

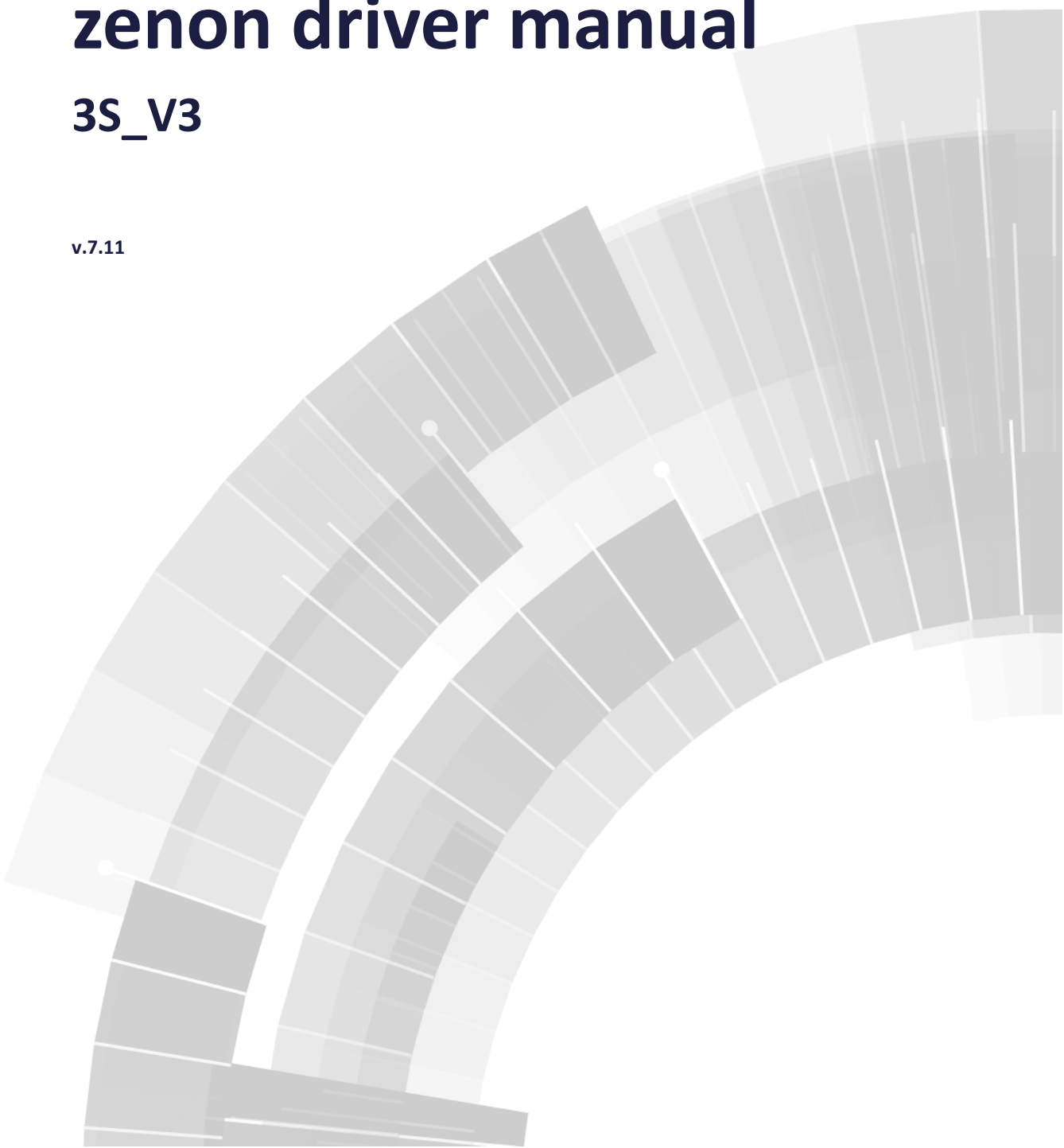


COPADATA
do it your way

zenon driver manual

3S_V3

v.7.11





©2014 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

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

Contents

1. Welcome to COPA-DATA help	5
2. 3S_V3.....	5
3. 3S_V3 - Data sheet.....	7
4. Driver history	9
5. Requirements.....	9
5.1 PC	10
5.2 Control	10
6. Configuration	11
6.1 Creating a driver.....	11
6.2 Settings in the driver dialog	13
6.2.1 General	14
6.2.2 Options	17
6.2.3 Information.....	21
7. Creating variables.....	21
7.1 Creating variables in the Editor.....	22
7.2 Addressing.....	25
7.3 Driver objects and datatypes	26
7.3.1 Driver objects	26
7.3.2 Mapping of the data types	26
7.4 Creating variables by importing	27
7.4.1 XML import.....	28
7.4.2 DBF Import/Export	28
7.4.3 Online- and Offline-Import.....	35
7.5 Driver variables	41
8. Driver-specific functions	47
9. Driver commands	50

10. Error analysis.....	52
10.1 Analysis tool	52
10.2 Error numbers	55
10.3 Check list	56

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>).

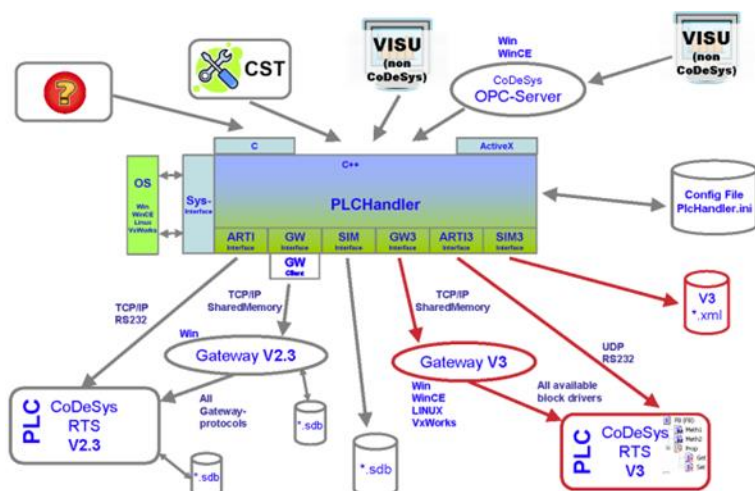
LICENSES AND MODULES

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_V3

GENERAL

The `3s_v3` driver uses the PLC handler from 3S to connect to controllers with `CoDeSys v3`. The PLC handler is an interface for version V2.3 and V3 controllers. The driver currently only supports version V3 controllers.



Version V3 controllers are addressed via the `node address`, which is unique in the network that can be contacted.

A connection to the control can be established.

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

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

SYMBOLIC ADDRESSING

The PLC handler only addresses variables in the controller using its symbolic name. Addressing using the offset of the variables is not possible. However, for complex variables (structures, arrays) the whole (binary) data block of the variable can be addressed using the `base name`. This is used by the driver for RDA (on page 47).

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

In order for the PLC handler to find the variables via the symbolic name in the controller, you must select the variables used in the zenon project in the CoDeSys project configuration software using a

symbol configuration object and transfer this to the controller. In doing so, a symbol file in XML format is created at the same time. This can be used for offline importing of variables.

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

3. 3S_V3 - Data sheet

General:	
Driver file name	3S_V3.exe
Driver name	3S v3 driver for PLC Handler
PLC types	3S CoDeSys v3 based PLCs ; Schneider Electric PAC Drive LMC
PLC manufacturer	3S; Elau; Schneider; Bosch Rexroth;

Driver supports:	
Protocol	3S-Arti; 3S-Gateway;
Addressing: Address-based	-
Addressing: Name-based	x
Spontaneous communication	-
Polling communication	x
Online browsing	x
Offline browsing	x
Real-time capable	-
Blockwrite	-

Modem capable	-
Serial logging	-
RDA numerical	x
RDA String	-

Requirements:	
Hardware PC	-
Software PC	PLC Handler from 3S; The file 'PLCHandlerDll.dll' is delivered with the installation media, you find it in the subdirectory: Installationmedia\AdditionalSoftware\3S PLC Handler\ Please copy the file to the system directory: 64 bit: C:\Windows\SysWOW64 32bit: C:\Windows\System32
Hardware PLC	-
Software PLC	-
Requires v-dll	-

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

4. Driver history

Date	Build number	Change
6/21/2013	7.11.0.7498	Created driver documentation

DRIVER VERSIONING

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

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



Example

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

5. Requirements

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

5.1 PC

ADDITIONAL SOFTWARE

PC

The driver needs the `PLCHandlerD11.dll`, which is produced by 3S and contains `PLCHandler-SDK`. The DLL must be version 3.5 or newer. It must be copied to the installation folder of the driver or to the standard Windows folder.

Standard folder:

- ▶ 32-Bit: `C:\Windows\System32`
- ▶ 64-Bit: `C:\Window\SysWOW64`

The DLL is on the zenon installation medium in the subfolder: `AdditionalSoftware\3S PLC Handler`.

CE

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

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

5.2 Control

CONTROLS

The `3s_v3` driver is for connection to a PLC that uses `3s CoDeSys v3` as its operating system, for example `3S CoDeSys Control Win V3` or `Schneider LMC PacDrive`.

The connection to the controller can be chosen as either the ARTI interface (Asynchronous RunTime Interface) or a local or remote gateway.

6. Configuration

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



Information

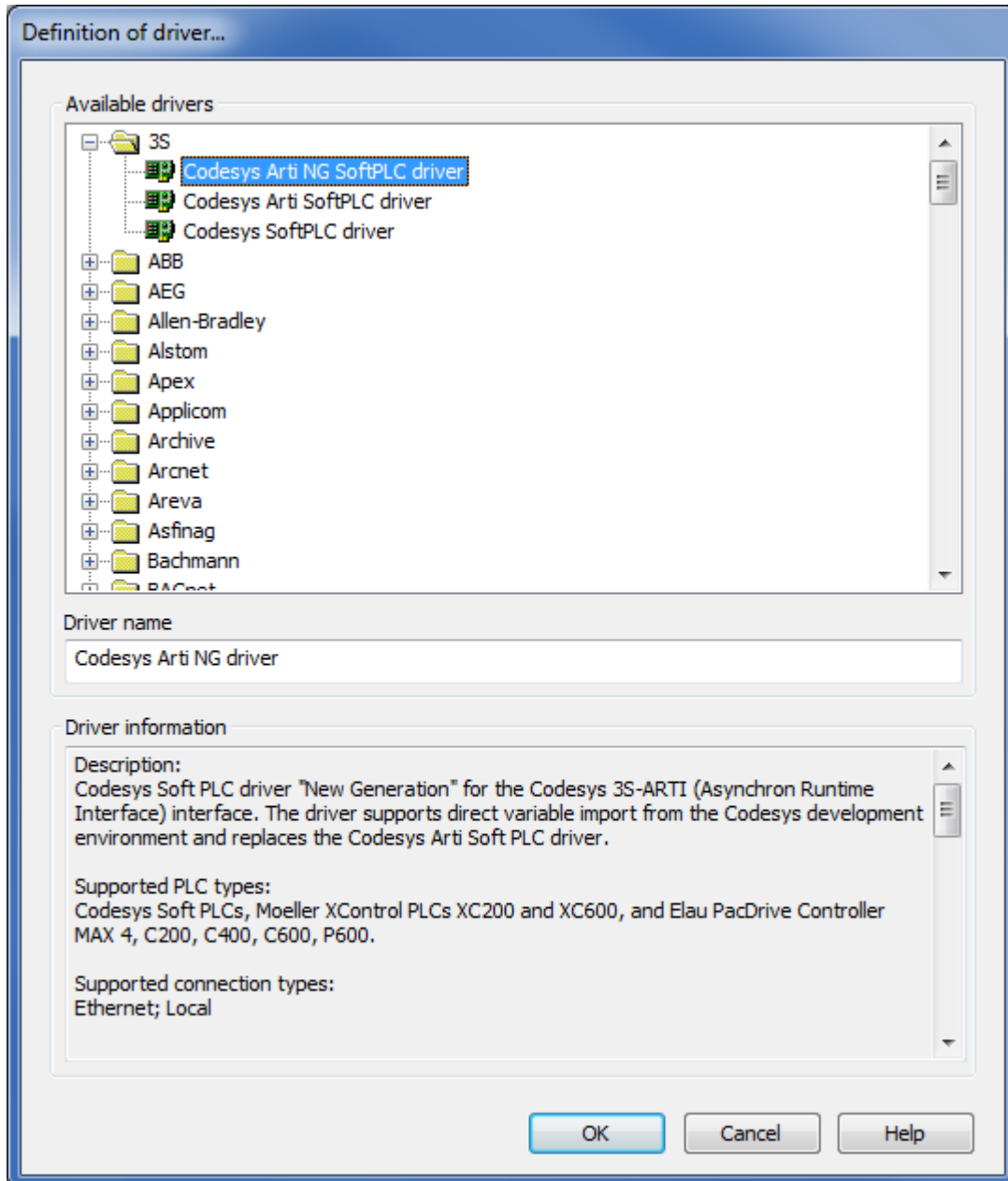
Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.

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.

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



- Select the desired driver and give it a name:
 - The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.
 - The driver name is part of the file name. Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (_).

- **Attention:** This name cannot be changed later on.
4. Confirm the dialog with **OK**. In the following dialog the single configurations of the drivers are defined.

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



Information

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

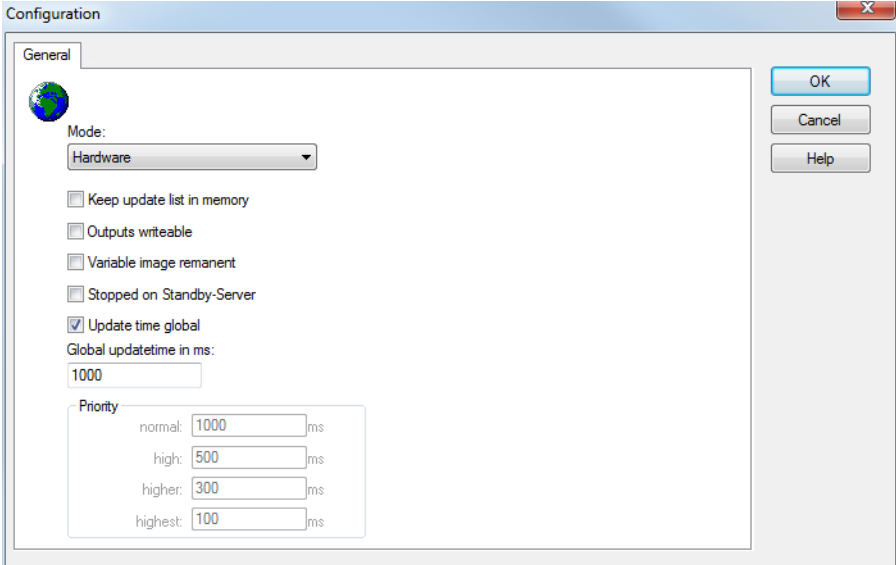
- ▶ Internal
- ▶ MathDr32
- ▶ SysDrv.

▶

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General



Parameters	Description
Mode	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> ▶ Hardware: <p>A connection to the control is established.</p> ▶ Simulation static <p>No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by straton. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver.</p> ▶ Simulation - counting <p>No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically.</p> ▶ Simulation - programmed <p>N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the straton Workbench and runs in a straton Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).</p>
Keep update list in the memory	<p>Variables which were requested once are still requested from the control even if they are currently not needed.</p> <p>This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.</p>
Outputs writeable	<p>Active: Outputs can be written.</p> <p>Inactive: Writing of outputs is prevented.</p> <p>Note: Not available for every driver.</p>

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

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

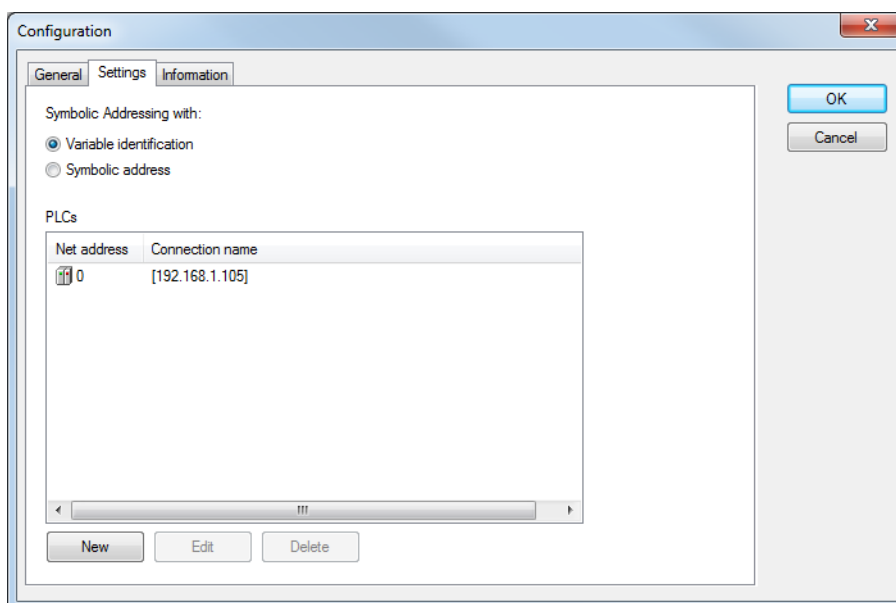
UPDATE TIME FOR CYCLICAL DRIVER

The following applies for cyclical drivers:

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

6.2.2 Options

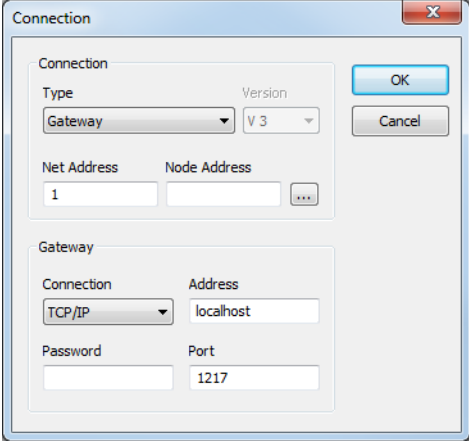
Selection of controllers that are to be used. For each controller, it is possible to select whether the connection is established via ARTI or a gateway.



Parameters	Description
Symbolic addressing via	<p>Selection of whether the <code>variable identification</code> or the <code>symbolic address</code> option is to be used for symbolic addressing of variables.</p> <p>This selection can be made from zenon version 7.10 onwards.</p> <p>The <code>variable identification</code> is always used up to and including version 7.00. In this case, the options <code>variable identification</code> and <code>symbolic address</code> are not displayed.</p>
Variable identification	Active: Symbolic addressing of variables is carried out using the <code>variable identification</code> .
Symbolic address	Active: Symbolic addressing of the variables is carried out using the <code>symbolic address</code> .
controls	List of configured connections.
New	Opens the dialog (on page 19) for configuring a connection.
Edit	Opens the dialog (on page 19) for editing the selected connection.
Clear	Deletes the selected connection after a confirmation message.
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.

Connection

Configuration of the connection to a controller.



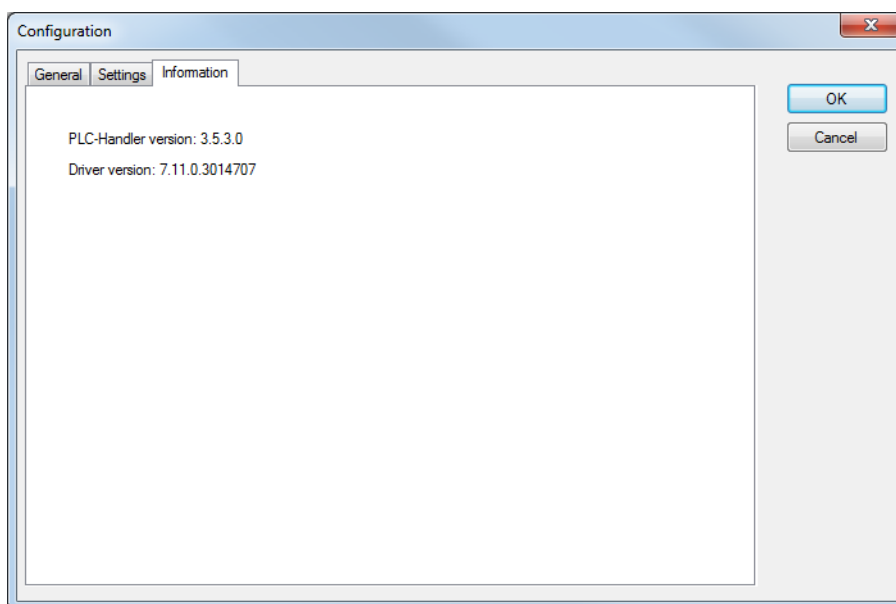
Parameters	Description
Connection type	<p>Selection of the connection type from drop-down list:</p> <ul style="list-style-type: none"> ▶ ARTI: via the local network card ▶ Gateway: via a gateway on the local or remote computer
Version	<p>Selection of the version to be used from a drop-down list.</p> <p>Can only be configured if several versions are supported.</p>
Net address	<p>Entry of the net address.</p> <p>This corresponds to the zenon Net address property.</p>
node address	<p>The node address of the PLC.</p> <p>Make an entry in the field by clicking on the ... button and then scanning the network for controllers that can be contacted via the selected connection type.</p> <p>All controllers that can be contacted with the selected connection settings are offered for selection.</p> <p>The node address can, depending on the connection type, be different for the same controller.</p>
Gateway	<p>Configuration of the gateway.</p>
Connection	<p>Selection of the connection type from the the PLC handler to the gateway via a drop-down list:</p> <ul style="list-style-type: none"> ▶ TCP/IP: TCP/IP connection ▶ Shared Memory: Use of the shared memory on the local computer
Address	<p>IP address of the gateway.</p> <p>Only necessary for communication via TCP/IP.</p>
Password	<p>Password of the gateway.</p> <p>For communication via TCP/IP only.</p>
Port	<p>Port of the gateway.</p> <p>For communication via TCP/IP only.</p> <p>Default: 1217</p>
OK	<p>Applies settings and closes the dialog.</p>

Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

6.2.3 Information

Information on the versions of:

- ▶ PLC-Handler
- ▶ Drivers



7. Creating variables

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

7.1 Creating variables in the Editor

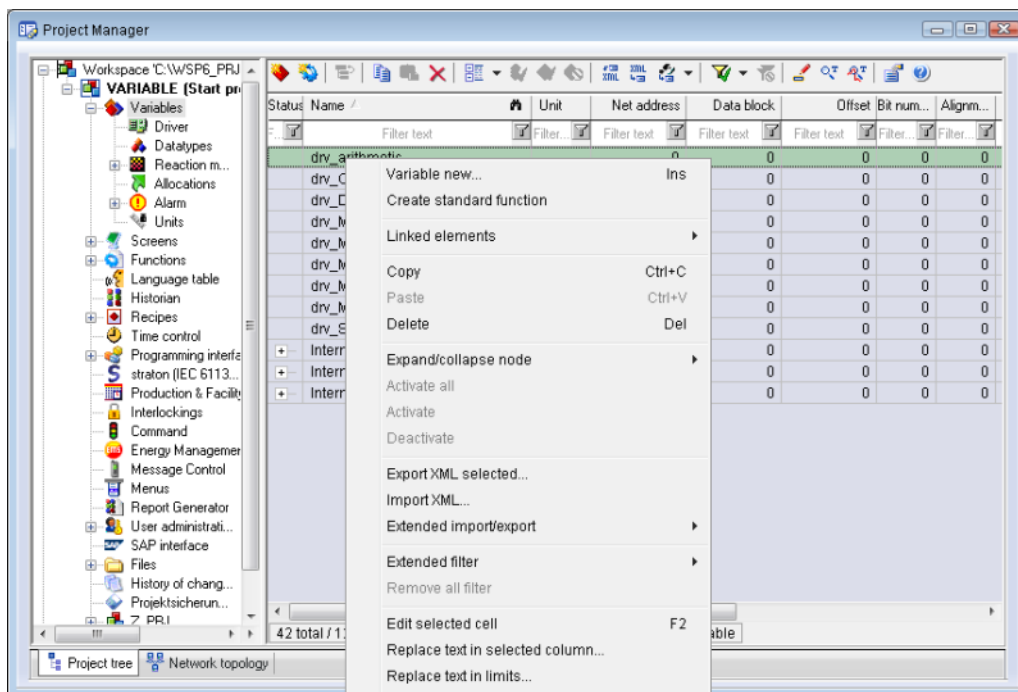
Variables can be created:

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

VARIABLE DIALOG

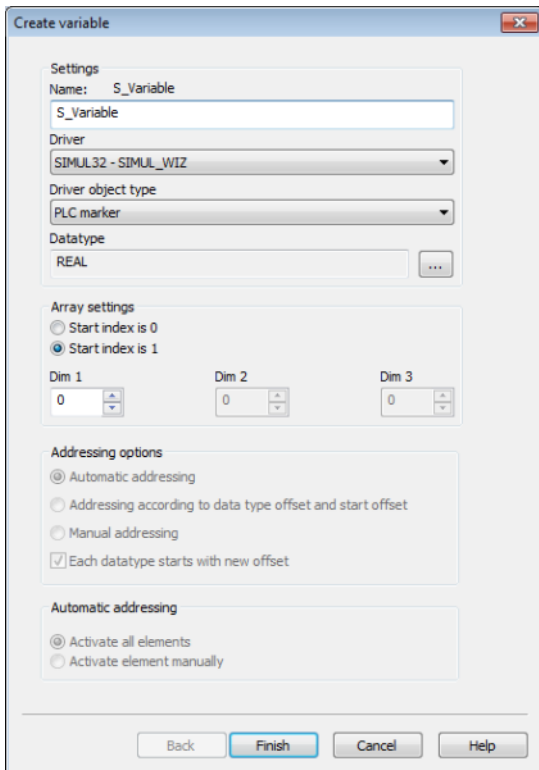
To create a new variable, regardless of which type:

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



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

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



Parameters	Description
Name	<p>Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name.</p> <p>Maximum length: 128 characters</p> <p>Attention: The characters # and @ are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the Finish button remains inactive.</p>
Drivers	<p>Select the desired driver from the drop-down list.</p> <p>Note: If no driver has been opened in the project, the driver for internal variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is automatically loaded.</p>
Driver object type (cti.chm::/28685.htm)	<p>Select the appropriate driver object type from the drop-down list.</p>

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

INHERITANCE FROM DATA TYPE

Measuring range, Signal range and Set value are always:

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

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

7.2 Addressing

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

7.3 Driver objects and datatypes

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

7.3.1 Driver objects

The following object types are available in this driver:

Driver object type	Channel type	Read / Write	Supported data types	Description
Marker	8	R / W	BOOL, USINT, SINT, BYTE, UINT, WORD, INT, UDINT, DWORD, DINT, ULINT, LINT, LWORD, REAL, LREAL, STRING, WSTRING, DT, TOD, DATE, TIME	
Driver variable	35	R / W	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	<p>Variable for the static analysis of the communication; is transferred between driver and Runtime (not to the PLC).</p> <p>Note: The addressing and the behavior is the same for most zenon drivers.</p> <p>Find out more in the chapter about the Driver variables (on page 41)</p>

7.3.2 Mapping of the data types

All variables in zenon are derived from IEC data types. The following table compares the IEC datatypes with the datatypes of the PLC.

Control	zenon	Data type
BIT, BOOL	BOOL	8
USINT	USINT	9
BYTE	BYTE	22
SINT	SINT	10
WORD	WORD	23
UINT, UINT16	UINT	2
INT, INT16, ENUM	INT	1
UDINT	UDINT	4
DINT	DINT	3
ULINT	ULINT	27
DWORD	DWORD	24
LINT	LINT	26
REAL	REAL	5
LREAL	LREAL	6
STRING	STRING	12
WSTRING	WSTRING	21
DATE	DATE	18
TIME	TIME	17
DT, DATE_AND_TIME	DT	20
TOD, TIME_OF_DAY	TOD (Time of Day)	19

Data type: The 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

Description	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the MAX_LAENGE entry in project.ini .
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in project.ini .
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the MAX_LAENGE entry in project.ini .
Unit	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Bus address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADDRESS	N	5	Offset

BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipe Group Manager
LES_SCHR	R	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-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 DEFINITION

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

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

EXPRESSIONS IN THE COLUMN "COMMENT" REFER TO THE EXPRESSIONS USED IN THE DIALOG BOXES FOR THE DEFINITION OF VARIABLES. FOR MORE INFORMATION, SEE CHAPTER VARIABLE DEFINITION.

7.4.3 Online- and Offline-Import

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

REQUIREMENTS FOR THE IMPORT

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

Depending on the settings regarding whether the `variable identification` or the `symbolic address` should be used for addressing, the `symbolic name` is entered in the corresponding property.

The imported `symbolic name` with the node address prefixed is entered as a variable name in order to keep any symbol names that may have the same name unique. The variable name can therefore not be used for symbolic addressing.

When importing a variable, the variables to be imported can be filtered. Either all existing or only new variables are offered for selection.

In addition, orphaned variables can be searched for and deleted. In the selection list, all variables that no longer have a symbol name or are no longer in the controller are offered for deletion.

IMPORTING VARIABLES WITH THE WIZARD

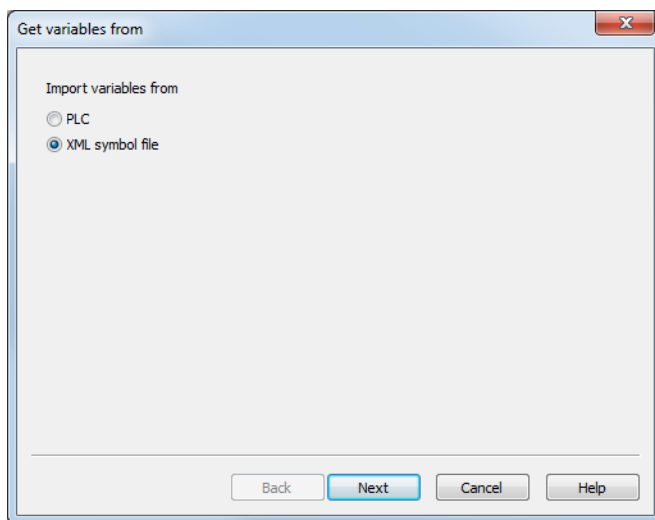
To import variables:

1. select `Import variables from driver` from the context menu of the driver
2. The wizard is opened.
3. Select, in the start window, whether the variables are to be imported from the controller or from an XML file.
4. click on `Next`.

5. Select the controller and, for offline import, the XML file.
6. click on **Next**.
7. configure the filter settings.
8. Click on **Finish**.
9. The selection list is displayed
10. Select the variables to be imported.
11. Click on **OK**.

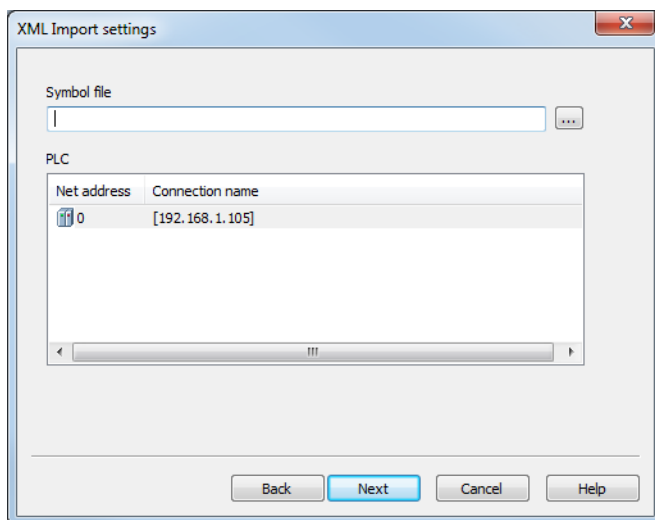
The variables are imported If an error occurs, a corresponding message is displayed.

SELECTING THE IMPORT SOURCE



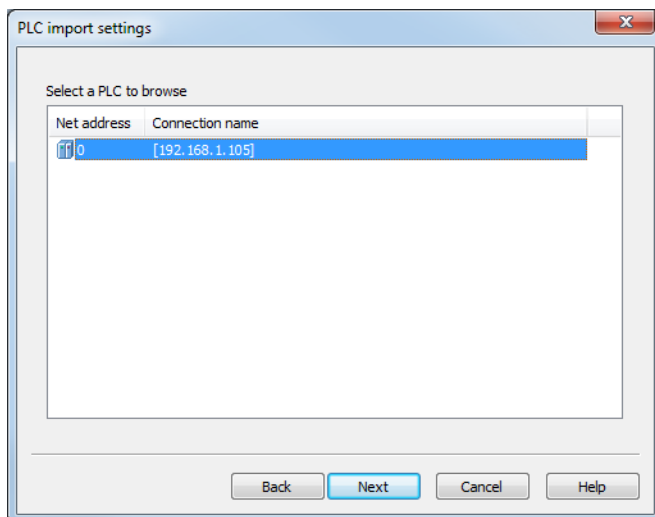
Parameters	Description
Import variables from	Selection of import source.
Control	Active: Variables are always imported from a controller.
XML symbol file	Active: Variables are imported from an XML file created with the CoDeSys software.
Next	Accepts setting and switches to the next window.
Cancel	Cancels the import.
Help	Opens online help.

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



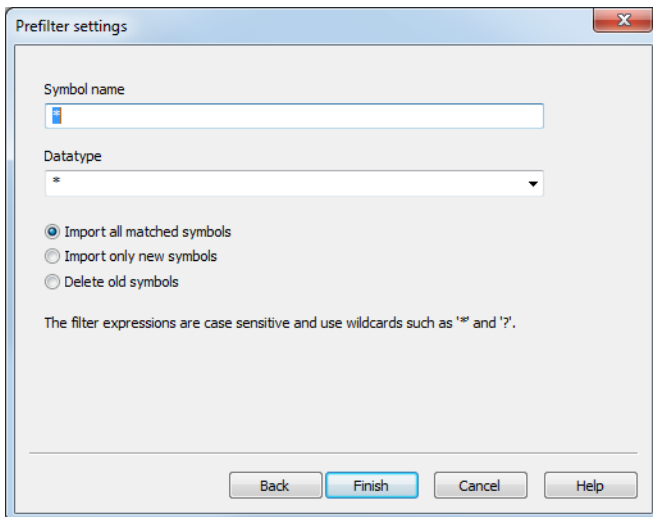
Parameters	Description
Symbol file	Selection of the symbol file. Only available if an XML file was selected as the source in the start window.
Control	Selection of the controller, if variables are imported, from a list.
Back	Switches to the previous window.
Next	Accepts setting and switches to the next window.
Cancel	Cancels the import.
Help	Opens online help.

CONTROLLER SOURCE: SELECTING THE CONTROLLER



Parameters	Description
Control	Selection of the controller from which variables are to be imported, from a list.
Back	Switches to the previous window.
Next	Accepts setting and switches to the next window.
Cancel	Cancels the import.
Help	Opens online help.

CONFIGURATION OF THE FILTER



Prefilter settings

Symbol name

Datatype

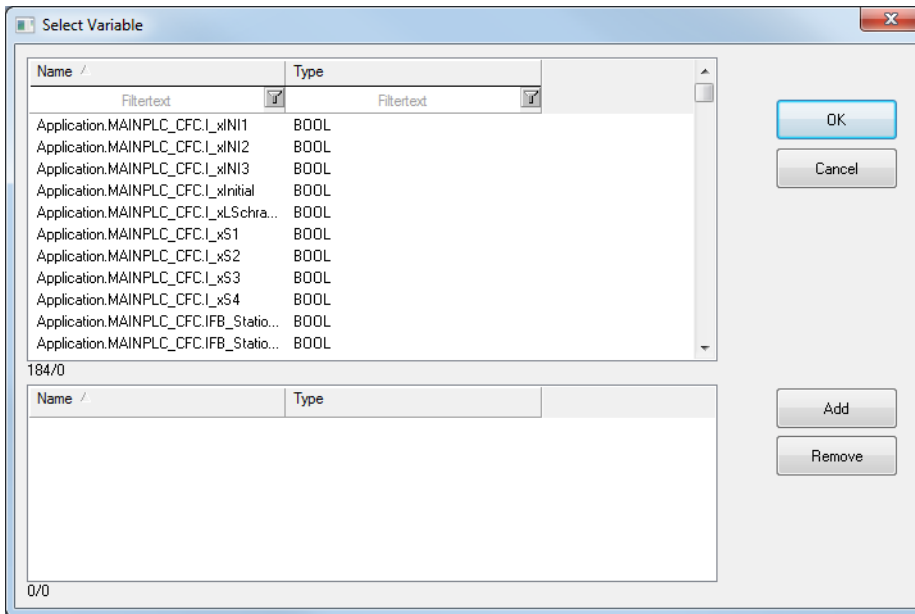
Import all matched symbols
 Import only new symbols
 Delete old symbols

The filter expressions are case sensitive and use wildcards such as '*' and '?'.

Back Finish Cancel Help

Parameters	Description
Symbol name	<p>Entry of the symbol name as a filter criteria.</p> <p>The name can be entered with the following parameters:</p> <ul style="list-style-type: none"> ▶ * ▶ ?
Data type	<p>Entry of a data type in the combobox or selection from a drop-down list.</p> <p>The following placeholders can be used for input:</p> <ul style="list-style-type: none"> ▶ * ▶ ? <p>* Selects all projects.</p>
Import all corresponding symbols	Active: Offers all symbols that correspond to the filter for import.
Import new symbols only	Active: Offers only symbols that are not yet present in the zenon project for import.
Delete all symbols	Active: Offers all symbols that exist in the zenon project that do not exist in the controller for deletion.
Back	Switches to the previous window.
Finish	Accepts setting, starts the import and opens the window with the variables that are can be imported or deleted.
Cancel	Cancels the import.
Help	Opens online help.

SELECTION OF THE VARIABLES



Parameters	Description
List of variables that can be imported	Displays all variables that can be imported or deleted depending on the settings in the wizard.
Display of existing/highlighted variables	Displays the number of existing and currently-highlighted variables.
List of variables to be imported	Displays all variables that are selected for import or deletion.
Display of existing/highlighted variables	Displays the number of variables selected for import or deletion and the currently-highlighted variables.
Add	Adds the variables highlighted in the Importable variables list to the Variables to be imported list.
Delete	Deletes highlighted variables from the Variables to be imported list.
OK	Accepts selection and imports variables or deletes variables.
Cancel	Cancels import process.

7.5 Driver variables

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

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

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

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



Information

Not every driver supports all driver variants.

For example:

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

INFORMATION

Name from import	Type	Offset	Description
MainVersion	UINT	0	Main version number of the driver.
SubVersion	UINT	1	Sub version number of the driver.
BuildVersion	UINT	29	Build version number of the driver.
RTMajor	UINT	49	zenon main version number
RTMinor	UINT	50	zenon sub version number
RTSp	UINT	51	zenon service pack number
RTBuild	UINT	52	zenon build number
LineStateIdle	BOOL	24.0	TRUE, if the modem connection is idle
LineStateOffering	BOOL	24.1	TRUE, if a call is received
LineStateAccepted	BOOL	24.2	The call is accepted
LineStateDialtone	BOOL	24.3	Dialtone recognized
LineStateDialing	BOOL	24.4	Dialing active
LineStateRingBack	BOOL	24.5	While establishing the connection
LineStateBusy	BOOL	24.6	Target station is busy
LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
LineStateProceeding	BOOL	24.9	Dialing completed
LineStateOnHold	BOOL	12:00 AM	Connection in hold
LineStateConferenced	BOOL	12:00 AM	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	12:00 AM	Connection in hold for conference
LineStateOnHoldPendTransfer	BOOL	24.13	Connection in hold for transfer
LineStateDisconnected	BOOL	24.14	Connection terminated.
LineStateUnknow	BOOL	24.15	Connection status unknown

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

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If <code>TRUE</code> , the modem is automatically reconnected for reading
ApplyCom	BOOL	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method <code>SrvDrvVarApplyCom</code> being called (which currently has no further function).
ApplyModem	BOOL	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method <code>SrvDrvVarApplyModem</code> . This closes the current connection and opens a new one according to the settings <code>PhoneNumberSet</code> and <code>ModemHwAdrSet</code> .
PhoneNumberSet	STRING	38	Telephone number, that should be used
ModemHwAdrSet	DINT	39	Hardware address for the telephone number
GlobalUpdate	UDINT	3	Update time in milliseconds (ms).
BGlobalUpdaten	BOOL	4	<code>TRUE</code> , if update time is global
TreiberSimul	BOOL	5	<code>TRUE</code> , if driver in sin simulation mode
TreiberProzab	BOOL	6	<code>TRUE</code> , if the variables update list should be

			kept in the memory
ModemActive	BOOL	7	TRUE, if the modem is active for the driver
Device	STRING	8	Name of the serial interface or name of the modem
ComPort	UINT	9	Number of the serial interface.
Baud rate	UDINT	10	Baud rate of the serial interface.
Parity	SINT	11	Parity of the serial interface
ByteSize	USINT	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.
StopBit	USINT	13	Number of stop bits of the serial interface.
Autoconnect	BOOL	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	STRING	17	Current telephone number
ModemHwAdr	DINT	21	Hardware address of current telephone number
RxIdleTime	UINT	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	UDINT	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	UDINT	20	Number of ringing tones before a call is accepted
ReCallIdleTime	UINT	53	Waiting time between calls in seconds (s).
ConnectTimeout	UINT	54	Time in seconds (s) to establish a connection.

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	UDINT	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	UDINT	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	UDINT	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	UDINT	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	UDINT	33	Number of writing errors
ReadSucceedCount	UDINT	35	Number of successful reading attempts
MaxCycleTime	UDINT	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	UDINT	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	UDINT	26	Number of writing attempts
ReadErrorCount	UDINT	34	Number of reading errors
MaxUpdateTimeNormal	UDINT	56	Time since the last update of the priority group Normal in milliseconds (ms).
MaxUpdateTimeHigher	UDINT	57	Time since the last update of the priority group Higher in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group High in milliseconds (ms).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group Highest in milliseconds (ms).
PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

ERROR MESSAGES

Name from import	Type	Offset	Description
ErrorTimeDW	UDINT	2	Time (in seconds since 1.1.1970), when the last error occurred.
ErrorTimeS	STRING	2	Time (in seconds since 1.1.1970), when the last error occurred.
RdErrPrimObj	UDINT	42	Number of the PrimObject, when the last reading error occurred.
RdErrStationsName	STRING	43	Name of the station, when the last reading error occurred.
RdErrBlockCount	UINT	44	Number of blocks to read when the last reading error occurred.
RdErrHwAdresse	DINT	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	UDINT	46	Block number when the last reading error occurred.
RdErrMarkerNo	UDINT	47	Marker number when the last reading error occurred.
RdErrSize	UDINT	48	Block size when the last reading error occurred.
DrvError	USINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

8. Driver-specific functions

The driver supports the following functions:

RDA

PROBLEM

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

SOLUTION

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

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

EXAMPLES

RDA-TYP 1:

```

TYPE RDA_1 :                               // Structure for RDA type 1
  STRUCT
    Trigger : DINT;                         // trigger variable
    Count : UDINT;                          // number of data sets
    Cycle : UDINT;                          // cycle time in [ms] (only relevant for type 1
and 4)
    RDA_Type : UDINT;                       // RDA-type, 1 - 4
    Oldest : UDINT;                         // index of the oldest value (only relevant for
type 1)
    Data : ARRAY[0..19] OF DINT;           // Payload
  END_STRUCT
END_TYPE

```

RDA-TYP 2:

```

TYPE RDA_DATA_2 :                          // Structure for RDA type 2 payload
  STRUCT
    Value : DINT;                           // value
    TimeStamp : ARRAY[0..3] OF BYTE;       // Time stamp (hour, minute, second, 1/100th
second)
  END_STRUCT
END_TYPE

TYPE RDA_2 :                               // Structure for RDA type 2
  STRUCT

```



```

    Trigger : DINT;                // trigger variable
    Count : UDINT;                 // Number of data sets
    Cycle : UDINT;                // Cycle time in [ms] (only relevant for
type 1 and 4)
    RDA_Type : UDINT;             // RDA-type, 1 - 4
    Oldest : UDINT;              // Index of the oldest value (only relevant for
type 1)
    Data : ARRAY[0..19] OF RDA_DATA_2; // Payload
END_STRUCT
END_TYPE

```

RDA-TYP 3:

```

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

```

```

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

```

RDA-TYP 4:

```

TYPE RDA_4 :                     // Structure for RDA type 4
STRUCT
    Trigger : DINT;             // trigger variable
    Count : UDINT;              // Number of data sets

```

```

    Cycle : UDINT;                // Cycle time in [ms] (only relevant for
type 1 and 4)
    RDA_Type : UDINT;            // RDA-type, 1 - 4
    Oldest : UDINT;             // Index of the oldest value (only relevant for
type 1)
    TimeStamp : ARRAY[0..7] OF BYTE; // Time stamp of the first value (year, month, day,
hour, minute, second, 1/100th second, reserve)
    Data : ARRAY[0..19] OF DINT;   // Payload
END_STRUCT
END_TYPE

```

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

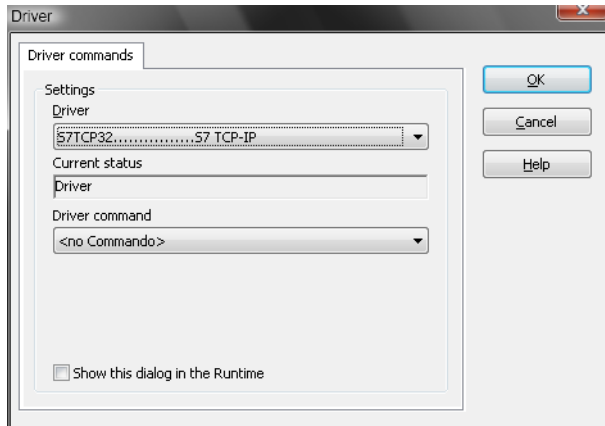
9. Driver commands

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

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

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

- ▶ The dialog for configuration is opened



Parameters	Description
Drivers	Drop-down list with all drivers which are loaded in the project.
Current state	Fixed entry which has no function in the current version.
Driver commands	Drop-down list for the selection of the command.
▶ Start driver (online mode)	Driver is reinitialized and started.
▶ Stop driver (offline mode)	Driver is stopped. No new data is accepted. Note: If the driver is in offline mode, all variables that were created for this driver receive the status switched off (OFF; Bit 20).
▶ Driver in simulation mode	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▶ Driver in hardware mode	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▶ Driver-specific command	Enter driver-specific commands. Opens input field in order to enter a command.
▶ Activate driver write set value	Write set value to a driver is allowed.
▶ Deactivate driver	Write set value to a driver is prohibited.

write set value	
▶ Establish connection with modem	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
▶ Disconnect from modem	Terminate connection (for modem drivers)
Show this dialog in the Runtime	The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

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

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

10. Error analysis

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

10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (`main.chm::/12464.htm`) program that was also installed with zenon. You can find it under *Start/All programs/zenon/Tools 7.11 -> Diagviewer*.

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

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

and events.

In the Diagnosis Viewer you can also:

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

Hints:

1. In Windows CE even errors are not logged per default due to performance reasons.
2. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
3. The Diagnosis Viewer does not display all columns of a log file per default. To display more columns activate property **Add all columns with entry** in the context menu of the column header.
4. If you only use **Error logging**, the problem description is in column **Error text**. For other diagnosis level the description is in column **General text**.
5. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** and/or **Error code** and/or **Driver error parameter (1 and 2)**. Hints on the meaning of error codes can be found in the driver documentation and the protocol/PLC description.
6. At the end of your test set back the diagnosis level from **Debug** Or **Deep Debug**. At **Debug** and **Deep Debug** there are a great deal of data for logging which are saved to the hard drive and which can influence your system performance. They are still logged even after you close the **Diagnosis Viewer**.



Information

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

10.2 Error numbers

Error text in the log	no.	Meaning
RESULT_FAILED	-1	Action erroneous.
RESULT_OK	0	Action successful
RESULT_PLC_NOT_CONNECTED	1	PLC not connected
RESULT_PLC_LOGIN_FAILED	2	Login to PLC has failed
RESULT_PLC_NO_CYCLIC_LIST_DEFINED	3	No cyclic list has been found
RESULT_PLCHANDLER_INACTIVE	4	PLCHandler instance was set inactive
RESULT_LOADING_SYMBOLS_FAILED	5	Loading of the symbols has failed
RESULT_ITF_NOT_SUPPORTED	6	The defined communication interface is not valid or not supported
RESULT_COMM_FATAL	7	Communication error occurred during action
RESULT_NO_CONFIGURATION	8	Wrong or erroneous configuration of the PLCHandler
RESULT_INVALID_PARAMETER	9	At least one parameter is invalid
RESULT_ITF_FAILED	10	Communication interface could not be initialized correctly (e. g. Gateway DLLs not available)
RESULT_NOT_SUPPORTED	11	Method not yet supported resp. implemented for this configuration (e. g. not supported for this interface)
RESULT_EXCEPTION	12	Handled exception in a low layer occurred during action
RESULT_TIMEOUT	13	Timeout exceeded
RESULT_STILL_CONNECTED	14	PLC already connected (at a further ::Connect() call)
RESULT_RECONNECTTHREAD_STILL_ACTIVE	15	Reconnect Thread already active (started at a further ::Connect() call)
RESULT_PLC_NOT_CONNECTED_SYMBOLS_LOADED	16	No connection to the PLC, but symbols available offline
RESULT_NO_UPDATE	17	Asynchronous operation (e. g. cyclic read of variables) has not yet finished

RESULT_OCX_CONVERSION_FAILED	18	Error during conversion of values inside the PLCHandler's ActiveX interface occurred
RESULT_TARGETID_MISMATCH	19	PLC does not match to the passed target id etc.
RESULT_NO_OBJECT	20	No object found for the required action (e. g. tried to get an element beyond the end of the list)
RESULT_COMPONENTS_NOT_LOADED	21	PLCHandler instantiation has failed, because of missing components
RESULT_BUSY	22	Last action still in progress, cannot start the required one
RESULT_DISABLED	23	Feature is disabled by the configuration (e. g. Logging)
RESULT_PLC_FAILED	24	Communication to the PLC was successful, but the PLC has returned a bad result

10.3 Check list

Questions and hints for fault isolation:

GENERAL TROUBLESHOOTING

- ▶ Is the PLC connected to the power supply?
- ▶ Analysis with the **Diagnosis Viewer** (on page 52):
-> Which messages are displayed?
- ▶ Are the participants available in the **TCP/IP** network?
- ▶ Can the PLC be reached via the **Ping** command?

Ping: *Open command line -> ping <IP address> (e.g. ping 192.168.0.100) -> press Enter.*

Do you receive an answer with a time or a time-out?

- ▶ Can the PLC be reached via **Telnet**?

Telnet: *Command line Enter open, telnet <IP address port number> (for example for Modbus: telnet 192.168.0.100 502) -> Press Return key.*

If the monitor display turns black, a connection could be established.

- ▶ Did you configure the Net address in the address properties of the variable correctly?
 - Does the addressing match with the configuration in the driver dialog?
 - Does the net address match the address of the target station?
- ▶ Did you use the right object type for the variable?

Example: Driver variables are purely statistics variables. They do not communicate with the PLC. (See chapter Driver variable (on page 41).)

- ▶ Does the offset addressing of the variable match the one in the PLC?

SOME VARIABLES REPORT INVALID.

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

VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY

Driver is not working. Check the:

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

VARIABLES ARE DISPLAYED WITH A BLUE DOT

The communication in the network is faulty:

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

VALUES ARE DISPLAYED INCORRECTLY

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

DRIVER FAILS OCCASIONALLY

Analysis with the `Diagnosis Viewer` (on page 52):

-> Which messages are displayed?