



zenon
by COPA-DATA

zenon driver manual

OPCUA32

v.8.10



COPADATA

© 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.....	5
2	OPCUA32.....	5
3	OPCUA32 - data sheet	6
4	Driver history.....	7
5	Requirements	8
5.1	PC	8
5.2	Control.....	8
6	Configuration	8
6.1	Creating a driver.....	9
6.2	Settings in the driver dialog.....	12
6.2.1	General.....	13
6.2.2	Connections.....	17
7	Creating variables.....	30
7.1	Creating variables in the Editor.....	30
7.2	Addressing	34
7.3	Driver objects and datatypes.....	35
7.3.1	Driver objects	35
7.3.2	Mapping of the data types	36
7.4	Creating variables by importing.....	37
7.4.1	XML import	37
7.4.2	DBF Import/Export	38
7.4.3	Online import	44
7.5	Communication details (Driver variables).....	45
8	Driver-specific functions.....	51
8.1	DataChangeFilter.....	52
8.2	Configurable DataChangeTrigger.....	52
8.3	Several subscriptions	52
8.4	Writing of arrays.....	53
9	Driver command function.....	53

10 Error analysis	58
10.1 Analysis tool	58
10.2 Check list	59

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 OPCUA32

The **OPCUA driver** is for communication with OPC UA servers and is based on the official stack from the OPC Foundation. OPC UA is the abbreviation for **OPC Unified Architecture**.

Main features of the driver:

- ▶ Communication is spontaneous, i.e. amended variables are automatically reported by the server,
- ▶ The driver supports several servers
- ▶ The variables can be read from the server directly

- Variables are addressed by means of the **Browse Name** defined in the standard



Information

The OPCU UA driver for zenon 810 uses version 1.02-336-1 of the Ansi C stack.

3 OPCUA32 - data sheet

General:	
Driver file name	OPCUA32.exe
Driver name	OPC UA Client Treiber
PLC types	All OPC-UA servers with Data Access communication
PLC manufacturer	OPC; straton; COPA-DATA

Driver supports:	
Protocol	OPC-UA
Addressing: Address-based	Name based
Addressing: Name-based	--
Spontaneous communication	X
Polling communication	--
Online browsing	X
Offline browsing	--
Real-time capable	X
Blockwrite	--
Modem capable	--
RDA numerical	--
RDA String	--
Hysteresis	X

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

Requirements:	
Hardware PC	--
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	X

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
07.07.08	100	Created driver documentation

DRIVER VERSIONING

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

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

Example

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

5 Requirements

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

5.1 PC

This driver supports a connection via the standard network card of the PC. In order for PC and PLC to communicate:

- ▶ the PLC and the PC must be in the same network range
- ▶ the subnet masks must be configured accordingly on both devices
- ▶ the driver file **OPCUA32.exe** must be in the current zenon installation folder

5.2 Control

The PLC has to support the OPC Unified Architecture Protocol with the OPC Binary Transport.

Note: OPC UA Web services are not supported.

6 Configuration

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

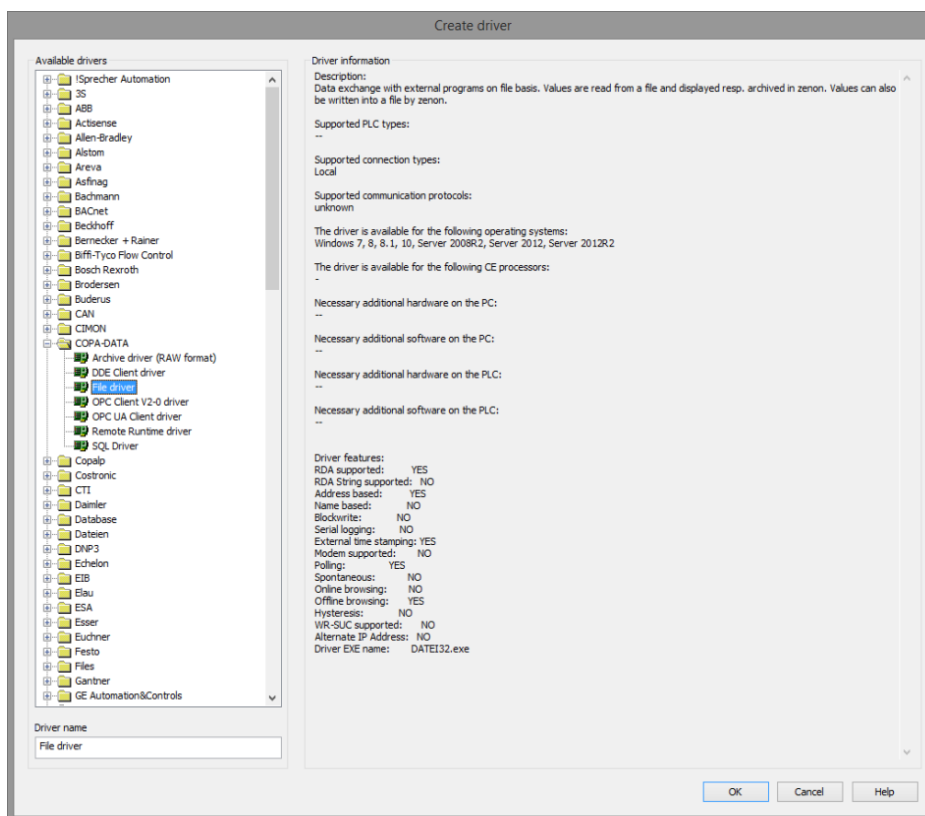


Information

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

6.1 Creating a driver

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



Parameter	Description
Available drivers	<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>

Parameter	Description
	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: <i>Empty</i> 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:

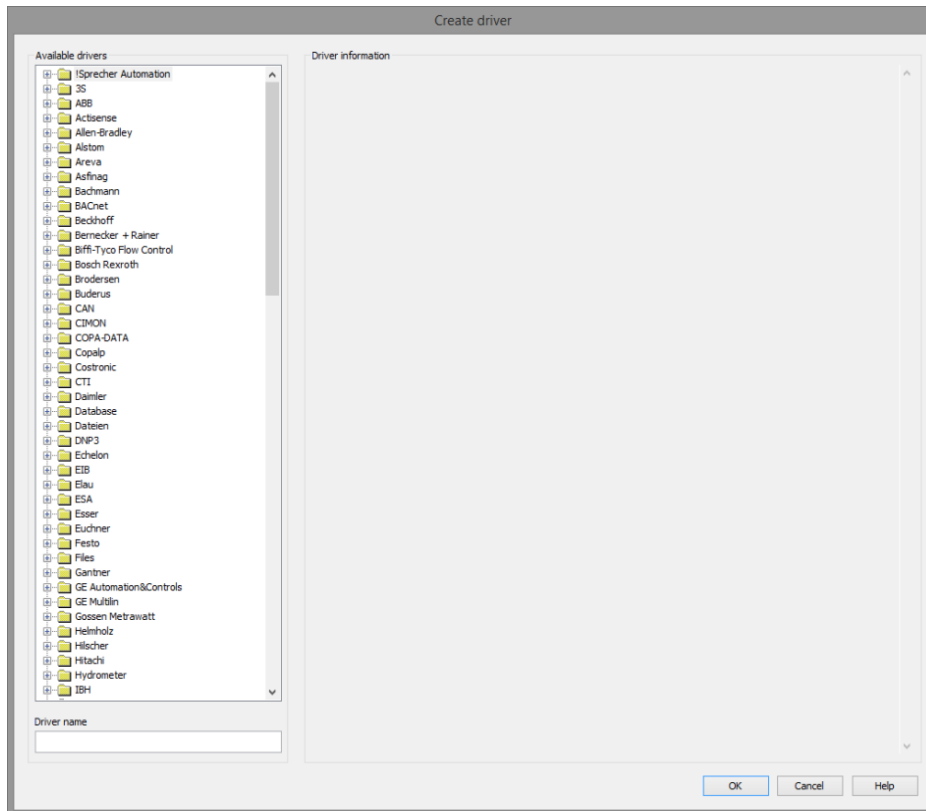
C:\ProgramData\COPA-DATA\zenon[version number].

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.

- The dialog offers a list of all available drivers.

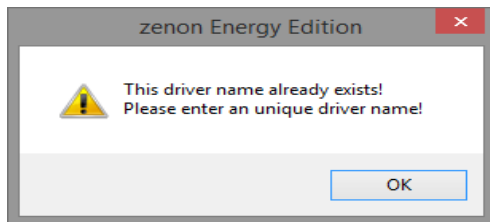


- Select the desired driver and name it in the **Driver name** input field.
This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.
The following is applicable for the **Driver name**:
 - ▶ The **Driver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

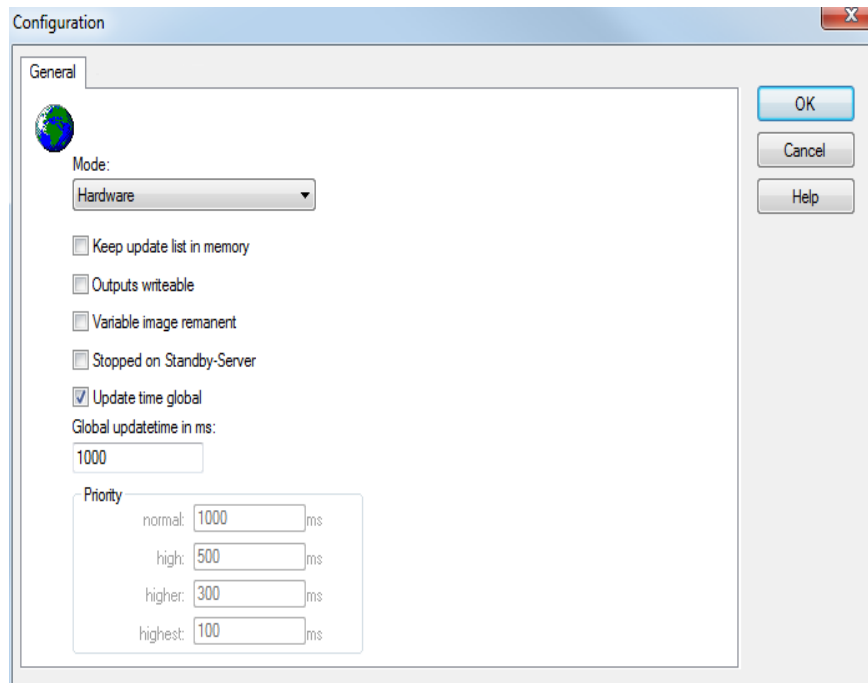
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

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



Option	Description
Mode	<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.

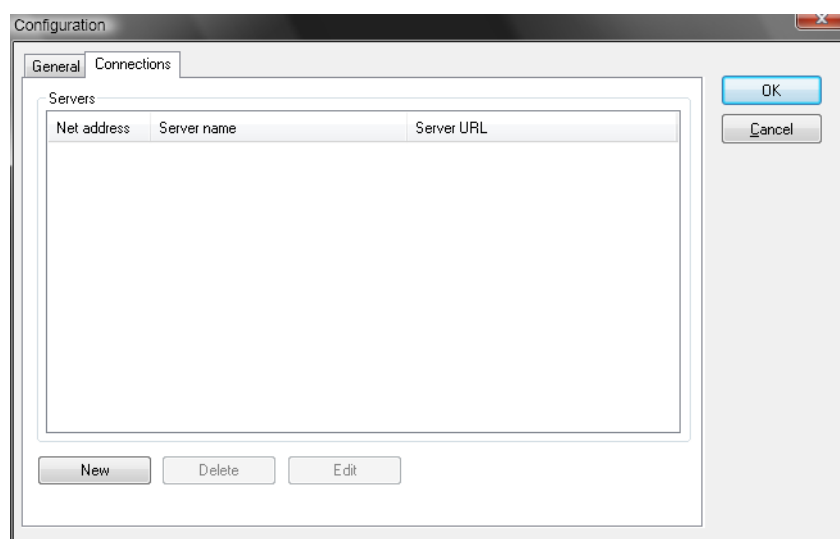
UPDATE TIME FOR SPONTANEOUS DRIVERS

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**.

6.2.2 Connections

Configure the OPC UA **Connections** to one or several PLCs in the tab.



SERVERS

List of configured connections. Select a connection to delete or modify it.

Parameter	Description
Net adress	The net address identifies the connection. Every connection must have a unique net address, which are assigned automatically. Variables are assigned to a connection via the net address.
Server name	Freely definable name for the easier distinction of connections.
Server URL	The network address which is used to contact the connection terminal of the server. e.g. <i>opc.tcp://server:4840</i>
New	Opens the dialog for creating a new connection.
Delete	Deletes highlighted entry from the list.
Edit	Opens highlighted entry for editing.

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.

CREATE NEW CONNECTION

Click on the **New** button. In the following dialog:

- ▶ Define the connection details in the **Communication Settings** (on page 19) tab
- ▶ Set the advanced options under **Advanced settings** (on page 21)
- ▶ Configure the certificates under **Certificates** (on page 26)

EDIT CONNECTION

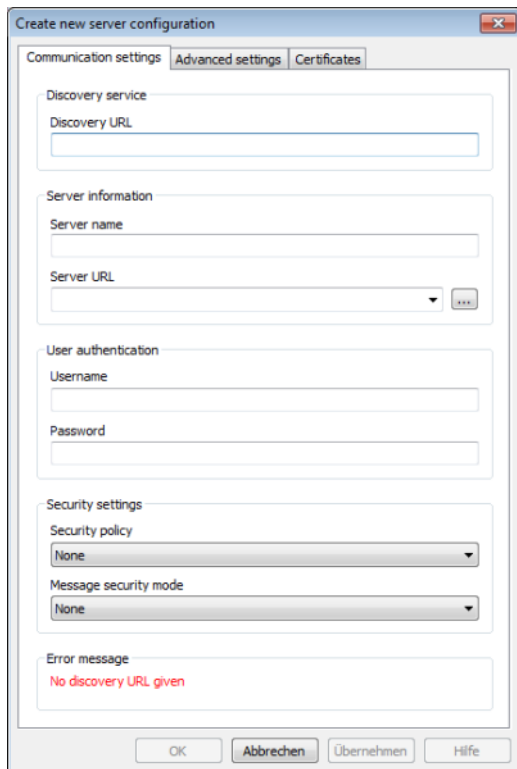
Select an existing connection in the connection dialog and click the **Edit** button to modify it. The properties for this are identical with the fields displayed when creating a new connection.

DELETE CONNECTION

To delete a connection:

- ▶ select the connection in the connection list
- ▶ click on the button **Delete**
- ▶ the connection will be removed

6.2.2.1 Communication settings



Parameter	Description
Discovery URL	<p>Input field for the address of the <i>Discovery service</i>. Existing OPC UA servers in the network can be identified and queried with this <i>Discovery service</i>.</p> <p>Default: <i>empty</i></p> <ul style="list-style-type: none"> ▶ Example: opc.tcp://192.168.0.1:4840
Server name	Freely-definable name of the configured connection.
Server URL	<p>The network address under which an OPC UA server in the network can be reached.</p> <p>IP addresses and DNS names can be configured as follows: <i>opc.tcp://IP address or DNS name:Port/Server Name</i></p> <ul style="list-style-type: none"> ▶ <i>Entry of the address in the input field</i> ▶ <i>Selection from drop-down list</i> <p>Clicking on the ... button starts the detection of the available servers on the basis of the configured Discovery URL.</p> <p>Example: opc.tcp://192.168.0.1:4841</p>

Parameter	Description
	Example: opc.tcp://PC1:4841/SimulationServer
Username	<p>Optional user name with authentication activated on the server (endpoint with UserIdentityToken Username & Password).</p> <p>Note: Upper-case and lower case letters are taken into account when entering the password and user name.</p>
Password	Optional password with authentication activated on the server.
Security Policy	<p>Entry of the algorithm with encrypted communication.</p> <p>Select from drop-down list:</p> <ul style="list-style-type: none"> ▶ <i>None</i> ▶ <i>Basic128</i> ▶ <i>Basic128RSA15</i> ▶ <i>Basic256</i> <p>Default: <i>None</i></p> <p>The values of the selection list correspond to the OPC UA Specification, Part 7.</p> <p>Note: During reading, all supported values are provided by the discovery service.</p>
Message Security Mode	<p>Message security defines the security level when sending messages.</p> <p>Select from drop-down list:</p> <ul style="list-style-type: none"> ▶ <i>None</i> All messages are sent in plain text (not signed and not encrypted). ▶ <i>Sign</i> All messages are signed but not sent in encrypted form. ▶ <i>Sign & Encrypt</i> All messages are signed and sent in encrypted form. <p>Default: none</p>

Error Message

Display of incorrect or missing configurations. This output field is applicable for all **Advanced settings** and is not limited to the current tab.

Example: *Incomplete configuration on another tab.*

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.

CONFIGURATION

For a complete and valid configuration, it is sufficient to enter, in the **Discovery URL** input field, the address of a correctly-configured *Discovery Server*.

If this *Discovery Server* has been configured correctly, a click on the ... button (next to the **Server URL** field) will cause the necessary configuration data from this *Discovery Server* to be read and applied.

Attention: If **Sign** or **Sign & Encrypt** is used, the server certificate must be imported in the **Certificates** tab. You can find details on this authentication in the **Part 4 - 5.6.3 ActivateSession** norm.

Incorrect entries in the discovery URL are shown in the dialog by clicking on the ... button in a dialog:

- ▶ *A value had an invalid Syntax (0x80b60000)*
Incorrect syntax for discovery URL
- ▶ *The requested operation is not supported (0x803d0000)"*
opc.tcp is missing for the entry for the discovery URL

6.2.2.2 Advanced settings

Subscription

Options for the connection.

Parameter	Description
Publishing interval [ms]	<p>Defines the interval (in milliseconds) within which the server must send a message for the <i>Subscription</i>.</p> <p>The server can also change this interval upwards in order to adhere to technical limitations.</p> <p>Default: 500</p>
Lifetime count	<p>States how often the Publishing interval can be exceeded before the <i>Subscription</i> is deleted by the server.</p> <p>Values: ≥ 1 and $\geq 3 \times \text{Maximum keep-alive count}$</p> <p>Default: 30</p>
Maximum keep-alive count	<p>States how often the Publishing interval can expire before a <i>keep-alive</i> message is sent.</p> <p>Values: ≥ 1</p> <p>Default: 5</p>

MONITORED ITEMS

Parameter	Description
Sampling interval	<p>Sampling interval in milliseconds, in which the server must query MonitoredItems.</p> <p>The server can also change this interval upwards in order to adhere to technical limitations. Selection from drop-down list or direct entry in field. Possible values:</p> <ul style="list-style-type: none"> ▶ <i>Fastest practical rate (= 0)</i> ▶ <i>Default: sampling interval defined by publishing rate (= -1)</i> ▶ <i>Manual entry of values in milliseconds</i>
Data-Change trigger	<p>Drop-down list to select the trigger that sends a notification.</p> <ul style="list-style-type: none"> ▶ <i>Report notification on status code or value change</i> If the status is changed or a value is changed, a <i>notification</i> is sent. ▶ <i>Report notification on timestamp change also</i> This option contains the notification as it is also transferred for <i>Report notification on status code or</i>

Parameter	Description
	<p><i>value change.</i></p> <p>In addition, a notification is also sent in the event of changes to the time stamp (value or status).</p> <p>Default: <i>Report notification on status code or value change</i></p>
Use positive hysteresis of variables as AbsoluteDeadBand	<p>Checkbox to support the Positive for signal variable property.</p> <ul style="list-style-type: none"> ▶ <i>Not activated:</i> Configured Dead bands are not taken into account. ▶ <i>Activated:</i> Only values of the configured Dead bands are shown. <p>If this checkbox is activated, the configuration in the Positive for signal property is taken into account. You can find this property in the Value calculation variable property, in the Hysteresis properties group.</p> <p>Default: <i>inactive</i></p> <p>Note: You can find further information about this in the Variables (main.chm::/15247.htm) manual in the Hysteresis chapter.</p>

ADDRESSING

Parameters	Description
Use persistent Node IDs	<p>Behavior of the driver when signing in variables in Runtime.</p> <ul style="list-style-type: none"> ▶ <i>Activated:</i> For the following access (creation of <i>monitored Items</i>, read and write access), the saved <i>NodeID</i> and <i>NamespaceIndex</i> from the Symbolic address property is used. When creating variables through online import, the property is set accordingly. A requirement is that the <i>NodeID</i> does not change on the server. ▶ <i>Not activated:</i> Executes the <i>TranslateBrowsePathToNodeid</i> function once. The driver then uses the <i>NodeID</i> <p>Note: This option can, with some (especially smaller) OPCA UA servers, lead to connection problems if the <i>TranslateBrowsePathToNodeid</i> query leads to a higher</p>

Parameters	Description
	server load. Default: <i>inactive</i>

CONNECTION

Parameters	Description
Read initial values on startup	<ul style="list-style-type: none"> ▶ <i>Active:</i> In addition to signing in the variables (advise) a read request is also transferred. ▶ <i>Inactive:</i> No additional read request is sent to the server. The values may be received with a delay, depending on the server. <p>Default: <i>active</i></p> <p>Recommendation: Select this option if a large amount of variables are to communicate with the server.</p>
Access array variables on index range	<p>Working with OPC UA array variables. In addition to the Net address and addressing using the Browse name property, the offset of the variable is also relevant. The offset is always one higher than the array index (in OPC UA, the offset of an array starts at 0)</p> <ul style="list-style-type: none"> ▶ <i>Active:</i> Each array element is signed in as its own variable for a value change. <i>IndexRange</i> is used for <i>CreateMonitoredItems</i>. <p><i>Inactive:</i> The complete array is registered for value changes. As a result of this, large arrays for which most or all array elements are needed sign in more quickly on the server. In doing so, it is always the complete array that is transferred, even with just a few value changes. The network load can be increased as a result.</p> <p>Default: <i>active</i></p> <p>Recommendation: Select this option if only a few elements from large arrays are needed.</p>

Error Message

Display of incorrect or missing configurations. This output field is applicable for all **Advanced settings** and is not limited to the current tab.

Example: *Incomplete configuration on another tab.*

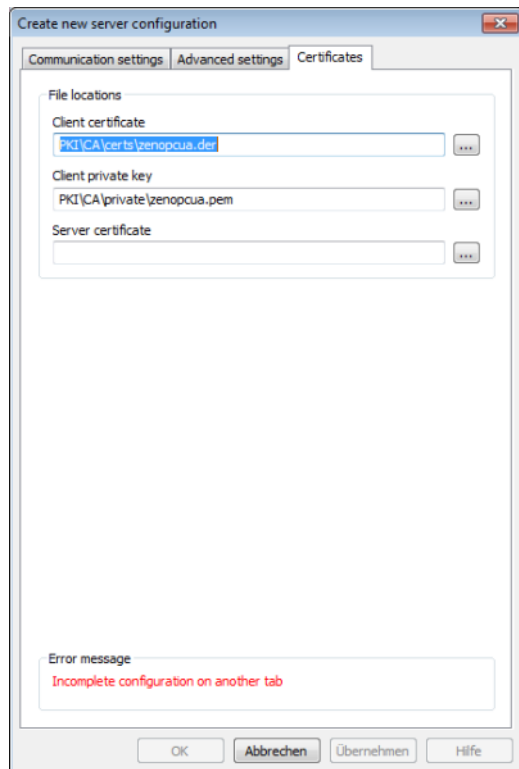
CLOSE DIALOG

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.

Hint: The connection failure to the OPCUA is detected by the OPCUA client driver using the methods recommended in the standard: It is expected that the server sends a *PublishResponse* after *RevisedMaxKeepAliveCount* * *RevisedPublishingInterval* at the latest, after the time when the last *PublishResponse* was sent. If no value of *MonitoredItems* has changed, an empty *KeepAlive PublishResponse* is expected. The OPCUA client driver waits another 5 seconds after the time has elapsed for the response from the server and then establishes a connection failure. The connection is separated at the client, no *DeleteSubscriptionRequest* is sent because it cannot be guaranteed that the server will still react.

6.2.2.3 Certificates

You configure the certificates for encrypted communication in the **Certificates** tab:



FILE LOCATIONS

Parameter	Description
Client Certificate	<i>Application Instance Certificate</i> of the client. This certificate is used by the server to encrypt messages. Path in the project folder: "..\Custom\Drivers\PKI\CA\certs\".
Client Private Key	Private key of the client. This is used to decrypt server messages. Path in the project folder: "..\Custom\Drivers\PKI\CA\certs\"
Server Certificate	<i>Application Instance Certificate</i> with public key of the server. This key is used to encrypt messages that are sent to the server.

Error Message

Display of incorrect or missing configurations. This output field is applicable for all **Advanced settings** and is not limited to the current tab.

Example: *Incomplete configuration on another tab.*

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.

CERTIFICATES FOR COMMUNICATION

INITIAL SETTING

As soon as a connection is created in the driver configuration, the driver automatically creates an *Application Instance Certificate* certificate with a public key and the appropriate private key, provided these do not yet exist.

VALIDITY

This self-created certificate is valid for one year. The OPCUA client driver does not check the validity of its own certificate. A tolerant server also accepts, under certain circumstances, a certificate that has already expired. For a server with a strict check, it is therefore advisable to create your own *Application Instance* certificate with a long period of validity.

DIFFERENT CERTIFICATES

By default, each connection and each driver uses the same *Application Instance Certificate* within a project. However, by definition, each OPCUA client should use its own *Application Instance Certificate*. In this case, or if an OPCUA server requires it, a self-created certificate can also be configured for each connection in the driver or in the respective drivers.

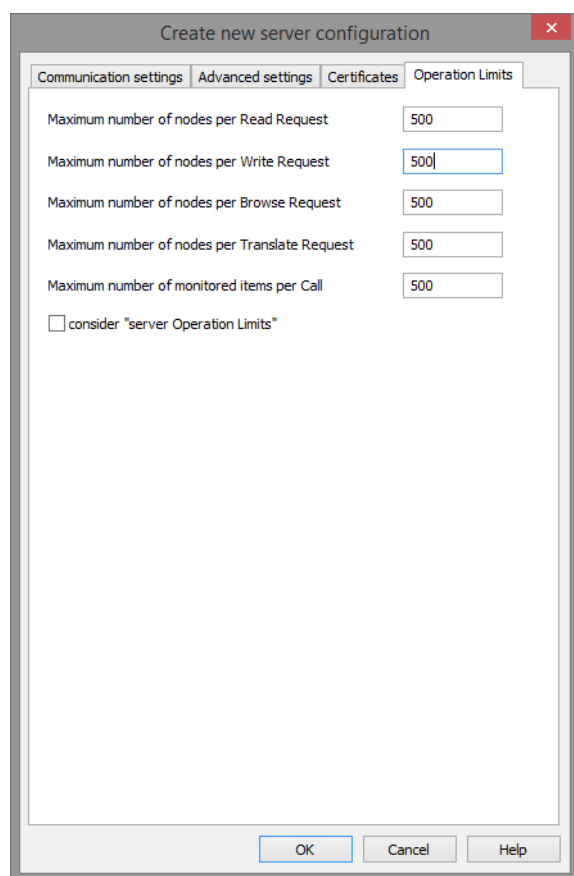
CONFIGURATION

A self-created *X509 OPC UA Application Instance Certificate*, in DER format, must be added in the Editor under Files -> Drivers -> PKI -> CA -> Certs. The appropriate private key, in PEM format, must be added in the Editor under Files -> Drivers -> PKI -> CA -> Private. The correct file for each connection must then be selected in the driver configuration.

Example

X509 OPCUA Application Certificates can be created with the **OPCUA Configuration Tool** from the **OPC Foundation** or with the **XCA** software.

6.2.2.4 Operation Limits



This tab is available for each projected connection.

Parameter	Description
Maximum number of nodes per Read Request	<p>Maximum number of <i>Nodes</i> that are taken into account in a <i>ReadRequest</i>.</p> <p>Default: 500</p> <p>Minimum: 1</p> <p>Maximum: 4294967295</p>
Maximum number of nodes per Write	Maximum number of <i>Nodes</i> that are taken into

Parameter	Description
Request	<p>account in a <i>WriteRequest</i>.</p> <p>Default:500</p> <p>Minimum: 1</p> <p>Maximum: 4294967295</p>
Maximum number of nodes per Browse Request	<p>Maximum number of <i>Nodes</i> that are taken into account in a <i>BrowseRequest</i>.</p> <p>Default:500</p> <p>Minimum: 1</p> <p>Maximum: 4294967295</p>
Maximum number of nodes per Translate Request	<p>Maximum number of <i>Nodes</i> that are taken into account in a <i>TranslateBrowsePathRequest</i>.</p> <p>Default:500</p> <p>Minimum: 1</p> <p>Maximum: 4294967295</p>
Maximum number of monitored items per Call	<p>Maximum number of <i>Nodes</i> that are included in a corresponding <i>Request</i> during the creation or deletion of Monitored Items.</p> <p>Default:500</p> <p>Minimum: 1</p> <p>Maximum: 4294967295</p>
consider "Server Operation Limits"	<p>Source of the configuration of the Operation Limits.</p> <ul style="list-style-type: none"> ▶ activated: at connection buildup the limits are read from the server. Operation Limits configured in this dialog are not considered. ▶ inactive: The limits are used as configured in this dialog. <p>Default: <i>inactive</i></p> <p>In doing so, the following applies:</p> <ul style="list-style-type: none"> ▶ At connection buildup to the server the limits are read from the server.

Parameter	Description
	<ul style="list-style-type: none"> ▶ If the server provides these limits and if the limits can be read successfully, they replace the configured Operation Limits. ▶ This applies to both, the variable import and the communication to the Runtime.

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

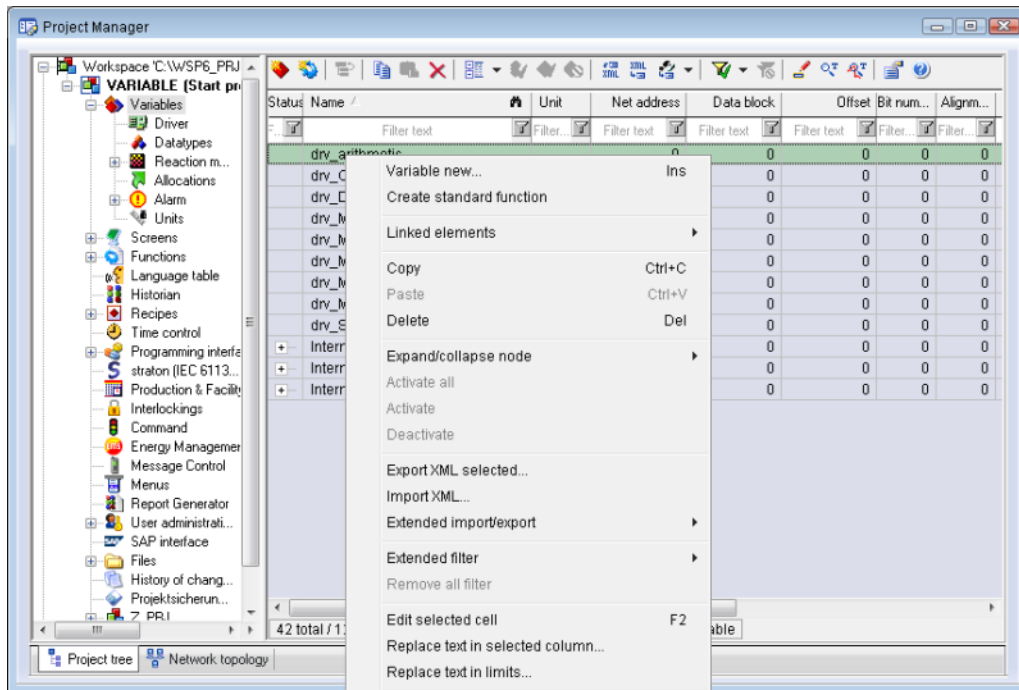
Variables can be created:

- ▶ 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:

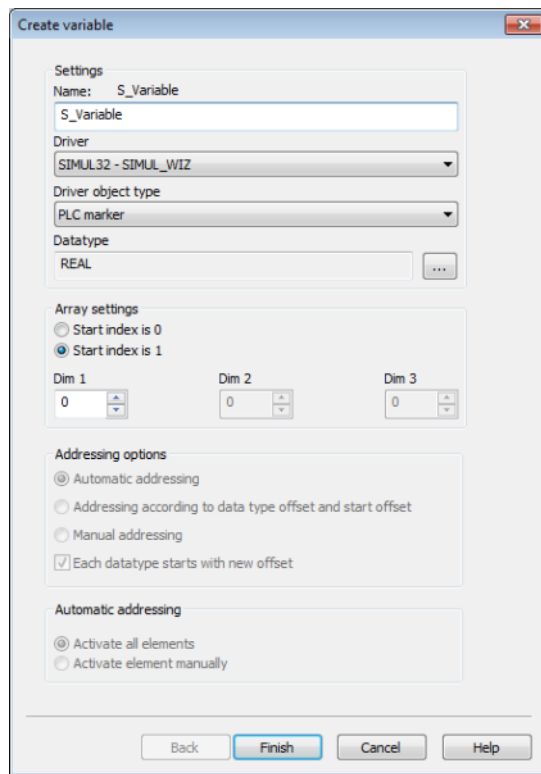
1. Select the **New variable** command in the **Variables** node in the context menu



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



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.

7.2 Addressing

You define the addressing of the variables in the property window:

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 variable. This address refers to the bus address in the connection configuration of the driver. This defines on which OPC UA Server the variable resides.
Data block	not used for this driver
Offset	Offset of variables. Equal to the memory address of the variable in the PLC. Adjustable from 0 to 4294967295.
Alignment	Alignment for variables with byte length of 1. You can choose between low byte and high byte.
Bit number	not used for this driver
String length	Only available for String variables. Maximum number of characters that the variable can take.
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.
Driver connection/Driver	Object type of the variables. Depending on the driver used, is selected

Group/Property	Description
Object Type	when the variable is created and can be changed here.
Driver connection/Data Type	<p>Data type of the variable. Is selected during the creation of the variable; the type can be changed here.</p> <p>Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.</p>
Browse name	<p>Equals the Browse name of the OPC UA specification. Only hierarchic references (forward) starting from the object folder are allowed. For example: <i>9:Data/9:Dynamic/9:Scalar/9:UInt32Value</i> or <i>Server/ServerStatus/StartTime</i>.</p> <p>The preceding number in the first example specifies the used name space index of the variable.</p> <p>Is automatically set during online import.</p>

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
PLC marker	8	X	X	<i>BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, STRING</i>	
<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 (not

Driver Object Type	Channel type	Read	Write	Supported data types	Description
					<p>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 45) chapter.</p>

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
OpcUa_Boolean	BOOL	8
OpcUa_Byte	USINT	9
OpcUa_SByte	SINT	10
OpcUa_UInt16	UINT	2
OpcUa_Int16	INT	1
OpcUa_UInt32	UDINT	4
OpcUa_Int32	DINT	3
OpcUa_UInt64	ULINT	27
OpcUa_Int64	LINT	26
OpcUa_Float	REAL	5

PLC	zenon	Data type
OpcUa_Double	LREAL	6
OpcUa_String	STRING	12
-	WSTRING	21
-	DATE	18
-	TIME	17
OpcUa_DateTime	DATE_AND_TIME	20
-	TOD (Time of Day)	19

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.

Attention: OpcUa_DateTime based on 100ns parts since 01.01.1601 00:00:00. The zenon data type DATE_AND_TIME based on Unix time (ms since 01.01.1970 00:00:00). OPC UA time stamp before 01.01.1970 00:00:00 is shown in zenon with the value "0".

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	Character	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

Identification	Type	Field size	Comment
			data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager
LES_SCHR	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-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

Identification	Type	Field size	Comment
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix
ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in project.ini .
ADJWVBA	R	1	Non-linear value adaption: 0: Non-linear value adaption is used 1: Non-linear value adaption is not used
ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
ZWREMA	N	16	Linked counter REMA.

Identification	Type	Field size	Comment
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

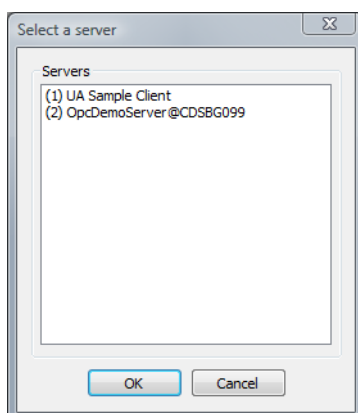
Identification	Type	Field size	Comment
			(if "-1" is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/Event Group
A_KLASSE1	N	10	Alarm/Event Class
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	C	66	Limit value text
A_DELAY1	N	10	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

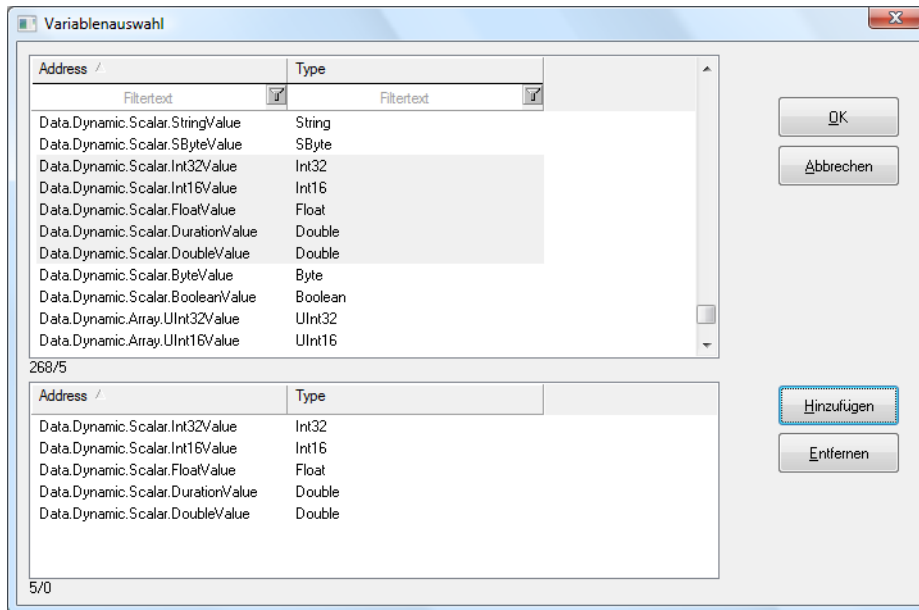
In order to create variables via the online import function:

1. select the driver in the detail view of the project manager
2. open the context menu with a right-click
3. click on **Import variables from driver**
4. now the dialog for the online import opens.



5. select the server from which you want to import the variables

6. after loading the available variables, they will be displayed



7. select the desired variables and add them to the zenon project via **Add** and **OK**



Information

When importing a large amount of variables, it is possible that not all variables have been imported correctly. In this case, you are informed by a corresponding message.

7.5 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.
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

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

Name from import	Type	Offset	Description
			Otherwise there is no communication with the corresponding controller.

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. 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

Name from import	Type	Offset	Description
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)
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.

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

ERROR MESSAGE

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

Name from import	Type	Offset	Description
RdErrBlockCount	<i>UINT</i>	44	Number of blocks to read when the last reading error occurred.
RdErrHwAdresse	<i>DINT</i>	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	<i>UDINT</i>	46	Block number when the last reading error occurred.
RdErrMarkerNo	<i>UDINT</i>	47	Marker number when the last reading error occurred.
RdErrSize	<i>UDINT</i>	48	Block size when the last reading error occurred.
DrvError	<i>USINT</i>	25	Error message as number
DrvErrorMsg	<i>STRING</i>	30	Error message as text
ErrorFile	<i>STRING</i>	15	Name of error log file

8 Driver-specific functions

The driver supports the following functions:

MAPPING SPECIAL STATUS

Mapping of NAN and NULL to the zenon status INVALID. If one of the statuses from the table is received, the status INVALID is set in zenon.

OPCUA	Value	NT_870	OV_870
<i>NAN</i>	<i>0</i>	<i>1</i>	<i>1</i>
<i>NULL</i>	<i>0</i>	<i>1</i>	<i>0</i>
<i>+INF</i>	<i>+MAXFLOAT</i>	<i>0</i>	<i>1</i>
<i>-INF</i>	<i>-MAXFLOAT</i>	<i>0</i>	<i>1</i>

8.1 DataChangeFilter

DATACHANGEFILTER CAN BE CONFIGURED:

- ▶ The **Data-Change trigger** property supports the "STATUS_VALUE_1" and "STATUS_VALUE_TIMESTAMP_2" status.
If STATUS_VALUE_TIMESTAMP_2 is configured, the driver includes a *DataChangeFilter* in the *CreateMonitoredItemRequest* and requests the server to send a *DataChangeNotification* accordingly if the time stamp has also changed. The driver sends an updated value in the Runtime if the time stamp changes a monitored element.
- ▶ The new **Use positive hysteresis of variable as AbsoluteDeadband** property causes the driver to include a *DataChangeFilter* in *CreateMonitoredItemRequest* with a correspondingly-configured *AbsoluteDeadband*. In the project configuration in the Editor, for the variable in **Positive for signal** property (**Value calculation** variable property group), there must be a value that does not equal 0 defined.

8.2 Configurable DataChangeTrigger

Up to zenon 7.60, no configurable *DataChangeTriggers* are supported.

For compatibility with older versions with zenon 7.60, the following is applicable:

- ▶ If no *DataChangeFilter* has been set, the **COT_4** and **COT_5** status bits are set to 0 in the Runtime when sending data.
- ▶ If existing *DataChangeFilter* variables are already set, in the event of a value change or a change to the status of the status bits **COT_4** and **COT_5** are set to 0 in the Runtime when sending data.



Information

Changes to the time stamp are also shown in the CEL as well as during zenon archiving.

8.3 Several subscriptions

The **OPCUA32 driver** normally creates a single **Subscription** for spontaneous communication in the zenon Runtime.

If the OPC UA server supports a limited number of *MonitoredItems* per **Subscription**, it can be necessary to split the variables to several subscriptions.

For this you can use the **Data block** property of the variable in property group **Addressing** in the zenon Editor. Per default the data block has the value *0*. As a result all variables of a **subscription** are added. For some variables of **Data block** projected with value *1*, the driver creates two **subscriptions**.



Attention

The project engineer is responsible for considering the maximum number of **subscriptions** supported by the server. If for the variables more **subscriptions** are projected than supported by the server, it may lead to impairment of the communication.

8.4 Writing of arrays

WRITING OF ARRAYS AS A COMPLETE ARRAY OR WITH INDIVIDUAL ARRAY ELEMENTS

- ▶ The **Access array variables on index range** driver option takes array elements into account when writing.
If the option is inactive, the **IndexRange** is set to "*Null*". This option is intended for OPC UA servers from third-party manufacturers, which do not support the writing of individual array elements.
Compatibility with OPC UA servers that do not support the optional "**Attribute Write Index**" (in the "**Attribute Services**" of the "**Core Server Facet**") is thus guaranteed. A requirement is that the **Permanently read variable** property is set to active for all array elements. The driver thus ensures that the current value from the complete array is present. When writing a variable from an array, the driver changes according to this element from the array and sends a *Write Request* with the complete array to the server. Writing fails if there is not yet a current value present for the complete array.
- ▶ If the **Access array variables on index range** option is active, the driver uses the *IndexRange* field and writes individual array elements as required.

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

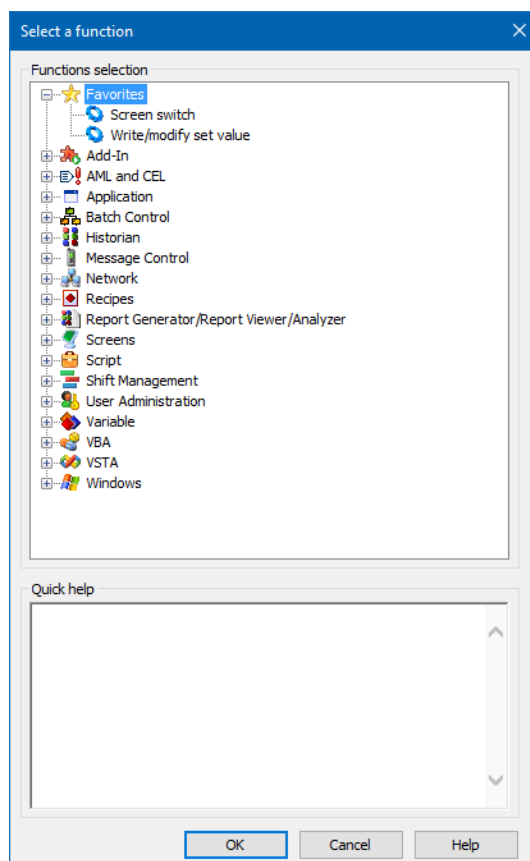
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

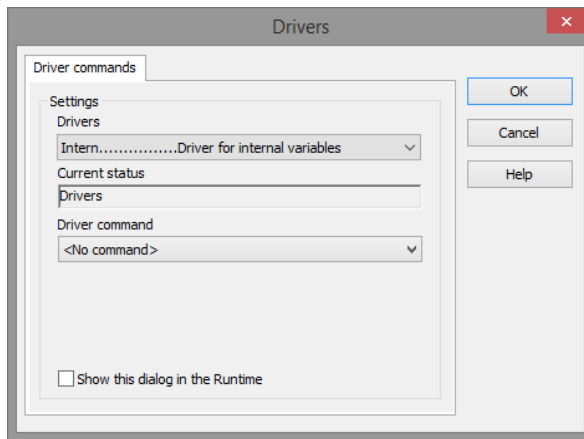
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



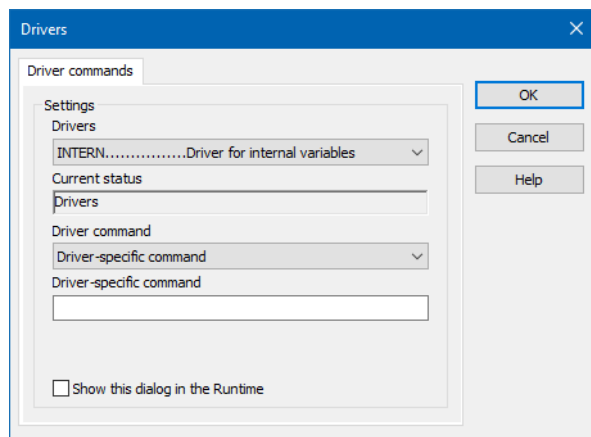
2. Navigate to the node **Variable**.
3. Select the **Driver commands** entry.

The dialog for configuration is opened



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

Option	Description
	the available driver commands section.
Driver-specific command	<p>Entry of a command specific to the selected driver.</p> <p>Note: Only available if, for the driver command option, the <i>driver-specific command</i> has been selected.</p>
Show this dialog in the Runtime	<p>Configuration of whether the configuration can be changed in the Runtime:</p> <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. <p>Default: <i>inactive</i></p>

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>	<p>No command is sent.</p> <p>A command that already exists can thus be removed from a configured function.</p>
<i>Start driver (online mode)</i>	<p>Driver is reinitialized and started.</p> <p>Note: If the driver has already been started, it must be stopped. Only then can the driver be re-initialized and started.</p>
<i>Stop driver (offline mode)</i>	<p>Driver is stopped. No new data is accepted.</p> <p>Note: If the driver is in offline mode, all variables that</p>

Driver command	Description
	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) 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.

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

QUESTIONS FOR LOCATING ERRORS

- ▶ Is the PLC connected to the power supply?
- ▶ Are the participants available in the TCP/IP network?
- ▶ Can the PLC be reached via the **Ping** command?
- ▶ Can the PLC be reached at the respective port via **TELNET**?
- ▶ Are the PLC and the PC connected with the right cable?
- ▶ Did you configure the net address correctly, both in the driver dialog and in the address properties of the variables?
- ▶ Did you use the right object type for the variable?
- ▶ Does the offset addressing of the variable match the one in the PLC?
- ▶ Analysis with the Diagnosis Viewer (on page 58): Which messages are displayed?
- ▶ Can you communicate with another OPC UA client?