSize	UDINT	48	Block size when the last reading error occurred.
DrvError	USINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

8. Driver-specific functions

The driver supports the following functions:

EXTENDED ERROR FILE

No

SERIAL LOGGING

No

BLOCKWRITE

Yes

REDUNDANCY

Yes

RDA

No

REAL TIME STAMPING

No

BROWSING

Yes

INI ENTRIES

ZENON6.INI

None

PROJECT.INI

None

WINCE

Yes

9. Driver commands

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

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

> ▸   create a new function

> ▸   select Variables -> Driver commands

▶   The dialog for configuration is opened
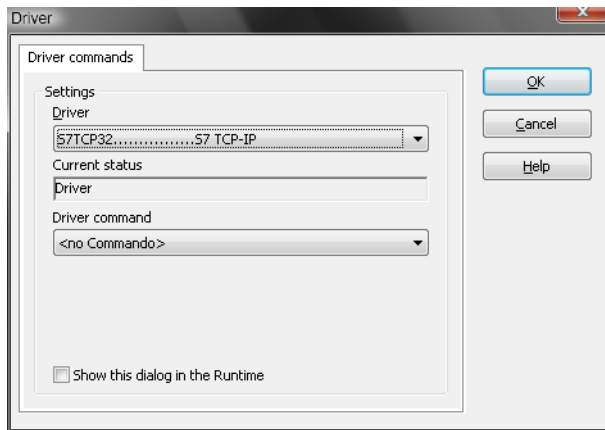
Parameter	Description
Drivers	Drop-down list with all drivers which are loaded in the project.
Current status	Fixed entry which has no function in the current version.
Driver command	Drop-down list for the selection of the command.
▶ `Start driver (online mode)`	Driver is reinitialized and started.
▶ `Stop driver (offline mode)`	Driver is stopped. No new data is accepted. Note: If the driver is in offline mode, all variables that were created for this driver receive the status `switched off` (OFF; Bit 20).
▶ `Driver in simulation mode`	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▶ `Driver in hardware mode`	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▶ `Driver-specific command`	Enter driver-specific commands. Opens input field in order to enter a command.
▶ `Driver - activate set setpoint value`	Write set value to a driver is allowed.
▶ `Driver - deactivate set setpoint value`	Write set value to a driver is prohibited.
▶ `Establish connecton with modem`	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
▶ `Disconnect from modem`	Terminate connection (for modem drivers)
Show this dialog in the Runtime	The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

If the computer, on which the **driver command** function is executed, is part of the zenon network, additional actions are carried out. A special network command is sent from the computer to the project server, which then executes the desired action on its driver. In addition, the Server sends the same driver command to the project standby. The standby also carries out the action on its driver.

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

10. Error analysis

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

10.1   Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under Start/All programs/zenon/Tools 7.50 -> Diagviewer.

zenon driver log all errors in the LOG files. The default folder for the LOG files is subfolder **LOG** in directory `ProgramData`, example:

`%ProgramData%\COPA-DATA\LOG`. LOG files are text files with a special structure.

Attention: With the default settings, a driver only logs error information. With the **Diagnosis Viewer** you can enhance the diagnosis level for most of the drivers to "Debug" and "Deep Debug". With this the driver also logs all other important tasks and events.

In the Diagnosis Viewer you can also:

- ▸ Follow 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

Error code	Description
-1	Common ARTI error.
-101	The type of protocol is not supported.
-102	The communication channel is not valid or open.
-103	A channel with the specified parameters is already open.
-104	The wrong type of message was received.
-105	There was not any message received.
-106	There was not enough data for this message type received.
-201	There is too much data in the send queue.
-202	There is too much data in the receive queue.
-203	Only in the synchronous mode if last service has not finished yet.
-301	Common communication error in the system –dependent level.
-401	There wasn't such a file on the target or it could not be opened.
-402	The file data doesn't fit completely in the provided buffer.
-501	A strictly necessary function parameter is NULL or invalid.
-502	The maximum number of open channels is exceeded.
-503	There is no SDD assigned to the channel.
-504	There is no type table assigned to the SDD.
-505	The end of symbol table is reached.
-506	There is no symbol with that name found in the SDD.
-507	The data stream for reading the variables is bigger than the target's buffer.
-508	The data stream for writing the variables in the VarList is bigger than the target's buffer.
-509	Another VarList is still active with reading / writing its values.
-510	Writing various VarList blocks is not supported.
-511	The variable's swap size doesn't fit with the number of bytes to be written
-512	Error in parsing the symbol file.
-513	There is a new project on the runtime system so symbols data have changed.

| -514 | Error in sorting the symbol file. |
| -515 | There is no project downloaded to the runtime system. |

10.3   Check list

▶ Have you analyzed the error text file (which errors did occur)?

▶ Is the option "Send symbol file" activated? See the general target system settings for that.

▶ Can you reach the remote station in the network (ping)?

▶ Send the zenon project to support@copadata.com (mailto:support@copadata.com)