zenon

by COPA-DATA

zenon driver manual

SqlDrv

v.8.20

Contents

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 customer service team, which 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 SqlDrv

The driver sqldrv.exe serves to connect data to an SQL database via an ODBC driver.

3 SQLDRV - data sheet

General:	
Driver file name	SQLDRV.exe

General:	
Driver name	SQL driver
PLC types	ODBC-compatible SQL database
PLC manufacturer	SQL; Database; COPA-DATA

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

Requirements:	
Hardware PC	--

Requirements:	
Software PC	ODBC driver for SQL database
Hardware PLC	--
Software PLC	--
Requires v-dll	--

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

4 Driver history

Date	Driver version	Change
07.07.08	1300	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

The following applies for setup and operation of the SqlDrv driver:

- The driver **sqldrv.exe** must be present in the zenon directory
- ODBC database (on page 24) must be defined
- SQL database (on page 24) must be defined
- Tables RECEIVE and SEND (on page 8) must be present

6 SQL database:

In the SQL database, two tables for data exchange with zenon have to be created:

- *RECEIVE*: zenon receives data from this table
- *SEND*: This table receives data from zenon

Both have the same format. The database and the names of the tables can be configured.

THE DEFINED FORMAT OF THE TABLES:

Name	Data type	Use
ID	*LINT*	ID with autoincrement (is not set when pasting).
NAME	*STRING*	Name of the variable. Must be exactly as in the variable list (on page 25).
DATUMZEIT	*DATE/TIME*	Time stamp without milliseconds. Format: *DD.MM.YYYY HH:mm:ss* If the value *1970-01-01 00:00:00* is used, the driver sends the

Name	Data type	Use
		current date and the current time to Runtime.
ZEIT_MS	*INT*	Milliseconds for the time stamp.
WERT	*STRING*	Value of the variable (format depends on data type).
STATUS	*INT*	Represents the 32 status bits of the variables as a decimal figure. For example, *131072* corresponds to the status **SPONTANEOUS (bit 17)** ▸ *0*: driver sends *SPONTAN* status in the Runtime. **Note:** Must not be empty.

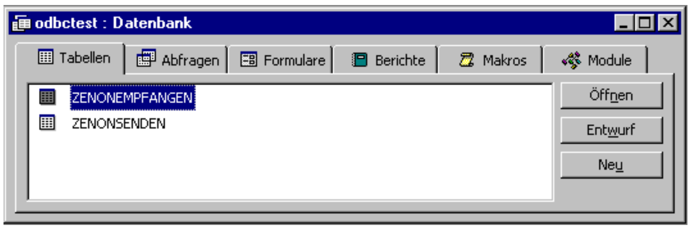
EXPANSION FOR REDUNDANT OPERATION (ONLY CONCERNS RECEIPT TABLE):

ACK_SRV	*INT*	Is set to 1 if the server has read
ACK_SB	*INT*	Is set to 1 if the standby server has read.
INSERTZEIT	*DATE/TIME*	Default: *Now()*, time of insertion of the line.

6.1 Examples

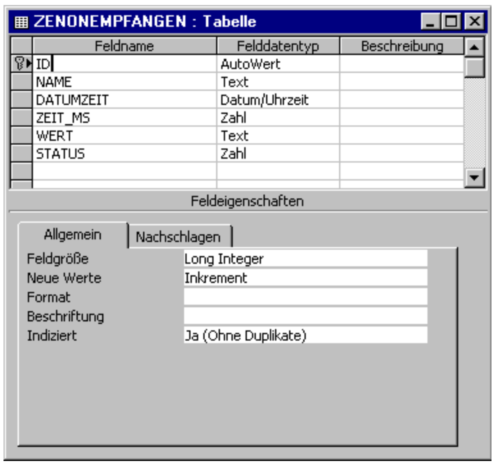
1ST ACCESS

- Table definition



ZENONEMPFANGEN
ZENONSENDEN

- Creation of receive table and send table



ZENONEMPFANGEN : Tabelle

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	
NAME	Text	
DATUMZEIT	Datum/Uhrzeit	
ZEIT_MS	Zahl	
WERT	Text	
STATUS	Zahl	

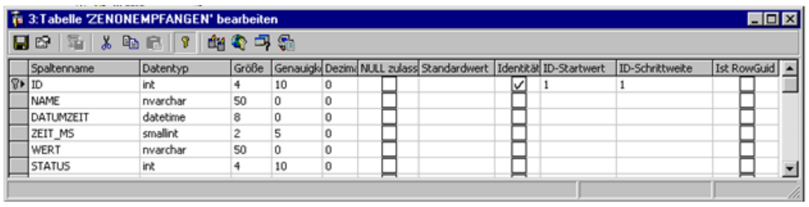
Feldeigenschaften

Allgemein | Nachschlagen

Feldgröße	Long Integer
Neue Werte	Inkrement
Format	
Beschriftung	
Indiziert	Ja (Ohne Duplikate)

2ND MS SQL SERVER

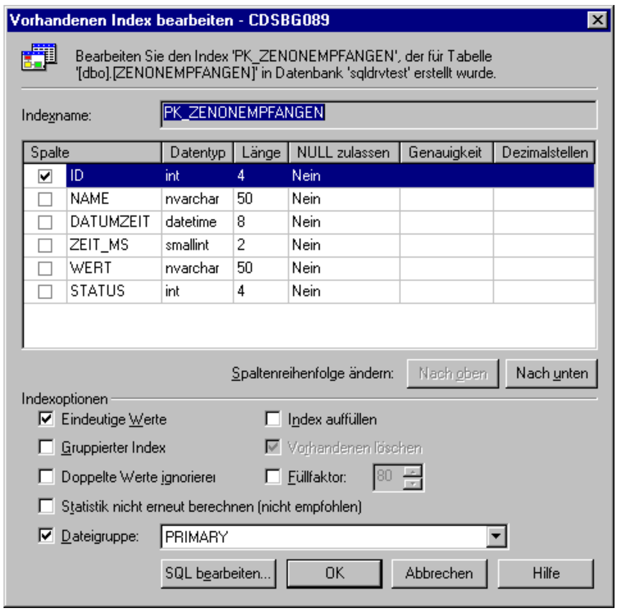
1. Create table
2. Format of the receive table



3:Tabelle 'ZENONEMPFANGEN' bearbeiten

Spaltenname	Datentyp	Größe	Genauigk	Dezima	NULL zulass	Standardwert	Identität	ID-Startwert	ID-Schrittweite	Ist RowGuid
ID	int	4	10	0	☐		✓	1	1	☐
NAME	nvarchar	50	0	0	☐		☐			☐
DATUMZEIT	datetime	8	0	0	☐		☐			☐
ZEIT_MS	smallint	2	5	0	☐		☐			☐
WERT	nvarchar	50	0	0	☐		☐			☐
STATUS	int	4	10	0	☐		☐			☐

3. Settings for the index of the receive table



Vorhandenen Index bearbeiten - CDSBG089

Bearbeiten Sie den Index 'PK_ZENONEMPFANGEN', der für Tabelle '[dbo].[ZENONEMPFANGEN]' in Datenbank 'sqldrvtest' erstellt wurde.

Indexname: PK_ZENONEMPFANGEN

Spalte		Datentyp	Länge	NULL zulassen	Genauigkeit	Dezimalstellen
✓	ID	int	4	Nein		
☐	NAME	nvarchar	50	Nein		
☐	DATUMZEIT	datetime	8	Nein		
☐	ZEIT_MS	smallint	2	Nein		
☐	WERT	nvarchar	50	Nein		
☐	STATUS	int	4	Nein		

Spaltenreihenfolge ändern: Nach oben | Nach unten

Indexoptionen
- ✓ Eindeutige Werte
- ☐ Index auffüllen
- ☐ Gruppierter Index
- ✓ Vorhandenen löschen
- ☐ Doppelte Werte ignorieren
- ☐ Füllfaktor: 80
- ☐ Statistik nicht erneut berechnen (nicht empfohlen)
- ✓ Dateigruppe: PRIMARY

SQL bearbeiten... | OK | Abbrechen | Hilfe

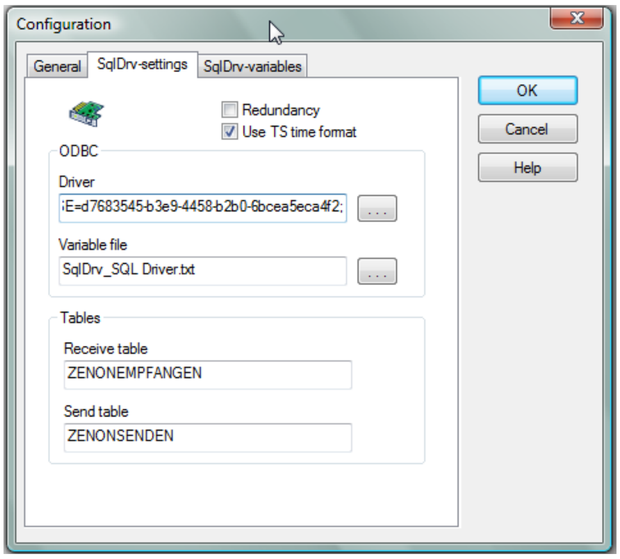
4. Similar to creating a table for SENDING.

If the database does not support the needed date format, the following definition can be used.

Parameter	Type	Description
DATUMZEIT	Text	Time stamp without milliseconds

If the database supports the operating system's standard date, the formatting can be changed under **START -> Control panel-> Region and language-> Formats**.

3RD MICROSOFT SQL SERVER MANAGEMENT STUDIO EXPRESS


Configuration

General | SqlDrv-settings | SqlDrv-variables

☐ Redundancy
☑ Use TS time format

ODBC

Driver: ;E=d7683545-b3e9-4458-b2b0-6bcea5eca4f2;

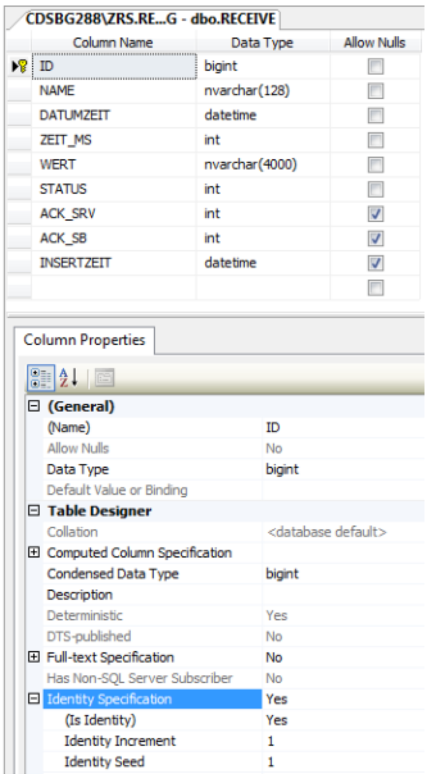
Variable file: SqlDrv_SQL Driver.txt

Tables

Receive table: ZENONEMPFANGEN

Send table: ZENONSENDEN

EMPFANGEN



CDSBG288\ZRS.RE...G - dbo.RECEIVE

Column Name	Data Type	Allow Nulls
ID	bigint	☐
NAME	nvarchar(128)	☐
DATUMZEIT	datetime	☐
ZEIT_MS	int	☐
WERT	nvarchar(4000)	☐
STATUS	int	☐
ACK_SRV	int	✓
ACK_SB	int	✓
INSERTZEIT	datetime	✓
		☐

Column Properties

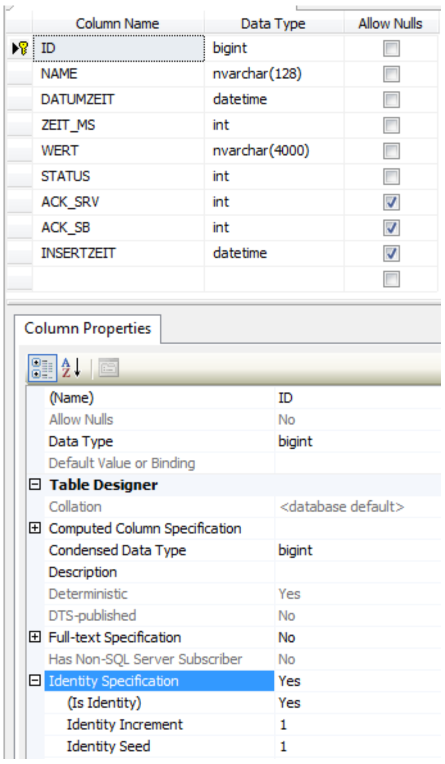
(General)	
(Name)	ID
Allow Nulls	No
Data Type	bigint
Default Value or Binding	
Table Designer	
Collation	<database default>
Computed Column Specification	
Condensed Data Type	bigint
Description	
Deterministic	Yes
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

If this query is used in SQL Management Studio to create the table, the *database name* must be entered in "**USE[]**" .

Query:

```
USE [Please put in the database name (e.g. project GUID) here]

GO

/****** Object: Table [dbo].[SEND] Script Date: 06/18/2011 15:52:29 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[SEND](

[ID] [bigint] IDENTITY(1,1) NOT NULL,

[NAME] [nvarchar](128) NOT NULL,

[DATUMZEIT] [datetime] NOT NULL,

[ZEIT_MS] [int] NOT NULL,

[WERT] [nvarchar](4000) NOT NULL,

[STATUS] [int] NOT NULL,

CONSTRAINT [PK_SEND] PRIMARY KEY CLUSTERED

([ID] ASC )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY] )
ON [PRIMARY]
```

SEND



	Column Name	Data Type	Allow Nulls
	ID	bigint	☐
	NAME	nvarchar(128)	☐
	DATUMZEIT	datetime	☐
	ZEIT_MS	int	☐
	WERT	nvarchar(4000)	☐
	STATUS	int	☐
	ACK_SRV	int	✓
	ACK_SB	int	✓
	INSERTZEIT	datetime	✓
			☐

Column Properties

(Name)	ID
Allow Nulls	No
Data Type	bigint
Default Value or Binding	
Table Designer	
Collation	<database default>
Computed Column Specification	
Condensed Data Type	bigint
Description	
Deterministic	Yes
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

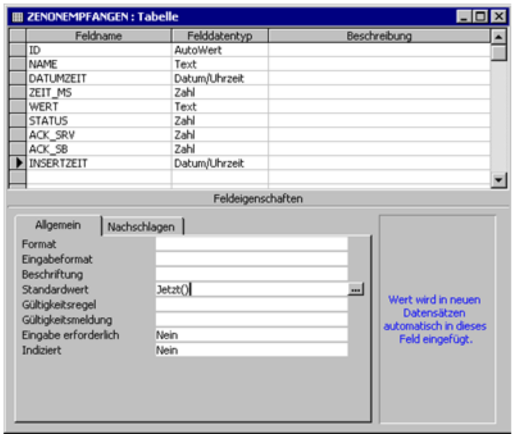
If this query is used in SQL Management Studio to create the table, the *database name* must be entered in "**USE[]**" .

Query:

```
USE [Please put in the database name (e.g. project GUID) here]
GO
/****** Object: Table [dbo].[RECEIVE] Script Date: 06/18/2011 15:52:29 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[RECEIVE](
[ID] [bigint] IDENTITY(1,1) NOT NULL,
[NAME] [nvarchar](128) NOT NULL,
[DATUMZEIT] [datetime] NOT NULL,
[ZEIT_MS] [int] NOT NULL,
[WERT] [nvarchar](4000) NOT NULL,
[STATUS] [int] NOT NULL,
[ACK_SRV] [int] NULL,

[ACK_SB] [int] NULL,

[INSERTZEIT] [datetime] NULL,

CONSTRAINT [PK_RECEIVE] PRIMARY KEY CLUSTERED

([ID] ASC )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)

ON [PRIMARY] )

ON [PRIMARY]
```

REDUNDANCY

If the driver is used in redundant operation, the following settings must be considered:


ZENONEMPFANGEN : Tabelle

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	
NAME	Text	
DATUMZEIT	Datum/Uhrzeit	
ZEIT_MS	Zahl	
WERT	Text	
STATUS	Zahl	
ACK_SRV	Zahl	
ACK_SB	Zahl	
INSERTZEIT	Datum/Uhrzeit	

Feldeigenschaften

Allgemein | Nachschlagen

Format	
Eingabeformat	
Beschriftung	
Standardwert	Jetzt()
Gültigkeitsregel	
Gültigkeitsmeldung	
Eingabe erforderlich	Nein
Indiziert	Nein

Wert wird in neuen Datensätzen automatisch in dieses Feld eingefügt.

The following additional entries in the receive table are required:

Parameter	Type	Description
ACK_SRV	Number	Is set to 1 if the configured server has read
ACK_SB	Number	Is set to 1 if the configured standby has read
INSERTZEIT	Date/Time	Default: *Now()*, time the line was inserted.

Additional settings:

- The **Redundancy** field must be activated.
- The variable file **SQLDRV.txt** must be on the server and on the standby server. (to be specified separately for Remote Transport)
- You must make sure that the database path is the same for both server and standby server, especially when using Access databases. The same applies for the DNS file (e.g.: path statement with computer names).
- In redundancy operation, a data set is then deleted if

- either **ACK_SRV** or **ACK_SB** are *1*
- or if **INSERTZEIT** is older than *5* minutes

7 Configuration

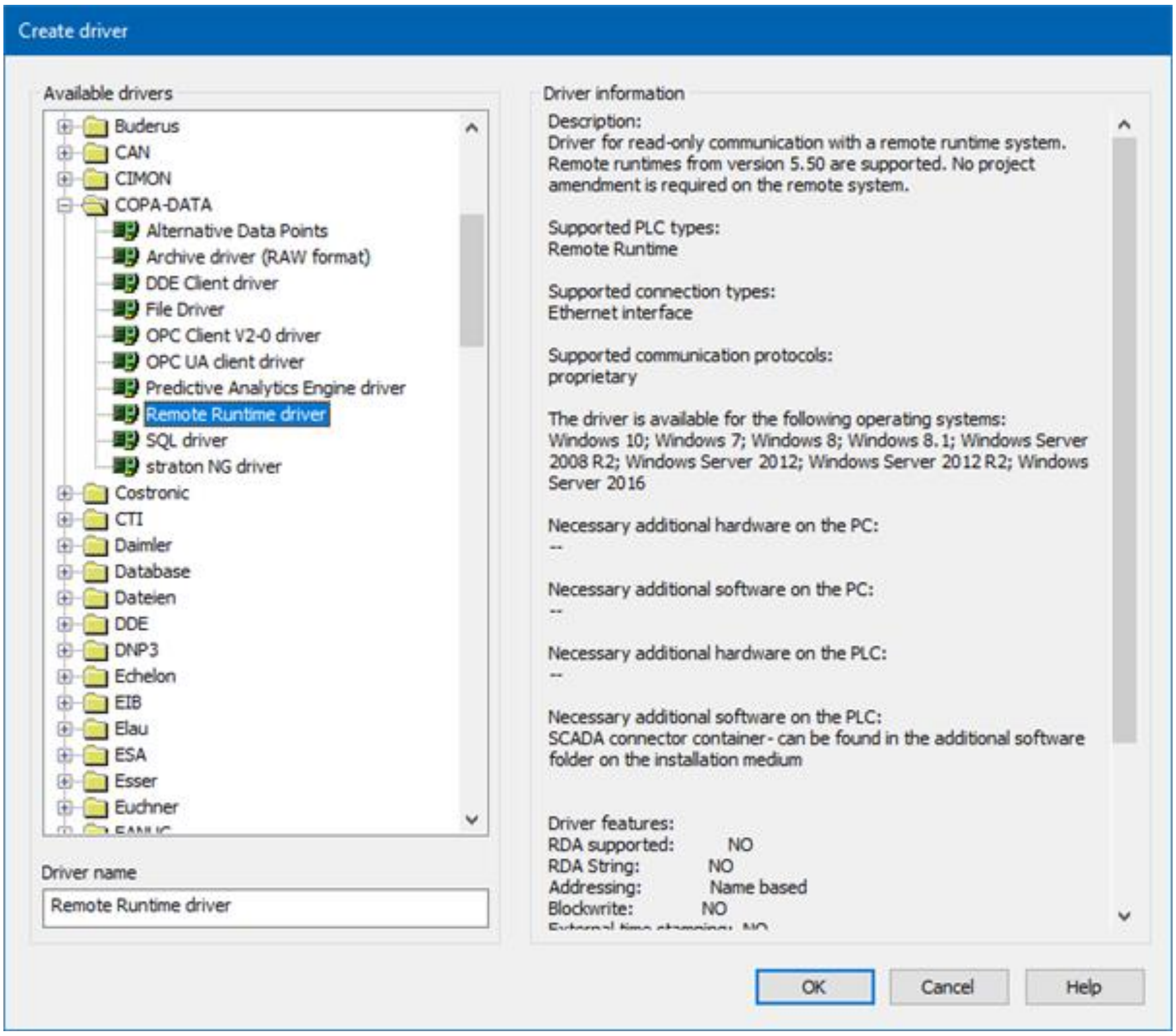
In this chapter you will learn how to use the driver in a project and which settings you can change.

Information

Find out more about further settings for zenon variables in the chapter Variables of the online manual.

7.1 Creating a driver

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



Driver name: Remote Runtime driver

Description:
Driver for read-only communication with a remote runtime system. Remote runtimes from version 5.50 are supported. No project amendment is required on the remote system.

Supported PLC types:
Remote Runtime

Supported connection types:
Ethernet interface

Supported communication protocols:
proprietary

The driver is available for the following 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

Necessary additional hardware on the PC:
--

Necessary additional software on the PC:
--

Necessary additional hardware on the PLC:
--

Necessary additional software on the PLC:
SCADA connector container - can be found in the additional software folder on the installation medium

Driver features:
RDA supported: NO
RDA String: NO
Addressing: Name based
Blockwrite: NO

Parameter	Description
Available drivers	List of all available drivers. The display is in a tree structure: *[+]* expands the folder structure and shows the drivers contained therein. [-] reduces the folder structure Default: *No selection*
Driver name	Unique **Identification** of the driver. Default: *empty* The input field is pre-filled with the pre-defined

Parameter	Description
	Identification after selecting a driver from the list of available drivers.
Driver information	Further information on the selected driver. Default: *empty* The information on the selected driver is shown in this area after selecting a driver.

CLOSE DIALOG

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

Information

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

CREATE NEW DRIVER

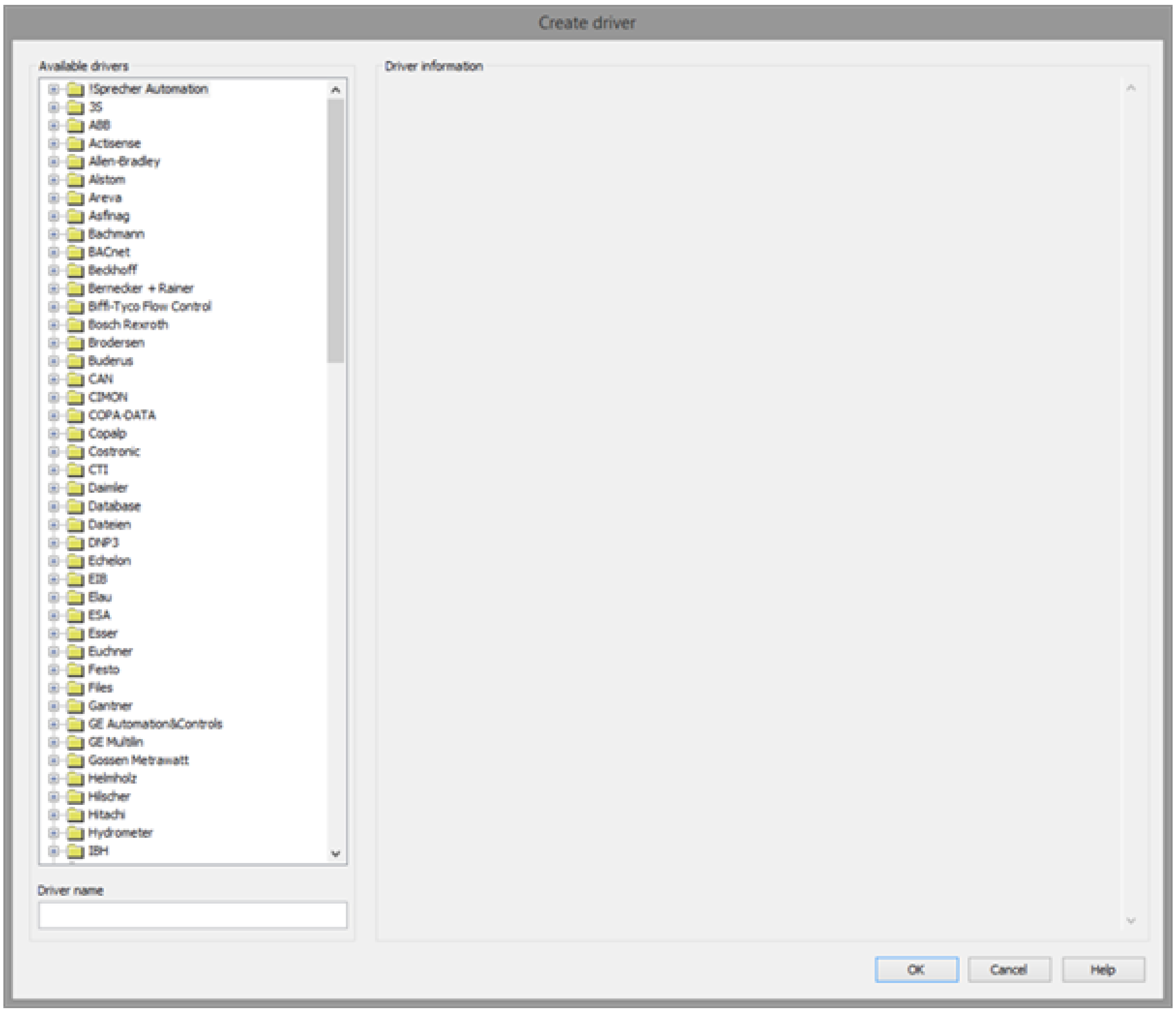
In order to create a new driver:

1. Right-click on **Driver** in the Project Manager and select **New driver** in the context menu.

   Optional: Select the **New driver** button from the toolbar of the detail view of the **Variables**.The Create driver dialog is opened.

   The **Create simple data type** dialog is opened.

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



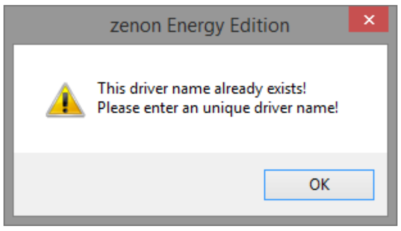
!Sprecher Automation
3S
ABB
Actisense
Allen-Bradley
Alstom
Areva
Asfinag
Bachmann
BACnet
Beckhoff
Bernecker + Rainer
Biffi-Tyco Flow Control
Bosch Rexroth
Brodersen
Buderus
CAN
CIMON
COPA-DATA
Copalp
Costronic
CTI
Daimler
Database
Dateien
DNP3
Echelon
EIB
Elau
ESA
Esser
Euchner
Festo
Files
Gantner
GE Automation&Controls
GE Multilin
Gossen Metrawatt
Helmholz
Hilscher
Hitachi
Hydrometer
IBH

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

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

DRIVER NAME DIALOG ALREADY EXISTS

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



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

ZENON PROJECT

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

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

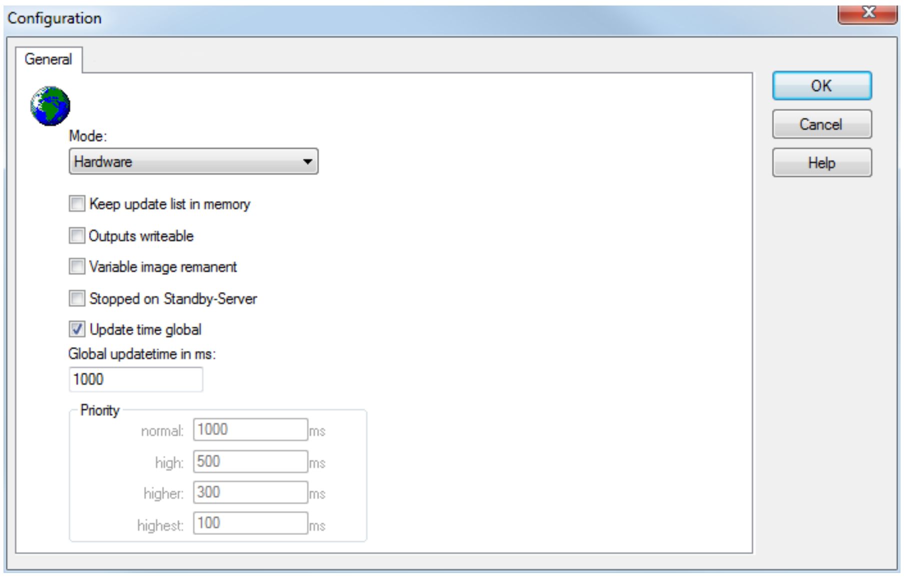
> **Information**
>
> Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

7.2 Settings in the driver dialog

You can change the following settings of the driver:

7.2.1 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.



Mode: Hardware

- [ ] Keep update list in memory
- [ ] Outputs writeable
- [ ] Variable image remanent
- [ ] Stopped on Standby-Server
- [x] Update time global

Global updatetime in ms: 1000

Priority

normal:	1000	ms
high:	500	ms
higher:	300	ms
highest:	100	ms

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

Option	Description
	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.
Keep update list in the memory	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	‣ *Active*: Outputs can be written. ‣ *Inactive*: Writing of outputs is prevented. **Note**: Not available for every driver.
Variable image remanent	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in hardware mode if one of these statuses is active: ‣ User status *M1* (*0*) to *M8* (*7*) ‣ *REVISION(9)* ‣ *AUS(20)* ‣ *ERSATZWERT(27)* The variable image is always saved if: ‣ the variable is of the **Communication details** object type ‣ the driver runs in simulation mode. (not programmed simulation) The following states are not restored at the start of the Runtime:

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

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

CLOSE DIALOG

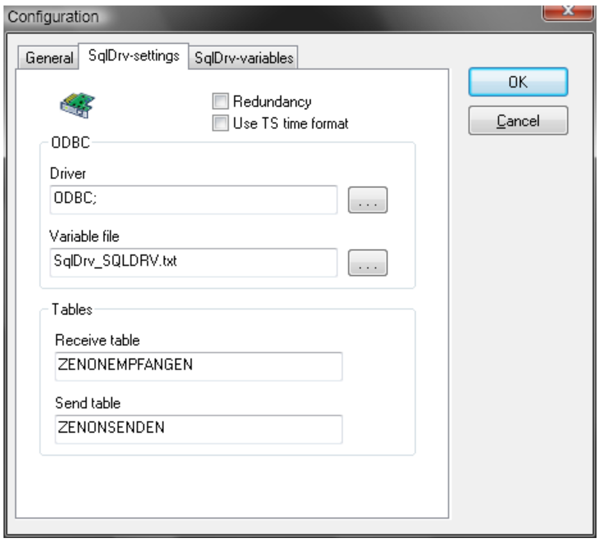
Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR SPONTANEOUS DRIVERS

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

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

7.2.2 Driver dialog SqlDrv_Einstellungen



Redundancy: unchecked
Use TS time format: unchecked

ODBC
Driver: ODBC;
Variable file: SqlDrv_SQLDRV.txt

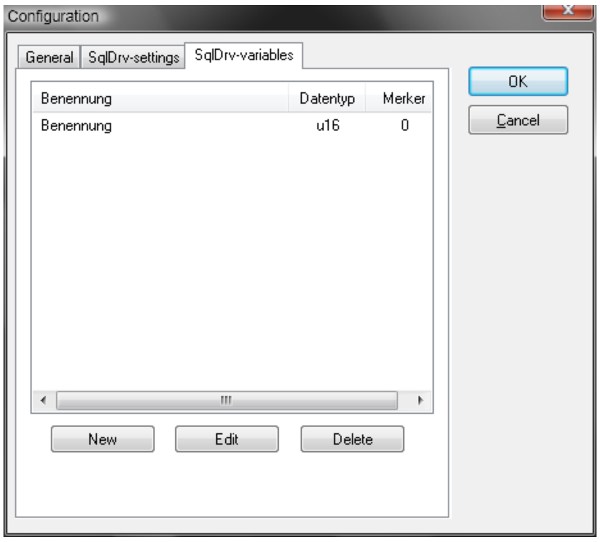
Tables
Receive table: ZENONEMPFANGEN
Send table: ZENONSENDEN

Parameter	Description
Drivers	ODBC connection including all necessary parameters. Click on the **...** button to open the dialog to select an existing connection or create a new connection. You can find help on creation in the Microsoft documentation by clicking on the **Help** button.
Variable file	File in which the variable assignment is saved. Clicking on the **...** button opens the dialog to select an existing file. storage location: *zenon project directory*.
Receive table	Name of the database table from which zenon receives data. Recommendation: *RECEIVE*
Send table	Name of the database table from which zenon receives data. Recommendation: *SEND*
Redundancy	Must be activated if server and standby server are supposed to access the same receive table. In this case, there must be 3 additional columns in the receive table (on page 8): ▸ **ACK_SRV** ▸ **ACK_SB** ▸ **INSERTZEIT**
Use TS time format	Time stamp: ODBC date and time.

Parameter	Description
	Must be activated if using a Microsoft SQL server data base.

7.2.3 Driver dialog SqlDrv-Variablen

In the variable list, the allocation between the database variable (plain text name) and the zenon variable (memory number) is defined.



Benennung	Datentyp	Merker
Benennung	u16	0

Parameter	Description
New	Creates new variables.
Name	Variable name. **Attention:** - Is created using a unique relation between zenon variable and database variable and must therefore be unique. - The string must be exactly identical to the string in the table for *RECEIVE* in the *NAME* column (including any possible spaces). Otherwise the row from the SQL database is read but the driver can then not assign this row any variables in Runtime. (error message in the LOG file "Variable not Defined: 'variable name from SQL table in the log file').
Marker number	Marker number with which the variable is defined using the **Offset** property. Is created using a unique relation between zenon variable and

Parameter	Description
	database variable and must therefore be unique.
Data type	For possible data types, see "Objects for process variables in zenon"
Change	Opens list elements for editing.
Delete	Deletes elements from the list.

8 Creating variables

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

8.1 Variables

Variables are created directly in the SqlDrv driver using the settings in Driver dialog SqlDrv variables (on page 25).

8.2 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.

8.2.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
PLC marker	8	X	X	*REAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING*	
Communication details	35	X	X	*BOOL, SINT, USINT, INT, UINT, DINT,*	Variables for the static analysis of the communication; Values

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
				UDINT, REAL, STRING	are transferred between driver and Runtime (not to the PLC). **Note**: The addressing and the behavior is the same for most zenon drivers. You can find detailed information on this in the Communication details (Driver variables) (on page 35) chapter.

Channel name	Data type	Channel type	Object	Read	Write
String	12	8	7	X	--
float32	5	8	6	X	--
i/u32Bit_mV	3	8	5	X	--
i/u32Bit	4	8	5	X	--
i/u16Bit	2	8	4	X	--
i/u16Bit_mV	1	8	4	X	--
i/u8Bit_mV	10	8	3	X	--
i/u8Bit	9	8	3	X	--
boolean	8	8	2	X	--

Key:

X: supported

--: not supported

CHANNEL TYPE

The term **Kanaltyp** is the internal numerical name of the driver object type. It is also used for the extended DBF import/export of the variables.

"**Kanaltyp**" is used for advanced CSV import/export of variables in the "**HWObjectType**" column.

8.2.2 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.

EXAMPLE FOR ALL POSSIBLE ZENON DATA TYPES:

SPS	zenon
I8	i/u8Bit (signed)
I16	i/u16Bit (signed)
I32	i/u32Bit (signed)
U8	i/u8Bit
U16	i/u16Bit
U32	i/u32Bit
F32	float32
Boolean	Boolean
string	String

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.

8.3 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 manual in the Variables section.

8.3.1 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** chapter.

8.3.2 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 instructions of 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 instructions of the import assistant.

Attention

DBF files:

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

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

Information

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

FILE STRUCTURE OF THE DBASE EXPORT FILE

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

Attention

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

DBF files must:

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

STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered

Identification	Type	Field size	Comment
			manually). The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_D	Log	1	The variable is deleted with the *1* entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	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-OBJEKTTYP and DATENTYP

Identification	Type	Field size	Comment
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix
ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists.

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

Attention

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

LIMIT VALUE DEFINITION

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

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

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

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

8.4 Communication details (Driver variables)

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

- Information
- Configuration
- Statistics and
- Error message

The definitions of the variables implemented in the driver kit are available in the import file **DRVVAR.DBF** and can be imported from there.
Path to file: *%ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables*

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

Information

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

For example:

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

INFORMATION

Name from import	Type	Offset	Description
MainVersion	*UINT*	0	Main version number of the driver.
SubVersion	*UINT*	1	Sub version number of the driver.
BuildVersion	*UINT*	29	Build version number of the driver.
RTMajor	*UINT*	49	zenon main version number
RTMinor	*UINT*	50	zenon sub version number
RTSp	*UINT*	51	zenon Service Pack number
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

Name from import	Type	Offset	Description
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
LineStateOnHoldPendTransfe r	*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 state 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: **<Net address>:<Connection status>;…;…;** A connection is only known after a variable

Name from import	Type	Offset	Description
			has first signed in. In order for a connection to be contained in a string, a variable of this connection must be signed in once. The status of a connection is only updated if a variable of the connection is signed in. Otherwise there is no communication with the corresponding controller.

CONFIGURATION

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

Name from import	Type	Offset	Description
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.

Name from import	Type	Offset	Description
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.

Name from import	Type	Offset	Description
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

9 Driver-specific functions

The driver supports the following functions:

The driver gets real-time values or real-time archive values from a receive table in the database and writes set values to a send table in the database. The identification of the variables is realized with a unique allocation between the zenon internal memory number and a plain text name in the tables. This allocation has to be projected in the driver. In zenon, the time entered in the database, i.e. realtime, will be used (e.g. CEL, AML,...).

LIMITATIONS

- At least one non-RDA variable must be registered for it to work without problems. This means: To be able to query real-time archive data, at least one variable with real-time values to query must be registered. To do this, a screen that contains real-time values must be switched, or at least one (dummy) variable with limit monitoring must be created.
- The database is never polled if only RDA variables are created.
- The connection to an MySql database with "MySQL ODBC Driver 3.51" does not work; the ODBC driver of this version does not support any dynasets.

MAXIMUM STRING LENGTH

The maximum string length for reading and writing is 32767 characters. There are also limits due to the database used.

10 Driver command function

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

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

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

Attention

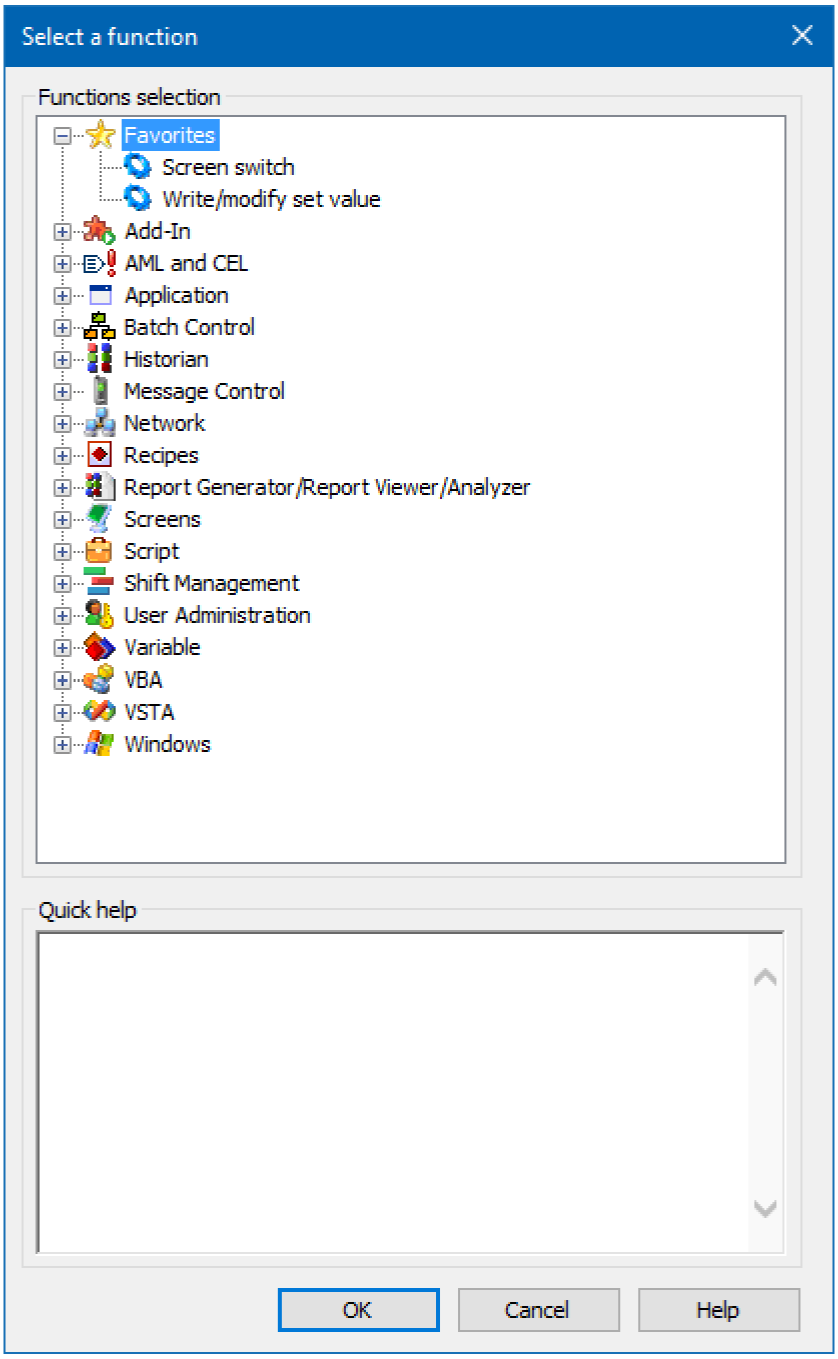
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

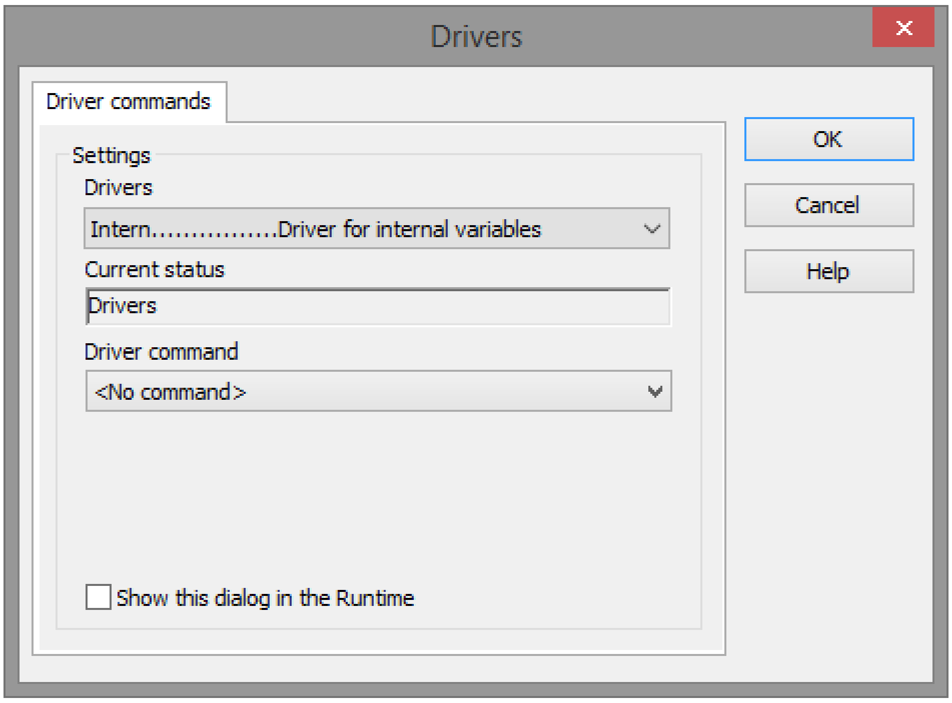
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



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

The dialog for configuration is opened



Settings

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

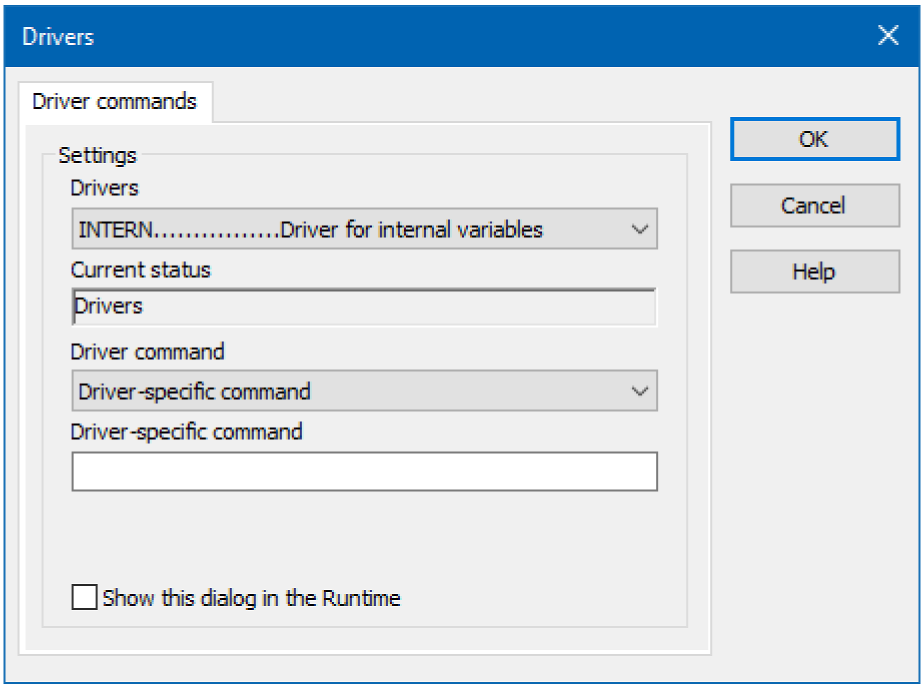
Current status: Drivers

Driver command: <No command>

☐ Show this dialog in the Runtime

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

DRIVER COMMAND DIALOG



Settings

Drivers: INTERN...............Driver for internal variables

Current status: Drivers

Driver command: Driver-specific command

Driver-specific command

☐ Show this dialog in the Runtime

Option	Description
Driver	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
Current condition	Fixed entry that is set by the system. no function in the current version.
Driver command	no function in the current version. For details on the configurable driver commands, see the **available driver commands** section.
Driver-specific command	Entry of a command specific to the selected driver. **Note:** Only available if, for the **driver command** option, the *driver-specific command* has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime: ▸ *Active*: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. ▸ *Inactive*: The Editor configuration is applied in the Runtime when executing the function. Default: *inactive*

CLOSE DIALOG

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

AVAILABLE DRIVER COMMANDS

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

Driver command	Description
No command	No command is sent. A command that already exists can thus be removed from a configured function.

Driver command	Description
Start driver (online mode)	Driver is reinitialized and started. **Note:** If the driver has already been started, it must be stopped. Only then can the driver be re-initialized and started.
Stop driver (offline mode)	Driver is stopped. No new data is accepted. **Note:** If the driver is in offline mode, all variables that were created for this driver receive the status *switched off* (*OFF*; Bit *20*).
Driver in simulation mode	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
Driver in hardware mode	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
Driver-specific command	Entry of a driver-specific command. Opens input field in order to enter a command.
Driver - activate set setpoint value	Write set value to a driver is possible.
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)
Driver in counting simulation mode	Driver is set into counting simulation mode. All values are initialized with *0* and incremented in the set update time by *1* each time up to the maximum value and then start at *0* again.
Driver in static simulation mode	No communication to the controller is established. All values are initialized with *0*.
Driver in programmed simulation mode	The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime.

DRIVER COMMAND FUNCTION IN THE NETWORK

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

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

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

11 Error analysis

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

11.1 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 program that was also installed with zenon. You can find it under **Start/All programs/zenon/Tools 8.20 -> 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 manual.

11.2 Driver monitoring

Runtime monitors the availability of the driver by means of a watchdog. If a driver is no longer available, the *INVALID* status bit is also set for all checked-in variables.

Possible causes for a triggering of the watchdog:

- The driver process is no longer running.

  Check whether the driver EXE file is still running in the Task Manager.
- Operating system is busy with processes that have a higher priority.

  Check the configuration of your system to see whether there is sufficient memory and CPU power. In this case, the driver only resets the *INVALID* status bit if there is a value change on the connected party. Static values retain the *INVALID* status bit until the next time the Runtime or the driver is started.

CONFIGURATION OF WATCHDOG

For the monitoring of communication in the Runtime, the connection to the driver is checked in a fixed, prescribed time period of 60 seconds. This process is repeated several times. If, within 5 attempts (= within 5 minutes), no valid connection to the driver is detected, the *INVALID* bit is set for the checked-in (*advised*) variables. In addition, the *INVALID* bit is also set when new variables are advised. The *INVALID* bit will no longer be reset.

Corresponding LOG entries are created for this.

LOG ENTRY

An error message is logged in the LOG when the watchdog is triggered:

Parameter	Description
Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.	No communication with driver within the given time. ▶ *<time>*: Time (in milliseconds) ▶ *<drvDesc>*: Driver name ▶ *<drvExe>*: Driver EXE name ▶ *<drvId>*: Driver ID in the zenon project
Communication with %s timed out. Invalid-Bit will be set.	Communication to the *%s* driver could not be established after 5 attempts within 60 seconds. The *INVALID* bit is set for the variable.
Communication with %s timed out. Timeout happened %d times	Communication to the *%s* driver could not be established after *%d* times within 60 seconds.

11.3 Check list

Check the following in the event of errors:

- Are sqldrv.txt on the server and standby?
- Have you analyzed the error text file (which errors did occur)?

For further analysis of errors, send the following to your support department in charge:

- Project Backup
- "Error text file" (this is in the prject path under *RT\FILES\zenon\custom\log*)

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

%ProgramData%\COPA-DATA\LOG.

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

The following is required for further analysis of errors:

- The project backup
- LOG files

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