COPA-DATA
do it your way

zenon driver manual

AllenDF1

v.7.60

©2017 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. AllenDF1 5

3. ALLENDF1 - Data sheet 6

4. Driver history 7

5. Requirements 8

5.1 PC 8

5.2 Control 8

6. Configuration 9

6.1 Creating a driver 9

6.2 Settings in the driver dialog 12

6.2.1 General 13

6.2.2 Com 16

6.2.3 Settings 20

6.2.4 Network stations 22

7. Creating variables 24

7.1 Creating variables in the Editor 24

7.2 Addressing 28

7.3 Driver objects and datatypes 29

7.3.1 Driver objects 30

7.3.2 Mapping of the data types 31

7.4 Creating variables by importing 32

7.4.1 XML import 32

7.4.2 DBF Import/Export 33

7.5 Communication details (Driver variables) 39

8. Driver-specific functions 44

9. Driver commands 45

10. Error analysis 47

10.1 Analysis tool 47

10.2 Error numbers 48

10.3 Check list 48

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 (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. AllenDF1

The general driver configuration can be found in the separate chapter "General driver documentation"

3. ALLENDF1 - Data sheet

General:	
Driver file name	ALLENDF1.exe
Driver name	Allen Bradley DF1-driver
PLC types	All Allen Bradley PLCs with DF1 interface; Micro Logix 1400 PLCs with Ethernet port.
PLC manufacturer	Allen-Bradley; Rockwell;

Driver supports:	
Protocol	DF1;
Addressing: Address-based	X
Addressing: Name-based	--
Spontaneous communication	--
Polling communication	X
Online browsing	--
Offline browsing	--
Real-time capable	--
Blockwrite	--
Modem capable	--
Serial logging	X
RDA numerical	--
RDA String	--
Hysteresis	X
extended API	--
Supports status bit **WR-SUC**	--
alternative IP address	--

Requirements:	
Hardware PC	Serial interface RS232; standard networkcard
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	--

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

4. Driver history

Date	Driver version	Comment
9.8.2005	11	The driver now supports the modem connection implemented in the driver kit. The serial interface can be reconfigured during Runtime. This can be done with the according variables of the driver object type **Communication details**.
14.08.2008	1300	
1.01.2016	7.50	The driver supports now the object types `timer` and `counter`.

DRIVER VERSIONING

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

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

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

5. Requirements

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

5.1 PC

HARDWARE

Serial interface RS232.

The PLC is connected to the serial port RS232 (COM) of the PC with a serial cable.

SOFTWARE

Copy the driver file AllenDF1.EXE into the current program directory (unless it is already there) and enter it into the TREIBER_EN.XML file with the tool DriverInfo.exe.

5.2 Control

HARDWARE

SLC 500 PLC with SLC 5/03 or SLC 5/04 processors.

Used option: "CRC" error check

The PLC is connected to the serial port RS232 (COM) of the PC with a serial cable.

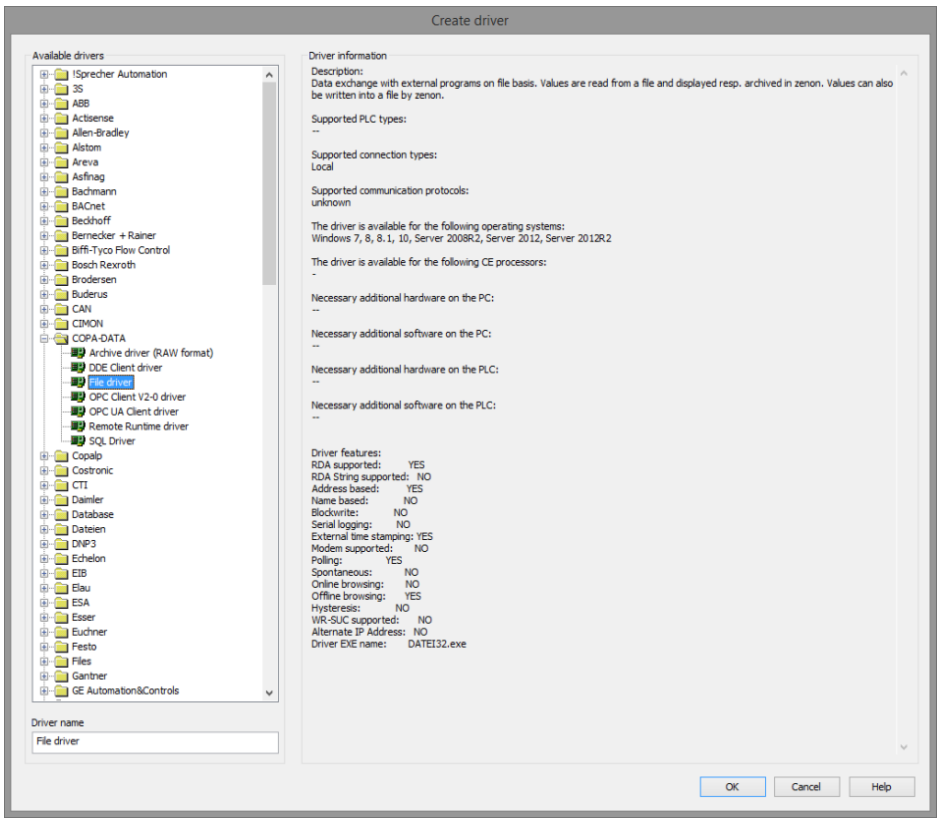
6. Configuration

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

CLOSE DIALOG

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

Information

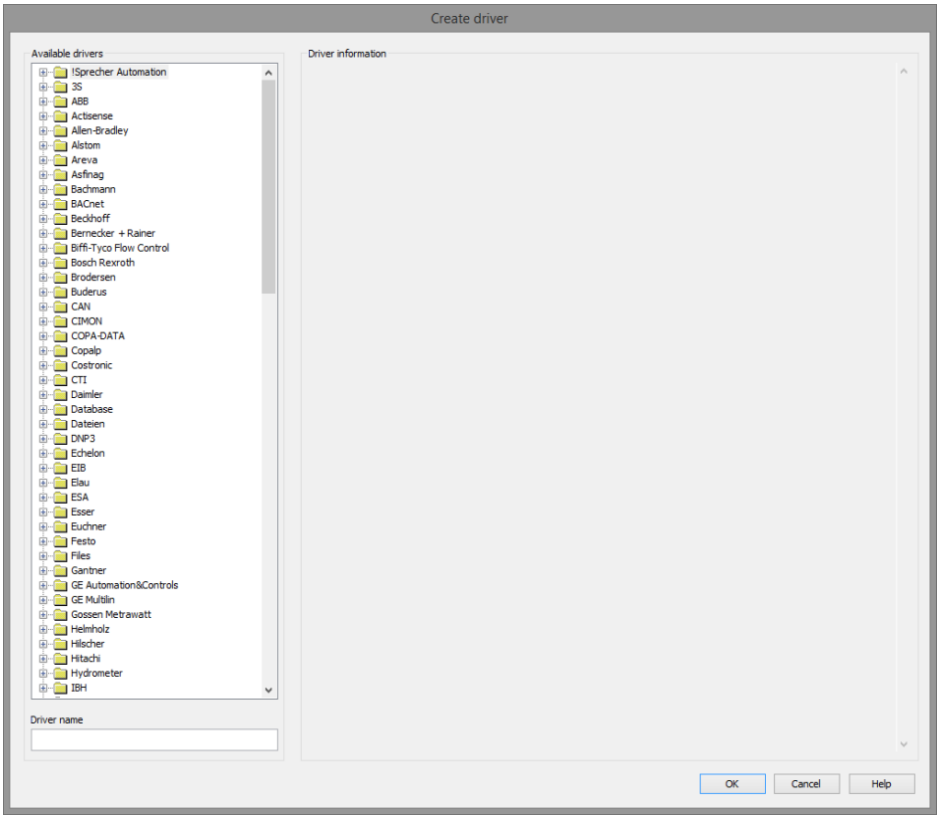
The content of this dialog is saved in the file called Treiber_[Language].xml. You can find this file in the following folder: `C:\ProgramData\COPA-DATA\zenon[version number]`.

CREATE NEW DRIVER

In order to create a new driver:

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

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



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

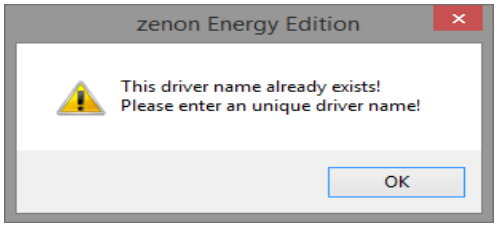
   - The **Driver name** must be unique.
     If a driver is used more than once in a project, a new name has to be given each time.
     This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
   - The **Driver name** is part of the file name.
     Therefore it may only contain characters which are supported by the operating system.
     Invalid characters are replaced by an underscore (_).
   - Attention: This name cannot be changed later on.

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

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

DRIVER NAME DIALOG ALREADY EXISTS

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



<CD_PRODUCNTAME> PROJECT

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

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

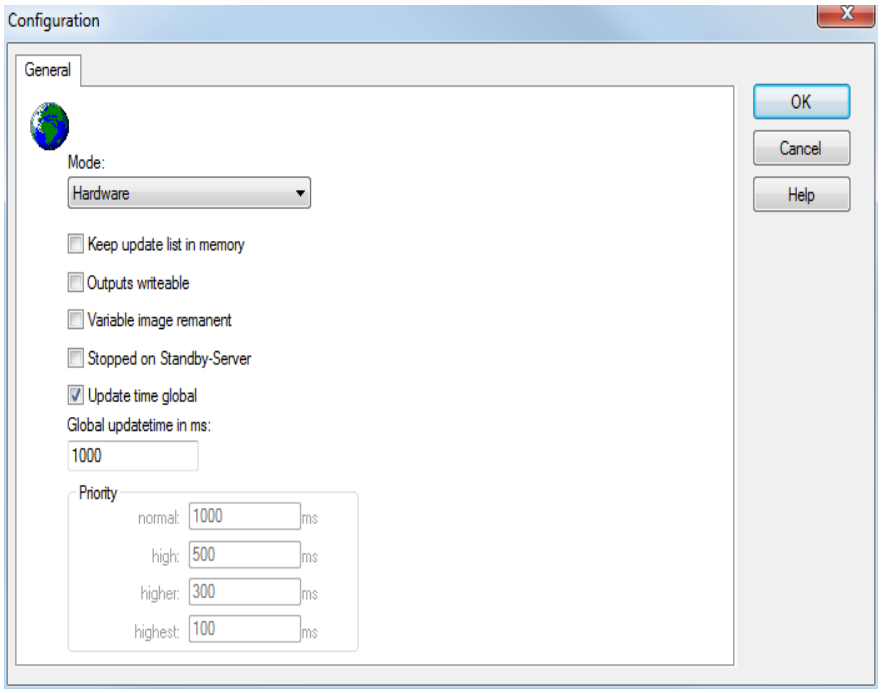
> **Information**
>
> Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

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	Allows to switch between hardware mode and simulation mode ▸ `Hardware`: A connection to the control is established. ▸ Simulation - static: No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▸ `Simulation - counting`: No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically. ▸ `Simulation - programmed`: No communication is established to the PLC. The 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 driver object type **Communication details** ▸ 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 one communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. **Attention:** If this option is active, the gapless archiving is no longer guaranteed. Active: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **switched off** (**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 variables are allocated separately in the settings of the variable properties. The communication of the individual variables can be graded according to importance or required topicality using the priority classes. Thus the communication load is distributed better. **Attention:** Priority classes are not supported by each driver For example, drivers that communicate spontaneously do not support it.

CLOSE DIALOG

Options	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.

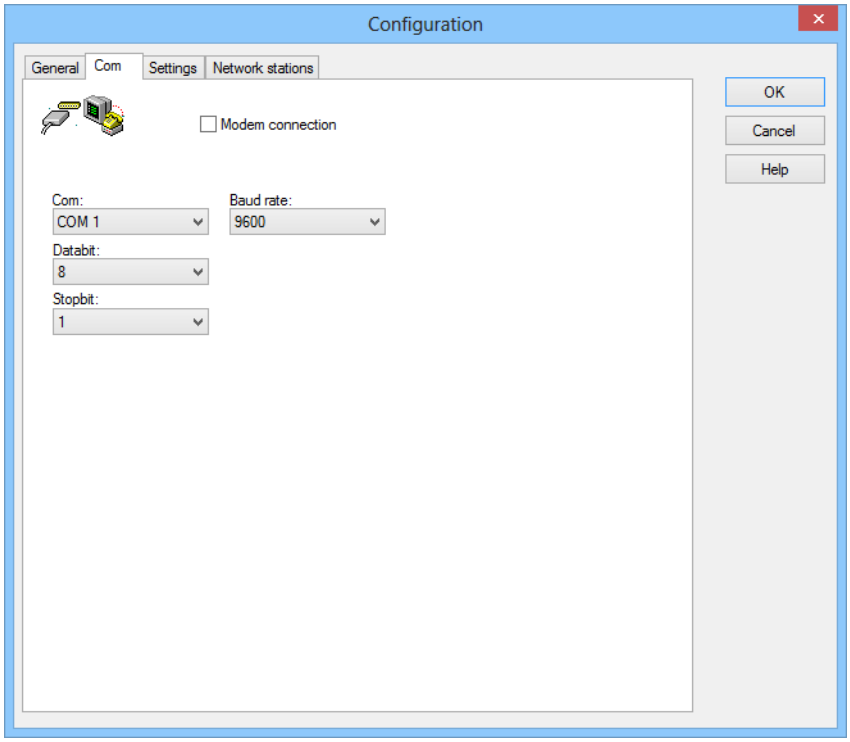
6.2.2 Com

The connection via the COM interface can be established via:

- Cable
- Modem

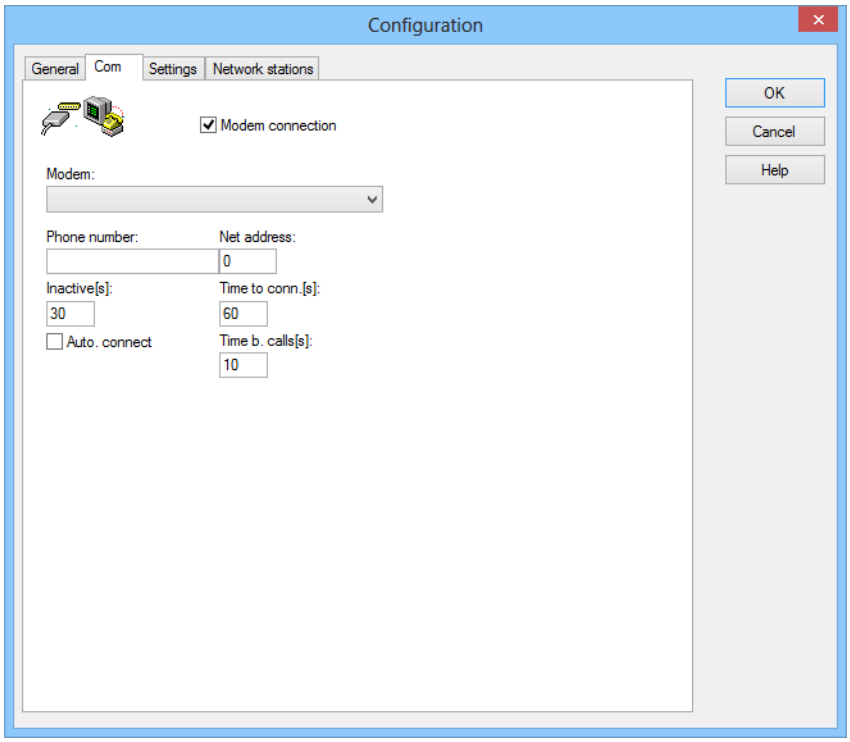
Depending on the selection, different properties can be shown for configuration.

CABLE


Configuration
General
Com
Settings
Network stations
Modem connection
OK
Cancel
Help
Com:
COM 1
Baud rate:
9600
Databit:
8
Stopbit:
1

Parameters	Description
Modem connection	Only available if a modem is connected to the computer. ▸ `Active`: The connection is established using a modem. The corresponding properties are displayed. ▸ `Inactive`: The connection is established using a direct cable. The corresponding properties are shown. Note: Parity and protocol are automatically set by the driver depending on the specifications.
Com	Selection of the serial interface from a drop-down list. Default: `COM 1`
Data bit	Selection of the data word length from a drop-down list. Default: `8`
Stop bit	Selection of the stop bit from a drop-down list. Default: `1`
Baud rate	Selection of the transfer speed from a drop-down list. Custom settings of the control.
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.

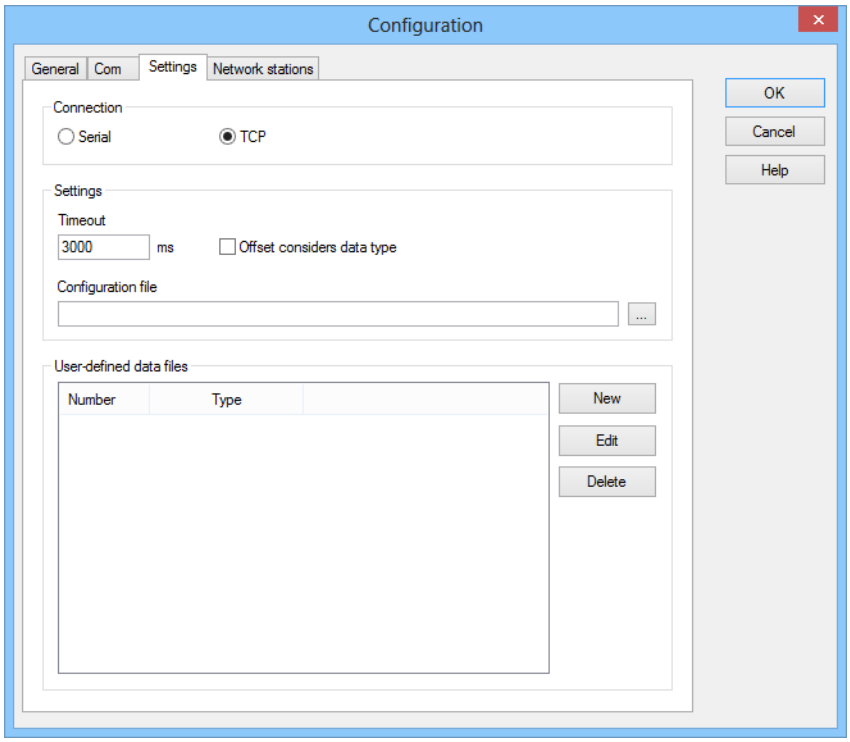
MODEM



Parameters	Description
Modem connection	Only available if a modem is connected to the computer. ▸ `Active`: The connection is established using a modem. The defined project parts will now be generated. ▸ `Inactive`: The connection is established using a direct cable. The corresponding properties are shown. Note: Parity and protocol are automatically set by the driver depending on the specifications.
Modem	Selection of the modem from the drop-down list.
Telephone number	Entry of the telephone number that the modem needs for the connection.
Net address	Entry of the zenon **Net address**.
Inactive(s)	
Time to connect	Waiting time for establishing the connection in seconds.
Automatic connection	`Active`: The connection is made automatically.
Time between calls	Waiting time between two connections in seconds.
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

6.2.3 Settings

The connection settings are configured in this tab.



CONNECTION

Parameters	Description
Connection	Configuration of the connection
Serial	`Active`: A serial connection is made. Configuration is carried out in the **Com** (on page 16) tab.
TCP	`Active`: Connection is made via TCP/IP. Configuration is carried out in the **Network stations** (on page 22) tab.

SETTINGS

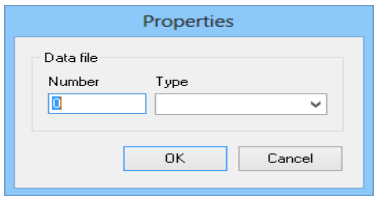
Parameter	Description
Settings	General settings for the connection.
Timeout	Within this time, the PLC must answer to a request; otherwise, the query will be terminated with a communication error.
Offset considers data type	If this checkbox is activated, the offsets of the data points have to be configured according to the data type size. The advantage of this is that the reading of the data points is optimized, which results in a higher data transmission rate. If this option is deactivated, every data point will have its own offset.
Configuration file	The configuration of the data files will be stored under the file name specified here. If using a CE driver, this file also has to be on the CE device. Default: `AllenDF1_Allen Bradley DF1-driver.txt` Note: The proposed name of the file consists of the driver names and their description.

USER-DEFINED DATA FILES

Parameter	Description
User-defined data files	Display and administration of user-defined data files.
New	Opens the dialog to add a user-defined data file.
Edit	Opens the dialog to edit a user-defined data file.
Delete	Deletes user-defined data file from the list.
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.

USER-DEFINED DATA FILE

Clicking on New button opens the dialog to configure the user-defined data file.



Parameter	Description
Data file	Configuration of the data file. For further information, see the Addressing (on page 28) chapter.
Number	Entry of the number. ▸ Minimum: 9 ▸ Maximum: 255
Type	Selection of the type from the drop-down list: ▸ Bit ▸ Integer ▸ Float ▸ String
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

6.2.4 Network stations

An IP address of a zenon **Net address** is allocated in this tab. Variables in zenon then use the **Net address** for communication with the controller. The variables are also informed of the **Net address**; this is linked to the IP address. You can also find information on addressing in the **Addressing** (on page 28) chapter.

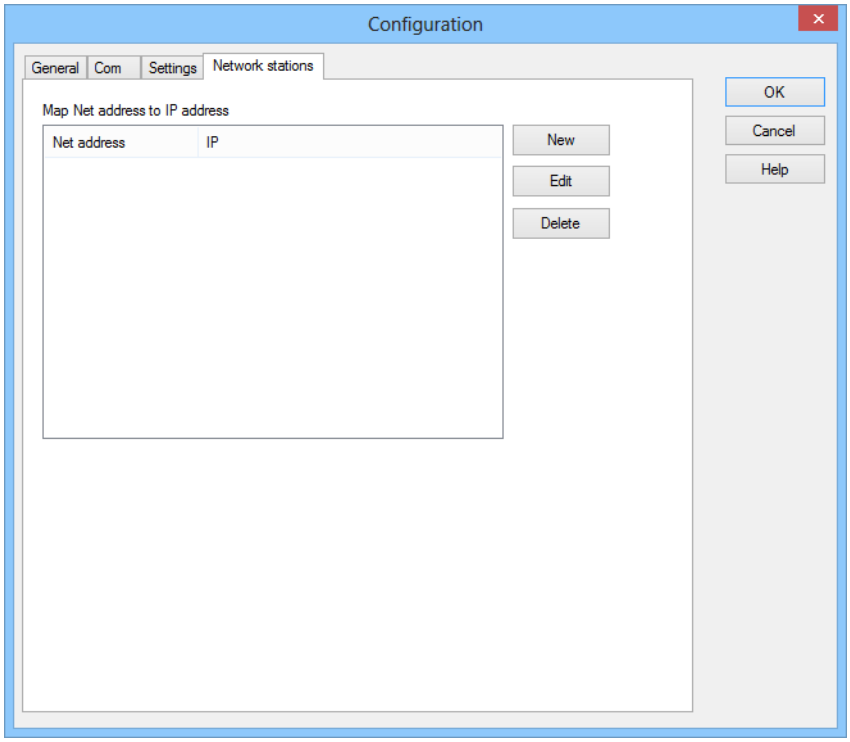
Note: Can only be configured if TCP/IP has been selected as the connection type in the Settings (on page 20) tab.

To configure network stations:

1. click on the button New
2. The dialog to allocate a **Net address** is opened

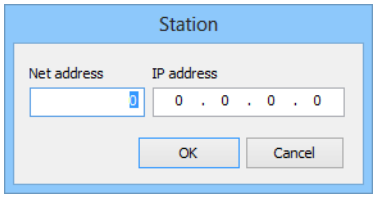
3. Enter the **Net address** and IP address
4. Confirm the configuration by clicking on the OK button

CONFIGURATION OF NETWORK STATIONS



Parameters	Description
Allocation of network address to IP address	Display and administration of the connection.
New	Opens dialog to configure a station by allocating an IP address to a zenon **Net address**.
Edit	Opens the dialog to configure a station that has previously been configured.
Delete	Deletes selected entry from the list.
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.

STATION CONFIGURATION



Parameters	Description
Net address	Entry of the zenon **Net address**.
IP address	Entry of the IP address that is assigned to the **Net address**.
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

7. Creating variables

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

7.1 Creating variables in the Editor

Variables can be created:

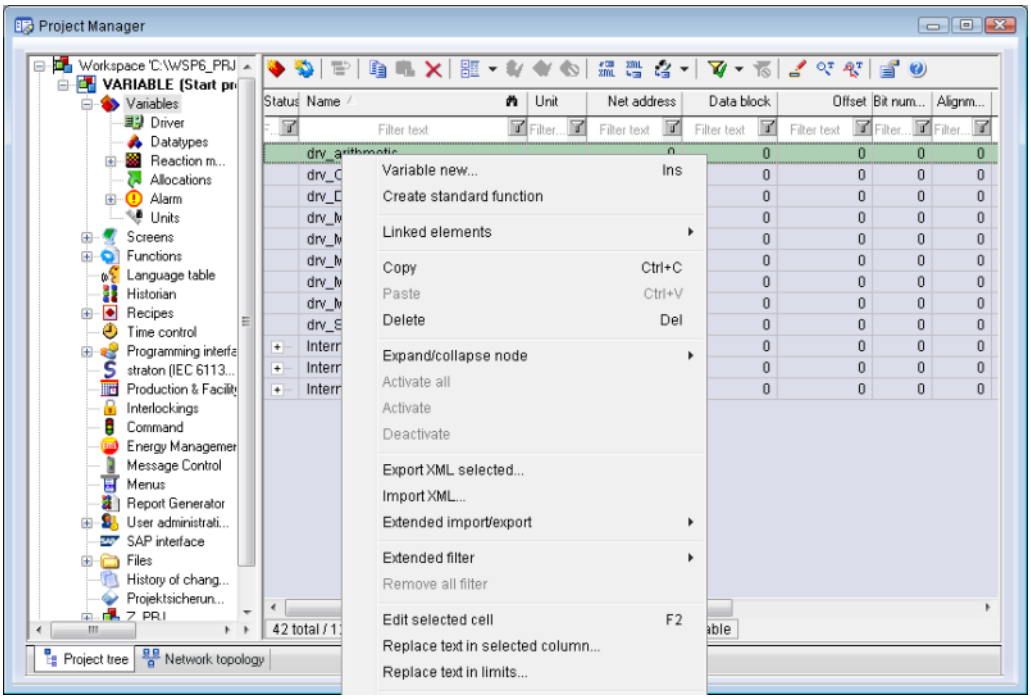
- as simple variables

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

VARIABLE DIALOG

To create a new variable, regardless of which type:

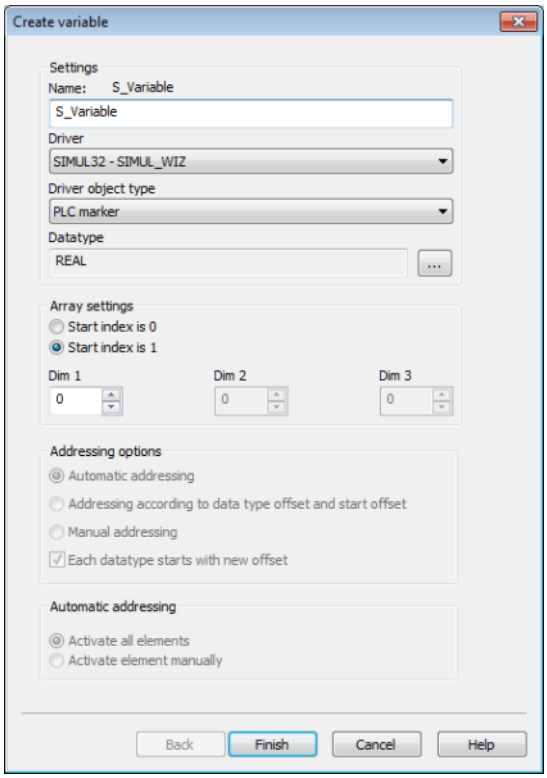
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

2. Configure the variable

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



Property	Description
Name	Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name. Maximum length: 128 characters Attention: The characters **#** and **@** are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive. Note: 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

Property	Description
Name	Freely definable name. Attention: For every zenon project the name must be unambiguous.
Identification	Freely definable identification. E.g. for Resources label, comments, ...
Net address	Network address of variables. This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.
Data block	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 variables. Equal to the memory address of the variable in the PLC. Adjustable from `0` to `4294967295`.
Alignment	not used for this driver
Bit number	Offset of variables. Equal to the memory address of the variable in the PLC. Adjustable from `0` to `4294967295`.
String length/String length	Only available for String variables. Maximum number of characters that the variable can take.
Driver connection/Driver Object Type	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Driver connection/Data Type	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.
Driver connection/Priority	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. 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.

ASSIGNMENT OF ADDRESSES OF ALLEN BRADLEY PLCS TO ZENON

AB control units administer addresses in this form: **Data_type_short+numeral:offset/bit number**.
For example: **B3:100/7**

Assignment of data types and addresses:

Data type	Short form	Digit
BINARY	B	3
TIMER (Only DF1)	T	4
COUNTER (Only DF1)	C	5
CONTROL (Only DF1)	R	6
INTEGER	N	7
FLOATING PT	F	8

To transfer these addresses in zenon correctly:

- Enter the **digit** (3, for example) in the **Data block** property
- The zenon **Offset** corresponds to the address within this bandwidth
- Bit number corresponds to **Bit number**

EXAMPLE

The AB address **B3:100/7** corresponds to the following in zenon:

- **Data block**: 3
- **Offset**: 100
- **Bit number**: 7

ALLOCATION OF IP ADDRESS

IP addresses are configured for AB controllers using zenon **Net address**. To do this, allocate, in the driver configuration in the **Network stations** (on page 22) tab, the IP addresses of the respective **Net address**.

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 driver object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
Timer	22	X	X	BOOL, INT, SINT, UINT, USINT	
Ext. Data block	34	X	X	REAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING	
Communication details	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. You can find detailed information on this in the Communication details (Driver variables) (on page 39) chapter.
Counter	23	R / W	X	BOOL, INT, SINT, UINT, USINT	

Key:

X => supported

-- => not supported

The channel type or the data type is used in the driver for mapping the correct object types or data types. This information is also used for the "Extended variable import/export" via DBF files.

OBJECTS FOR PROCESS VARIABLES IN ZENON

The variables in zenon are defined according to the following aspects:

Net Address: not needed

Data block: The specified data block corresponds to the list number in the PLC

(Lists 0-8 are standard lists that are preprogrammed into the PLC.

The user-defined lists start with number 9).

Offset: The offset corresponds with the address in the variable list on the PLC.

Bit: Corresponds with the bit address in a bit list of the PLC.

Channel name	Data type	Channel type	Object	Read	Write
n.V.	o	o	o	o	o

In the variable definition the data file number has to be entered under "Datenbaustein".

E.g. if a data word with the address 10 should be defined and stored in the data file Integer 7, a word variable with the following settings has to be created: Data block=7 and Offset=10;

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 DATA TYPES

Data type	Description
PLC	zenon
I8	i/u8Bit (signed)
I16	i/u16Bit (signed)
I32	i/u32Bit (signed)
U8	i/u8Bit
U16	i/u16Bit
U32	i/u32Bit
F32	float32
Boolean	Boolean
String	String

zenon data types	Allen Bradley data types
data bit	bit or binary
data byte	Byte

data word	Integer
data doubleword	doubleword
data float	Float
data string	String

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

7.4 Creating variables by importing

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

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

7.4.1 XML import

During XML import of variables or data types, these are first assigned to a driver and then analyzed. Before import, the user decides whether and how the respective element (variable or data type) is to be imported:

- `Import`: The element is imported as a new element.
- `Overwrite`: The element is imported and overwrites a pre-existing element.
- `Do not import`: The element is not imported.

Note: The actions and their durations are shown in a progress bar during import.

REQUIREMENTS

The following conditions are applicable during import:

- Backward compatibility

  At the XML import/export there is no backward compatibility. Data from older zenon versions cannot be taken over. The handover of data from newer to older versions is not supported.

- Consistency

  The XML file to be imported has to be consistent. There is no plausibility check on importing the file. If there are errors in the import file, this can lead to undesirable effects in the project.

  Particular attention must be paid to this, primarily if not all properties exist in the XML file and these are then filled with default values. E.g.: A binary variable has a limit value of 300.

- Structure data types

  Structure data types must have the same number of structure elements.
  Example: A structure data type in the project has 3 structure elements. A data type with the same name in the XML file has 4 structure elements. Then none of the variables based on this data type in the export 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	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	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-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	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix
ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in **project.ini**.
ADJWVBA	R	1	Non-linear value adaption: 0: Non-linear value adaption is used

			1: Non-linear value adaption is not used
ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
ZWREMA	N	16	Linked counter REMA.
MAXGRAD	N	16	Gradient overflow for counter REMA.

Attention

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

LIMIT VALUE DEFINITION

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

Identification	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	technical value or ID number of a linked variable for a dynamic limit value (see VARIABLEx) (if VARIABLEx is `1` and here it is `-1`, the existing variable linkage is not overwritten)
SCHWWERT1	F	16	Threshold value for limit value
HYSTERESE1	F	14	Is not used
BLINKEN1	R	1	Set blink attribute
BTB1	R	1	Logging in CEL
ALARM1	R	1	Alarm
DRUCKEN1	R	1	Printer output (for CEL or Alarm)
QUITTIER1	R	1	Must be acknowledged
LOESCHE1	R	1	Must be deleted
VARIABLE1	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	R	1	Functions linking
ASK_FUNC1	R	1	Execution via Alarm Message List
FUNC_NR1	N	10	ID number of the linked function (if “-1” is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/event group
A_KLASSE1	N	10	Alarm/event class
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	C	66	Limit value text
A_DELAY1	N	10	Time delay
INVISIBLE1	R	1	Invisible

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

7.5 Communication details (Driver variables)

The driver kit implements a number of driver variables. This variables are part of the driver object type **Communication details**. These are divided into:

- Information
- Configuration
- Statistics and
- Error message

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

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

Information

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

For example:

- Variables for modem information are only supported by modem-compatible drivers
- 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.

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	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:

INI ENTRIES

ZENON6.INI

None

PROJECT.INI

None

LIMITATIONS

For correct communication, the PLC has to be set to DF1 Full-Duplex Master and the error check has to be set to CRC (cyclic redundancy check). In the driver configuration the baud rate can be adapted to that of the PLC.

The driver generally does not support the Real Time Data Acquisition functionality.

As the PLC uses lists fixed to data types, it is not possible to access one and the same datapoint in the PLC with different zenon data types.

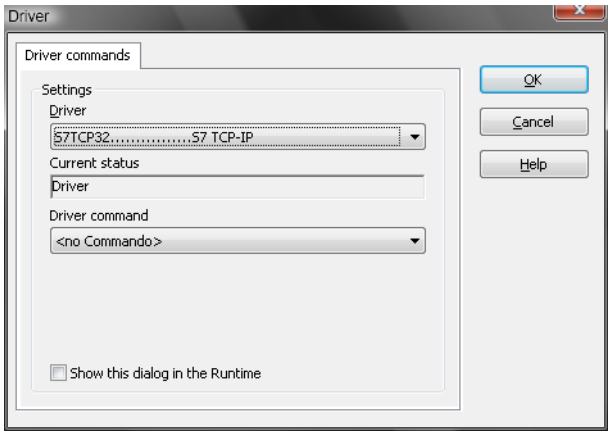
The driver does not support array variables.

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 7.60 -> 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

Error code	Description
-10	A request got no response.
-11	The confirmation for the request has not been received.
-12	The response telegram for the request has not fully been received within the timeout.
-13	The CRC checksum has not been received.
-14	The CRC checksum of the response is incorrect.
-15	Computer or PLC address in the response telegram are incorrect.
-16	The telegram number does not fit with that of the response.
-17	The response "No message existing" received.
-18	The response has an invalid length.
-19	The number of received characters exceeds the receive buffer.
-21	After a repeated request no NAK has been received.
-22	After a repeated request other characters than NAK have been received.
-262144 to -196609	PLC specific error message on reading. STS in LSW/Lsb Ext-STS in LSW/Msb
-524288 to -458753	PLC specific error message on writing. STS in LSW/Lsb Ext-STS in LSW/Msb

10.3 Check list

Check the following in the event of errors:

- Is the COM port in use by another application or are the settings incorrect?
- Is the PLC that you are trying to communicate with connected to the power supply?
- Is the cable between PLC and PC/IPC connected correctly?
- Is the option "CRC" activated in the communication monitoring of the PLC?
- Do the lists, which are queried as zenon variables, exist in the PLC?
- Did you enter all user-defined lists that the driver is supposed to query in the driver configuration?
- Did you configure different variable types for one PLC data list? See chapter "Limitations"
- Did you define variables in zenon that have no address in the PLC?

  This should be checked with the standard lists 0-8. E.g. Integer List N7 addressable up to offset 100 (default).
- Have you analyzed the "driver communication error file" (which errors have occurred)?

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. You can find more information on the Diagnosis Viewer in the Diagnosis Viewer (main.chm::/12464.htm) manual.

The following is required for further analysis of errors:

- The project backup
- LOG files

Send these to your support person after agreement with the customer service department.