



**COPADATA**  
do it your way

# zenon driver manual

**RemoteRT**

**v.7.50**





©2016 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

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

# Contents

<b>1. Welcome to COPA-DATA help .....</b>	<b>5</b>
<b>2. RemoteRT .....</b>	<b>5</b>
<b>3. REMOTERT - Data sheet .....</b>	<b>6</b>
<b>4. Driver history .....</b>	<b>7</b>
<b>5. Requirements.....</b>	<b>8</b>
5.1 Installation and procedure.....	8
5.2 Connector.....	9
<b>6. Configuration .....</b>	<b>12</b>
6.1 Creating a driver.....	13
6.2 Settings in the driver dialog .....	15
6.2.1 General .....	15
6.2.2 Remote Runtime connections .....	18
<b>7. Creating variables.....</b>	<b>20</b>
7.1 Creating variables in the Editor.....	21
7.2 Addressing.....	24
7.3 Driver objects and datatypes .....	24
7.3.1 Driver objects .....	25
7.3.2 Mapping of the data types .....	25
7.4 Creating variables by importing .....	27
7.4.1 XML import.....	27
7.4.2 DBF Import/Export .....	28
7.5 Driver variables .....	33
<b>8. Polling in Runtime .....</b>	<b>40</b>
<b>9. Driver-specific functions .....</b>	<b>41</b>
<b>10. Driver commands .....</b>	<b>41</b>
<b>11. Error analysis.....</b>	<b>43</b>

11.1	Analysis tool .....	43
11.2	Check list .....	44

# 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) (<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) (<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) (<mailto:sales@copadata.com>).

# 2. RemoteRT

The Remote Runtime driver (**RemoteRT.exe**) makes it possible to read variable values in Runtime from a different Runtime by means of a connector container, and to apply these. The driver is fundamentally different from other zenon drivers in terms of its connector concept. The requesting of data from the source Runtime roughly corresponds to the teaching of a recipe.

The driver only addresses on the basis of names via the **Symbolic address**.

**Note:** The Remote Runtime driver is most of all intended for connection to Runtimes of version 5.50 or lower. From version 6.00, the use of Process Gateway is recommended.

### 3. REMOTERT - Data sheet

General:	
Driver file name	REMOTERT.exe
Driver name	Remote Runtime driver
PLC types	Remote Runtime
PLC manufacturer	zenon system driver; COPA-DATA;

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

Requirements:	
Hardware PC	--
Software PC	--
Hardware PLC	--
Software PLC	SCADA Connector Container - located in additional software folder on installation media
Requires v-dll	X

Platforms:	
Operating systems	Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2;
CE platforms	-;

## 4. Driver history

Date	Build number	Change
04.09.14	7.20.0.14252	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 Installation and procedure

#### INSTALLATION

All necessary files are installed with the installation of zenon.



#### Attention

*In order for the driver to be able to be started, the **zrsProvider.dll** must be installed.*

The Connector Container may need to be installed manually on the source computer. To do this, there is a separate setup available on the zenon installation medium. Path: **AdditionalSoftware\COPA-DATA SCADA Runtime Connector\setup86\_connectors.exe**.



#### Information

*The Connector Container is entered as an Autostart application during setup. It is started in this context when a user is logged in. It must also correspond to the user context in which Runtime is running. If Runtime as a service is started, the Connector Container must also be started as a service using the *Startup Tool*.*

#### PROCEDURE

- Systems involved:



- Local Runtime (SCADA): The Remote Runtime driver runs here.
- Remote Runtime (PLC): The connector container runs here.
- ▶ Addressing:
  - Local Runtime (SCADA): **Symbolic address**.
  - Remote Runtime (SPS): **Name** of the variable.
- ▶ The variable on the local Runtime gives the driver all information that it needs:
  - **Net address**: Is mapped to the IP address and project name of Remote Runtime in the driver configuration.
  - **Symbolic address**: Must correspond to the **Name** of the variables in the zenon project that runs on Remote Runtime.



#### Attention

*For the Remote Runtime driver, the **Symbolic address** property must be configured on the local system for each desired variable. The variable is ignored if this is empty.*

## CONNECTION AND UPDATE TIME

The driver establishes a connection to the connector container on the respective target computer, to TCP port 50778. This port must be contactable and enabled in the firewall.

There is one read attempt per **Net address** in each update cycle. A connection is established for each read attempt, the query is handled and the connection is closed.

Depending on the basic load of the target system, and the number of variables requested or the cycle of queried values, it is possible that there is a considerable load placed on the source system.

**Recommendation:** The **Global update time** should be greater than 1000 ms.

**Note:** The polling driver connection with update times of 1 second or more needs time accordingly. Plan slower reactions.

## 5.2 Connector

A connector acts as the link between the data source. Connectors can consist of:

- ▶ Connector stub
- ▶ Connector container
- ▶ Connector plug-in

A distinction is made between:

- ▶ SQL-based connectors: These are executed in the connector stub directly.
- ▶ C++ DLLs: These work as plugins for the connector container.

External Runtime data are requested by the source system via a TCP connection. This connection is established between the connector stub at the SQL server or the connector container for plugins.

## VORAUSSETZUNGEN

Für den Einsatz von Connectoren müssen folgende Voraussetzungen erfüllt werden:

- ▶ In der zenon Runtime muss der Event-Mechanismus der COM Schnittstelle aktiviert sein. Dazu muss in der **zenon6.ini** folgender Eintrag gesetzt sein:  
`[VBA]`  
`EVENT=1`
- ▶ Der Port 50778 muss frei und offen sein (z. B. in der Firewallkonfiguration).



### Information

Connectors cannot be started multiple times.

The connector is ended if one of the following events occurs when the network connection is made:

- ▶ Error when creating the socket
- ▶ Error when opening the listening port
- ▶ Error when starting the listening
- ▶ Error when accepting a client connection

## LIMITATIONS AND PERFORMANCE

The following applies for connectors:

- ▶ Timeout: 5 minutes (independent of report timeout)
- ▶ Variables: Only variables that are listed in metadata are requested
- ▶ String variable: maximum of 4000 characters

The performance of a connector depends on the:

- ▶ Performance of the Analyzer server
- ▶ Performance of the Runtime server
- ▶ Load of the Runtime servers (connector runs with low priority)
- ▶ Network performance and network load

## RULES FOR TIME FILTER

Time filters are applied as follows:

- ▶ Time stamp:
  - Start time: set filter time plus one millisecond
  - End time: set filter time
- ▶ Aggregation archives: Value at the start of an interval always represents the aggregation value of the previous interval.
- ▶ Time stamp in basis archives: Lag behind archive cycle in millisecond range.

Example time stamp:

- ▶ Filter 10:00 to 11:00: is interpreted as 10:00:001 to 11:00:000.
- ▶ Filter 1 day: starts one millisecond after midnight.

## CONNECTOR STUB

The connector stub is a DLL which is used by the **Table Valued User Defined Functions** in SQL Server in order to request data from the connector container from a source computer. It

- ▶ reads in necessary metadata (project name, server, standby, etc.) from SQL server
- ▶ establishes a TCP connection to the connector container at queries
- ▶ sends a request
- ▶ receives the answer

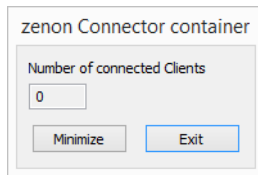
Several queries from different threads can be carried out at the same time. The connector stub is able to establish a TCP connection to an alternative source computer if the primary source computer cannot be reached. The names of the primary and alternative source computers are entered in the project table in column `SERVER` and `STANDBY`.

## CONNECTOR CONTAINER

The connector container is an application (EXE) which runs at the source system and which loads and executes the connector plug-ins (DLLs). The connector container is a normal user process (no service) which is normally started together with the application which should deliver the data. The connector container opens a TCP port and waits for query requests from the connector stub whereon it loads the requested connector plug-in and invokes the fitting access function for the request. The return data is then sent to the connector stub. Several queries from different TCP connections can be executed in parallel if the source system supports this.

In normal operation the connector container is displayed as icon in the task tray and does not have an own main window. Additional status information can be displayed via a status dialog.

## DIALOG



Parameters	Description
<b>Number of connected Clients</b>	Displays the number of clients connected.
<b>Minimize</b>	Minimizes the dialog into the info area of the task bar.
<b>Exit</b>	Closes the connector container.

## RESTART

If the connector container has been closed, it can be restarted by:

- ▶ Restarting the computer.
- ▶ Manual start.
  - From Windows 8: Task-Manager -> Tab -> Autostart -> Connector-Container -> Open file path-> Double-click on **zrsConnector.exe**.
  - Other operating systems: Open file path-> Double-click on **zrsConnector.exe**.  
32-bit path: %Program Files (x86)%\Common Files\COPA-DATA\Connectors

## CONNECTOR PLUG-IN

Connector plug-ins create the link between stub and container.

The **SCADA Runtime** connector plug-in uses the zenon API to connect to zenon Runtime and can query runtime data from there. Historical shift data and recipes cannot be queried.

/References:

- ▶ Project reference: zenon project name
- ▶ Variable reference: zenon variable name

# 6. Configuration

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



### Information

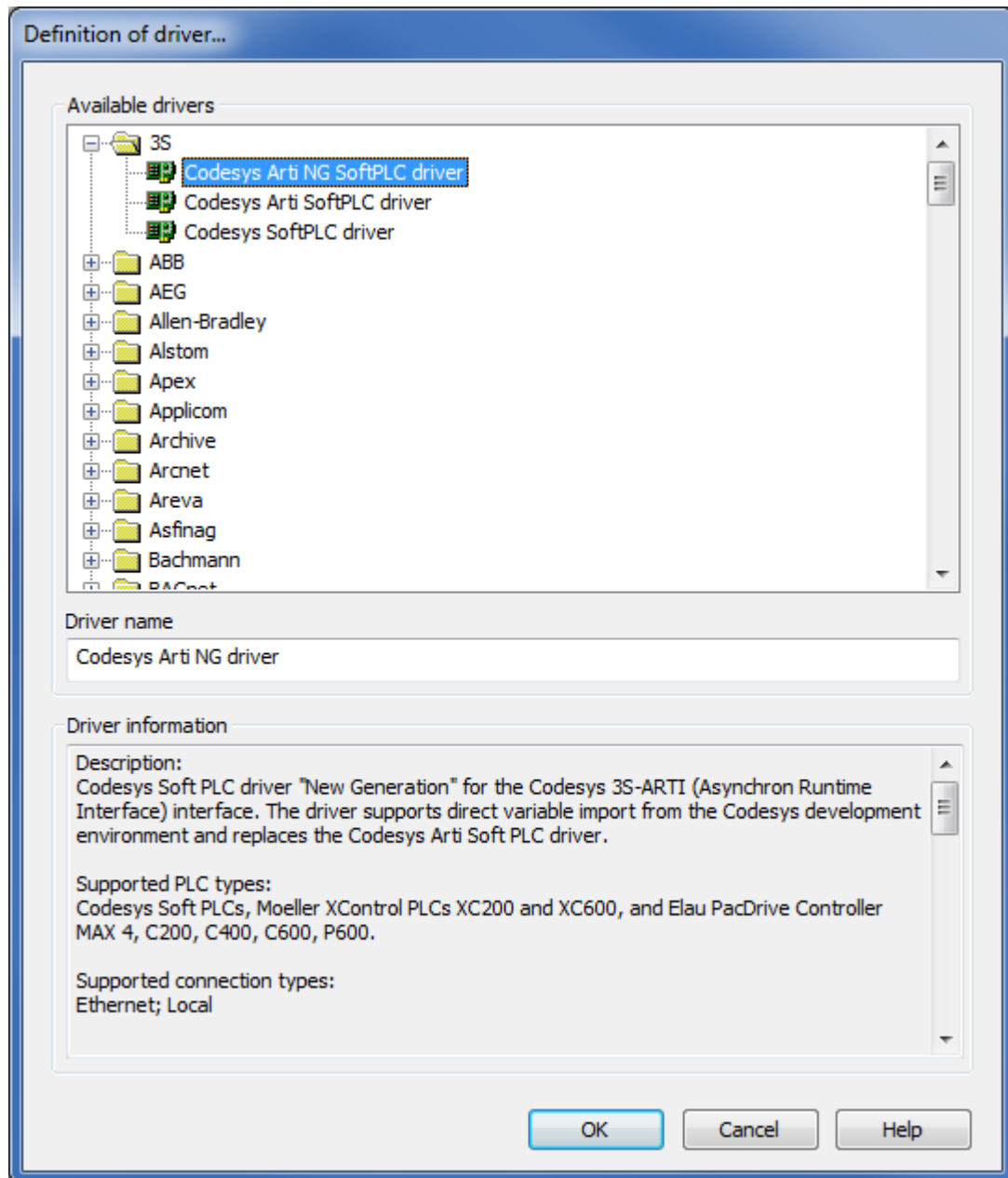
*Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.*

## 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.

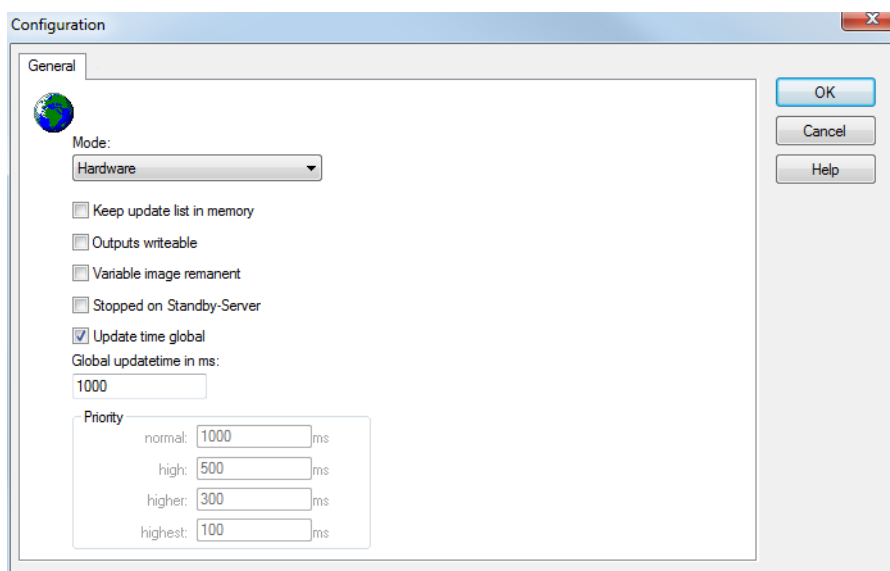
▶

## 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
<b>Mode</b>	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> <li>▶ Hardware: <p>A connection to the control is established.</p> </li> <li>▶ Simulation static <p>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.</p> </li> <li>▶ Simulation - counting <p>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.</p> </li> <li>▶ Simulation - programmed <p>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).</p> </li> </ul>
<b>Keep update list in the memory</b>	<p>Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.</p>
<b>Output can be written</b>	<p>Active: Outputs can be written.</p> <p>Inactive: Writing of outputs is prevented.</p> <p>Note: Not available for every driver.</p>
<b>Variable image remanent</b>	<p>This option saves and restores the current value, time stamp and the states of a data point.</p> <p>Fundamental requirement: The variable must have a valid value and time stamp.</p>



	<p>The variable image is saved in mode hardware if:</p> <ul style="list-style-type: none"> <li>▶ one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active</li> </ul> <p>The variable image is always saved if:</p> <ul style="list-style-type: none"> <li>▶ the variable is of the object type <b>Driver variable</b></li> <li>▶ the driver runs in simulation mode. (not programmed simulation)</li> </ul> <p>The following states are not restored at the start of the Runtime:</p> <ul style="list-style-type: none"> <li>▶ SELECT(8)</li> <li>▶ WR-ACK(40)</li> <li>▶ WR-SUC(41)</li> </ul> <p>The mode <b>Simulation - programmed</b> at the driver start is not a criterion in order to restore the remanent variable image.</p>
<b>Stop on Standby Server</b>	<p>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.</p> <p><b>Attention:</b> If this option is active, the gapless archiving is no longer guaranteed.</p> <p><b>Active:</b> 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 <b>switched off (statusverarbeitung.chm::/24150.htm)</b> but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.</p> <p><b>Note:</b> Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.</p>
<b>Global Update time</b>	<p><b>Active:</b> The set <b>Global update time</b> in ms is used for all variables in the project. The priority set at the variables is not used.</p> <p><b>Inactive:</b> The set priorities are used for the individual variables.</p>
<b>Priority</b>	<p>The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.</p> <p>The allocation to the variables takes place separately in the settings of the variable properties.</p> <p>The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities.</p>

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

## CLOSE DIALOG

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

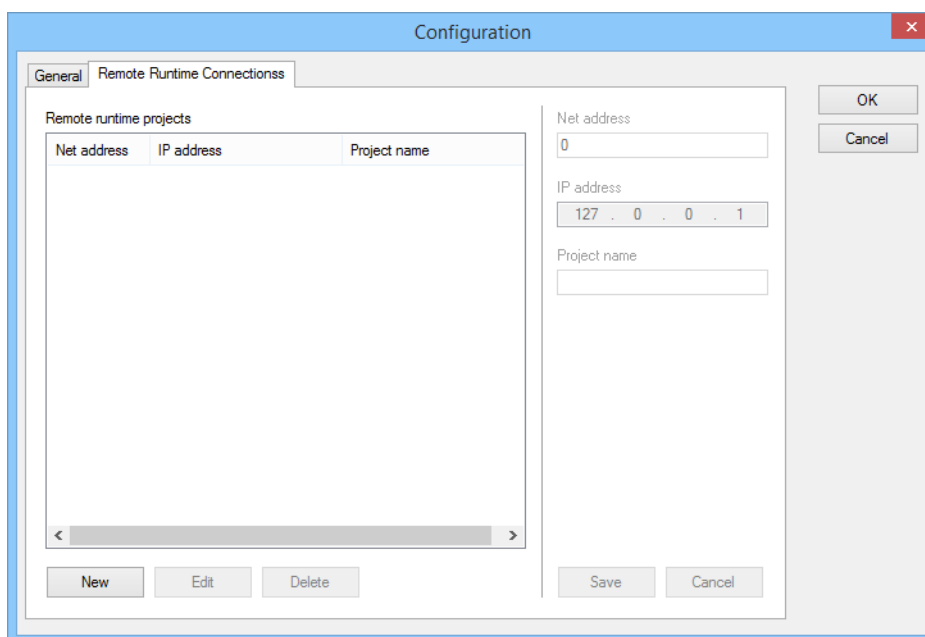
## UPDATE TIME FOR CYCLICAL DRIVERS

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 Remote Runtime connections

Configuration of the connections to Remote Runtime:



Parameters	Description
<b>Remote Runtime projects</b>	<p>Contains all defined connections. For these, the respective <b>network address</b>, the <b>IP address</b> and the <b>project name</b> are displayed.</p> <p>Selection of the connection to edit or delete by clicking on the entry. Only active if the dialog is not in edit mode.</p> <p>Maximum: 256 connections</p>
<b>New</b>	<p>Clicking on this switches to edit mode and allows the creation of a new connection.</p> <p>Only active if the dialog is not in edit mode and less than 256 connections are contained in the list.</p>
<b>Edit</b>	<p>Clicking on this switches to edit mode and allows editing of the selected connection.</p> <p>Is only active if the dialog is not in edit mode and a connection in the list has been selected.</p>
<b>Remove</b>	<p>Clicking deletes the selected connection from the list.</p> <p>Only active if the dialog is not in edit mode and a connection in the list has been selected.</p>
<b>Net address</b>	<p>Entry of the zenon <b>Net address</b>.</p> <p>Only active if the dialog is in edit mode.</p> <p>Default: Lowest free network address or the network address of the selected connection.</p>
<b>IP address</b>	<p>Entry of the IP address.</p> <p>Only active if the dialog is in edit mode.</p> <p>Default: 127.0.0.1 or the IP address of the selected connection.</p>
<b>Project name</b>	<p>Entry of the project name for the connection. The name is automatically displayed in capital letters.</p> <p>Only active if the dialog is in edit mode.</p> <p>Default: Empty or name of the selected connection.</p>
<b>Save</b>	<p>Clicking validates the entries. If validation is successful, the changes are accepted and shown in the list. Edit mode is ended.</p> <p>Validations:</p> <ul style="list-style-type: none"> <li>▶ Net address: Positive whole number less than 256.</li> <li>▶ IP address: Formal valid IP address.</li> <li>▶ Project name: not empty.</li> </ul> <p>Only active if the dialog is in edit mode.</p>
<b>Cancel</b>	Discards all changes and ends edit mode.

	Only active if the dialog is in edit mode.
--	--

#### CLOSE DIALOG

Parameters	Description
<b>OK</b>	Applies all changes in all tabs and closes the dialog.
<b>Cancel</b>	Discards all changes in all tabs and closes the dialog.

#### PROCEDURES

##### CREATE NEW CONNECTION

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

##### EDIT CONNECTION

1. Select the connection in the connection list.
2. Click on the **Edit** button.
3. Change the connection parameters.
4. Click on **Save**.

##### DELETE CONNECTION

1. Select the connection in the connection list.
2. Click on the button **Delete**.
3. The connection will be removed from the list

## 7. Creating variables

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

## 7.1 Creating variables in the Editor

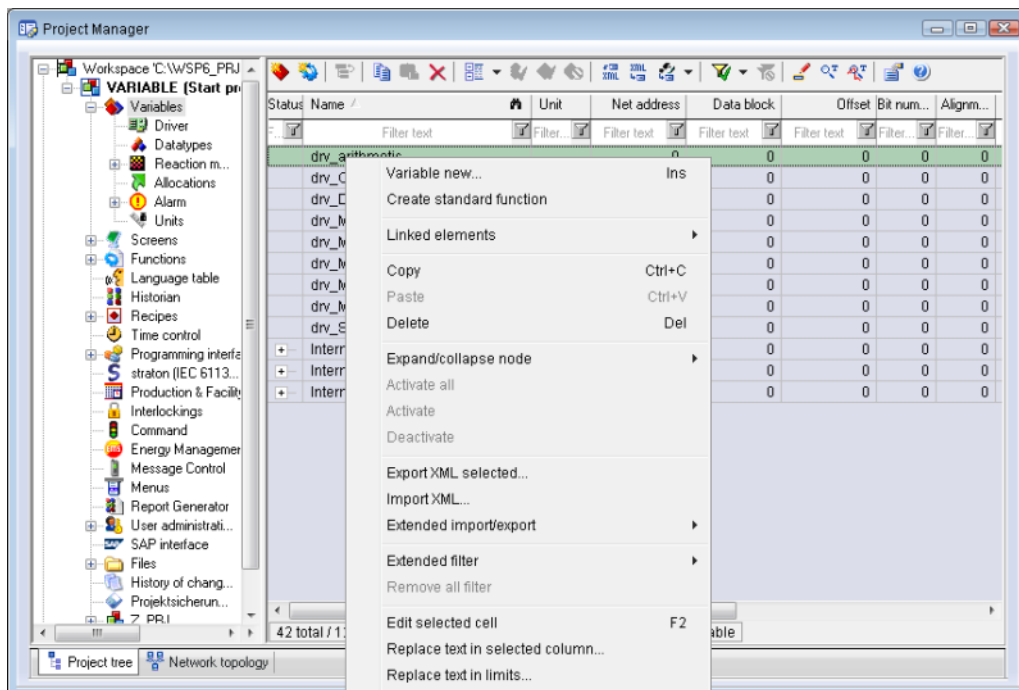
Variables can be created:

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

### VARIABLE DIALOG

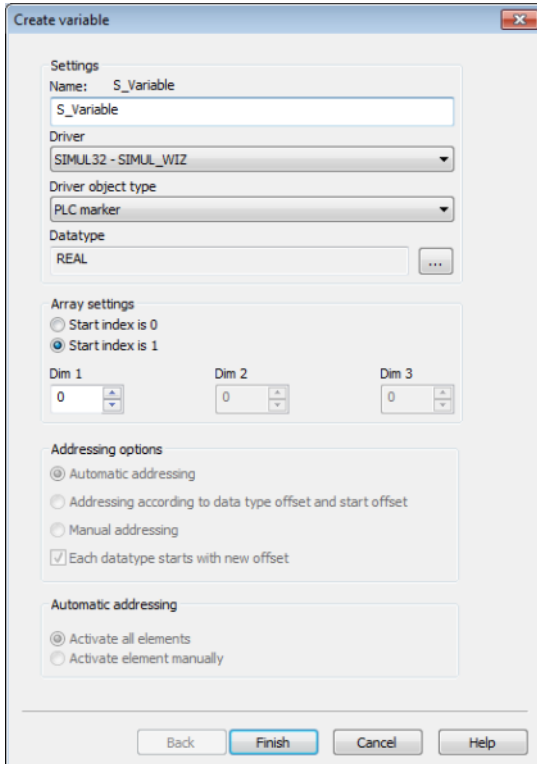
To create a new variable, regardless of which type:

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



The screenshot shows the 'Create variable' dialog box with the following settings:

- Settings**
  - Name: S\_Variable
  - Driver: SIMUL32 - SIMUL\_WIZ
  - Driver object type: PLC marker
  - Datatype: REAL
- Array settings**
  - ☐ Start index is 0
  - ☒ Start index is 1
  - Dim 1: 0
  - Dim 2: 0
  - Dim 3: 0
- Addressing options**
  - ☒ Automatic addressing
  - ☐ Addressing according to data type offset and start offset
  - ☐ Manual addressing
  - ☒ Each datatype starts with new offset
- Automatic addressing**
  - ☒ Activate all elements
  - ☐ Activate element manually

Buttons at the bottom: Back, Finish, Cancel, Help.

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

Group/Property	Description
<b>General</b>	
<b>Name</b>	Freely definable name. <b>Attention:</b> For every zenon project the name must be unambiguous.
<b>Identification</b>	Any text can be entered here, e.g. for resource labels, comments.
<b>Addressing</b>	The name of the variables in the project on Remote Runtime are only read from the <b>Symbolic address</b> property.
<b>Net address</b>	Bus address or net address of the variable.  This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.
<b>Data block</b>	not used for this driver
<b>Offset</b>	not used for this driver
<b>Alignment</b>	not used for this driver
<b>Bit number</b>	not used for this driver
<b>String length</b>	Only available for String variables: Maximum number of characters that the variable can take.
<b>Driver connection/Driver Object Type</b>	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.  Only <code>driver variable</code> and <code>Remote Runtime variable</code> are available for the Remote Runtime driver.
<b>Driver connection/Data Type</b>	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. The following are available for the Remote Runtime driver: <ul style="list-style-type: none"> <li>▶ <code>BOOL</code></li> <li>▶ <code>LREAL</code></li> <li>▶ <code>WSTRING</code></li> </ul> <b>Attention:</b> If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

## 7.3 Driver objects and datatypes

Driver objects are areas available in the PLC, such as markers, data blocks etc. Here you can find out which driver objects are provided by the driver and which IEC data types can be assigned to the respective driver objects.



### 7.3.1 Driver objects

The following object types are available in this driver:

Driver object type	Channel type	Read	Write	Supported data types	Description
<b>Driver variable</b>	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the static analysis of the communication; is transferred between driver and Runtime (not to the PLC).  <b>Note:</b> The addressing and the behavior is the same for most zenon drivers.  Find out more in the chapter about the Driver variables (on page 33)
<b>Remote Runtime variable</b>	64	X	--	BOOL, LREAL, WSTRING	

### 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.

Remote Runtime	Local Runtime	Data type
BOOL	BOOL	8
-	USINT	9
-	SINT	10
-	UINT	2
-	INT	1
-	UDINT	4
-	DINT	3
-	ULINT	27
-	LINT	26
-	REAL	5
LREAL	LREAL	6
-	STRING	12
WSTRING	WSTRING	21
-	DATE	18
-	TIME	17
-	DATE_AND_TIME	20
-	TOD (Time of Day)	19

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

The Remote Runtime driver does not carry out any type conversions. The following breakdown shows how the connector container supplies data for individual zenon data types:

- ▶ **BOOL:**
  - Numerical value:
    - 0: False
    - 1: True i
  - Text values: empty
- ▶ **Numerical data types:**
  - Numeric value: Value
  - Text value: empty
- ▶ **String data types:**
  - Numeric value: empty
  - Text value: Text

Based on this, the driver determines the value for each data type from the data provided by the connector client as follows:

- ▶ **BOOL**: `True` if the field for the numerical value is not 0, otherwise `False`.
- ▶ **LREAL**: Value of the field for numerical values.
- ▶ **WSTRING**: Value of the field for text values.

If the data types from your own Runtime do not correspond to those of Remote Runtime, there will be errors when calculating values:

Remote\target	BOOL	LREAL	WSTRING
BOOL on Remote Runtime	No error	0 if <code>False</code> on remote Runtime. 1 if <code>True</code> on Remote Runtime	Always empty.
Numeric on Remote Runtime	<code>False</code> if precisely 0 on Remote Runtime. Otherwise <code>True</code> .	There may be a rounding error, because the value is mapped to LREAL (limited precision).	Always empty.
Text on Remote Runtime	Always <code>False</code> .	Always 0.	The string may be shortened.

## 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 `INT` with 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 <b>project.ini</b> .
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 <b>project.ini</b> .
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 <b>project.ini</b> .
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

<b>LES_SCHR</b>	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
<b>MIT_ZEIT</b>	L	1	time stamp in zenon (only if supported by the driver)
<b>OBJEKT</b>	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTYP and DATENTYP
<b>SIGMIN</b>	Float	16	Non-linearized signal - minimum (signal resolution)
<b>SIGMAX</b>	F	16	Non-linearized signal - maximum (signal resolution)
<b>ANZMIN</b>	F	16	Technical value - minimum (measuring range)
<b>ANZMAX</b>	F	16	Technical value - maximum (measuring range)
<b>ANZKOMMA</b>	N	1	Number of decimal places for the display of the values (measuring range)
<b>UPDATERATE</b>	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
<b>MEMTIEFE</b>	N	7	Only for compatibility reasons
<b>HDRATE</b>	F	19	HD update rate for historical values (in sec, one decimal possible)
<b>HDTIEFE</b>	N	7	HD entry depth for historical values (number)
<b>NACHSORT</b>	L	1	HD data as postsorted values
<b>DRRATE</b>	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
<b>HYST_PLUS</b>	F	16	Positive hysteresis, from measuring range
<b>HYST_MINUS</b>	F	16	Negative hysteresis, from measuring range
<b>PRIOR</b>	N	16	Priority of the variable
<b>REAMATRIZE</b>	C	32	Allocated reaction matrix
<b>ERSATZWERT</b>	F	16	Substitute value, from measuring range
<b>SOLLMIN</b>	F	16	Minimum for set value actions, from measuring range
<b>SOLLMAX</b>	F	16	Maximum for set value actions, from measuring range
<b>VOMSTANDBY</b>	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
<b>RESOURCE</b>	C	128	Resources label. Free string for export and display in lists.  The length can be limited using the MAX_LAENGE entry in <b>project.ini</b> .
<b>ADJWVBA</b>	L	1	Non-linear value adaption: 0: Non-linear value adaption is used 1: Non-linear value adaption is not used

<b>ADJZENON</b>	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
<b>ADJWVBA</b>	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
<b>ZWREMA</b>	N	16	Linked counter REMA.
<b>MAXGRAD</b>	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
<b>AKTIV1</b>	L	1	Limit value active (per limit value available)
<b>GRENZWERT1</b>	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)
<b>SCHWWERT1</b>	F	16	Threshold value for limit value
<b>HYSTERESE1</b>	F	14	Is not used
<b>BLINKEN1</b>	L	1	Set blink attribute
<b>BTB1</b>	L	1	Logging in CEL
<b>ALARM1</b>	L	1	Alarm
<b>DRUCKEN1</b>	L	1	Printer output (for CEL or Alarm)
<b>QUITTIER1</b>	L	1	Must be acknowledged
<b>LOESCHE1</b>	L	1	Must be deleted
<b>VARIABLE1</b>	L	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
<b>FUNC1</b>	L	1	Functions linking
<b>ASK_FUNC1</b>	L	1	Execution via Alarm Message List
<b>FUNC_NR1</b>	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
<b>A_GRUPPE1</b>	N	10	Alarm/Event Group
<b>A_KLASSE1</b>	N	10	Alarm/Event Class
<b>MIN_MAX1</b>	C	3	Minimum, Maximum
<b>FARBE1</b>	N	10	Color as Windows coding
<b>GRENZTXT1</b>	C	66	Limit value text
<b>A_DELAY1</b>	N	10	Time delay
<b>INVISIBLE1</b>	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.

## 7.5 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 ErrorMessage only for drivers that only edit one connection at a time

## INFORMATION

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

LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
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 <code>driver stop</code> , the variable has the value <code>TRUE</code> and an <b>OFF</b> bit. After the driver has started, the variable has the value <code>FALSE</code> and no <b>OFF</b> bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

## CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If <code>TRUE</code> , 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 <code>SrvDrvVarApplyCom</code> 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 <code>SrvDrvVarApplyModem</code> . This closes the current connection and opens a new one according to the settings <b>PhoneNumberSet</b> and <b>ModemHwAdrSet</b> .

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 <b>Normal</b> in milliseconds (ms).
MaxUpdateTimeHigher	UDINT	57	Time since the last update of the priority group <b>Higher</b> in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group <b>High</b> in milliseconds (ms).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group <b>Highest</b> 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. Polling in Runtime

When polling, all registered variables of a network address are obtained in a query at once.

Polling procedure:

1. The driver triggers polling for a network address, depending on global update time, the update time of the highest-priority variable of the network address and an error waiting time after a failed polling.
2. The connection definition of the network is searched for in the driver configuration. If this cannot be found, the polling is ended with an error.
3. In the driver, all variables of this network address that are currently registered are obtained. If there are not currently any variables registered, the polling is ended as successful with no further action.
4. The configured IP address is broken down into a structure that can be used by the connector client. If an error occurs in the process, the polling is ended with an error.
5. the current variable values of all variables to be polled are queried using the connector client with the previously-obtained IP address structure and the project name given in the connection. The variable name on Remote Runtime corresponds to the entry in the **Symbolic address** property of the variables on the local system.  
If an error occurs in the process, the polling is ended with an error. If a variable on the Remote Runtime cannot be queried, this is not considered an error.
6. The values received for each variable are sent to Runtime by the driver:



- a) If a value has come, is sent to Runtime in accordance with the data type set on its own Runtime.
- b) If no value has come, the individual variables are set to I-bit.

## 9. Driver-specific functions

The driver supports the following functions:

- Import of variable values of a remote zenon Runtime to a local zenon Runtime.

**Attention:** In order to not overload the Remote Runtime, a global update time of at least one second is recommended. Note that polling is slower than is usual for other drivers.

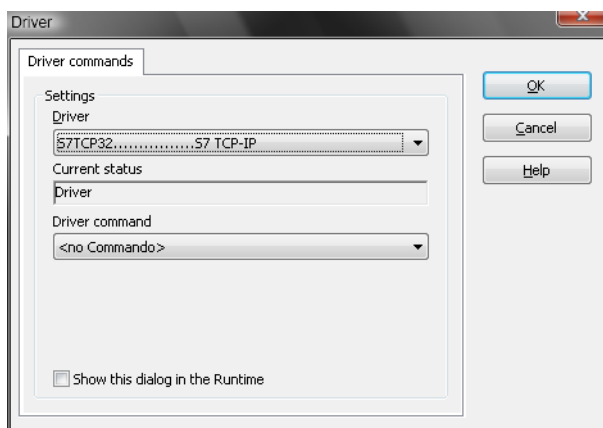
Error timeout: 20 seconds

## 10. 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
<b>Drivers</b>	Drop-down list with all drivers which are loaded in the project.
<b>Current status</b>	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. <b>Note:</b> If the driver is in offline mode, all variables that were created for this driver receive the status <code>switched off (OFF; Bit 20)</code> .
▶ 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)
<b>Show this dialog in the Runtime</b>	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.

## 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 (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.

## 11.2 Check list

Questions and hints for fault isolation:

### GENERAL TROUBLESHOOTING

- ▶ Is the Runtime connected to the power supply?
- ▶ Analysis with the **Diagnosis Viewer** (on page 43):  
-> Which messages are displayed?
- ▶ Are the participants available in the **TCP/IP** network?
- ▶ Can the Runtime be reached via the `Ping` command?  
Ping: Open command line -> ping < IP address> (e.g. ping 192.168.0.100) -> press Enter.  
Do you receive an answer with a time or a time-out?
- ▶ Can the Runtime be reached via `Telnet`?  
Telnet: Command line Enter open, telnet <IP address port number> (for example for Modbus: telnet 192.168.0.100 502) -> Press Return key.  
If the monitor display turns black, a connection could be established.
- ▶ Has a **Symbolic address** been issued for each variable whose value is to be imported?
- ▶ Did you configure the Net address in the address properties of the variable correctly?
  - Does the addressing match with the configuration in the driver dialog?
  - Does the net address match the address of the target station?
- ▶ Did you use the right object type for the variable?

### SOME VARIABLES REPORT INVALID.

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

**VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY**

Driver is not working. Check the:

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

## SOLUTIONS TO PROBLEMS

Problem	Reason	Solution
the driver does not start. An error message is shown and the process is ended immediately.	The <b>zrsProvider.dll</b> could not be loaded.	<p>Install the container components for zenon. If the problem remains, then:</p> <ul style="list-style-type: none"> <li>▶ Ensure that the registry entries exist. 32-bit directory under AnalyzerWrapperDir32, 64-bit directory under AnalyzerWrapperDir64</li> <li>▶ Ensure that zrsProvider.dll exists in the paths stated in the registry entries. 32-bit DLL in the 32-bit directory; 64-bit DLL in the 64-bit directory.</li> <li>▶ Ensure that all dependencies of zrsProvider.dll are installed (the correct version of vcredist; can be checked with Dependency Walker).</li> </ul>
Variables from the Remote Runtime driver are on I-bit.	<p>Which cause exactly is shown by the log entries in the Diagnosis Viewer. Possible reasons:</p> <ul style="list-style-type: none"> <li>▶ Network address of the variables is not correct.</li> <li>▶ Network connection is not possible. For example due to an incorrect IP address, target IP cannot be reached due to a network failure,</li> </ul>	<p>Please check:</p> <ul style="list-style-type: none"> <li>▶ Network connectivity and firewall settings.</li> <li>▶ Install connector container on the target system and start it in the same user context as Runtime.</li> <li>▶ Check project configuration: <ul style="list-style-type: none"> <li>- IP address</li> <li>- project name of the connections in the</li> </ul> </li> </ul>

	<p>firewall settings etc.</p> <ul style="list-style-type: none"><li>▶ Connector container does not run on the target system.</li><li>▶ Connector container on the target system runs in a different user context than Runtime on the target system.</li><li>▶ Project name of the Remote Runtime project is not correct.</li><li>▶ <b>Symbolic address</b> of the variables are not variable names of the project on Remote Runtime.</li></ul>	<p>driver configuration</p> <ul style="list-style-type: none"><li>- Symbolic addresses</li><li>Network addresses of the variables</li></ul>
--	--	---