COPADATA
do it your way

zenon driver manual
OPCUA32

v.7.50

COPADATA
do it your way

Contents

1. Welcome to COPA-DATA help

GENERAL HELP

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

PROJECT SUPPORT

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

LICENSES AND MODULES

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

2. OPCUA32

OPC UA stands for **OPC Unified Architecture**. The driver is used for communicating with OPC UA servers and is based on the official stack of the OPC foundation. Its main features:

▶ spontaneous communication, i.e. modified variables are automatically reported by the server,

▶ the driver supports several servers

▶ the variables can be read directly from the server

▶ variables are addressed via the **Browse Name** defined in the standard

3. OPCUA32 - Data sheet

General:	
Driver file name	OPCUA32.exe
Driver name	OPC UA Client driver
PLC types	All OPC-UA server with Data Access comunication
PLC manufacturer	OPC; straton; COPA-DATA;

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

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

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

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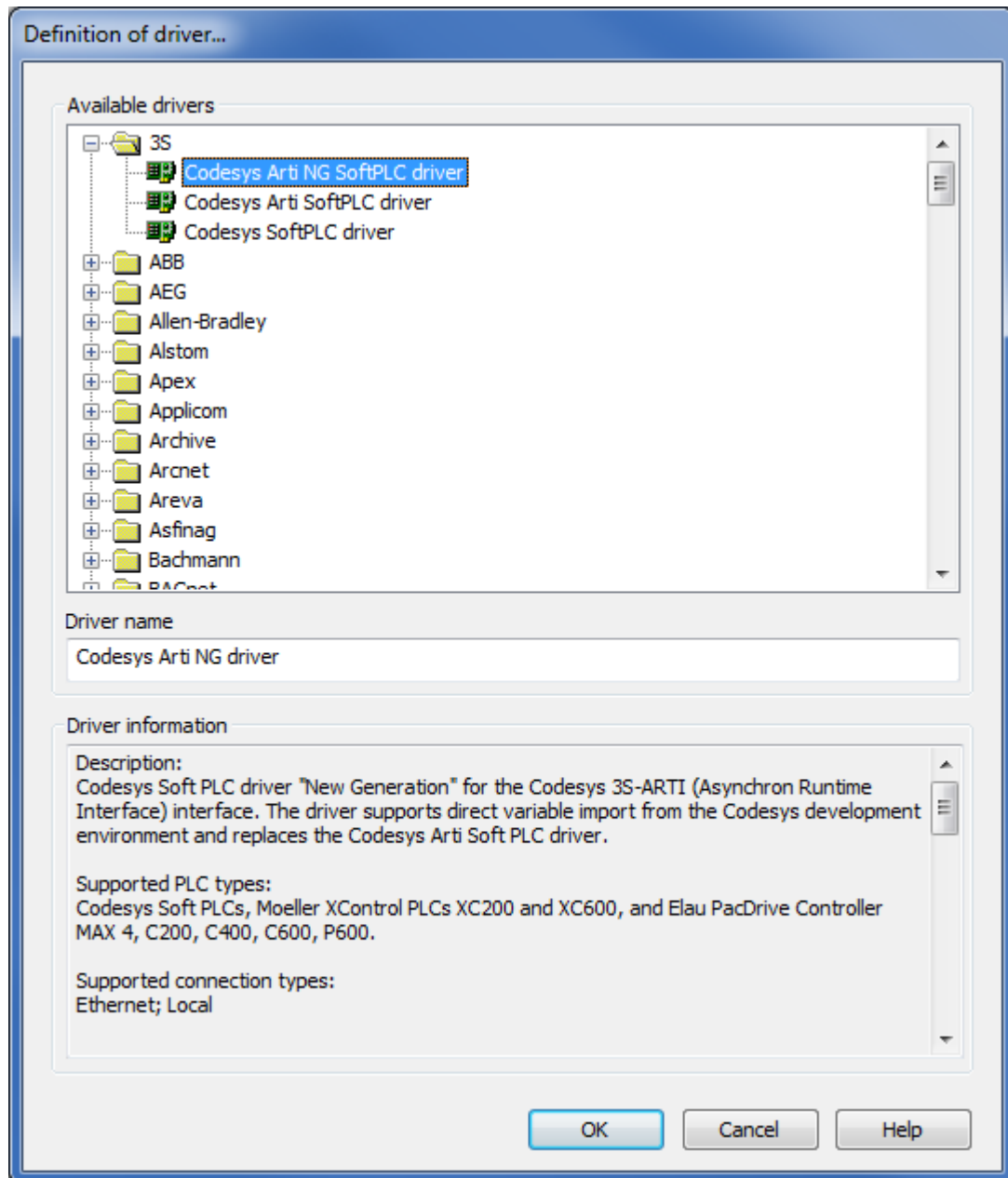
> **Information**
>
> *Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.*

6.1 Creating a driver

In order to create a new driver:

1. Right-click on **Driver** in the Project Manage and select **Driver new** in the context menu.

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

3. Select the desired driver and give it a name:

- The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.

- The driver name is part of the file name. Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (_).

- Attention: This name cannot be changed later on.

4. Confirm the dialog with **OK**. In the following dialog the single configurations of the drivers are defined.

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

💡 **Information**

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

▸ Internal

▸ MathDr32

▸ SysDrv.

▸

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

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.

Parameters	Description
Mode	Allows to switch between hardware mode and simulation mode ▶ Hardware: A connection to the control is established. ▶ Simulation static No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▶ Simulation - counting No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically. ▶ Simulation - programmed N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).
Keep update list in the memory	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	`Active`: Outputs can be written. `Inactive`: Writing of outputs is prevented. Note: Not available for every driver.
Variable image remanent	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp.

	The variable image is saved in mode hardware if:
	▸ one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active
	The variable image is always saved if:
	▸ the variable is of the object type **Driver variable**
	▸ the driver runs in simulation mode. (not programmed simulation)
	The following states are not restored at the start of the Runtime:
	▸ SELECT(8)
	▸ WR-ACK(40)
	▸ WR-SUC(41)
	The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.
Stop on Standby Server	Setting for redundancy at drivers which allow only on communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. Attention: If this option is active, the gapless archiving is no longer guaranteed. `Active`: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **switched off** (**statusverarbeitung.chm::/24150.htm**) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian. Note: Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.
Global Update time	`Active`: The set **Global update time** in `ms` is used for all variables in the project. The priority set at the variables is not used. `Inactive`: The set priorities are used for the individual variables.
Priority	The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time. The allocation to the variables takes place separately in the settings of the variable properties. The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities.

| | Thus the communication load is distributed better. |
| | Attention: Priority classes are not supported by each driver For example, drivers that communicate spontaneously do not support it. |

CLOSE DIALOG

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

UPDATE TIME FOR CYCLICAL DRIVERS

The following applies for cyclical drivers:

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

6.2.2   Connections

Configure the OPC UA connections to the PLCs in the Connections tab.

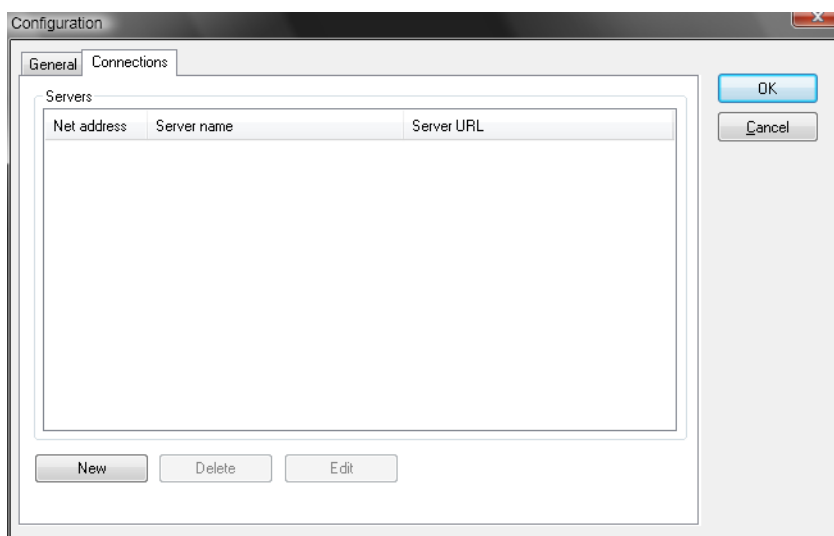| Parameters	Description
Servers	Contains the configured connections. Select a connection to delete or modify it.
Net adress	The net address identifies the connection. Every connection must have a unique net address, which are assigned automatically. Variables are assigned to a connection via the net 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. `opc.tcp://server:4840`
New	Opens the dialog for creating a new connection.
Delete	Deletes highlighted entry from the list.
Edit	Opens highlighted entry for editing.
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 16) tab

> ▶ Set the advanced optins undr **Avanced settings** (on page 18)

> ▶ Configure the certificates under **Certificates** (on page 21)

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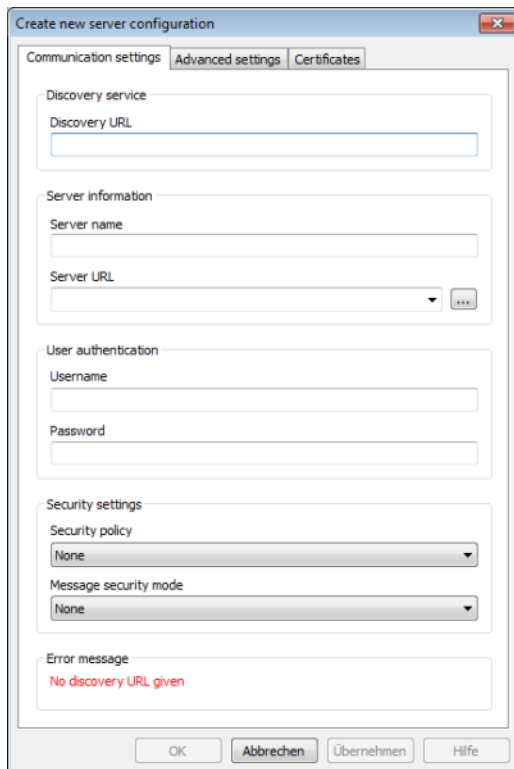
Select an existing connection in the connection dialog and click the **Edit** button to modify it. The properties for this are identical with the fields displayed when creating a new connection.

DELETE CONNECTION

To delete a connection:

> ▶ select the connection in the connection list

> ▶ click on the button **Delete**

> ▶ the connection will be removed

Communication settings

Parameters	Description
Discovery URL	The address of the discovery server, from which the information about existing servers can be requested. Example: **opc.tcp://192.168.0.1:4840**
Server name	Freely definable name for identifying the connection.
Server URL	The network address which is used to contact the connection terminal of the server. Select from drop-down list. Clicking on the **...** button starts Discovery via the **Discovery URL** entered. Example: **opc.tcp://192.168.0.1:4841** Example:**opc.tcp://PC1:4841/SimulationServer**
Username	Optional user name for auhentification via User Identity Token.
Password	Optional password for auhentification via User Identity Token.
Security Policy	Security algorithm used for the connection with the server. Look up part 7 of the OPC UA specification for allowed values. The discovery server provides all supported values when reading.
Message Security Mode	The **message security mode**specifies the security level that is used for the transmission of messages. Possible values: ▶ `None` ▶ `Sign` All messages are signed but not encrypted. ▶ `Sign & Encrypt` All messages are signed and encrypted.
Error Message	Notification about missing or incorrect details.
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all unsaved changes in all tabs and closes the dialog.
Apply	Saves all changes for this tab.
Help	Opens online help.
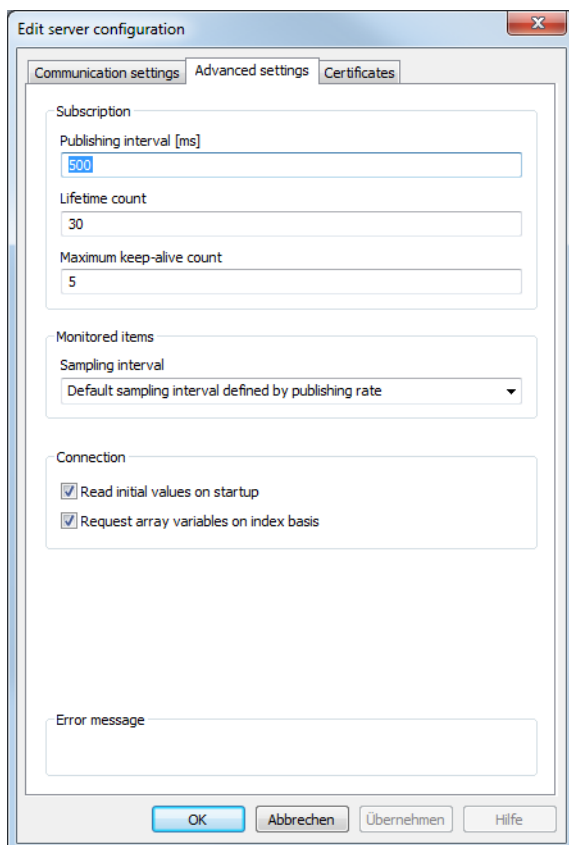
Attention: If **Sign** or **Sign & Encrypt** is used, the server certificate must be imported in the **Certificates** tab. You can find details on this authentication in the **Part 4 - 5.6.3 ActivateSession** norm.

Hint: Usually only the **Discovery URL** is filled out. The server names and URLs, as well as the allowed security settings, are read off from the discovery server by clicking on the **...** button next to the **Server URL** field.

Hint: If an incorrect syntax is used for the Discovery URL, the following error message appears: **"A value had an invalid Syntax (0x80b60000)"** after clicking on the **...** button

Hint: If "opc.tcp" is missing on the Discovery URL, the following error message appears: **"The requested operation is not supported (0x803d0000)"** after a click on the **...** button

Advanced settings

Parameters	Description
Subscription	Options for the connection.
Publishing interval [ms]	Defines the interval (in milliseconds) within which the server must send a message for the **subscription**. The server can also change this interval upwards in order to adhere to technical limitations. Default: `500`
Lifetime count	States how often the **publishing interval** can be exceeded before the **subscription** is deleted by the server. Values: `>=1 and >=3*`**Maximum keep-alive count** Default: `30`
Maximum keep-alive count	States how often the **publishing interval** can run out before a **keep-alive** message is sent. Values: `>= 1` Default: `5`
Monitored items	
Sampling interval	Sampling interval in milliseconds, in which the server must query **MonitoredItems**. 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: ▸ `Fastest practical rate (= 0)` ▸ `Default: sampling interval defined by publishing rate (=-1)` ▸ `Manual entry of values in milliseconds`
Connection	
Read initial values on startup	`Active`: In addition to signing in the variables (advise) a read request is also transferred. `Inactive`: No additional read request is sent to the server. The values may be received with a delay, depending on the server. Default: `active` Recommendation: Select this option if a large amount of variables are to communicate with the server.
Request array variables on index basis	Working with OPC UA array variables. In addition to the network address and addressing via 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) `Active`: Each array element is signed in as its own variable for a value assessment. IndexRange is used for CreateMonitoredItems.

	`Inactive`: The complete array is signed in for value changes. As a result of this, large arrays for which most or all array elements are needed sign in to the server more quickly. However in doing so - even for just a few value changes - the complete array is always transferred. The network load can be increased as a result. Default: `active` Recommendation: Select this option if only a few elements from large arrays are needed.
Error message	Notification about missing or incorrect details.
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all unsaved changes in all tabs and closes the dialog.
Apply	Saves all changes for this tab.
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.

Certificates

Use the tab **Certificates** to configure the certificates:
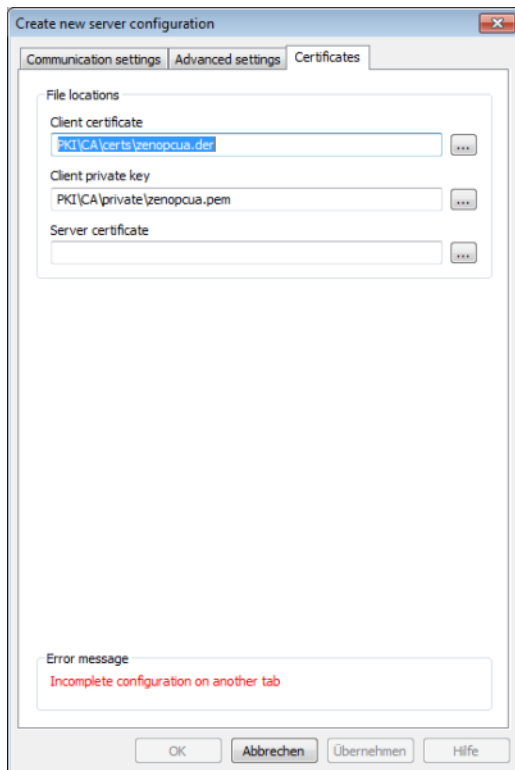
Parameters	Description
Client Certificate	Application instance certificate of the client. Used by the server for encrypting messages. This file must be in the project folder under `"..\Custom\Drivers\PKI\CA\certs\"`.
Client Private Key	Private key of the client. Used for decrypting server messages. Must be in the certificate path in the ..\Custom\Drivers\PKI\CA\certs\ subfolder.
Server Certificate	Application instance certificate with public server key. Used for encrypting messages that are sent to the server.
Error Message	Notification about missing or incorrect details.
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all unsaved changes in all tabs and closes the dialog.
Apply	Saves all changes for this tab.
Help	Opens online help.

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

The self-created certificate is valid for one year. The OPCUA client driver itself does not check the validity of its own certificate. A tolerant server also accepts, under certain circumstances, a certificate that has expired. For a server that carries out a more strict check, it is therefore advisable to create your own application instance certificates, with a longer period of validity.

It is also the case that each connection and each driver within a project uses the same application instance certificate by default because this is easier to manage. By definition, each OPCUA client should use a unique application instance certificate. If an OPCUA server requires it, a self-created certificate can also be configured for each connection in the driver or in the individual driver.

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.

X509 OPCUA application certificates can, for example, be created using the OPCUA Configuration Tool from the OPC Foundation or with the XCA software.

7. Creating variables

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

7.1 Creating variables in the Editor

Variables can be created:

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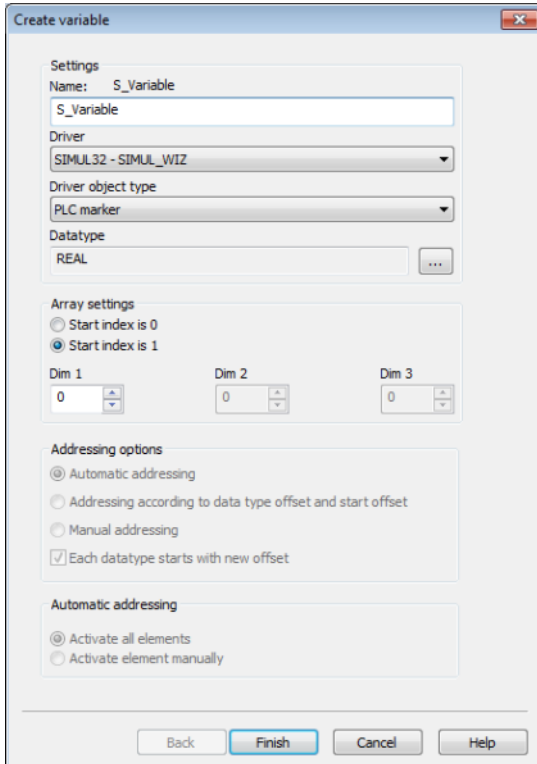- ▶ as structure variables (main.chm::/15278.htm)

VARIABLE DIALOG

To create a new variable, regardless of which type:

1. Select the **New variable** command in the **Variables** node in the context menu
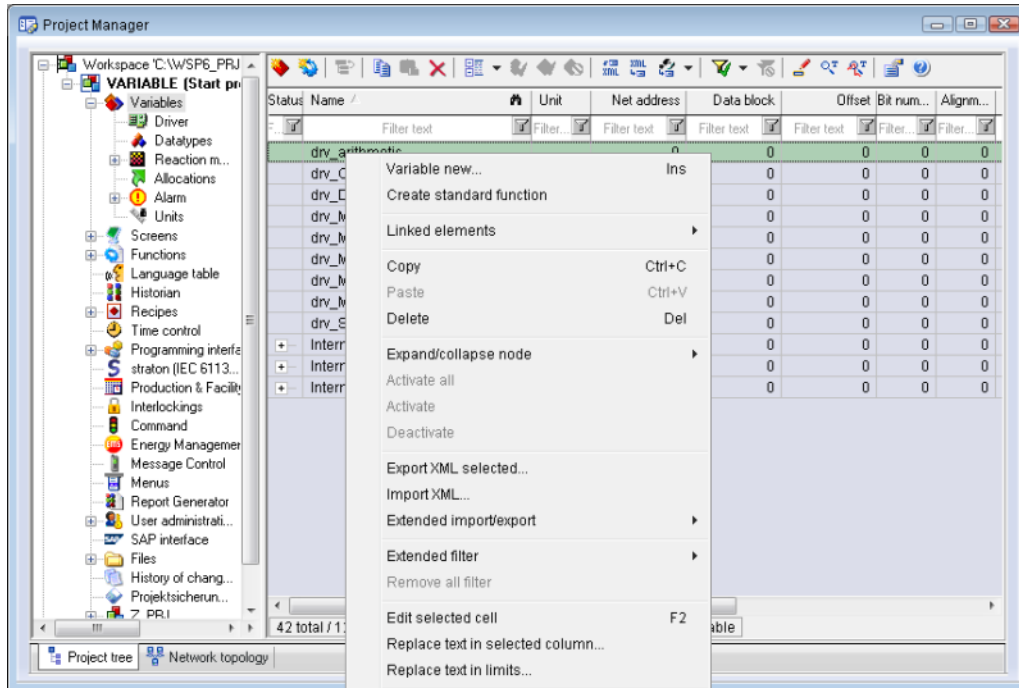
2. The dialog for configuring variables is opened

3. configure the variable

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

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 character Attention: The characters **#** and **@** are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive. Note: For some drivers, the addressing is possible over the property **Symbolic address**, as well.
Drivers	Select the desired driver from the drop-down list. Note: If no driver has been opened in the project, the driver for internal variables (**Intern.exe** (**Main.chm::/Intern.chm::/Intern.htm**)) is automatically loaded.
Driver object type (cti.chm::/28685.htm)	Select the appropriate driver object type from the drop-down list.
Data type	Select the desired data type. Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the Arrays chapter.
Addressing options	Expanded settings for arrays and structure variables. You can find details in the respective section.
Automatic element activation	Expanded settings for arrays and structure variables. You can find details in the respective section.

SYMBOLIC ADDRESS

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

Maximum length: `1024` characters.

INHERITANCE FROM DATA TYPE

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

▶ derived from the datatype

▶ Automatically adapted if the data type is changed

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

7.2   Addressing

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

Group/Property	Description
General	
Name	Freely definable name.
	Attention: The name must be unique for every zenon project.
Identification	Any text can be entered here, e.g. for resource labels, comments ...
Addressing	
Net address	This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.
Data block	not used for this driver
Offset	Offset of the variable; the memory address of the variable in the PLC. Adjustable from `0` to `4294967295`.
Alignment	Alignment for variables with byte length 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.
Driver connection/Driver Object Type	Depending on the employed driver, an object type is selected during the creation of the variable; the type can be changed here later.
Driver connection/Data Type	Data type of the variable, which is selected during the creation of the variable; the type can be changed here later.
	Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.
Browse name	Equals the **Browse name** of the OPC UA specification. Only hierarchic references (forward) starting from the object folder are allowed. For example: `9:Data/9:Dynamic/9:Scalar/9:UInt32Value` or `Server/ServerStatus/StartTime`.
	The preceding number in the first example specifies the used name space index of the variable.
Browse name	Is automatically set during online import.

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	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, STRING	
Driver variable	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. Find out more in the chapter about the Driver variables (on page 36)

7.3.2 Mapping of the data types

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

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

7.4   Creating variables by importing

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

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

7.4.1 XML import

For the import/export of variables the following is true:

- ▶ The import/export must not be started from the global project.
- ▶ The start takes place via:
  - Context menu of variables or data typ in the project tree
  - or context menu of a variable or a data type
  - or symbol in the symbol bar variables

> ⚠ **Attention**
>
> *When importing/overwriting an existing data type, all variables based on the existing data type are changed.*
>
> *Example:*
>
> *There is a data type XYZ derived from the type INTwith variables based on this data type. The XML file to be imported also contains a data type with the name XYZ but derived from type STRING. If this data type is imported, the existing data type is overwritten and the type of all variables based on it is adjusted. I.e. the variables are now no longer INT variables, but STRING variables.*

7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

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

IMPORT DBF FILE

To start the import:

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

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

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

EXPORT DBF FILE

To start the export:

1. right-click on the variable list

2. in the drop-down list of **Extended export/import...** select the **Export dBase...** command

3. follow the export assistant

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

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

> 💡 **Information**
>
> *dBase does not support structures or arrays (complex variables) at export.*

File structure of the dBase export file

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

> ⚠ **Attention**
>
> dBase does not support structures or arrays (complex variables) at export.
>
> DBF files must:
>
> ▸ conform with their name to the 8.3 DOS format (8 alphanumeric characters for name, 3 characters for extension, no space)
>
> ▸ Be stored close to the root directory   (Root)

STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	Char	128	Variable name.
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (field/column must be entered manually).
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
KANAL_D	Log	1	The variable is deleted with the `1` entry (field/column has to be created by hand).
TAGNR	C	128	Identification.
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Bus 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	L	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTTYP and DATENTYP
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	L	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	L	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	L	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	L	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	L	1	Set blink attribute
BTB1	L	1	Logging in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Printer output (for CEL or Alarm)
QUITTIER1	L	1	Must be acknowledged
LOESCHE1	L	1	Must be deleted
VARIABLE1	L	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	L	1	Functions linking
ASK_FUNC1	L	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	L	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.
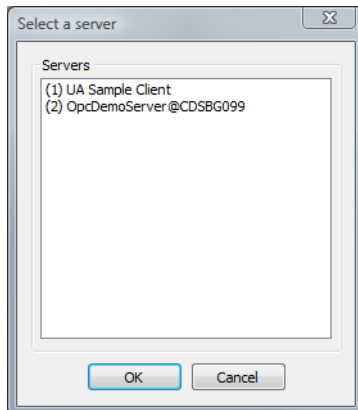
5. select the server from which you want to import the variables

6. after loading the available variables, they will be displayed
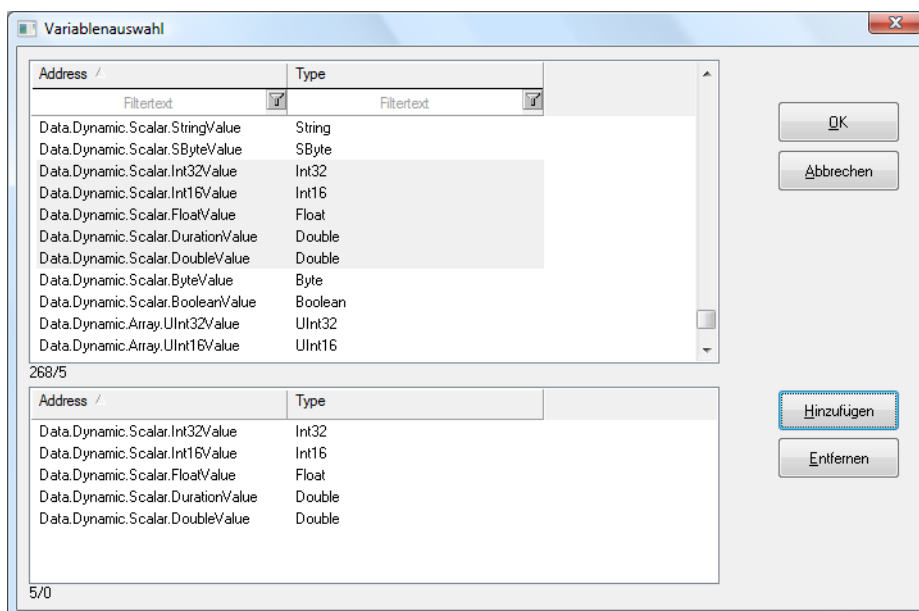
7. select the desired variables and add them to the zenon project via **Add** and **OK**

💡 **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 Driver variables

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

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

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

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

> **Information**
>
> *Not every driver supports all driver variants.*
>
> *For example:*
>
> ▸ Variables for modem information are only supported by modem-compatible drivers
>
> ▸ Driver variables for the polling cycle only for pure polling drivers
>
> ▸ Connection-related information such as ErrorMSG only for drivers that only edit one connection at a a time

INFORMATION

Name from import	Type	Offset	Description
MainVersion	UINT	0	Main version number of the driver.
SubVersion	UINT	1	Sub version number of the driver.
BuildVersion	UINT	29	Build version number of the driver.
RTMajor	UINT	49	zenon main version number
RTMinor	UINT	50	zenon sub version number
RTSp	UINT	51	zenon Service Pack number
RTBuild	UINT	52	zenon build number
LineStateIdle	BOOL	24.0	TRUE, if the modem connection is idle
LineStateOffering	BOOL	24.1	TRUE, if a call is received
LineStateAccepted	BOOL	24.2	The call is accepted
LineStateDialtone	BOOL	24.3	Dialtone recognized
LineStateDialing	BOOL	24.4	Dialing active
LineStateRingBack	BOOL	24.5	While establishing the connection
LineStateBusy	BOOL	24.6	Target station is busy

LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
LineStateProceeding	BOOL	24.9	Dialing completed
LineStateOnHold	BOOL	24.10	Connection in hold
LineStateConferenced	BOOL	24.11	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	BOOL	24.13	Connection in hold for transfer
LineStateDisconnected	BOOL	24.14	Connection terminated.
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped For `driver stop`, the variable has the value `TRUE` and an **OFF** bit. After the driver has started, the variable has the value `FALSE` and no **OFF** bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

CONFIGURATION

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

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

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

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	UDINT	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	UDINT	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	UDINT	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	UDINT	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	UDINT	33	Number of writing errors
ReadSucceedCount	UDINT	35	Number of successful reading attempts

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

ERROR MESSAGE

Name from import	Type	Offset	Description
ErrorTimeDW	UDINT	2	Time (in seconds since 1.1.1970), when the last error occurred.
ErrorTimeS	STRING	2	Time (in seconds since 1.1.1970), when the last error occurred.
RdErrPrimObj	UDINT	42	Number of the PrimObject, when the last reading error occurred.
RdErrStationsName	STRING	43	Name of the station, when the last reading error occurred.
RdErrBlockCount	UINT	44	Number of blocks to read when the last reading error occurred.

RdErrHwAdresse	DINT	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	UDINT	46	Block number when the last reading error occurred.
RdErrMarkerNo	UDINT	47	Marker number when the last reading error occurred.
RdErrSize	UDINT	48	Block size when the last reading error occurred.
DrvError	USINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

8. Driver-specific functions

The driver supports the following functions:

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
NAN	0	1	1
NULL	0	1	0
+INF	+MAXFLOAT	0	1
-INF	-MAXFLOAT	0	1

9. Driver commands

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

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

▶ create a new function

▶ select Variables -> Driver commands

▶ The dialog for configuration is opened
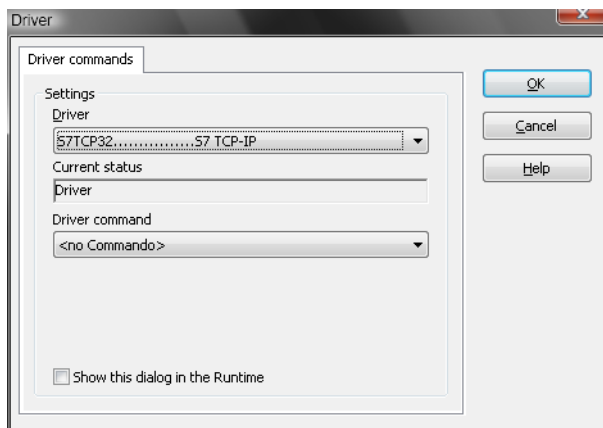
Parameter	Description
Drivers	Drop-down list with all drivers which are loaded in the project.
Current status	Fixed entry which has no function in the current version.
Driver command	Drop-down list for the selection of the command.
▸ `Start driver (online mode)`	Driver is reinitialized and started.
▸ `Stop driver (offline mode)`	Driver is stopped. No new data is accepted. Note: If the driver is in offline mode, all variables that were created for this driver receive the status `switched off` (OFF; Bit 20).
▸ `Driver in simulation mode`	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▸ `Driver in hardware mode`	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▸ `Driver-specific command`	Enter driver-specific commands. Opens input field in order to enter a command.
▸ `Driver - activate set setpoint value`	Write set value to a driver is allowed.
▸ `Driver - deactivate set setpoint value`	Write set value to a driver is prohibited.
▸ `Establish connecton with modem`	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
▸ `Disconnect from modem`	Terminate connection (for modem drivers)
Show this dialog in the Runtime	The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

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

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

10. Error analysis

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

10.1   Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under Start/All programs/zenon/Tools 7.50 -> Diagviewer.

zenon driver log all errors in the LOG files. The default folder for the LOG files is subfolder **LOG** in directory `ProgramData`, example:

`%ProgramData%\COPA-DATA\LOG`. LOG files are text files with a special structure.

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

In the Diagnosis Viewer you can also:

▶   Follow 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 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 45): Which messages are displayed?
- ▶ Can you communicate with another OPC UA client?