



zenon
by COPA-DATA

zenon driver manual

FRAPORT

v.8.10



COPA-DATA

© 2019 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

Distribution and/or reproduction of this document or parts thereof in any form are permitted solely with the written permission of the company COPA-DATA. Technical data is only used for product description and are not guaranteed qualities in the legal sense. Subject to change, technical or otherwise.

Contents

1	Welcome to COPA-DATA help.....	4
2	FRAPORT.....	4
3	FRAPORT - data sheet.....	5
4	Driver history.....	6
5	Configuration	7
5.1	Creating a driver.....	8
5.2	Settings in the driver dialog.....	11
5.2.1	General.....	12
5.2.2	Settings.....	16
5.2.3	Connections.....	20
6	Creating variables.....	22
6.1	Creating variables in the Editor.....	22
6.2	Addressing	26
6.2.1	Flight display for target tracks.....	28
6.2.2	Process variables	28
6.2.3	Status	30
6.3	Driver objects and datatypes.....	34
6.3.1	Driver objects	34
6.3.2	Mapping of the data types	35
6.4	Creating variables by importing.....	36
6.4.1	XML import	36
6.4.2	DBF Import/Export	37
6.5	Communication details (Driver variables).....	43
7	Driver-specific functions.....	49
8	Driver command function	50
9	Error analysis	54
9.1	Analysis tool	54
9.2	Check list	55

1 Welcome to COPA-DATA help

ZENON VIDEO-TUTORIALS

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

GENERAL HELP

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

PROJECT SUPPORT

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

LICENSES AND MODULES

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

2 FRAPORT

The FRAPORT driver makes it possible to display all messages and displays in the zenon interconnected baggage system (IBS) and to be able to control the system.

3 FRAPORT - data sheet

General:	
Driver file name	FRAPORT.exe
Driver name	FRAPORT Treiber
PLC types	FRAPORT interconnected baggage system
PLC manufacturer	Fraport AG

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

Requirements:	
Hardware PC	Standard network adapter
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	--

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

4 Driver history

Date	Driver version	Change
16.03.11	100	Created driver documentation
18.03.15	7.20.0.19127	API interface enhanced

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 Configuration

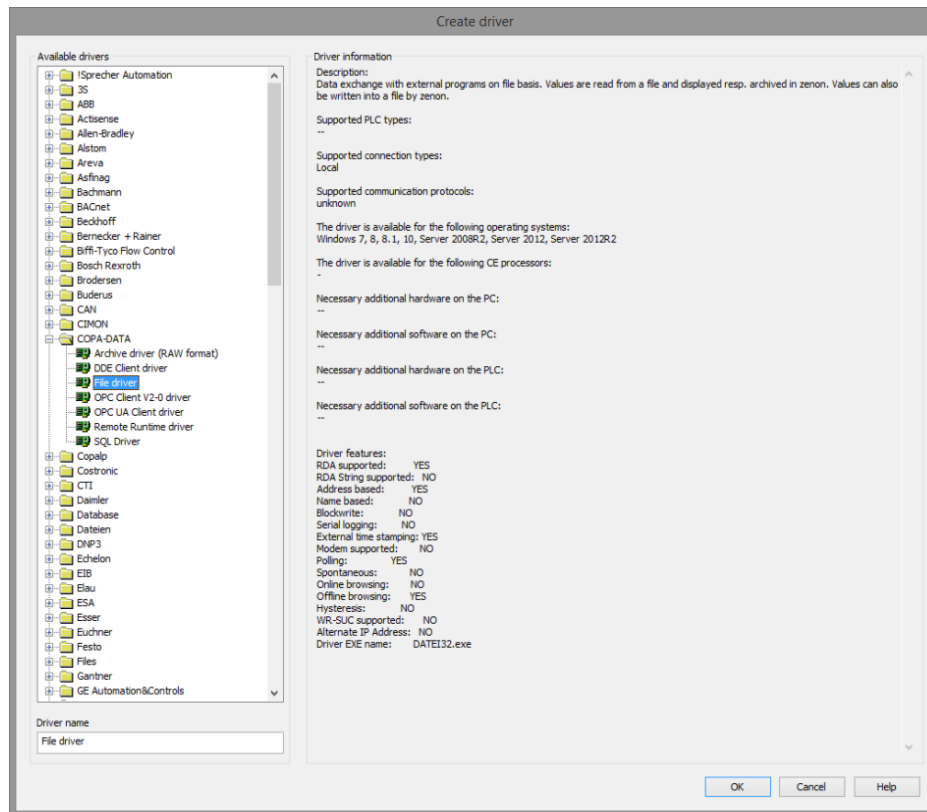
In this chapter you will learn how to use the driver in a project and which settings you can change.

Information

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

5.1 Creating a driver

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



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

CLOSE DIALOG

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



Information

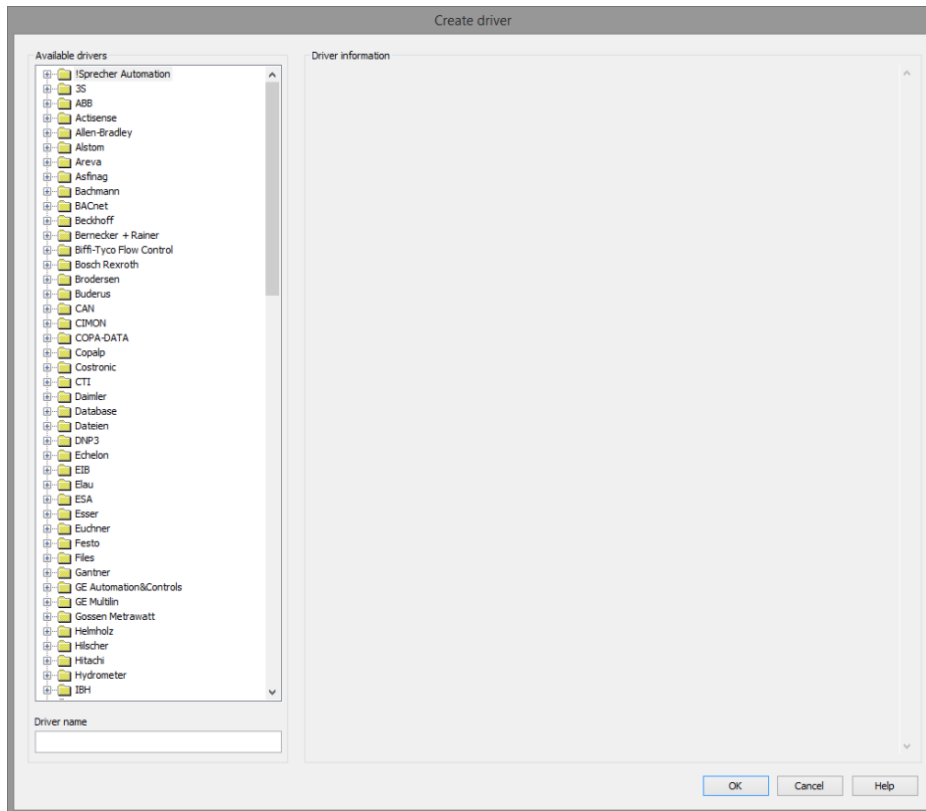
The content of this dialog is saved in the file called Treiber_[Language].xml. You can find this file in the following folder:
C:\ProgramData\COPA-DATA\zenon[version number].

CREATE NEW DRIVER

In order to create a new driver:

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

- The dialog offers a list of all available drivers.

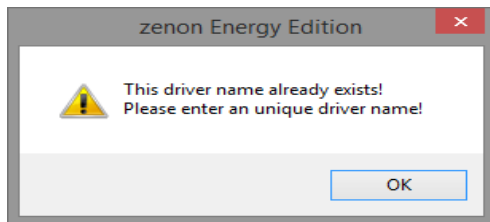


- Select the desired driver and name it in the **Driver name** input field.
This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.
The following is applicable for the **Driver name**:
 - ▶ The **Driver name** must be unique.
If a driver is used more than once in a project, a new name has to be given each time.
This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
 - ▶ The **Driver name** is part of the file name.
Therefore it may only contain characters which are supported by the operating system.
Invalid characters are replaced by an underscore (_).
 - ▶ **Attention:** This name cannot be changed later on.
- Confirm the dialog by clicking on the **OK** button.
The configuration dialog for the selected driver is opened.

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

DRIVER NAME DIALOG ALREADY EXISTS

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



ZENON PROJECT

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

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



Information

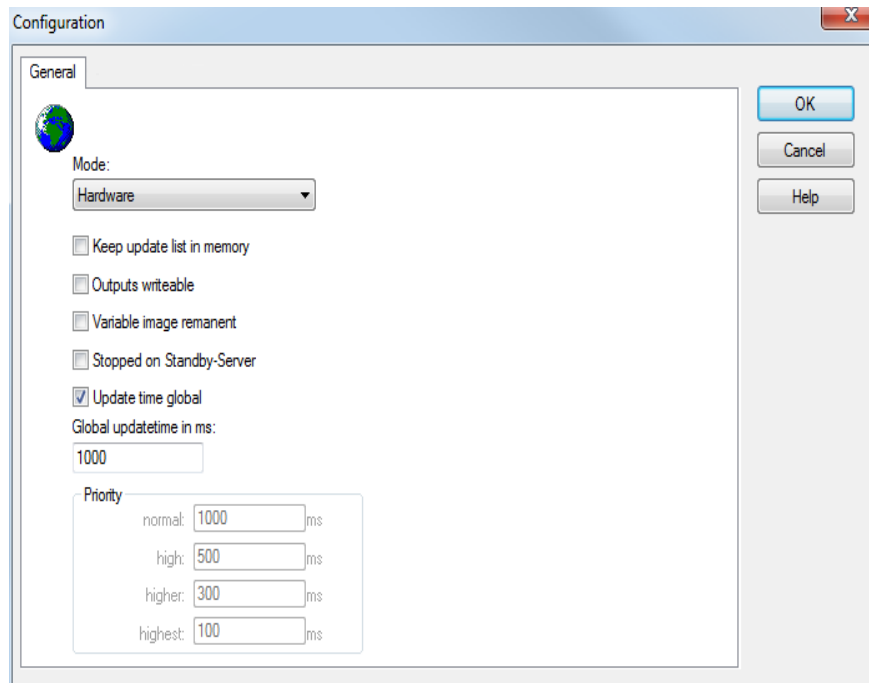
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

5.2 Settings in the driver dialog

You can change the following settings of the driver:

5.2.1 General

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



Option	Description
Mode	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> ▶ <i>Hardware:</i> A connection to the control is established. ▶ <i>Simulation - static:</i> No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▶ <i>Simulation - counting:</i> No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values

Option	Description
	<p>within a value range automatically.</p> <ul style="list-style-type: none"> ▶ <i>Simulation - programmed:</i> No communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).
Keep update list in the memory	<p>Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.</p>
Output can be written	<ul style="list-style-type: none"> ▶ <i>Active:</i> Outputs can be written. ▶ <i>Inactive:</i> Writing of outputs is prevented. <p>Note: Not available for every driver.</p>
Variable image remanent	<p>This option saves and restores the current value, time stamp and the states of a data point.</p> <p>Fundamental requirement: The variable must have a valid value and time stamp.</p> <p>The variable image is saved in hardware mode if one of these statuses is active:</p> <ul style="list-style-type: none"> ▶ User status <i>M1 (0)</i> to <i>M8 (7)</i> ▶ <i>REVISION(9)</i> ▶ <i>AUS(20)</i> ▶ <i>ERSATZWERT(27)</i> <p>The variable image is always saved if:</p> <ul style="list-style-type: none"> ▶ the variable is of the object type Driver variable ▶ the driver runs in simulation mode. (not

Option	Description
	<p>programmed simulation)</p> <p>The following states are not restored at the start of the Runtime:</p> <ul style="list-style-type: none"> ▶ <i>SELECT(8)</i> ▶ <i>WR-ACK(40)</i> ▶ <i>WR-SUC(41)</i> <p>The mode Simulation - programmed at the driver start is not a criterion in order to restore the remanent variable image.</p>
Stop on Standby Server	<p>Setting for redundancy at drivers which allow only one communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade.</p> <p>Attention: If this option is active, the gapless archiving is no longer guaranteed.</p> <ul style="list-style-type: none"> ▶ <i>Active:</i> Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status switched off (statusverarbeitung.chm::/24150.htm) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian. <p>Default: <i>inactive</i></p> <p>Note: Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.</p>
Global Update time	<p>Setting for the global update times in milliseconds:</p> <ul style="list-style-type: none"> ▶ <i>Active:</i> The set Global update time is used for all variables in the project. The priority set at the variables is not used. ▶ <i>Inactive:</i> The set priorities are used for the individual variables.

Option	Description
	Exceptions: Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the Spontaneous driver update time section.
Priority	<p>The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.</p> <p>The variables are allocated separately in the settings of the variable properties.</p> <p>The communication of the individual variables can be graded according to importance or required topicality using the priority classes. Thus the communication load is distributed better.</p> <p>Attention: Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.</p>

CLOSE DIALOG

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

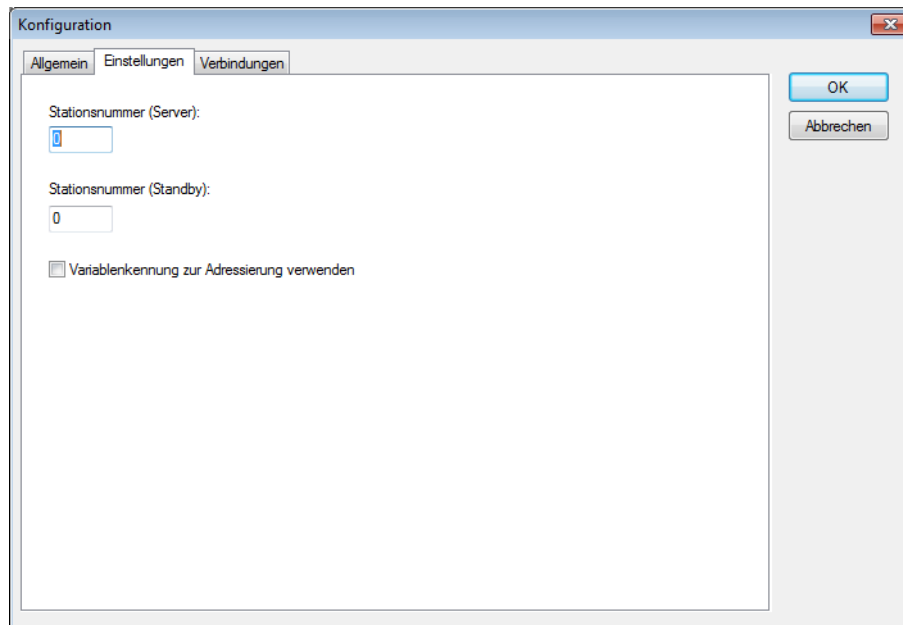
UPDATE TIME FOR SPONTANEOUS DRIVERS

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

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

5.2.2 Settings

General settings for communication are carried out in this tab. These settings apply to all connections (on page 20) that have been created in the driver.



Property	Description
Station number (server):	Station number of the zenon server.
Station number (standby):	Station number of the zenon standby server.
Use variable identification for addressing	<i>Active:</i> The identification instead of the name is used for symbolic addressing.



Information

The settings of these tabs can also be read and written to by means of the API (on page 16).

5.2.2.1 API for FRAPORT driver

EXAMPLE FOR READING THE SETTINGS

To read off the settings via API, use the following code:

Option Explicit

```
'Sample how to get all DynProperties of the FRAPORT driver...
Public Sub DrvPrintConfig()

    'Define objects...
    Dim obDriver As Driver
    Dim vProperties As Variant
    Dim i As Integer

    'allocate the Driver
    Set obDriver = thisProject.Drivers.Item("FRAPORT") 'FRAPORT = Drivername

    'enable driver configuration for editing
    Call obDriver.OpenConfig

    'get all DynProperties
    vProperties = obDriver.DynPropertiesEnum("")

    'iterate DynProperties and print to Immediate window
    Debug.Print "Properties:"
    For i = 0 To UBound(vProperties)
        Debug.Print vProperties(i)
    Next
    Debug.Print "-----"

    'print values of specified DynProperties
    Debug.Print obDriver.DynProperties("DrvConfig.Options.StationNumberServer")'0 - 65535
    Debug.Print obDriver.DynProperties("DrvConfig.Options.StationNumberStandBy")'0 - 65535
    Debug.Print obDriver.DynProperties("DrvConfig.Options.UseIdentification")'1 = active, 0 =
inactive

    'disable driver configuration
    Call obDriver.CloseConfig(False)
End Sub
```

CODE EXAMPLE TO SET THE SETTINGS

To configure the settings using API, use the following code:

Option Explicit

```
'Sample how to set specific DynProperties of the FRAPORT driver...
```

```
Public Sub DrvChangeConfig()

'Define objects...
Dim obDriver As Driver

'allocate the Driver
Set obDriver = thisProject.Drivers.Item("FRAPORT")'FRAPORT = Drivename

'enable driver configuration for editing
Call obDriver.OpenConfig

'assign values to specified DynProperties
obDriver.DynProperties("DrvConfig.Options.StationNumberServer") = 3 '0 - 65535
obDriver.DynProperties("DrvConfig.Options.StationNumberStandBy") = 2 '0 - 65535
obDriver.DynProperties("DrvConfig.Options.UseIdentification") = 1 '1 = active, 0 = inactive

'disable driver configuration
Call obDriver.CloseConfig(True)
End Sub
```

In the zenon VBA Editor (ALT + F11), add example code into "thisProject" and start with F5

5.2.2.2 New Topic (12)

5.2.2.3 Programming interfaces in the zenon Editor

OPEN VBA AND VSTA EDITOR

VBA EDITOR

VBA starts the same development environment for **Workspace** and **Project**.
To open the VBA Editor:

1. In the zenon Editor, navigate to the **Programming interface** node.
2. Expand the view of this node by clicking on [+].
The view of the node is expanded.
3. Right-click on **Macro list**
4. Select the entry **Open VBA Editor** in the context menu.

Alternatively: Press the short cut **Ctrl+F11**

VSTA EDITOR

VSTA provides separate development environments for **Workspace** and **project**. You can only use one of them at a time. At the start every other VSTA development environment which is open will be close.

To open the VSTA Editor for the workspace:

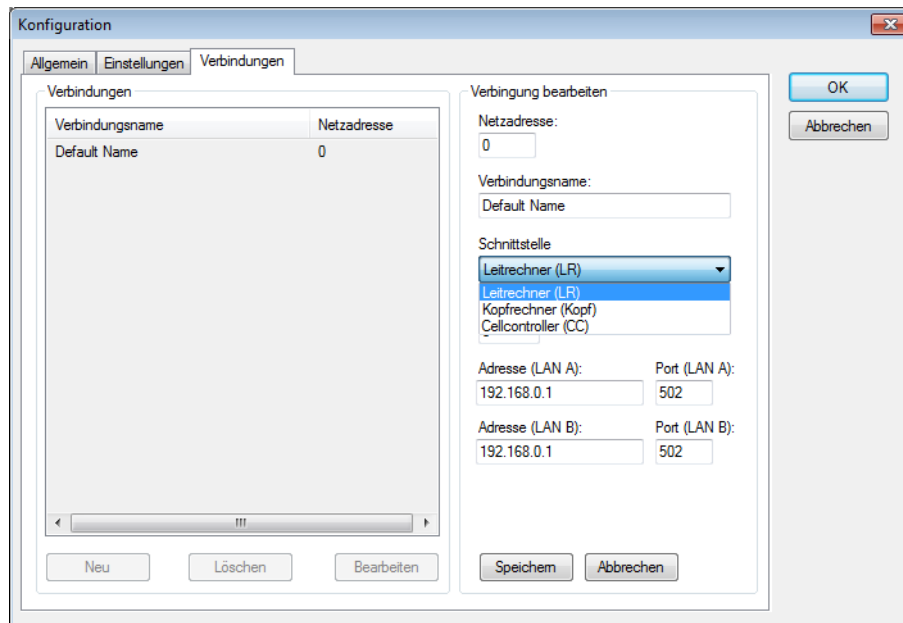
1. Press the **Alt+F10** keyboard shortcut.
A notice dialog is opened.
The code for the workspace and all loaded projects is displayed.

To open the VSTA Editor for the currently loaded project:

1. Navigate to the **Programming interfaces** node
2. Expand the view of this node by clicking on [+].
The view of the node is expanded.
3. Right-click on **VSTA**.
4. In the context menu, select **Open VSTA editor with ProjectAddin**.
A notice dialog is displayed.
5. Click on the **OK** button in order to continue with the VSTA project configuration.
The VSTA editor is opened for the project that is currently loaded.

5.2.3 Connections

On this tab you carry out the settings for the connection. You can create any number of connections in the driver. All connections use the global settings, which are defined in the Settings (on page 16) tab.



Property	Description
Connections	List of the connections created and their net addresses.
Connection name	Name of connection.
New	Adds new entry to the list. Settings are carried out in the field to the right of the list.
Delete	Deletes selected entry from the list.
Edit	Makes it possible to configure the selected entry. The area to the right of the list for editing the connection is activated. Attention! The driver dialog cannot be closed in editing mode. Only once the editing mode has been left using Save or Cancel can you close the driver dialog.
Edit connection	Connection settings for a new connection or the connection selected under Connection .
Net address	The net address identifies the connection. Therefore, every connection must have a unique net address. Variables are assigned to a connection via the net address. Attention! If you change the net address here, you must also

Property	Description
	change the net address of all other variables of the connection. Otherwise the allocation is no longer correct and variables cannot communicate in the Runtime!
Connection name	Connection name. Freely definable name.
Interface	Selection of the connection from the drop-down list: <ul style="list-style-type: none"> ▶ Control computer (LR) ▶ Cell controller (CC) ▶ Head computer (head)
Station number	Station number of the remote station.
Address (LAN A)	IP address or DNS name of the remote station in LAN A .
Port (LAN A)	TCP port of the remote station in LAN A .
Address (LAN B)	IP address or DNS name of the remote station in LAN B . If this entry is left empty, communication is carried out without redundancy.
Port (LAN B)	TCP port of the remote station in LAN B . If this entry is left empty, communication is carried out without redundancy.
Save	Saves changes.
Cancel	Discards changes.

Note: Driver messages can be read with the Diagnosis Viewer (main.chm::/12464.htm).

CREATE NEW CONNECTION

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

EDIT CONNECTION

1. select the connection in the connection list
2. Click on the **Edit** button

3. change the connection parameters
4. finish with **Save**

DELETE CONNECTION

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

6 Creating variables

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

6.1 Creating variables in the Editor

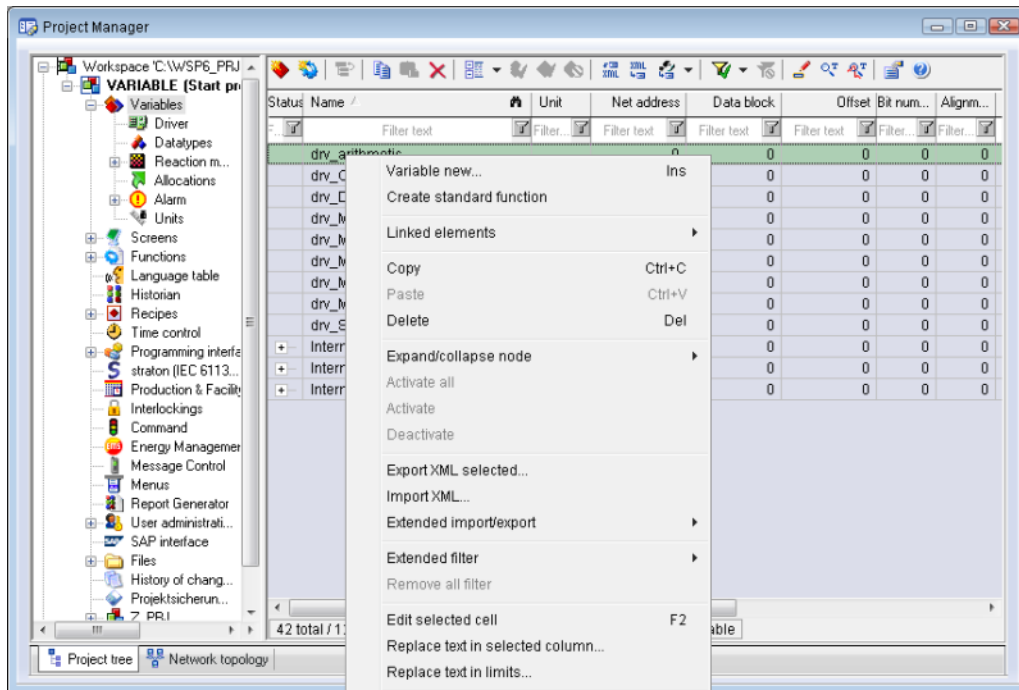
Variables can be created:

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

VARIABLE DIALOG

To create a new variable, regardless of which type:

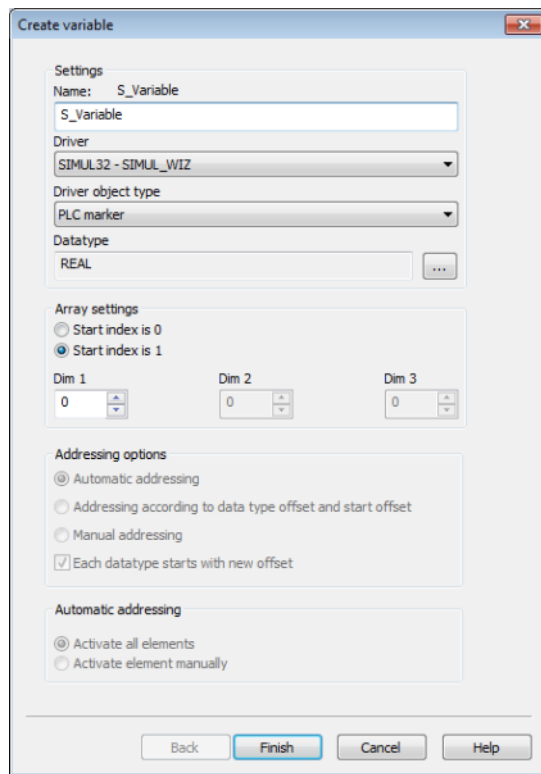
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

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

CREATE VARIABLE DIALOG



Property	Description
Name	<p>Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name.</p> <p>Maximum length: 128 characters</p> <p>Attention: the characters # and @ are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the Finish button remains inactive.</p> <p>Note: For some drivers, the addressing is possible over the property Symbolic address, as well.</p>
Drivers	<p>Select the desired driver from the drop-down list.</p> <p>Note: If no driver has been opened in the project, the driver for internal variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is automatically loaded.</p>
Driver Object Type (cti.chm::/28685.htm)	Select the appropriate driver object type from the drop-down list.
Data Type	Select the desired data type. Click on the ... button to open the

Property	Description
	selection dialog.
Array settings	Expanded settings for array variables. You can find details in the Arrays chapter.
Addressing options	Expanded settings for arrays and structure variables. You can find details in the respective section.
Automatic addressing	Expanded settings for arrays and structure variables. You can find details in the respective section.

SYMBOLIC ADDRESS

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

Maximum length: 1024 characters.

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

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

INHERITANCE FROM DATA TYPE

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

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

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

6.2 Addressing

Group/Property	Description
General	Property group for general settings.
Name	Freely definable name. Attention: For every zenon project the name must be unambiguous.
Identification	Freely definable identification. E.g. for Resources label, comments, ...
Addressing	
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 <i>Extended data block</i> , 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	Number of the bit within the configured offset. Possible entries: 0 to 65535.
String length	Only available for String variables. Maximum number of characters that the variable can take.
Driver connection/Data Type	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

Group/Property	Description
Driver connection/Driver Object Type	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Driver connection/Priority	not used for this driver The driver does not support cyclically-poling communication in priority classes.
Group/Property	Description
General	
Name	Name of the variable. Attention: For every zenon project the name must be unambiguous. Depending on the settings of Use identification as external name in the Settings (on page 16) dialog, the variable is addressed using its name or the identification set for it.
Identification	Name of the variable. Attention: For every zenon project the name must be unambiguous. Depending on the settings of Use identification as external name in the Settings (on page 16) dialog, the variable is addressed using its name or the identification set for it.
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 for this driver
Offset	not used for this driver
Alignment	not used for this driver
Bit number	Number of the bit within the configured offset. Possible entries: 0 ... 65535
String length	Only available for String variables: Maximum number of characters that the variable can take.
Driver connection/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	Data type of the variable. Is selected during the creation of the variable;

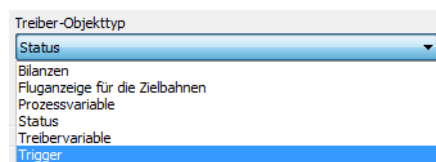
Group/Property	Description
connection/Datatype	the type can be changed here.
e	Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

ADDRESSING

Detailed information for addressing is available for the object types:

- ▶ Flight display for the target tracks (on page 28)
- ▶ Process variables (on page 28)
- ▶ Status (on page 30)

The following object types (on page 34) are generally available:



6.2.1 Flight display for target tracks

Flight displays are addressed using the target number in the system.

The complete address comprises:

- ▶ Any desired **prefix**
- ▶ An exclamation mark (!) as a separator
- ▶ The string **FLUG_**
- ▶ The target number

Example: WEO52!FLUG_1234

6.2.2 Process variables

The addressing of process variables is different on the control computer and on the cell controller.



Attention

If a variable is written that only displays part of a process variable, all other bits are set to 0.

CONTROL COMPUTER (LR)

The address comprises:

- ▶ Any desired **prefix**
- ▶ An exclamation mark (!) as a separator
- ▶ The **Area identification**
- ▶ An underscore (_) as a separator
- ▶ The actual **process variable names**
- ▶ A period (.) as a separator
- ▶ A **suffix**

Info: Prefix and suffix are optional

Example:

With prefix and suffix: **WEO52!PVXX1.QWord** or **WEO52!PVXX1._[0]**

In short: **W_PVXX1**

CELL CONTROLLER (CC)

The address comprises:

- ▶ Any desired **prefix**
- ▶ An exclamation mark (!) as a separator
- ▶ The actual **process variable names**
- ▶ A period (.) as a separator
- ▶ A **suffix**

Info: Prefix and suffix are optional

Example:

With prefix and suffix: **WEO52!PVXX1.QWord** or **WEO52!PVXX1._[0]**

In short: **PVXX1**

SUFFIXES

A part of the 64 bit process variable can be addressed using the suffix. The following suffixes can be used for the control computer and cell controller:

- ▶ **QWord** (64 bit)
- ▶ **_ [0], _ [1]** (32 bit, double word)
- ▶ **Word_ [0] ... Word_ [3]** (16 bit, word)
- ▶ **Byte_ [0] ... Byte_ [7]** (8 bit, byte)
- ▶ **Bit_ [0] ... Bit_ [63]** (bit)

6.2.3 Status

The address comprises:

- ▶ Any desired **prefix**
- ▶ An exclamation mark (!) as a separator
- ▶ The identifications for status information stated below

CONTROL COMPUTER (LR), HEAD COMPUTER (HEAD)

- ▶ LEITRECHNERKENNUNG
Addressing example: **WEO52!LEITRECHNERKENNUNG**

The control computer is set as follows:

- ▶ 'H', 72 ... HAUS
- ▶ 'O', 79 ... STO
- ▶ 'V', 86 ... V3
- ▶ 'W', 87 ... A-Plus
- ▶ 'K', 75 ... Head computer
- ▶ BETRIEBZUSTAND
Addressing example: **WEO52!BETRIEBZUSTAND**
The following control computer states are possible:
 - ▶ '1', 31 ... Ready for operation
 - ▶ '2', 32 ... Not ready for operation
- ▶ LOG_LINKZUSTAND[0] ... LOG_LINKZUSTAND[6]
addressing example: **WEO52!LOG_LINKZUSTAND[2]**

The logical link state displays whether data exchange is permitted with a computer or not. Only

the local connection state (local perspective) can be changed by command in the dialog. The character array contains a corresponding letter for each system. The letter of its own system is always set.

- ▶ 'H', 72 ... HAUS
- ▶ 'O', 79 ... STO
- ▶ 'V', 86 ... V3
- ▶ 'W', 87 ... A-Plus
- ▶ 'K', 75 ... Head computer
- ▶ 'I', 73 ... Info-Plus computer
- ▶ 'Z', 90 ... Visualization server
- ▶ '_', 95 ... Connection inactive
- ▶ PHYS_LINKZUSTAND[0] ... PHYS_LINKZUSTAND[6]
addressing example: **WEO52!PHYS_LINKZUSTAND[6]**
The physical connection status is kept internally in the PROCOM and cannot be influenced.
Provides information on whether data exchange with a computer is technically possible.
 - ▶ 'H', 72 ... HAUS
 - ▶ 'O', 79 ... STO
 - ▶ 'V', 86 ... V3
 - ▶ 'W', 87 ... A-Plus
 - ▶ 'K', 75 ... Head computer
 - ▶ 'I', 73 ... Info-Plus computer
 - ▶ 'Z', 90 ... Visualization server
 - ▶ '_', 95 ... Connection inactive
- ▶ STATUS_NODE_A
addressing example: **WEO52!STATUS_NODE_A**
 - ▶ 'M', 77 ... Node A in the master status
 - ▶ 'S', 83 ... Node A in standby status
- ▶ STATUS_NODE_B
addressing example: **WEO52!STATUS_NODE_B**
 - ▶ 'M', 77 ... Node A in the master status
 - ▶ 'S', 83 ... Node A in standby status
- ▶ GSV_MODE_NODE_A
addressing example: **WEO52!GSV_MODE_NODE_A**
 - ▶ 'N', 78 ... Node A in restart status

- ▶ GSV_MODE_NODE_B
addressing example: **WEO52!GSV_MODE_NODE_B**
 - ▶ 'N', 78 ... Node A in restart status
- ▶ REPL_NODE_A
addressing example: **WEO52!REPL_NODE_A**
 - ▶ 'H', 72 ... Replication active on node A (hot)
 - ▶ 'C', 67 ... Replication inactive on node A (cold)
- ▶ REPL_NODE_B
addressing example: Addressing example: **WEO52!REPL_NODE_B**
 - ▶ 'H', 72 ... Replication active on node A (hot)
 - ▶ 'C', 67 ... Replication inactive on node A (cold)
- ▶ ACTUAL_TIME
addressing example: Addressing example: **WEO52!AKTUELLE_ZEIT**
Current time on the system
- ▶ CONNECTION_STATUS
Addressing example: Addressing example: **WEO52!VERBINDUNGS_STATUS**
This variable provides the drive connection status to the control computer or head computer
 - ▶ 0 ... No action
 - ▶ 1 ... General query after new connection active
 - ▶ 2 ... General query initiated by user active
 - ▶ 3 ... Balance query active
- ▶ VERBINDUNGS_ZUSTAND:
Addressing example: Addressing example: **WEO52!VERBINDUNGS_ZUSTAND**
This variable provides the connection status of the driver to the control computer or head computer
 - ▶ 3 ... Unknown
 - ▶ 2 ... Both faulty
 - ▶ 1 ... One faulty
 - ▶ 0 ... Both connected

CELL CONTROLLER (CC)

- ▶ BETRIEBSZUSTAND
Addressing example: Addressing example: **WEO52!VERBINDUNGS_ZUSTAND**
The following control computer states are possible:
 - ▶ '1', 31 ... Ready for operation

- ▶ '2', 32 ... Not ready for operation
- ▶ '3', 33 ... Initialization active
- ▶ BETRIEBSZUSTAND_WEXPERTE[0] ... BETRIEBSZUSTAND_WEXPERTE[99]
addressing example: Addressing example: **WEO52!BETRIEBSZUSTAND_WEXPERTE[0]**
 - ▶ '1', 31 ... Ready for operation
 - ▶ '2', 32 ... Not ready for operation
- ▶ LOG_LINKZUSTAND
Addressing example: Addressing example: **WEO52!LOG_LINKZUSTAND**
The logical link state displays whether data exchange is permitted with a computer or not. Only the local connection state (local perspective) can be changed by command in the dialog. The character array contains a corresponding letter for each system. The letter of its own system is always set.
 - ▶ 'L', 76 ... Connection active
 - ▶ '?', 63... Connection state unknown
 - ▶ '_', 95 ... Connection inactive
- ▶ PHYS_LINKZUSTAND
Addressing example: Addressing example: **WEO52!PHYS_LINKZUSTAND**
The physical connection status is kept internally in the PROCOM and cannot be influenced. Provides information on whether data exchange with a computer is technically possible.
 - ▶ 'L', 76 ... Connection active
 - ▶ '?', 63... Connection state unknown
 - ▶ '_', 95 ... Connection inactive
- ▶ ACTUAL_TIME
addressing example: Addressing example: **WEO52!AKTUELLE_ZEIT**
Current time on the system
- ▶ CONNECTION_STATUS
Addressing example: Addressing example: **WEO52!VERBINDUNGS_STATUS**
This variable provides the drive connection status to the control computer or head computer
 - ▶ 0 ... No action
 - ▶ 1 ... General query after new connection active
 - ▶ 2 ... General query initiated by user active
- ▶ VERBINDUNGS_ZUSTAND:
Addressing example: Addressing example: **WEO52!VERBINDUNGS_ZUSTAND**
This variable provides the driver connection status to the cell controller.
 - ▶ 3 ... Unknown
 - ▶ 2 ... Both faulty

- ▶ 1 ... One faulty
- ▶ 0 ... Both connected

6.3 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.

6.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Description
Process variable	8	X	X	<i>BOOL, SINT, USINT, INT, UINT, DINT, UDINT, LINT, LWORD</i>	Process variables.
Status	64	X	--	<i>USINT, STRING, DATE_AND_TIME</i>	Status.
Flight display for the target tracks	65	X	--	<i>STRING</i>	Flight display for the target tracks.
Balancing	66	X	--	<i>STRING</i>	Balancing the head computer
Trigger	67	--	X	<i>UINT</i>	Trigger for manual execution of a general query or head query (balance query).
<i>Communication details</i>	35	X	X	<i>BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING</i>	Variables for the static analysis of the communication; is transferred between driver and Runtime

Driver Object Type	Channel type	Read	Write	Supported data types	Description
					<p>(not to the PLC).</p> <p>Note: The addressing and the behavior is the same for most zenon drivers.</p> <p>You can find detailed information on this in the Communication details (Driver variables) (on page 43) chapter.</p>

Key:

X: supported

--: not supported

6.3.2 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
ULINT	BOOL	8
ULINT	USINT	9
ULINT	SINT	10
ULINT	UINT	2
ULINT	INT	1
ULINT	UDINT	4
ULINT	DINT	3
ULINT	ULINT	27
ULINT	LINT	26

Control	zenon	Data type
-	REAL	5
-	LREAL	6
-	STRING	12
-	WSTRING	21
-	DATE	18
-	TIME	17
-	DATE_AND_TIME	20
-	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.

6.4 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.

6.4.1 XML import

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

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

Note: The actions and their durations are shown in a progress bar during import. The import of variables is described in the following documentation. Data types are imported along the same lines.

REQUIREMENTS

The following conditions are applicable during import:

- ▶ **Backward compatibility**

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

- ▶ **Consistency**

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

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

- ▶ **Structure data types**

Structure data types must have the same number of structure elements.

Example: A structure data type in the project has 3 structure elements. A data type with the same name in the XML file has 4 structure elements. Then none of the variables based on this data type in the file are imported into the project.

Hint

You can find further information on XML import in the **Import - Export** manual, in the **XML import (main.chm::/13046.htm)** chapter.

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

Identification	Type	Field size	Comment
			of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager
LES_SCHR	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTYP and DATENTYP
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MENTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values

Identification	Type	Field size	Comment
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

Identification	Type	Field size	Comment
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.

6.5 Communication details (Driver variables)

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

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

The definitions of the variables implemented in the driver kit are available in the import file **DRVVAR.DBF** and can be imported from there.

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

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



Information

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

For example:

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

INFORMATION

Name from import	Type	Offset	Description
MainVersion	UINT	0	Main version number of the driver.

Name from import	Type	Offset	Description
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 <i>driver stop</i> , the variable has the value <i>TRUE</i> and an OFF bit. After the driver has

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

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	<i>BOOL</i>	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	<i>BOOL</i>	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	<i>BOOL</i>	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem.

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

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

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	<i>UDINT</i>	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	<i>UDINT</i>	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	<i>UDINT</i>	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	<i>UDINT</i>	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	<i>UDINT</i>	33	Number of writing errors
ReadSucceedCount	<i>UDINT</i>	35	Number of successful reading attempts
MaxCycleTime	<i>UDINT</i>	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	<i>UDINT</i>	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	<i>UDINT</i>	26	Number of writing attempts
ReadErrorCount	<i>UDINT</i>	34	Number of reading errors
MaxUpdateTimeNormal	<i>UDINT</i>	56	Time since the last update of the priority group Normal in milliseconds (ms).
MaxUpdateTimeHigher	<i>UDINT</i>	57	Time since the last update of the priority group Higher in milliseconds (ms).

Name from import	Type	Offset	Description
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

7 Driver-specific functions

The driver supports the following functions:

MANUAL INITIATION OF A GENERAL QUERY

- ▶ Creation of a **trigger** driver object variable and the name or identification consisting of an optional prefix and the character sequence **GA**.
for example: **MyPrefix!GA** or **GA**
- ▶ Assignment to a connection is carried out using the network address.
- ▶ A general query is triggered by writing a value to a variable. In doing so, the value written is used as a **GA** type.
 - ▶ GA type: 1 ... All PVs
 - ▶ GA type: 2 ... Only special PVs



Information

The status of the general query can be queried using the VERBINDUNGS_STATUS status variable.

INITIATION OF A BALANCE QUERY FROM THE HEAD COMPUTER

- ▶ Creation of a **balances** driver object string variable.
- ▶ Creation of a **trigger** driver object variable and the name or identification consisting of an optional prefix and the character sequence **BILANZEN** for example: **MyPrefix!BILANZEN** or **BILANZEN**
- ▶ Assignment of variables to a connection is carried out using the network address.
- ▶ A head query is triggered by writing a value to a variable. In doing so, the value written is used as a **statistic** type.
 - ▶ Statistic type: 1 ... Statistic for live operation
 - ▶ Statistic type: 2 ... Operational transmission statistics



Information

The status of the head query can be queried using the VERBINDUNGS_STATUS status variable.

The balance information is stored in a **balance** driver object string variable.
Assignment to a connection is carried out using the network address.

8 Driver command function

The zenon **Driver commands** function is to influence drivers using zenon. You can do the following with a driver command:

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

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



Attention

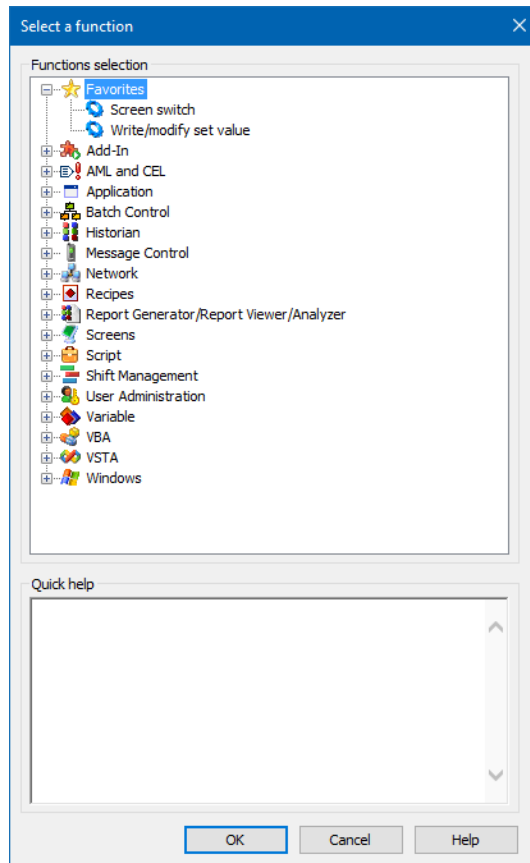
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

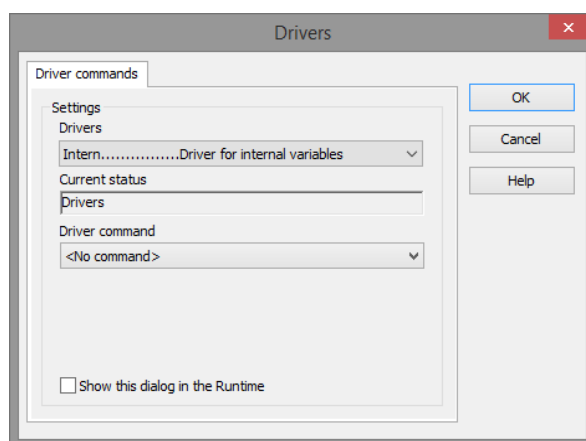
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



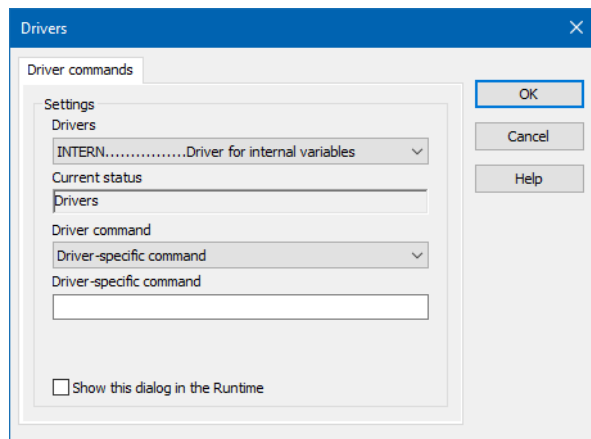
2. Navigate to the node **Variable**.
3. Select the **Driver commands** entry.

The dialog for configuration is opened



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

DRIVER COMMAND DIALOG



Option	Description
Driver	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
Current condition	Fixed entry that is set by the system. Has no function in the current version.
Driver command	Selection of the desired driver command from a drop-down list. For details on the configurable driver commands, see the available driver commands section.
Driver-specific command	Entry of a command specific to the selected driver. Note: Only available if, for the driver command option, the <i>driver-specific command</i> has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime: <ul style="list-style-type: none"> ▶ <i>Active</i>: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. ▶ <i>Inactive</i>: The Editor configuration is applied in the Runtime when executing the function. Default: <i>inactive</i>

CLOSE DIALOG

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

AVAILABLE DRIVER COMMANDS

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

Driver command	Description
<No command>	No command is sent. A command that already exists can thus be removed from a configured function.
<i>Start driver (online mode)</i>	Driver is reinitialized and started. Note: If the driver has already been started, it must be stopped. Only then can the driver be re-initialized and started.
<i>Stop driver (offline mode)</i>	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 <i>switched off</i> (OFF; Bit 20).
<i>Driver in simulation mode</i>	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
<i>Driver in hardware mode</i>	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
<i>Driver-specific command</i>	Entry of a driver-specific command. Opens input field in order to enter a command.
<i>Activate driver write set value</i>	Write set value to a driver is possible.
<i>Deactivate driver write set value</i>	Write set value to a driver is prohibited.
<i>Establish connection with modem</i>	Establish connection (for modem drivers)

Driver command	Description
	Opens the input fields for the hardware address and for the telephone number.
<i>Disconnect from modem</i>	Terminate connection (for modem drivers)
<i>Driver in counting simulation mode</i>	Driver is set into counting simulation mode. All values are initialized with 0 and incremented in the set update time by 1 each time up to the maximum value and then start at 0 again.
<i>Driver in static simulation mode</i>	No communication to the controller is established. All values are initialized with 0.
<i>Driver in programmed simulation mode</i>	The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime.

DRIVER COMMAND FUNCTION IN THE NETWORK

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

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

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

9 Error analysis

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

9.1 Analysis tool

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

zenon driver log all errors in the LOG files. LOG files are text files with a special structure. The default folder for the LOG files is subfolder **LOG** in the folder **ProgramData**. For example:

%ProgramData%\COPA-DATA\LOG.

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

In the Diagnosis Viewer you can also:

- ▶ Follow newly-created entries in real time
- ▶ customize the logging settings
- ▶ change the folder in which the LOG files are saved

Note:

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



Attention

In Windows CE errors are not logged per default due to performance reasons.

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

9.2 Check list

Questions and hints for fault isolation:

GENERAL TROUBLESHOOTING

- ▶ Analysis with the Diagnosis Viewer (on page 54):
-> Which messages are displayed?
- ▶ Are the participants available in the **TCP/IP** network?
- ▶ Can the PLC be reached via the *Ping* command?

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

Do you receive an answer with a time or a timeout?

- ▶ Can the PLC be reached at the respective port via *TELNET*?

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

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

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

- ▶ Did you use the right object type for the variable?

Example: Driver variables based on driver object type **Communication details** are purely statistics variables. They do not communicate with the PLC.

You can find detailed information on this in the Communication details (Driver variables) (on page 43) chapter.

SOME VARIABLES REPORT INVALID.

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

VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY

Driver is not working. Check the:

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

VARIABLES ARE DISPLAYED WITH A BLUE DOT

The communication in the network is faulty:

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

VALUES ARE DISPLAYED INCORRECTLY

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

DRIVER FAILS OCCASIONALLY

Analysis with the Diagnosis Viewer (on page 54):

-> Which messages are displayed?