zenon
by COPA-DATA

zenon driver manual

Predictive Analytics

v.8.10

Contents

1 Welcome to COPA-DATA help ... 4

2 Predictive Analytics Driver ... 4

3 PA_Drv - Datenblatt ... 5

4 Driver history ... 6

5 Requirements ... 6

5.1 PC ... 7

5.2 PLC ... 7

6 Configuration ... 7

6.1 Creating a driver ... 8

6.2 Settings in the driver dialog ... 11

6.2.1 General ... 12

6.2.2 Options ... 16

7 Creating variables ... 17

7.1 Creating variables in the Editor ... 18

7.2 Addressing ... 21

7.3 Driver objects and datatypes ... 23

7.3.1 Driver objects ... 23

7.3.2 Mapping of the data types ... 26

7.4 Creating variables by importing ... 26

7.4.1 XML import ... 27

7.4.2 DBF Import/Export ... 28

7.4.3 Online import ... 33

7.5 Communication details (Driver variables) ... 45

8 Driver-specific functions ... 50

9 Driver command function ... 52

10 Error analysis ... 56

10.1 Analysis tool ... 56

10.2 Check list ... 57

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 Predictive Analytics Driver

The **Predictive Analytics Driver** gets prediction data from the Analyzer server.
Predictions can be:

- spontaneous sent via schedule from the Analyzer Server and received in the driver
  or
- Polling from the driver with incoming values queried on the Analyzer server

In doing so, a package can contain:

- An individual value prediction: Prediction value is written to a simple variable or
- A value list: Prediction values are written to an array of simple variables

3 PA_Drv - Datenblatt

General:	
Driver file name	PA_Drv.exe
Driver name	Predictive Analytics Engine driver
PLC types	Predictive Analytics Engine
PLC manufacturer	COPA-DATA; zenon system driver

Driver supports:	
Protocol	MQTT
Addressing: Address-based	Name based
Addressing: Name-based	Symbolic Adress
Spontaneous communication	X
Polling communication	X
Online browsing	X
Offline browsing	--
Real-time capable	X
Blockwrite	--
Modem capable	--
RDA numerical	--
RDA String	--
Hysteresis	X
extended API	X

Driver supports:	
Supports status bit **WR-SUC**	X
alternative IP address	--

Requirements:	
Hardware PC	Ethernet Port
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	Build number	Change
13.12.18	53630	Created driver documentation

5 Requirements

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

5.1 PC

Certificate Bundles must be present for the connection to the **Service Hub**.

5.2 PLC

At least 1 prediction model must be available in a database on the Analyzer server. If spontaneous communication is required, there must be at least 1 schedule for at least 1 prediction model. Furthermore, the Analyzer server must be connected to a service hub.

Prediction models can be configured in the **Prediction Model Manager** of **zenon Analyzer**. The schedules can be configured in the zenon Analyzer Management Studio, whereby this must be started on the same computer as the Analyzer server.

6 Configuration

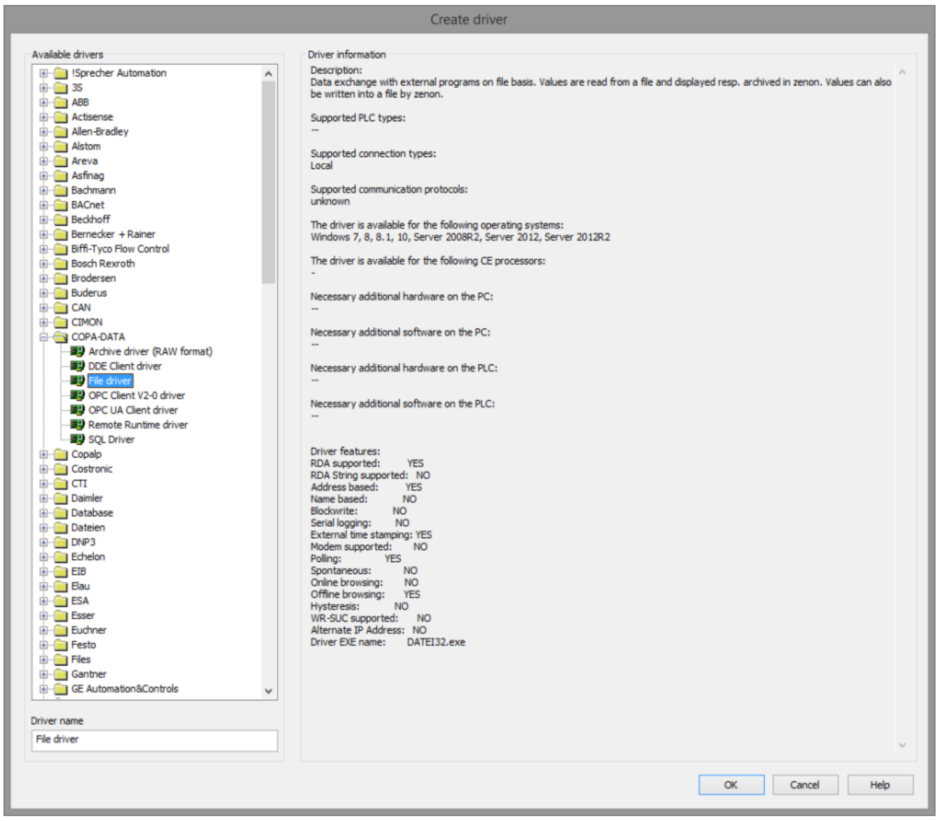
In this chapter you will learn how to use the driver in a project and which settings you can change.

Information

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

6.1 Creating a driver

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



Driver name
File driver

Driver information

```
Description:
Data exchange with external programs on file basis. Values are read from a file and displayed resp. archived in zenon. Values can also
be written into a file by zenon.

Supported PLC types:
--

Supported connection types:
Local

Supported communication protocols:
unknown

The driver is available for the following operating systems:
Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2

The driver is available for the following CE processors:
-

Necessary additional hardware on the PC:
--

Necessary additional software on the PC:
--

Necessary additional hardware on the PLC:
--

Necessary additional software on the PLC:
--


Driver features:
RDA supported:          YES
RDA String supported:   NO
Address based:          YES
Name based:             NO
Blockwrite:             NO
Serial logging:         NO
External time stamping: YES
Modem supported:        NO
Polling:                YES
Spontaneous:            NO
Online browsing:        NO
Offline browsing:       YES
Hysteresis:             NO
WR-SUC supported:       NO
Alternate IP Address:   NO
Driver EXE name:        DATEI32.exe
```

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

CLOSE DIALOG

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

Information

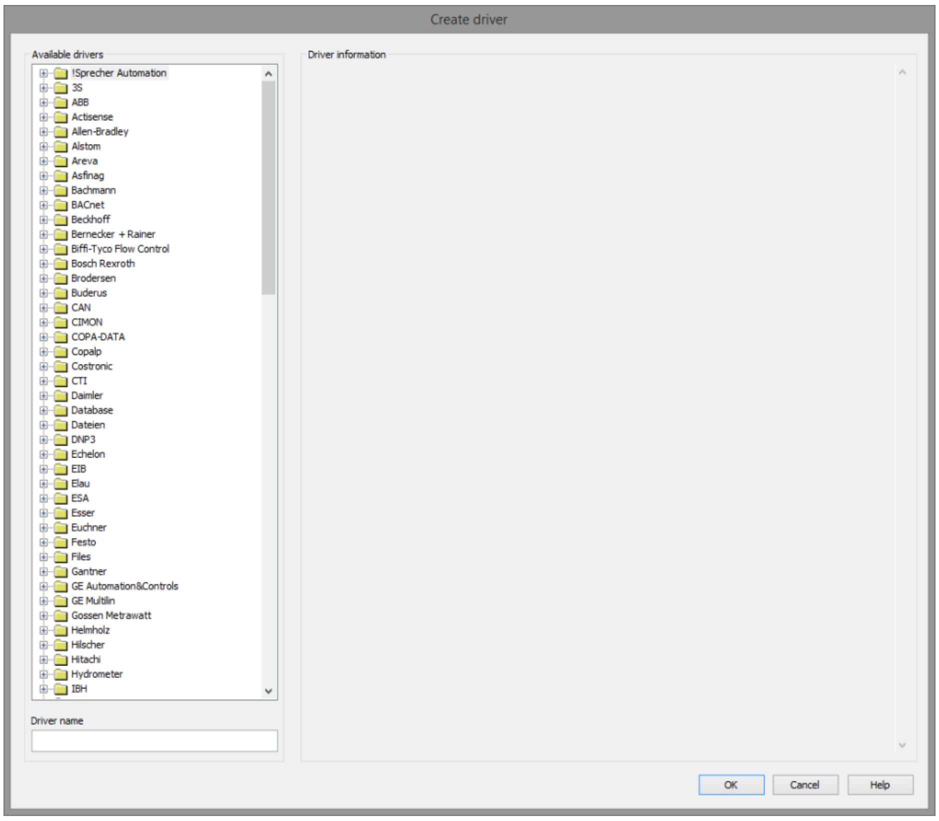
The content of this dialog is saved in the file called Treiber_[Language].xml. You can find this file in the following folder:
C:\ProgramData\COPA-DATA\zenon[version number].

CREATE NEW DRIVER

In order to create a new driver:

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

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



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

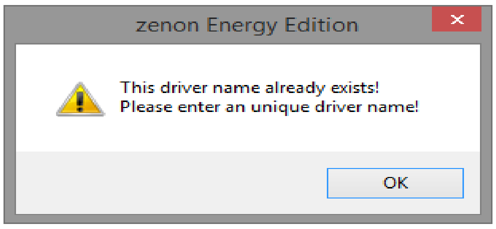
   - The **Driver name** must be unique.
     If a driver is used more than once in a project, a new name has to be given each time.
     This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
   - The **Driver name** is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

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

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

DRIVER NAME DIALOG ALREADY EXISTS

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



This driver name already exists!
Please enter an unique driver name!

ZENON PROJECT

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

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

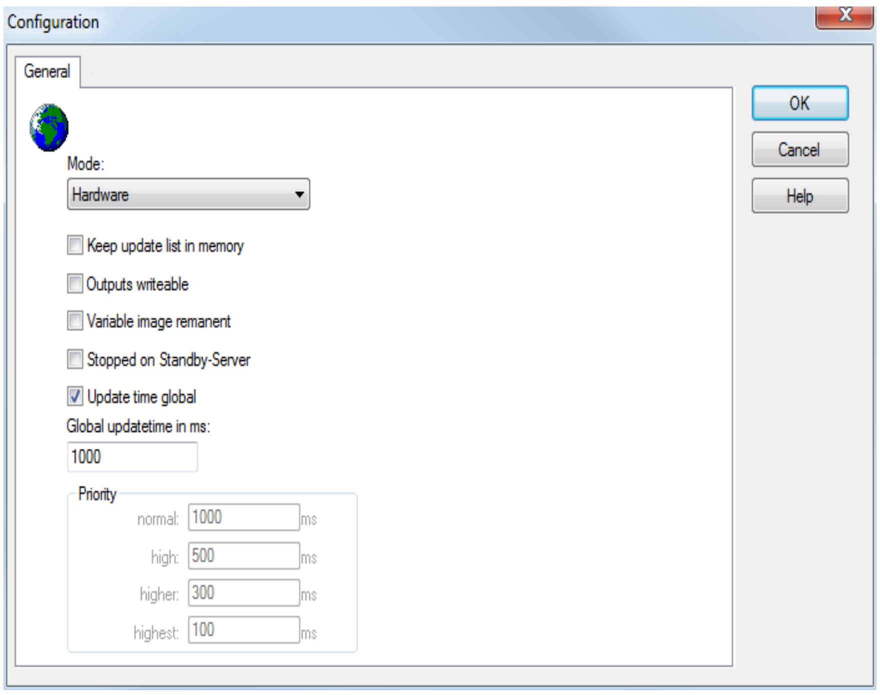
> **Information**
>
> Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

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


Configuration

General

Mode: Hardware

☐ Keep update list in memory

☐ Outputs writeable

☐ Variable image remanent

☐ Stopped on Standby-Server

☑ Update time global

Global updatetime in ms: 1000

Priority

normal: 1000 ms

high: 500 ms

higher: 300 ms

highest: 100 ms

Option	Description
Mode	Allows to switch between hardware mode and simulation mode ▶ *Hardware*: A connection to the control is established. ▶ Simulation - static: No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▶ *Simulation - counting*: No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values

Option	Description
	within a value range automatically. • *Simulation - programmed*: No communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).
Keep update list in the memory	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	• *Active*: Outputs can be written. • *Inactive*: Writing of outputs is prevented. **Note**: Not available for every driver.
Variable image remanent	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in hardware mode if one of these statuses is active: • User status *M1* (*0*) to *M8* (*7*) • *REVISION(9)* • *AUS(20)* • *ERSATZWERT(27)* The variable image is always saved if: • the variable is of the object type **Driver variable** • the driver runs in simulation mode. (not

Option	Description
	programmed simulation) The following states are not restored at the start of the Runtime: - *SELECT(8)* - *WR-ACK(40)* - *WR-SUC(41)* The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.
Stop on Standby Server	Setting for redundancy at drivers which allow only one communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. **Attention:** If this option is active, the gapless archiving is no longer guaranteed. - *Active*: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **switched off** (**statusverarbeitung.chm::/24150.htm**) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian. Default: *inactive* **Note:** Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.
Global Update time	Setting for the global update times in milliseconds: - *Active*: The set **Global update time** is used for all variables in the project. The priority set at the variables is not used. - *Inactive*: The set priorities are used for the individual variables.

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	The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time. The variables are allocated separately in the settings of the variable properties. The communication of the individual variables can be graded according to importance or required topicality using the priority classes. Thus the communication load is distributed better. **Attention:** Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.

CLOSE DIALOG

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

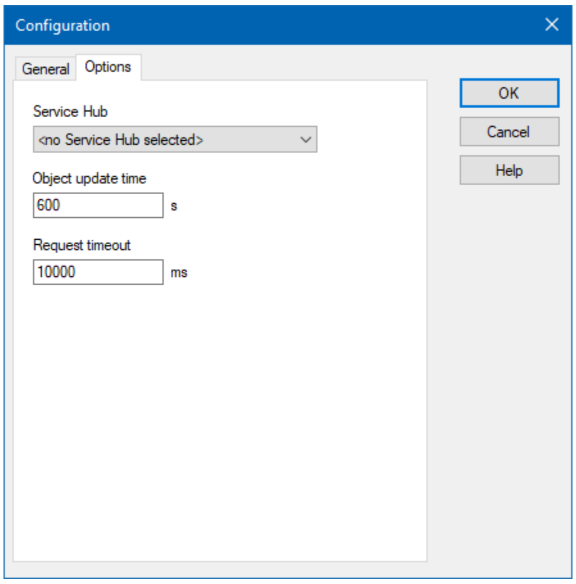
UPDATE TIME FOR SPONTANEOUS DRIVERS

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

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

6.2.2 Options

Configuration of the connection settings to the **Service Hub**:



Service Hub: <no Service Hub selected>

Object update time: 600 s

Request timeout: 10000 ms

Option	Description
Service Hub	Selection of the connection to the **Service Hub** from a drop-down list. All connections that have been created with the **Service Node Configuration Tool** and for which there is a **Certificate Bundle** are offered: ▶ *<no Service Hub selected>*: Empty string, no selection. ▶ *<configured connections>*: List of all available connections. Default:*<no Service Hub selected>* For access via API, the connection information is entered directly.
Object Update Time	Entry of the object refresh time in seconds. **Behavior:** ▶ If the databases of an Analyzer Server have been read successfully, there is a wait for this time until the existence of the database is checked again with a further call. ▶ If the prediction model and schedules of a database have been read successfully, there is a

Option	Description
	wait for this time until the existence of the objects and their properties have been checked with a further call. **Input time period:** ▸ Minimum: 60 seconds (1 minute) ▸ Maximum: 86400 seconds (1 day) Default:*600 s*
Request Timeout	Entry of the query timeout for polling and object updates in milliseconds: ▸ Polling is aborted after this time if there is no response. ▸ Object refreshes are aborted after this time if no response is received and then started again afterwards once this time has elapsed again. Possible values: ▸ Minimum: *1000* (1 second) ▸ Maximum: *3600000* (1 hour) Default:*10000* (10 seconds).

CLOSE DIALOG

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

7 Creating variables

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

7.1 Creating variables in the Editor

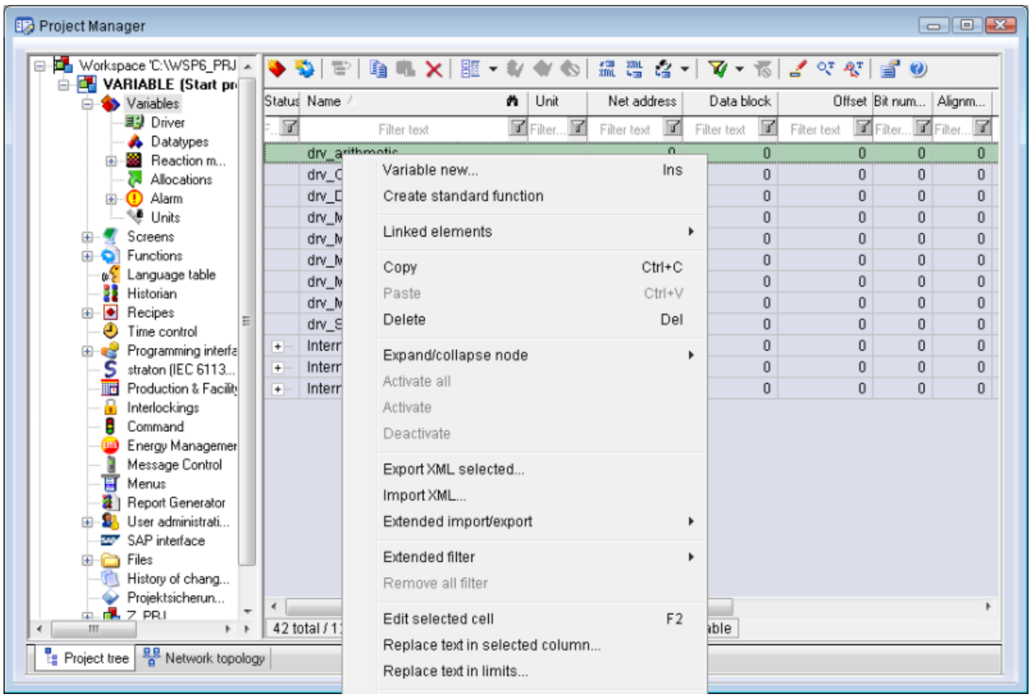
Variables can be created:

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

VARIABLE DIALOG

To create a new variable, regardless of which type:

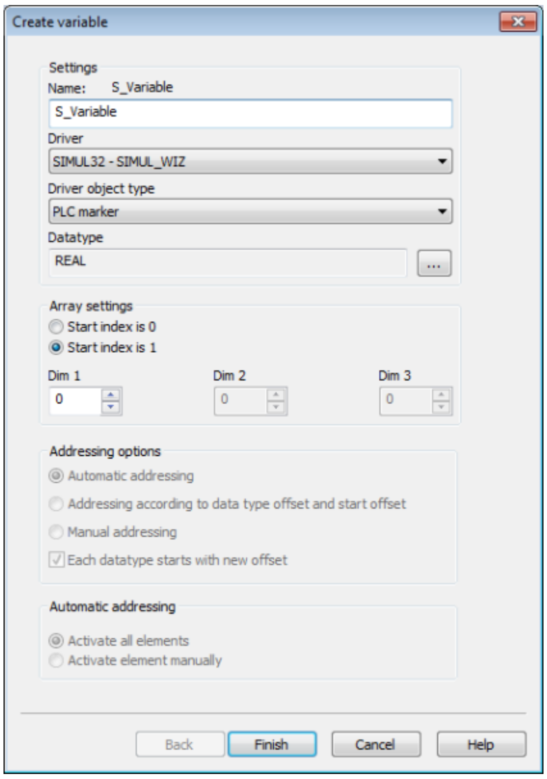
1. Select the **New variable** command in the **Variables** node in the context menu



   The dialog for configuring variables is opened

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

CREATE VARIABLE DIALOG


Create variable

Settings
Name: S_Variable
S_Variable
Driver
SIMUL32 - SIMUL_WIZ
Driver object type
PLC marker
Datatype
REAL

Array settings
Start index is 0
Start index is 1
Dim 1: 0
Dim 2: 0
Dim 3: 0

Addressing options
Automatic addressing
Addressing according to data type offset and start offset
Manual addressing
Each datatype starts with new offset

Automatic addressing
Activate all elements
Activate element manually

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

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
- KabaDPServer
- OPCUA32
- Phoenix32
- POZYTON
- RemoteRT
- S7TIA
- SEL
- SnmpNg32
- PA_Drv

INHERITANCE FROM DATA TYPE

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

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

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

7.2 Addressing

Group/Property	Description
General	
Name	Freely definable name. **Attention:** For every zenon project the name must be unambiguous.
Identification	Freely assignable identification, e.g. for resources label, comment ...
Addressing	Via symbolic address. Syntax: *[Identifier for communication type]::[Identifier for model type]::[Analyzer Server ID]::[Database name]::[Model ID]::[Identification]::[Name of the input parameter]* Description: - **Identifier for communication type**: - *S*: Spontaneous communication. Only initial values possible. Initial values must be *LREAL* type. - *T*: Polling communication. Triggered. Initial values must be *LREAL* type. Input values must be *LREAL* or *DATE_AND_TIME* depending on the prediction model used. Trigger variables must be *BOOL* type. - **Identifier for model type**: - *T*: Time based. Time stamp for which the prediction is created. Is written as an external time stamp of the variable. - *V*: Value-based. - **Analyzer Server ID**: Unique GUID of the Analyzer server. - **Database name**: Name of the database in which the prediction model is saved. - **Model ID**: Unique ID of the prediction model in the database.

Group/Property	Description
	- **Identification**: Is used depending on communication type: - *Spontaneous*: Unique GUID of the schedule on the Analyzer server. - *Polling*: Unique GUID in order to bundle variables for predicted value, input value and controller into a logical structure. - **Name of the input parameter:** Is only present if the variable is an input value for *polling*. The following input values are needed for predicted value: - Simple variables: **Input** for individual value prediction. - Array of simple variables: **From** and **To** for value range prediction. The array length determines the number of values in the process. The input values for the prediction models are distributed evenly over the range between **From** and **To**. **Recommendation:** Use online import to create variables. Background: The **Symbolic address** primarily uses information from the Analyzer servers and has a complex structure.
Net address	not used for this driver
Data block	not used for this driver
Offset	not used for this driver
Alignment	not used for this driver
Bit number	not used for this driver
String length	not used for this driver
Driver connection/Driver Object Type	Available driver object types: - **Predicted Value**: Predicted value (spontaneous and polling). - **Prediction Input Value**: Input value for polling predictions. - **Prediction Trigger**: Control variable for polling forecasts. **Note:** You can find a precise description of the events in the **driver objects** (on page 23) chapter.
Driver connection/Data	The driver supports the *BOOL*, *LREAL* and *DATE_AND_TIME* data types. **Attention:** If you change the data type later, all other properties of the

Group/Property	Description
Type	variable must be checked or 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
Predicted Value	8	x	--	*LREAL*	Variables that receive the predicted values. ▸ Simple variable with polling communication: With **Input** input values, precisely 1 value is obtained. ▸ Simple variable with spontaneous communication: The schedule on the Analyzer server must be set with an input value for precisely 1 value. ▸ Array of simple variables with polling communication: With the input values **From** and **To**, a range prediction is obtained with the array length as the number of values. On the Analyzer server, the predicted values for the set number of

Driver Object Type	Channel type	Read	Write	Supported data types	Description
					values distributed evenly over the range are determined. ▸ Array of simple variables with spontaneous communication: The number of values in the schedule on the Analyzer server must correspond to the number of variables in the array. The range of values must also be given. On the Analyzer server, the predicted values for the set number of values distributed evenly over the range are determined. ▸ Value-based prediction model The internal time stamp is set to the time of the arrival of the prediction value. The external time stamp remains empty. ▸ Time-based prediction model The internal time stamp is set to the time of the arrival of the predicted value. The external time stamp is to set the time for the predicted. And the time stamp external status bit is set.
Prediction Input Value	64	x	x	*DATE_AND_TIME, LREAL*	Input variable for polling prediction. If the model is time

Driver Object Type	Channel type	Read	Write	Supported data types	Description
					based, the data type must be *DATE_AND_TIME*. If the model is value based, the data type must be *LREAL*. Reading is only supported for status changes here. The name of the parameter is specified in **Symbolic address**. Possible names are: - **Input**: Input value for individual value prediction. - **From**: Lower limit for input value for range prediction. - **To**: Upper limit for input value for range predictions.
Prediction Trigger	10	x	x	*BOOL*	Control variable for polling communication. If the variable is set from *0* to *1* by Runtime, polling is started. If the polling has ended, (regardless of whether due to loss of connection, error or arrival of value), the variable is set back to *0* by the driver.
Communication details	35	x	x	*BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING*	Variables for the static analysis of the communication; Is transferred between driver and the Runtime. (Not to the PLC!) **Note**: The addressing and the behavior is the same for most zenon drivers. You can find detailed

Driver Object Type	Channe l type	Read	Write	Supported data types	Description
					information on this in the Communication details (Driver variables) (on page 45) chapter.

Key:

X: supported

--: not supported

7.3.2 Mapping of the data types

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

PLC	zenon	Data type
Not available Pure control variable on the driver.	*BOOL*	*8*
Prediction value for time-based or value-based prediction model or Input value for value-based prediction model	*LREAL*	*6*
Input value for time-based prediction model	*DATE_AND_TIME*	*20*

DATA TYPE

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

7.4 Creating variables by importing

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

Information

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

7.4.1 XML import

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

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

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

REQUIREMENTS

The following conditions are applicable during import:

- **Backward compatibility**

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

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

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

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

Hint

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

7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

Information

Import and Export via CSV or dBase supported; no driver specific variable settings, such as formulas. Use export/import via XML for this.

IMPORT DBF FILE

To start the import:

1. right-click on the variable list
2. in the drop-down list of **Extended export/import...** select the **Import dBase** command
3. follow the import assistant

The format of the file is described in the chapter File structure.

Information

Note:

- Driver object type and data type must be amended to the target driver in the DBF file in order for variables to be imported.
- dBase does not support structures or arrays (complex variables) at import.

EXPORT DBF FILE

To start the export:

1. right-click on the variable list
2. in the drop-down list of **Extended export/import...** select the **Export dBase...** command
3. follow the export assistant

Attention

DBF files:

- must correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
- must not have dots (.) in the path name.
  e.g. the path *C:\users\John.Smith\test.dbf* is invalid.
  Valid: *C:\users\JohnSmith\test.dbf*
- must be stored close to the root directory in order to fulfill the limit for file name length including path: maximum 255 characters

The format of the file is described in the chapter File structure.

Information

dBase does not support structures or arrays (complex variables) at export.

FILE STRUCTURE OF THE DBASE EXPORT FILE

The dBaseIV file must have the following structure and contents for variable import and export:

Attention

dBase does not support structures or arrays (complex variables) at export.

DBF files must:

- conform with their name to the 8.3 DOS format (8 alphanumeric characters for name, 3 characters for extension, no space)
- Be stored close to the root directory (Root)

STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	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

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

Identification	Typ e	Field size	Comment
SIGMIN	Floa t	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.

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

Attention

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

LIMIT VALUE DEFINITION

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

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

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

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

7.4.3 Online import

Online import is divided into 4 steps:

1. Preparation for the import.
2. Configuration of the variables to be imported in the dialog.
3. Checking of the entries in the dialog confirmation.
4. Execution of the import.

During each step, details on actions that have been carried out and errors that occurred can be read in the LOG.

PREPARATION FOR THE IMPORT

To prepare the import:

1. The connection to the **Service Hub** is established.
   If an error occurs in the process, the online import is aborted with an error message.
   Errors are, for example, missing communication or a timeout when waiting for a response from the service hub.
2. Once the connection has been established, there is a search for available Analyzer servers.
   If, within the configured timeout, there is not at least 1 available Analyzer server found, the online import is canceled with an error message.
3. The necessary API editor objects are obtained.
   The following is read in the process:
   - The names of all variables that exist in the project.
   - The **Symbolic address** of all the variables that exist in the project for this driver.

   This information is used for input validation in the import dialog. If an error occurs in the process, the online import is ended with an error. An error is, for example, if one of the necessary objects is not found.

   The preparation is completed and the online variable import dialog is displayed.

CONFIGURATION OF THE VARIABLES TO BE IMPORTED IN THE DIALOG

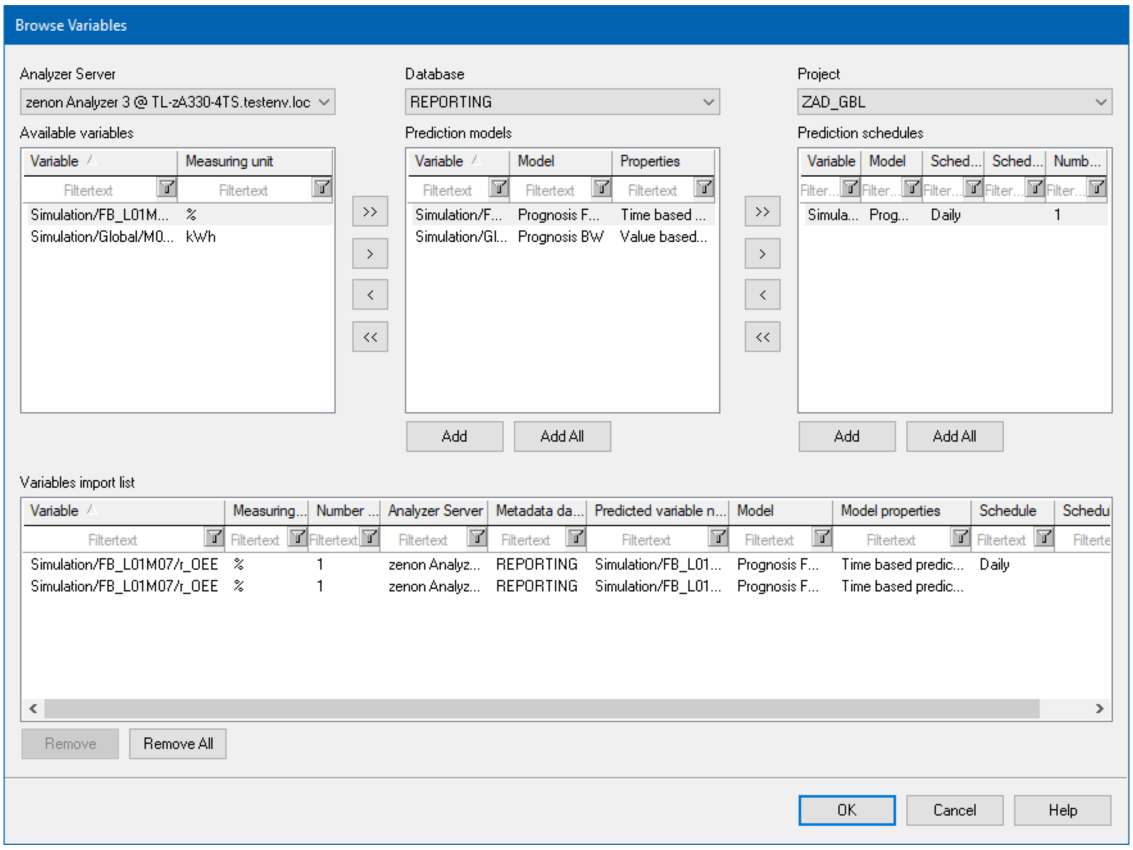
The import dialog allows the browsing of information to all available Analyzer servers.

Contents:

- Databases per Analyzer server
- Projects per database
- Variables with prediction models for each project
- Prediction model per variable
- Schedules for automated creation of predictions per prediction model

Variables for the driver can be derived and imported from the prediction models.

BROWSE VARIABLES DIALOG


Browse Variables

Analyzer Server: zenon Analyzer 3 @ TL-zA330-4TS.testenv.loc

Database: REPORTING

Project: ZAD_GBL

Available variables

Variable	Measuring unit
Simulation/FB_L01M...	%
Simulation/Global/M0...	kWh

Prediction models

Variable	Model	Properties
Simulation/F...	Prognosis F...	Time based ...
Simulation/Gl...	Prognosis BW	Value based...

Prediction schedules

Variable	Model	Sched...	Sched...	Numb...
Simula...	Prog...	Daily		1

Variables import list

Variable	Measuring...	Number ...	Analyzer Server	Metadata da...	Predicted variable n...	Model	Model properties	Schedule	Schedu
Simulation/FB_L01M07/r_OEE	%	1	zenon Analyz...	REPORTING	Simulation/FB_L01...	Prognosis F...	Time based predic...	Daily	
Simulation/FB_L01M07/r_OEE	%	1	zenon Analyz...	REPORTING	Simulation/FB_L01...	Prognosis F...	Time based predic...		

<u>This dialog consists of:</u>

1. Pre-filter selection
2. List of project details plus buttons
3. Action symbols for list operations between the list views
4. List of variables to be imported
5. Buttons to close the dialog and start the help

PRE-FILTER SELECTION

Selection list	Description
Analyzer Server	This list contains all available Analyzer servers. Only Analyzer servers that are online are available. If the online status of an Analyzer server changes, this is inserted into the selection list or removed from it. If the Analyzer server that is currently selected switches to offline status, an error message is displayed and the server selection is reset to *None*. If the name of the Analyzer server that is currently

Selection list	Description
	selected changes, an error message is shown and the server selection is reset to *None*. After selecting an Analyzer server, the available databases of the selected Analyzer server are queried. The database selection list is updated and the selected database is reset to *None*.
Database	This selection list contains all databases that are available on the selected Analyzer server. The list can only be operated if an Analyzer server has been selected. After selecting a database, the database content is queried by the Analyzer server, the project selection list is updated and the project selection is reset to *None*.
Project	This list contains all projects contained in the selected database for which there is at least one prediction model. The list can only be operated if a database has been selected. After selecting a project, the list views for the project details are emptied and the variable view list is populated with all variables of the project that exist for the prediction model.

PROJECT DETAILS

The project details show all three filtering in each column in three views. They support multiple selection. All list views can only be operated if a project has been selected.

With all insertion operations, the filter of the list view for variables to be imported is reset so that newly-inserted variables to be imported are guaranteed to be visible. If an object to be inserted is already in the list view, a new object is inserted as a copy. As a result, different variables with different numbers of values can be created in an online import call.

To remove all variables to be imported that correspond to the filter from the list view:

1. Configure the filter.
2. Select all variables to be imported.
3. Click on the button to remove the selected variables.

VARIABLES

Allows the selection of variables of the selected project:

Column	Description
Variable	Name of the variables in the zenon Analyzer database.
Measuring unit	Unit of the variables in the zenon Analyzer database.

PREDICTION MODELS

Allows the selection of the prediction models:

Column	Description
Variable	Name of the variables in the zenon Analyzer database. **Note:** Prediction models for different variables can have the same name.
Model	Name of the prediction model. Is configured in the **Prediction Model Manager** of the zenon Analyzer.
Properties	Properties of the prediction model. Are configured in the **Prediction Model Manager** of the zenon Analyzer and contain: ▸ An indication of whether the prediction model is time based or value based. ▸ Base cycle time of the prediction model if it is time based. It is the cycle time with which the variable is archived that is predicted.
Button	**Description**
Add	Adds variables for all prediction models selected in the list view for prediction models for variables to be imported. Only operable if at least one entry in the list has been selected.
Add all	Adds variables for all prediction models visible in the list view for prediction models for variables to be imported. Only operable if at least one entry in the list is present.

PREDICTION SCHEDULES

Allows the selection of schedules for automatic prediction:

Column	Description
Variable	Name of the variables in the zenon Analyzer database. **Note:** Prediction models for different variables can have the same name.
Model	Name of the prediction model. **Note:** Schedules for different prediction models can have the same name.
Schedule	Name of the schedule. Is configured in the **ZAMS** of the zenon Analyzer.
Schedule description	Description of the schedule. Is configured in the **ZAMS** of the zenon Analyzer.
Number of values	Number of values that are contained in each of the predictions triggered by this schedule. Is configured in the **ZAMS** of the zenon Analyzer.
Button	**Description**
Add	Adds variables for all schedules selected to the list view for schedules in the list view for variables to be imported. Only operable if at least one entry in the list has been selected.
Add all	Adds variables for all schedules visible (that correspond to the filter) to the list view for schedules for variables to be imported. Only operable if at least one entry in the list is present.

ACTION SYMBOLS

Allows actions between the lists.

Symbols	Description
>>	Inserts sub-objects of all currently-displayed elements of the list view to the left of the button on the right in the list view. Only operable if, in the list view left of the button, at least one element is displayed.
>	Adds sub-objects of all elements in the list view to the left

Symbols	Description
	of the button into the list view to the right of the button. Only operable if, in the list view to the left of the button, at least one element is displayed.
<	Removes selected objects from the list view to the right of the button. This button can only be operated if at least one element is selected in the list view to the right of the button. Only operable if, in the list view to the right of the button, at least one element is displayed.
<<	Removes all objects from the list view to the right of the button. This is applicable regardless of whether they correspond to the filter or not.-

Sub-objects are:

- For a variable: Prediction models of the variables.
- For a prediction model: Schedules of the prediction model.

When inserting into a list view, all filters of the list view are reset in order for the newly-inserted objects to be guaranteed to be displayed in the list. If an object is already in the selection list in which it is to be inserted, it is not inserted again.

When removing prediction models, all schedules of the prediction models to be removed are also removed from the schedule list view.
In order to remove the objects of a list view that correspond to the filter:

1. Configure the filter.
2. Select all displayed objects.
3. Click on the symbol to remove all selected objects from the list view.

LIST OF VARIABLES TO BE IMPORTED

You can find details for editing and for additionally-created parameter and command variables at the end of the table and in the section about **execution of the import**.

VARIABLES TO IMPORT

Column	Description
Variable	Name of variables to be imported. The target variable is created with this name. Any parameter or command

Column	Description
	variables that may be necessary receive a name that consists of the name set here and a suffix. With newly-inserted variables, the name of the variables is always entered into the zenon Analyzer database for which the prediction model was created (for which the schedule was created). The cells of this column can always be edited.
Measuring unit	Measuring unit of the variables to be imported. This is only used for target variables, not for parameter or command variables. With newly-inserted variables, the measuring unit of the variables is always entered into the zenon Analyzer database for which the prediction model was created (for which the schedule was created). The cells of this column can always be edited.
Number of values	Number of values for the target variable. This is only used for target variables, not for parameter or command variables. With newly-inserted variables, 1 is initially entered here if the variable has been created for a prediction model or the number of the values from the schedule if the variable has been created for a schedule. The cells of this column can only be edited for variables that have been created for prediction models.
Analyzer Server	Name of the zenon Analyzer instance from which the predictions come. This column is a column for information only. Its content is not relevant for the import and thus the cells of this column cannot be edited.
Metadata Database	Name of the database in which the prediction model is saved. This column is a column for information only. Its content is not relevant for the import and thus the cells of this column cannot be edited.

Column	Description
Predicte variable name	Name of the variables in the database for which the prediction model has been created. This column is a column for information only. Its content is not relevant for the import and thus the cells of this column cannot be edited.
Model	Name of the prediction model used. This column is a column for information only. Its content is not relevant for the import and thus the cells of this column cannot be edited.
Model properties	Properties of the prediction model used. This column is a column for information only. Its content is not relevant for the import and thus the cells of this column cannot be edited.
Schedule	Name of the schedule used. If the variable has been created for a prediction model and not for a schedule, it remains empty. This column is a column for information only. Its content is not relevant for the import and thus the cells of this column cannot be edited.
Schedule description	Description of the schedule used. If the variable has been created for a prediction model and not for a schedule, it remains empty. This column is a column for information only. Its content is not relevant for the import and thus the cells of this column cannot be edited.

Button	Description
Remove	Removes all selected entries from the list view for variables to be imported. Only operable if at least one entry in the list has been selected.
Remove all	Removes all entries from the list view for variables to be imported.

Column	Description
	Only operable if at least one entry in the list has been selected.

When clicking on a cell in this list, the cell is switched to edit mode, if the cell can be edited. Editing can be ended in different ways:

- Click on another element in the dialog: Editing is confirmed.
- Press the Enter key: Editing is confirmed.
- Press the Escape key: Editing is canceled and the previous value is restored.
- Press the Tab key: Editing is confirmed and the next editable cell is switched to edit mode. The next editable cell can be the cell to the right of the current cell ("**Measuring unit**" if "**Variable name**" is currently being edited; "**Number of values**" if "**Measuring unit**" is currently being edited and the variable has not been created for a schedule) or the "**Variable name**" column in the next line (if there is still a line underneath the cell currently being edited; if "**Measuring unit**" is currently being edited and the variable for a schedule has been created; if "**Number of values**" is currently being edited and the variable for a prediction model has been created).

CLOSING THE DIALOG

When confirming the editing, the currently-entered value is applied if it is not empty and can also be converted into a figure for the "**Number of values**" column. Otherwise the previous value is restored.

CLOSE DIALOG

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

CHECKING THE INPUTS WHEN CONFIRMING THE DIALOG

The entries are validated by clicking on **OK**.
The following error states can occur:

- Two or more variables in the list have the same name.
  Here, parameter and command variables to be explicitly created (see **executing the import** section) are taken into account.
- One or more of the variables present in the list has a name that has already been used for another variable in this project.

Here, parameter and command variables to be explicitly created (see **executing the import** section) are taken into account.

- Two or more variables present in the list have the same symbolic address.
  - With variables that are derived from schedules, this means that 2 or more variables should be created for the same schedule. This is an error, because several variables for the same schedule are not permitted. They would only provide redundant information.
  - Polling variables that are derived from a prediction model each receive a new unique GUID. With polling access, several variables are permitted for the same prediction model. This is because these different value figures can be executed with different input values.
- One or more of the variables in the list have a symbolic address that is already used for another variable for this driver instance in this project. The same detail indicators, such as two or more variables present in the list with the same symbolic address, are applicable.

If errors are discovered, notification is given by means of a pop-up message and the dialog is not closed. The error message contains information for each error category discovered. The names of the variables to be imported are contained in each paragraph. The dialog is only closed if no errors are discovered during this check.

Checks for clashes are always made without taking upper-case/lower-case letters into account. Only unique variable names are contained in the error message, but taking capitalization into account. It can therefore happen that several variable names are included in the error message where only the upper-case and lower-case letters differ.

EXECUTION OF THE IMPORT

All necessary variables are created during the import.
In doing so, the following applies:

- Only the target variables are created for spontaneous communication.
- For polling variables, the target variable, the necessary input variables and the command variables are created.
- Target variables are always the *Predicted Value* driver object type and *LREAL* data type. They always contain the measuring unit set in the dialog, the variable name set in the dialog and the number of values set in the dialog:
  - 1 value: A simple variable is created.
  - More than 1 value: An array variable with the set number of values in dimension 1 is created.
    The array is always set for *Startindex 1*, automatic addressing and new offset for each data type. All elements of the array are activated. In doing so, all array content variables get the measuring unit and the symbolic address.
- Input variables are always *Prediction Input Value* driver object type.
  The data type depends on the model:

    - *DATE_AND_TIME*, if the model is time based.
    - *LREAL*, if the model is value based.

  Variable names always have the format **[name of the target variables]_[name of the parameters]**.
  The unit of such variables always remains empty.
  The symbolic address for these variables is always **[symbolic address of the target variables]::[name of the parameter]**.
  The parameter variables that are created depend on the number of values of the target variables:

    - 1 value: Input variable for the **Input** parameter.
    - More than 1 value: Input variables for the parameters **From** and **To**.

- Command variables are always *Prediction Trigger* driver object type and *BOOL* data type.

  The unit of such variables always remains empty.
  They always receive their variable name as **[name of the target variables]_command]**.
  The symbolic address is always identical to the symbolic address of the target variables.

The following variables are created for the following setups:

- Variable was derived from a prediction model and has precisely 1 value:
    - Target variable (simple)
    - Input variable for **Input** (data type depends on whether the model is time based or value based)
    - Command variable
- Variable was derived from a prediction model and has more than 1 value:
    - Target variable (array)
    - Input variable for **From** (data type depends on whether the model is time based or value based)
    - Input variable for **To** (data type depends on whether the model is time based or value based)
    - Command variable
- Variable was derived from a schedule and has precisely 1 value:
    - Target variable (simple)
- Variable was derived from a schedule and has more than 1 value:
    - Target variable (array)

7.5 Communication details (Driver variables)

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

- Information
- Configuration
- Statistics and
- Error message

The definitions of the variables implemented in the driver kit are available in the import file **DRVVAR.DBF** 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 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

Name from import	Type	Offset	Description
RTBuild	*UINT*	52	zenon build number
LineStateIdle	*BOOL*	24.0	TRUE, if the modem connection is idle
LineStateOffering	*BOOL*	24.1	TRUE, if a call is received
LineStateAccepted	*BOOL*	24.2	The call is accepted
LineStateDialtone	*BOOL*	24.3	Dialtone recognized
LineStateDialing	*BOOL*	24.4	Dialing active
LineStateRingBack	*BOOL*	24.5	While establishing the connection
LineStateBusy	*BOOL*	24.6	Target station is busy
LineStateSpecialInfo	*BOOL*	24.7	Special status information received
LineStateConnected	*BOOL*	24.8	Connection established
LineStateProceeding	*BOOL*	24.9	Dialing completed
LineStateOnHold	*BOOL*	24.10	Connection in hold
LineStateConferenced	*BOOL*	24.11	Connection in conference mode.
LineStateOnHoldPendConf	*BOOL*	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	*BOOL*	24.13	Connection in hold for transfer
LineStateDisconnected	*BOOL*	24.14	Connection terminated.
LineStateUnknow	*BOOL*	24.15	Connection status unknown
ModemStatus	*UDINT*	24	Current modem status
TreiberStop	*BOOL*	28	Driver stopped For *driver stop*, the variable has the value *TRUE* and an **OFF** bit. After the driver has started, the variable has the value *FALSE* and no **OFF** bit.
SimulRTState	*UDINT*	60	Informs the status of Runtime for driver simulation.
ConnectionStates	*STRING*	61	Internal connection status of the driver to the PLC.

Name from import	Type	Offset	Description
			Connection statuses: *0:* Connection OK *1:* Connection failure *2:* Connection simulated Formating: *<Netzadresse>:<Verbindungszustand>;...;...;* A connection is only known after a variable has first signed in. In order for a connection to be contained in a string, a variable of this connection must be signed in once. The status of a connection is only updated if a variable of the connection is signed in. Otherwise there is no communication with the corresponding controller.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	*BOOL*	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	*BOOL*	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	*BOOL*	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings **PhoneNumberSet** and **ModemHwAdrSet**.
PhoneNumberSet	*STRING*	38	Telephone number, that should be used
ModemHwAdrSet	*DINT*	39	Hardware address for the telephone number

Name from import	Type	Offset	Description
GlobalUpdate	*UDINT*	3	Update time in milliseconds (ms).
BGlobalUpdaten	*BOOL*	4	TRUE, if update time is global
TreiberSimul	*BOOL*	5	TRUE, if driver in sin simulation mode
TreiberProzab	*BOOL*	6	TRUE, if the variables update list should be kept in the memory
ModemActive	*BOOL*	7	TRUE, if the modem is active for the driver
Device	*STRING*	8	Name of the serial interface or name of the modem
ComPort	*UINT*	9	Number of the serial interface.
Baudrate	*UDINT*	10	Baud rate of the serial interface.
Parity	*SINT*	11	Parity of the serial interface
ByteSize	*USINT*	14	Number of bits per character of the serial interface Value = *0* if the driver cannot establish any serial connection.
StopBit	*USINT*	13	Number of stop bits of the serial interface.
Autoconnect	*BOOL*	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	*STRING*	17	Current telephone number
ModemHwAdr	*DINT*	21	Hardware address of current telephone number
RxIdleTime	*UINT*	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	*UDINT*	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	*UDINT*	20	Number of ringing tones before a call is accepted
ReCallIdleTime	*UINT*	53	Waiting time between calls in seconds (s).
ConnectTimeout	*UINT*	54	Time in seconds (s) to establish a

Name from import	Type	Offset	Description
			connection.

STATISTICS

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

ERROR MESSAGE

Name from import	Type	Offse t	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:

- Receipt of predicted values that are created and sent by the Analyzer server in an automated manner by means of a schedule.
- Requests fro predicted values from the Analyzer server with specifiable input values.
- Predictions can be received and queried for time-based and value-based prediction models.
- Each prediction can be an individual prediction or a range prediction.

- Immediate detection of changes of the status of the Analyzer server: If this goes offline, all dependent variables are highlighted as *INVALID*.
- Checking the existence and properties of databases, prediction models and schedules in the event of a change to the status of a higher-level object. This check is carried our after expiry of a defined time after the last check: If an object cannot be found or its properties differ from the variable settings, all dependent variables are marked as *INVALID*. Hierarchy of the check: Analyzer instance -> Database -> Models and schedules.
  Examples of differing properties:
  - The model is time based, but it is given as value based in the symbolic address.
  - The number of variables in the array does not correspond to the number of the predicted values in the schedule.

PROCESS FOR SPONTANEOUS COMMUNICATION (FROM DRIVER START):

1. The variables are sent to the driver on starting. The administration structure for the required objects on the Analyzer server is built up from the symbolic addresses of all existing variables in the driver.
2. The Analyzer server is detected as online.
3. The database is found when checking the status.
4. The prediction model and the schedule are found when checking the status and the properties correspond to the configuration.
5. As soon as the variable is signed in, the receipt structure for schedule-based data is created and activated.
6. As soon as the Analyzer server sends a prediction value message, it is received in the driver via the receipt structure.
7. The values from the message are read and forwarded to the variables.

PROCESS FOR POLLING COMMUNICATION (FROM DRIVER START):

1. The variables are sent to the driver on starting. The administration structure for the required objects on the Analyzer server is built up from the symbolic addresses of all existing variables in the driver. In doing so, all variables required for polling communication - prediction value variable, input variables, control variable - are connected to one another via the GUID from the symbolic address.
2. The Analyzer server is detected as online.
3. The database is found when checking the status.
4. The prediction model is found when checking the status and the properties correspond to the configuration.
5. The necessary input values are written to the variables in the Runtime.

6. The control variable is set to *1* in the Runtime.
7. The polling process is started in the driver by changing the control variable.
8. The input values are read in the driver from the variables.
9. A query-response structure is created in the driver and the query is sent with the desired input values and number of values (see **driver objects** (on page 23)) via the structure.
10. The query is received in the Analyzer server. The predictions are created and sent as a response.
11. The response is received on the driver.
12. The values are read from the response and forwarded to the prediction variables.
13. The control variable is reset to *0* by the driver.

9 Driver command function

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

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

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

Attention

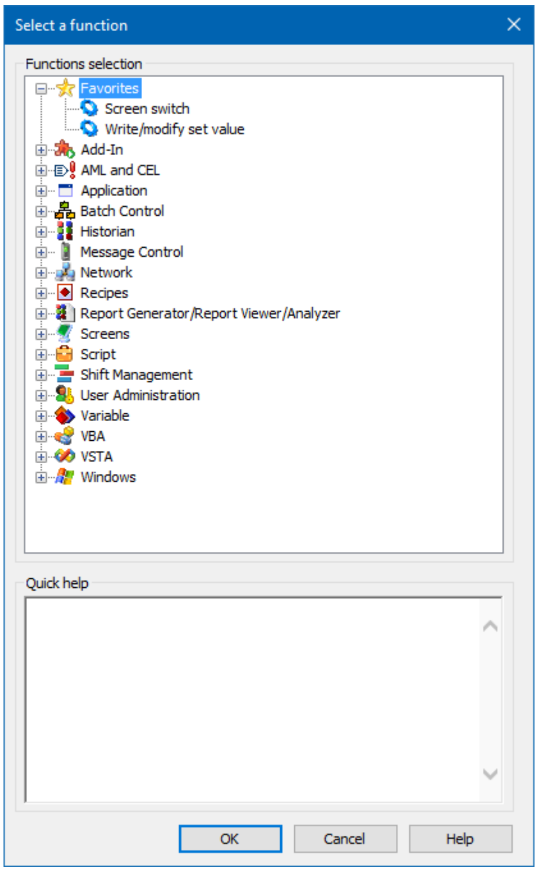
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

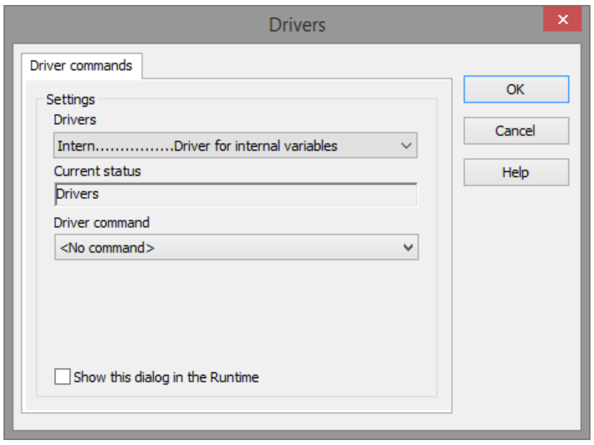
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



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

   The dialog for configuration is opened



Settings

Drivers: Intern................Driver for internal variables

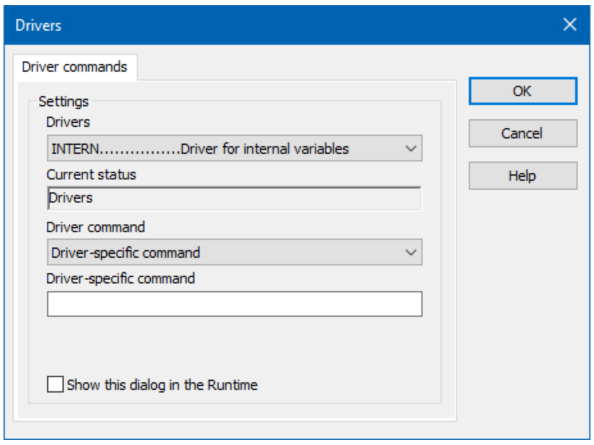
Current status: Drivers

Driver command: <No command>

☐ Show this dialog in the Runtime

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



Settings
Drivers
INTERN...............Driver for internal variables
Current status
Drivers
Driver command
Driver-specific command
Driver-specific command

☐ Show this dialog in the Runtime

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 *driver-specific command* has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime: - *Active*: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. - *Inactive*: The Editor configuration is applied in the Runtime when executing the function. Default: *inactive*

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.
Start driver (online mode)	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.
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	Entry of a driver-specific command. Opens input field in order to enter a command.
Activate driver write set value	Write set value to a driver is possible.
Deactivate driver write set value	Write set value to a driver is prohibited.
Establish connection with modem	Establish connection (for modem drivers)

Driver command	Description
	Opens the input fields for the hardware address and for the telephone number.
Disconnect from modem	Terminate connection (for modem drivers)
Driver in counting simulation mode	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.
Driver in static simulation mode	No communication to the controller is established. All values are initialized with *0*.
Driver in programmed simulation mode	The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime.

DRIVER COMMAND FUNCTION IN THE NETWORK

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

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

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

10 Error analysis

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

10.1 Analysis tool

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

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

%ProgramData%\COPA-DATA\LOG.

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

In the Diagnosis Viewer you can also:

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

Note:

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

Attention

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

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

10.2 Check list

GENERAL TROUBLESHOOTING

Check the following in the event of errors:

- Is the computer switched on with the zenon Analyzer and are the Analyzer server services running?
- Are the **Service-Hub** services running on the computer envisaged for this?
- Analysis with the **Diagnosis Viewer** (on page 56):
  -> Which messages are displayed?
- Is the computer available with the zenon Analyzer and the **Service Hub** in the **TCP/IP** network? Can it be contacted via the *Ping* command?
- Was the **Symbolic address** of the variable set correctly?
- Did you use the right object type for the variable

  Example: Driver variables are purely statistics variables. They do not communicate with the PLC.

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

%ProgramData%\COPA-DATA\LOG.

Attention: With the default settings, a driver only logs error information. With the Diagnosis Viewer you can enhance the diagnosis level for most of the drivers to "Debug" and "Deep Debug". With this the driver also logs all other important tasks and events. You can find more information on the Diagnosis Viewer in the Diagnosis Viewer (main.chm::/12464.htm) manual.

The following is required for further analysis of errors:

- The project backup
- LOG files

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

SOME VARIABLES REPORT INVALID.

- **Prediction Trigger** and **Prediction Input Value** driver object type variables go to INVALID if the desired prediction model is not available.
- **Predicted Value** driver object type values go to INVALID if the desired prediction model is no longer available or no prediction value has been received.

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 56):
-> Which messages are displayed?