zenon driver manual

FRAPORT

v.7.00

Contents

1. Welcome to COPA-DATA help

GENERAL HELP

If you miss any information in this help chapter or have any suggestions for additions, please feel free to contact us via e-mail: documentation@copadata.com (mailto:documentation@copadata.com).

PROJECT SUPPORT

If you have concrete questions relating to your project, please feel free to contact the support team via e-mail: support@copadata.com (mailto:support@copadata.com)

LICENSES AND MODULES

If you realize that you need additional licenses or modules, please feel free to contact the sales team via e-mail: sales@copadata.com (mailto:sales@copadata.com)

2. FRAPORT

The FRAPORT driver makes it possible to dispaly all mesages and displays in the zenon interconnected baggage system (IBS) and to be able to control the system.

3. FRAPORT - Data sheet

General:	
Driver file name	FRAPORT.exe
Driver description	FRAPORT Driver
PLC types	FRAPORT baggage system
PLC manufacturer	Fraport AG;

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

Prerequisites:	
Hardware PC	Standard Network Adapter
Software PC	-
Hardware PLC	-
Software PLC	-
Requires v-dll	-

Platforms:	
Operating systems	Windows XP, Vista, 7, Server 2003, Server 2008/R2;
CE platforms	-;

4. Driver history

Date	Driver version	Change
16.03.11	100	Created driver documentation

5. Configuration

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

Info

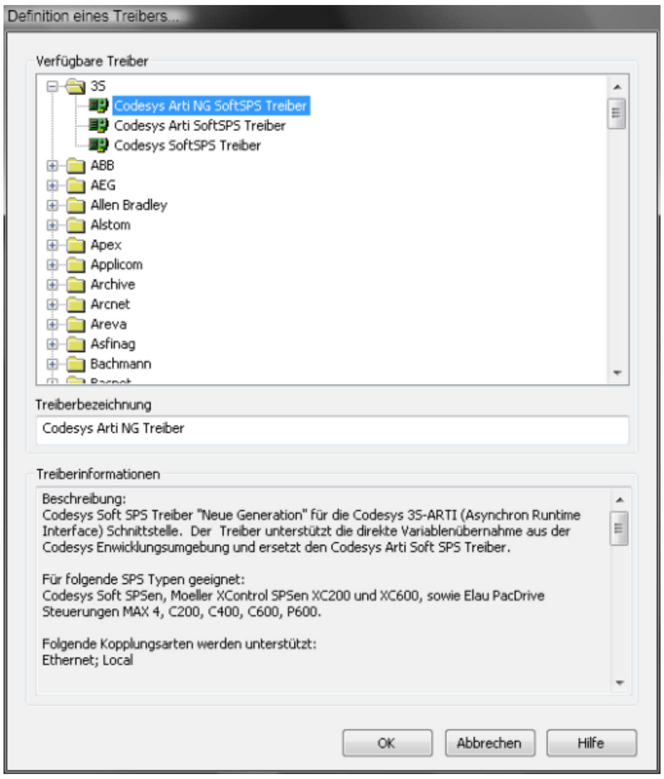
Find out more about further settings for zenon variables in the chapter Variables

> *(main.chm::/15247.htm) of the online manual.*

5.1 Creating a driver

In order to create a new driver:

- Right-click on **`Driver`** in the Project Manage and select **`Driver new`** in the context menu.
- In the following dialog the control system offers a list of all available drivers.



Definition eines Treibers...

Verfügbare Treiber

- 3S
  - Codesys Arti NG SoftSPS Treiber
  - Codesys Arti SoftSPS Treiber
  - Codesys SoftSPS Treiber
- ABB
- AEG
- Allen Bradley
- Alstom
- Apex
- Applicom
- Archive
- Arcnet
- Areva
- Asfinag
- Bachmann

Treiberbezeichnung: Codesys Arti NG Treiber

Treiberinformationen

Beschreibung:
Codesys Soft SPS Treiber "Neue Generation" für die Codesys 3S-ARTI (Asynchron Runtime Interface) Schnittstelle. Der Treiber unterstützt die direkte Variablenübernahme aus der Codesys Enwicklungsumgebung und ersetzt den Codesys Arti Soft SPS Treiber.

Für folgende SPS Typen geeignet:
Codesys Soft SPSen, Moeller XControl SPSen XC200 und XC600, sowie Elau PacDrive Steuerungen MAX 4, C200, C400, C600, P600.

Folgende Kopplungsarten werden unterstützt:
Ethernet; Local

- 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, every time 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.

- 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.

Info

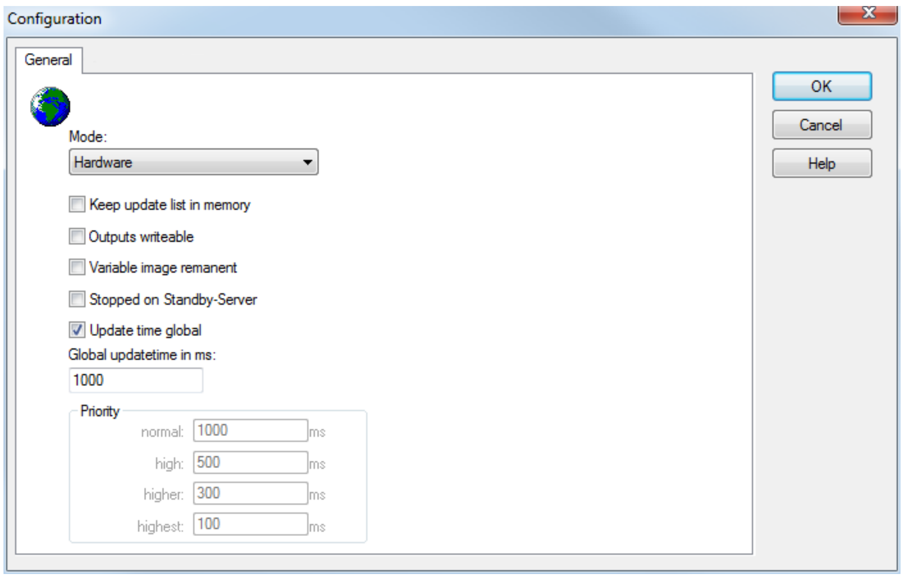
For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:

- Internal
- MathDr32
- SysDrv.

5.2 Settings in the driver dialog

You can change the following settings of the driver:

5.2.1 General


Configuration

General

Mode:
Hardware

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

Global updatetime in ms:
1000

Priority

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

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

`Variable image remanent`	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in mode hardware if: - one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active The variable image is always saved if: - the variable is of the object type `Driver variable` - the driver runs in simulation mode. (not programmed simulation) The following states are not restored at the start of the Runtime: - SELECT(8) - WR-ACK(40) - WR-SUC(41) The mode **`Simulation - programmed`** at the driver start is not a criterion in order to restore the remanent variable image.
`Stop at the Standby Server`	Setting for redundancy at drivers which allow only on communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. **Attention:** If this option is active, the gapless archiving is no longer guaranteed. `Active`: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **`switched off`** (**`statusverarbeitung.chm::/24150.htm`**) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.
`Global Update time`	`Active`: The set `Global update time` in `ms` is used for all variables in the project. The priority set at the variables is not used. `Inactive`: The set priorities are used for the individual variables.
`Priority`	Here you set the polling times for the individual priorities. All variables with the according priority are polled in the set time. The allocation is taken

	place for each variable separately in the settings of the variable properties. The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities. Thus the communication load is distributed better.

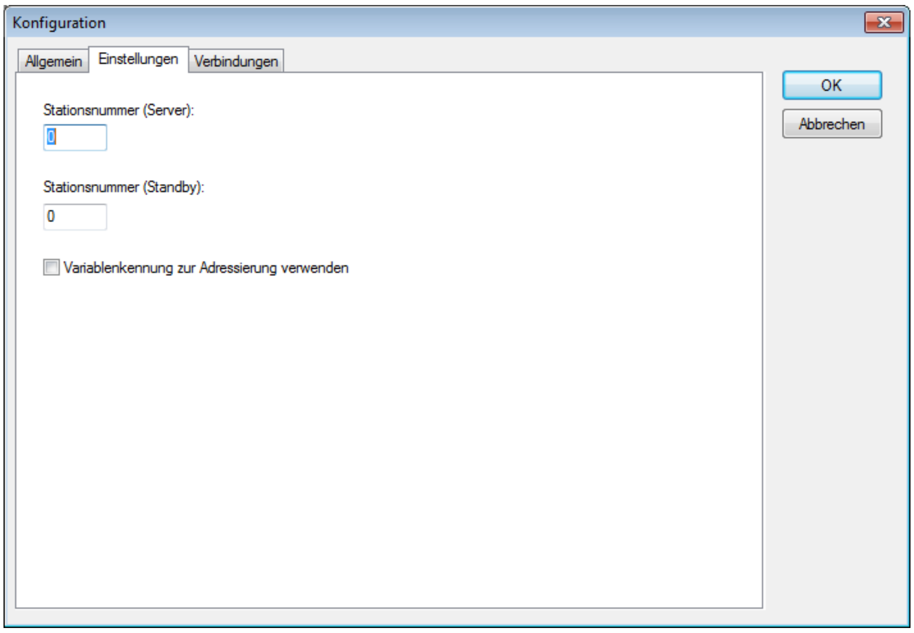
UPDATE TIME FOR CYCLICAL DRIVER

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.

5.2.2 Settings

General settings for communication are carried out in this tab. These settings apply to all connections (on page 13) that have been created in the driver.


Konfiguration

Allgemein | Einstellungen | Verbindungen

Stationsnummer (Server): 0

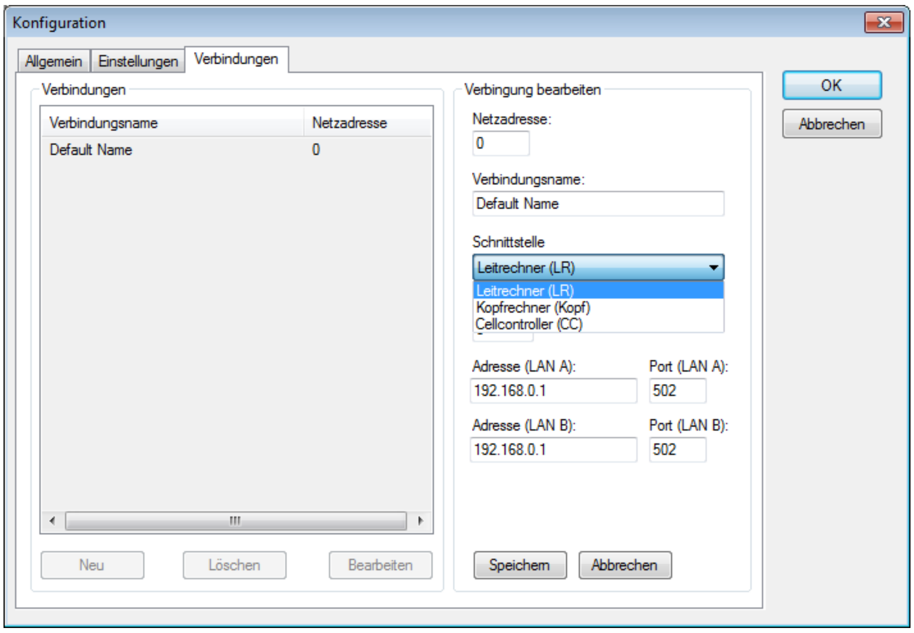
Stationsnummer (Standby): 0

☐ Variablenkennung zur Adressierung verwenden

Property	Description
`Station number (server):`	Station number of the zenon server.
`Station number (standby):`	Station number of the zenon standby server.
`Use variable identification for addressing`	`Active`: The identification instead of the name is used for symbolic addressing.

5.2.3 Connections

On this tab you carry out the settings for the connection. You can create any number of connections in the driver. All connections use the global settings, which are defined in the Settings (on page 12) tab.



Property	Description
Connections	List of the connections created and their net addresses.
Connection name	Name of connection.
New	Adds new entry to the list. Settings are carried out in the field to the right of the list.
Delete	Deletes selected entry from the list.
Edit	Makes it possible to configure the selected entry. The area to the right of the list for editing the connection is activated. **Attention!** The driver dialog cannot be closed in editing mode. Only once the editing mode has been left using **Save** or **Cancel** can you close the driver dialog.
Edit connection	Connection settings for a new connection or the connection selected under **Connection**.
Net address	The net address identifies the connection. Therefore, every connection must have a unique net address. Variables are assigned to a connection via the net address. **Attention!** If you change the net address here, you must also change the net address of all other variables of the connection. Otherwise the allocation is no longer correct and variables cannot communicate in the Runtime!
Connection name	Connection name. Freely definable name.
Interface	Selection of the connection from the drop-down list: ▶ Control computer (LR) ▶ Cell controller (CC) ▶ Head computer (head)
Station number	Station number of the remote station.
Address (LAN-A)	IP address or DNS name of the remote station in the LAN-A.

`Port (LAN-A)`	TCP port of the remote station in the LAN-A.
`Address (LAN-B)`	IP address or DNS name of the remote station in the LAN-B.
`Port (LAN-B)`	TCP port of the remote station in the LAN-B.
Save	Saves changes.
Cancel	Discards changes.

Hint: Driver messages can be read with the Diagnosis Viewer (main.chm::/12464.htm).

CREATE NEW CONNECTION

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

EDIT CONNECTION

1. select the connection in the connection list
2. click on the button **Edit**
3. change the connection parameters
4. finish with **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

6. Creating variables

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

6.1 Creating variables in the Editor

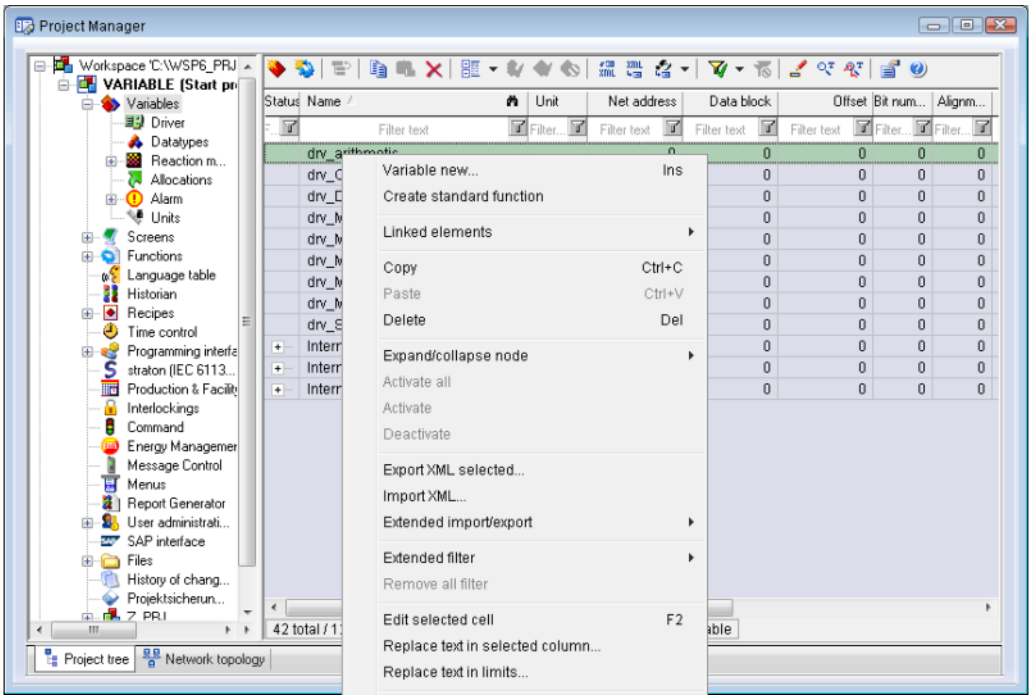
Variables can be created:

- 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:

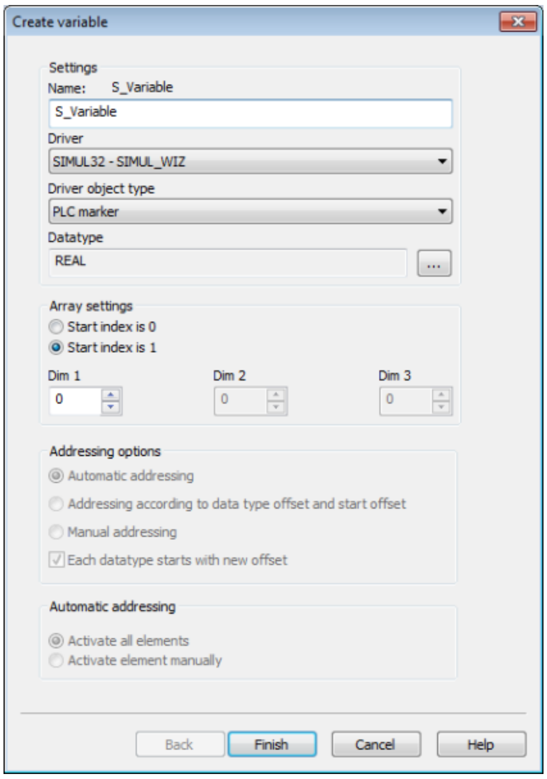
1. Select the **New variable** command in the **Variables** node in the context menu



2. The dialog for configuring variables is opened

3. configure the variable

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



Create variable

Settings

Name: S_Variable

S_Variable

Driver

SIMUL32 - SIMUL_WIZ

Driver object type

PLC marker

Datatype

REAL

Array settings

Start index is 0 [✓ Start index is 1]

Dim 1: 0 | Dim 2: 0 | Dim 3: 0

Addressing options

[✓ Automatic addressing] Addressing according to data type offset and start offset | Manual addressing

[✓] Each datatype starts with new offset

Automatic addressing

[✓ Activate all elements] Activate element manually

Property	Description
Name	Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name. **Attention:** The **#** character is not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive.
Drivers	Select the desired driver from the drop-down list. **Note:** If no driver has been opened in the project, the driver for internal variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is automatically loaded.
Driver object type (cti.chm::/28685.htm)	Select the appropriate driver object type from the drop-down list.

Data type	Select the desired data type. Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the Arrays chapter.
Addressing options	Expanded settings for arrays and structure variables. You can find details in the respective section.
Automatic element activation	Expanded settings for arrays and structure variables. You can find details in the respective section.

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.

6.2 Addressing

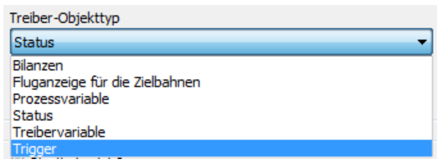
Group/Property	Description
General	
Name	Variable name. **ATTENTION:** For every zenon project the name must be unambiguous. Depending on the settings of Use identification as external name in the **Settings** (on page 12) dialog, the variable is addressed using its name or the identification set for it.
Identification	Variable name. **ATTENTION:** For every zenon project the name must be unambiguous. Depending on the settings of Use identification as external name in the **Settings** (on page 12) dialog, the variable is addressed using its name or the identification set for it.
Addressing	
Net address	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.
Data block	not used for this driver
Offset	not used for this driver
Alignment	not used for this driver
Bit number	Number of the bit within the configured offset. Possible entries: 0 ... 65535
String length	Only available for String variables: Maximum number of characters that the variable can take.
Driver connection/Driver object type	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Driver connection/Data type	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. **ATTENTION:** If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

ADDRESSING

Detailed information for addressing is available for the object types:

- Flight display for the target tracks (on page 20)
- Process variables (on page 20)
- Status (on page 22)

The following object types (on page 26) are generally available:



6.2.1 Flight display for target tracks

Flight displays are addressed using the target number in the system.

The complete address comprises:

- Any desired **`prefix`**
- An exclamation mark (**`!`**) as a separator
- The string **`FLUG_`**
- The target number

Example: **`WE052!FLUG_1234`**

6.2.2 Process variables

The addressing of process variables is different on the control computer and on the cell controller.

Attention

If a variable is written that only displays part of a process variable, all other bits are set to `0`.

CONTROL COMPUTER (LR)

The address comprises:

- Any desired **`prefix`**
- An exclamation mark (**`!`**) as a separator
- The **`Area identification`**
- An underscore (_) as a separator
- The actual **`process variable names`**
- A period (.) as a separator
- A **`suffix`**

Info: Prefix and suffix are optional

Example:

With prefix and suffix: **`WEO52!W_PVXX1.QWord`** or **`WEO52!W_PVXX1._[0]`**
In short: **`W_PVXX1`**

CELL CONTROLLER (CC)

The address comprises:

- Any desired **`prefix`**
- An exclamation mark (**`!`**) as a separator
- The actual **`process variable names`**
- A period (.) as a separator
- A **`suffix`**

Info: Prefix and suffix are optional

Example:

With prefix and suffix: **`WEO52!PVXX1.QWord`** or **`WEO52!PVXX1._[0]`**
In short: **`PVXX1`**

SUFFIXES

A part of the 64 bit process variable can be addressed using the suffix. The following suffixes can be used for the control computer and cell controller:

- `QWord` (64 bit)
- `_[0], _[1]` (32 bit, double word)
- `Word_[0] ... Word_[3]` (16 bit, word)
- `Byte_[0] ... Byte_[7]` (8 bit, byte)
- `Bit_[0] ... Bit_[63]` (bit)

6.2.3 Status

The address comprises:

- Any desired `prefix`
- An exclamation mark (`!`) as a separator
- The identifications for status information stated below

CONTROL COMPUTER (LR), HEAD COMPUTER (HEAD)

- LEITRECHNERKENNUNG
  Addressing example: `WEO52!LEITRECHNERKENNUNG`
  The control computer is set as follows:
  - 'H', 72 ... HAUS
  - 'O', 79 ... STO
  - 'V', 86 ... V3
  - 'W', 87 ... A-Plus
  - 'K', 75 ... Head computer
- BETRIEBSZUSTAND
  Addressing example: `WEO52!BETRIEBSZUSTAND`
  The following control computer states are possible:

- '1', 31 ... Ready for operation
- '2', 32 ... Not ready for operation

▶ LOG_LINKZUSTAND[0] ... LOG_LINKZUSTAND[6]

  Addressing example: **`WEO52!LOG_LINKZUSTAND[2]`**

  The logical link state displays whether data exchange is permitted with a computer or not. Only the local connection state (local perspective) can be changed by command in the dialog. The character array contains a corresponding letter for each system. The letter of its own system is always set.

  - 'H', 72 ... HAUS
  - 'O', 79 ... STO
  - 'V', 86 ... V3
  - 'W', 87 ... A-Plus
  - 'K', 75 ... Head computer
  - 'I', 73 ... Info-Plus computer
  - 'Z', 90 ... Visualization server
  - '_', 95 ... Connection inactive

▶ PHYS_LINKZUSTAND[0] ... PHYS_LINKZUSTAND[6]

  Addressing example: **`WEO52!PHYS_LINKZUSTAND[6]`**

  The physical connection status is kept internally in the PROCOM and cannot be influenced. Provides information on whether data exchange with a computer is technically possible.

  - 'H', 72 ... HAUS
  - 'O', 79 ... STO
  - 'V', 86 ... V3
  - 'W', 87 ... A-Plus
  - 'K', 75 ... Head computer
  - 'I', 73 ... Info-Plus computer
  - 'Z', 90 ... Visualization server
  - '_', 95 ... Connection inactive

▶ STATUS_NODE_A

  Addressing example: **`WEO52!STATUS_NODE_A`**

  - 'M', 77 ... Node A in the master status
  - 'S', 83 ... Node A in standby status

- STATUS_NODE_B
  Addressing example: **WEO52!STATUS_NODE_B**
  - 'M', 77 ... Node A in the master status
  - 'S', 83 ... Node A in standby status
- GSV_MODE_NODE_A
  Addressing example: **WEO52!GSV_MODE_NODE_A**
  - 'N', 78 ... Node A in restart status
- GSV_MODE_NODE_B
  Addressing example: **WEO52!GSV_MODE_NODE_B**
  - 'N', 78 ... Node A in restart status
- REPL_NODE_A
  Addressing example: **WEO52!REPL_NODE_A**
  - 'H', 72 ... Replication active on node A (hot)
  - 'C', 67 ... Replication inactive on node A (cold)
- REPL_NODE_B
  Addressing example: Addressing example: **WEO52!REPL_NODE_B**
  - 'H', 72 ... Replication active on node A (hot)
  - 'C', 67 ... Replication inactive on node A (cold)
- AKTUELLE_ZEIT
  Addressing example: Addressing example: **WEO52!AKTUELLE_ZEIT**
  Current time on the system
- VERBINDUNGS_STATUS
  Addressing example: Addressing example: **WEO52!VERBINDUNGS_STATUS**
  This variable provides the drive connection status to the control computer or head computer
  - 0 ... No action
  - 1 ... General query after new connection active
  - 2 ... General query initiated by user active
  - 3 ... Balance query active
- VERBINDUNGS_ZUSTAND:
  Addressing example: Addressing example: **WEO52!VERBINDUNGS_ZUSTAND**
  This variable provides the connection status of the driver to the control computer or head computer

- 3 ... Unknown
- 2 ... Both faulty
- 1 ... One faulty
- 0 ... Both connected

CELL CONTROLLER (CC)

- BETRIEBSZUSTAND
  Addressing example: Addressing example: **`WEO52!VERBINDUNGS_ZUSTAND`**
  The following control computer states are possible:
  - '1', 31 ... Ready for operation
  - '2', 32 ... Not ready for operation
  - '3', 33 ... Initialization active
- BETRIEBSZUSTAND_WEXPERTE[0] ... BETRIEBSZUSTAND_WEXPERTE[99]
  Addressing example: Addressing example: **`WEO52!BETRIEBSZUSTAND_WEXPERTE[0]`**
  - '1', 31 ... Ready for operation
  - '2', 32 ... Not ready for operation
- LOG_LINKZUSTAND
  Addressing example: Addressing example: **`WEO52!LOG_LINKZUSTAND`**
  The logical link state displays whether data exchange is permitted with a computer or not. Only the local connection state (local perspective) can be changed by command in the dialog. The character array contains a corresponding letter for each system. The letter of its own system is always set.
  - 'L', 76 ... Connection active
  - '?', 63... Connection state unknown
  - '_', 95 ... Connection inactive
- PHYS_LINKZUSTAND
  Addressing example: Addressing example: **`WEO52!PHYS_LINKZUSTAND`**
  The physical connection status is kept internally in the PROCOM and cannot be influenced. Provides information on whether data exchange with a computer is technically possible.
  - 'L', 76 ... Connection active
  - '?', 63... Connection state unknown
  - '_', 95 ... Connection inactive

- AKTUELLE_ZEIT

  Addressing example: Addressing example: **`WEO52!AKTUELLE_ZEIT`**

  Current time on the system

- VERBINDUNGS_STATUS

  Addressing example: Addressing example: **`WEO52!VERBINDUNGS_STATUS`**

  This variable provides the drive connection status to the control computer or head computer

  - 0 ... No action
  - 1 ... General query after new connection active
  - 2 ... General query initiated by user active

- VERBINDUNGS_ZUSTAND:

  Addressing example: Addressing example: **`WEO52!VERBINDUNGS_ZUSTAND`**

  This variable provides the driver connection status to the cell controller.

  - 3 ... Unknown
  - 2 ... Both faulty
  - 1 ... One faulty
  - 0 ... Both connected

6.3 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.

6.3.1 Driver objects

The following object types are available in this driver:

Driver object type	Channel type	Read / Write	Supported data types	Description
Process variable	8	R / W	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, LINT, LWORD	Process variables.
Status	64	R	USINT, STRING, DATE_AND_TIME	Status.
Flight display for the target tracks	65	R	STRING	Flight display for the target tracks.
Balancing	66	R	STRING	Balancing the head computer
Trigger	67	S	UINT	Trigger for manual execution of a general query or head query (balance query).
Driver variable	35	R / W	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variable for the static analysis of the communication; is transferred between driver and Runtime (not to the PLC). **Note**: The addressing and the behavior is the same for most zenon drivers. Find out more in the chapter about the Driver variables (on page 35)

6.3.2 Mapping of the data types

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

PLC	zenon	Data type
ULINT	BOOL	8
ULINT	USINT	9
ULINT	SINT	10
ULINT	UINT	2
ULINT	INT	1
ULINT	UDINT	4
ULINT	DINT	3
ULINT	ULINT	27
ULINT	LINT	26
-	REAL	5
-	LREAL	6
-	STRING	12
-	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.

6.4 Creating variables by importing

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

6.4.1 XML import of variables from another zenon project

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.*

6.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

IMPORT DBF FILE

To start the import:

1. right-click on the variable list
2. in the drop-down menu 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.

Info

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 menu 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.

Info

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) when exporting.

DBF files must:

- correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
- Be stored close to the root directory (Root)

DESIGN

Description	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the MAX_LAENGE entry in **project.ini** .
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "KANALNAME" (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in **project.ini** .
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the MAX_LAENGE entry in **project.ini** .
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Bus address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset

BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipe Group Manager
LES_SCHR	R	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 KANALTYP and DATENART
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	Resource label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in **project.ini** .
ADJWVBA	R	1	Non-linear value adaption: 0: Non-linear value adaption is used 1: non linear value adaption is not used
ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	Linked 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 DEFINITION

Limit definition for limit values 1 to 4, and status 1 to 4:

Description	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 (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
HYSTERESE1	F	14	Hysteresis in %
BLINKEN1	R	1	Set blink attribute
BTB1	R	1	Logging in CEL
ALARM1	R	1	Alarm
DRUCKEN1	R	1	Printer output (for CEL or Alarm)
QUITTIER1	R	1	Must be acknowledged
LOESCHE1	R	1	Must be deleted
VARIABLE1	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	R	1	Function linking
ASK_FUNC1	R	1	With interrogation before execution
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.

6.5 Driver variables

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

- Information
- Configuration
- Statistics and
- Error messages

The definitions of the variables defined in the driver kit are available in the import file **`drvvar.dbf`** (on the CD in the directory: `CD_Drive:/Predefined/Variables`) and can be imported from there.

Hint: 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.

Info

Not every driver supports all driver variants.

For example:

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

INFORMATION

Name from import	Type	Offset	Description
MainVersion	UINT	0	Main version number of the driver.
SubVersion	UINT	1	Sub version number of the driver.
BuildVersion	UINT	29	Build version number of the driver.
RTMajor	UINT	49	zenon main version number
RTMinor	UINT	50	zenon sub version number
RTSp	UINT	51	zenon service pack number
RTBuild	UINT	52	zenon build number
LineStateIdle	BOOL	24.0	TRUE, if the modem connection is idle
LineStateOffering	BOOL	24.1	TRUE, if a call is received
LineStateAccepted	BOOL	24.2	The call is accepted
LineStateDialtone	BOOL	24.3	Dialtone recognized
LineStateDialing	BOOL	24.4	Dialing active
LineStateRingBack	BOOL	24.5	While establishing the connection
LineStateBusy	BOOL	24.6	Target station is busy
LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
LineStateProceeding	BOOL	24.9	Dialing completed
LineStateOnHold	BOOL	24.10	Connection in hold
LineStateConferenced	BOOL	24.11	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	BOOL	24.13	Connection in hold for transfer
LineStateDisconnected	BOOL	24.14	Connection stopped
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped

			For `driver stop`, the variable has the value `TRUE` and an **`OFF`** bit. After the driver has started, the variable has the value `FALSE` and no **`OFF`** bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	BOOL	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	BOOL	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings **`PhoneNumberSet`** and **`ModemHwAdrSet`**.
PhoneNumberSet	STRING	38	Telephone number, that should be used
ModemHwAdrSet	DINT	39	Hardware address for the telephone number
GlobalUpdate	UDINT	3	Update time in milliseconds (ms).
BGlobalUpdaten	BOOL	4	TRUE, if update time is global
TreiberSimul	BOOL	5	TRUE, if driver in sin simulation mode
TreiberProzab	BOOL	6	TRUE, if the variables update list should be kept in the memory
ModemActive	BOOL	7	TRUE, if the modem is active for the driver

Device	STRING	8	Name of the serial interface or name of the modem
ComPort	UINT	9	Number of the serial interface.
Baud rate	UDINT	10	Baud rate of the serial interface.
Parity	SINT	11	Parity of the serial interface
ByteSize	SINT	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.
StopBit	SINT	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	UDINT	54	Time in seconds (s) to establish a connection.

STATISTICS

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

PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

ERROR MESSAGES

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	UDINT	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	SINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

7. Driver-specific functions

This driver supports the following functions:

MANUAL INITIATION OF A GENERAL QUERY

- Creation of a `trigger` driver object variable and the name or identification consisting of an optional prefix and the character sequence `GA`.
  For example: `MyPrefix!GA` or `GA`
- Assignment to a connection is carried out using the network address.
- A general query is triggered by writing a value to a variable. In doing so, the value written is used as a `GA` type.
  - GA type: 1 ... All PVs
  - GA type: 2 ... Only special PVs

  Info

  The status of the general query can be queried using the VERBINDUNGS_STATUS status variable.

INITIATION OF A BALANCE QUERY FROM THE HEAD COMPUTER

- Creation of a `balances` driver object string variable.
- Creation of a `trigger` driver object variable and the name or identification consisting of an optional prefix and the character sequence `BILANZEN`.
  For example: `MyPrefix!BILANZEN` or `BILANZEN`
- Assignment of variables to a connection is carried out using the network address.
- A head query is triggered by writing a value to a variable. In doing so, the value written is used as a `statistic` type.
  - Statistic type: `1` ... statistic for live operation
  - Statistic type: `2` ... operational transmission statistics

  Info

  The status of the head query can be queried using the VERBINDUNGS_STATUS status variable.

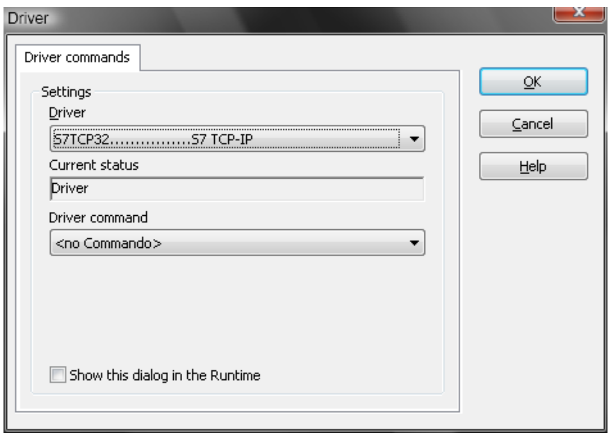
  The balance information is stored in a `balance` driver object string variable.
  Assignment to a connection is carried out using the network address.

8. 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


Driver
Driver commands
Settings
Driver
S7TCP32...............S7 TCP-IP
Current status
Driver
Driver command
<no Commando>
Show this dialog in the Runtime

Parameters	Description
`Drivers`	Drop-down list with all drivers which are loaded in the project.
`Current state`	Fixed entry which has no function in the current version.
`Driver commands`	Drop-down list for the selection of the command.
▸ `Start driver (online mode)`	Driver is reinitialized and started.
▸ `Stop driver (offline mode)`	Driver is stopped. No new data is accepted. **Note:** If the driver is in offline mode, all variables that were created for this driver receive the status `switched off` (`OFF`; Bit 20).

▸ Driver in simulation mode	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▸ Driver in hardware mode	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▸ Driver-specific command	Enter driver-specific commands. Opens input field in order to enter a command.
▸ Activate driver write set value	Write set value to a driver is allowed.
▸ Deactivate driver write set value	Write set value to a driver is prohibited.
▸ Establish connection with modem	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
▸ Disconnect from modem	Terminate connection (for modem drivers)
Show this dialog in the Runtime	The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

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

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

9. Error analysis

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

9.1 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 7.00 -> Diagviewer.*

zenon driver log all errors in the log files. The default folder for the log files is subfolder **LOG** in directory `ProgramData`, example: `C:\ProgramData\zenon \zenon700\LOG` for zenon version 7.00 SP0. 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 currently created entries live
- customize the logging settings
- change the folder in which the log files are saved

Hints:

1. In Windows CE even errors are not logged per default due to performance reasons.
2. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
3. 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.
4. If you only use **Error logging**, the problem description is in column **Error text**. For other diagnosis level the description is in column **General text**.
5. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** and/or **Error code** and/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.

6. 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**.

Info

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

9.2 Check list

Questions and hints for fault isolation:

GENERAL TROUBLESHOOTING

- Analysis with the **Diagnosis Viewer** (on page 44):
  -> Which messages are displayed?
- Are the participants available in the **TCP/IP** network?
- Can the PLC 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 PLC be reached via `Telnet`?

Telnet: *Open command line, telent <enter IP address port number> (e.g. for Modbus: telnet 192.168.0.100 502) -> press Enter.*

If the monitor display turns black, a connection could be established.

- 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?

  Example: Driver variables are purely statistics variables. They do not communicate with the PLC. (See chapter Driver variable (on page 35).)

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.

VARIABLES ARE DISPLAYED WITH A BLUE DOT

The communication in the network is faulty:

- With a network project:
  Is the network project also running on the server?
- With a stand-alone project or a network project which is also running on the server:
  Deactivate the property `Only read from standby` in node `Driver connection`/`Addressing`.

VALUES ARE DISPLAYED INCORRECTLY

Check the information for the calculation in node `Value calculation` of the variable properties.

DRIVER FAILS OCCASIONALLY

Analysis with the **`Diagnosis Viewer`** (on page 44):
-> Which messages are displayed?