zenon driver manual

BrTcp32

v.7.00

© 2012 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. The technical data contained herein has been provided solely for informational purposes and is not legally binding. Subject to change, technical or otherwise.

Contents

1. Welcome to COPA-DATA help 5

2. BrTcp32 5

3. BRTCP32 - Data sheet 5

4. Driver history 7

5. Requirements 7

5.1 PC 8

5.2 PLC 8

6. Configuration 9

6.1 Creating a driver 9

6.2 Settings in the driver dialog 10

6.2.1 General 11

6.2.2 Basic setting 14

6.2.3 Conn. TCP/IP 15

7. Creating variables 16

7.1 Creating variables in the Editor 17

7.2 Addressing 20

7.3 Driver objects and datatypes 20

7.3.1 Driver objects 21

7.3.2 Mapping of the data types 23

7.4 Creating variables by importing 23

7.4.1 XML import of variables from another zenon project 23

7.4.2 DBF Import/Export 24

7.5 Driver variables 30

8. Driver-specific functions 35

9. Driver commands 36

10. Error analysis 38

10.1 Analysis tool 38
10.2 Error numbers 41
10.3 Check list 56

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. BrTcp32

3. BRTCP32 - Data sheet

General:	
Driver file name	BRTCP32.exe
Driver description	BR TCP-IP
PLC types	B+R System 2000 and B+R Burklimat
PLC manufacturer	Bernecker + Rainer;

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

Prerequisites:	
Hardware PC	Standard network card
Software PC	-
Hardware PLC	-
Software PLC	Needs special software in the B+R System 2000 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
07.07.08	1800	Created driver documentation

5. Requirements

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

5.1 PC

HARDWARE

Network card

Interface: Ethernet

Protocol: B&R TCP

SOFTWARE

Copy the driver file BrTcp32.exe into the current program directory (unless it is already there) and enter it into the TREIBER_EN.XML file with the tool DriverInfo.exe.

The configuration file (according to the driver settings) must be sent to the project directory on the target PC. For this, the file has to be entered in the Remote Transport explicitly.

5.2 PLC

HARDWARE

B&R System 2000

Interface: Ethernet

Protocol: B&R TCP

Ethernet connection (TCP/IP). The handling function block must be installed and configured on the PLC.

6. Configuration

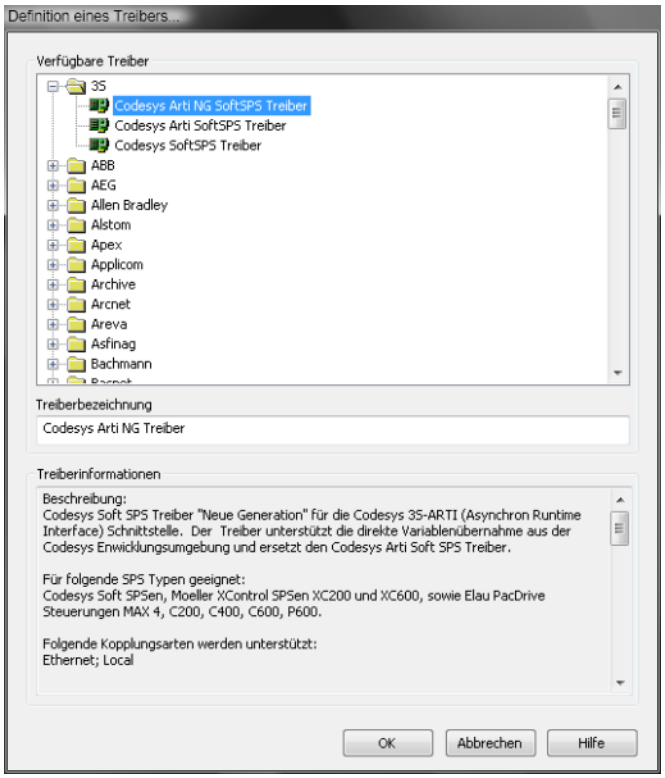
In this chapter you will learn how to use the driver in a project and which settings you can change.

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

6.1 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.



- 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.

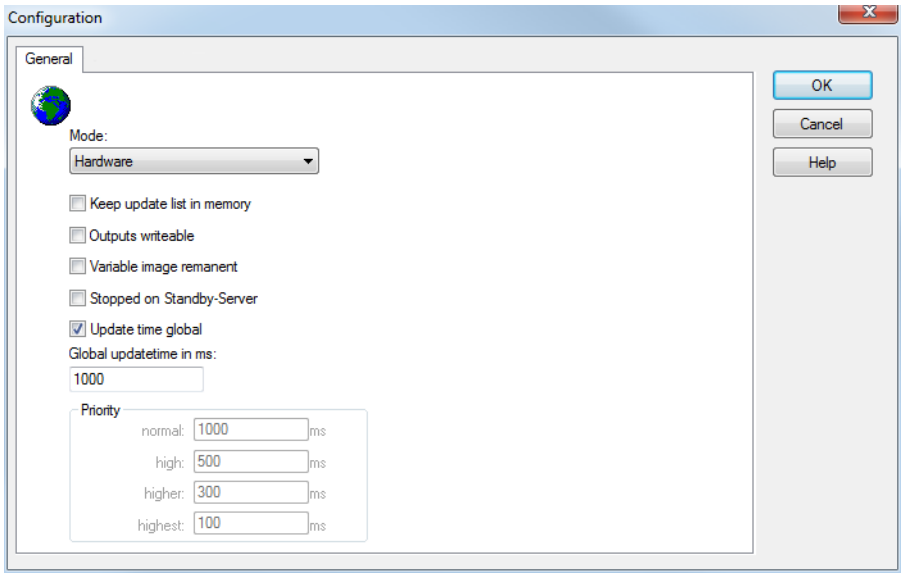
> **Info**
>
> *For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:*
>
> - Internal
> - MathDr32
> - SysDrv.

▸

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 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
OK
Cancel
Help

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.

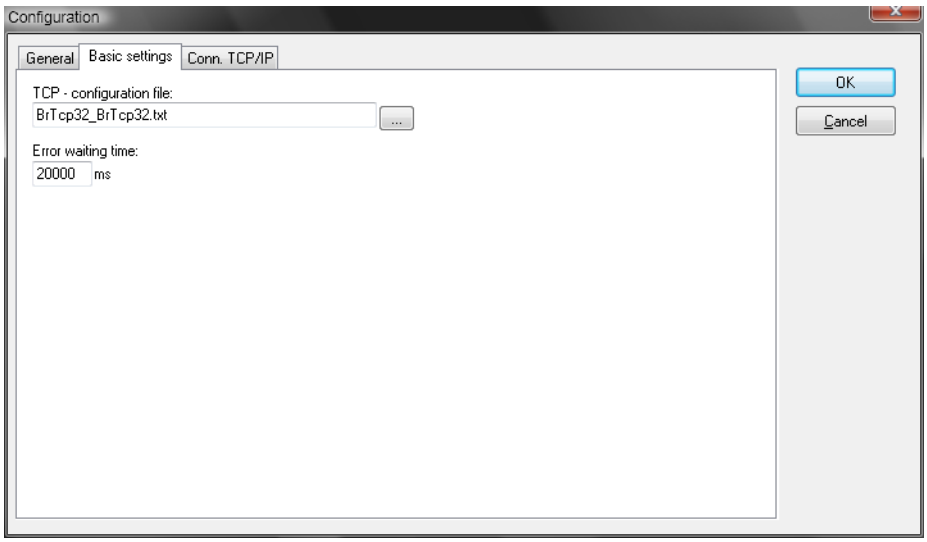
UPDATE TIME FOR CYCLICAL DRIVER

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.

6.2.2 Basic setting

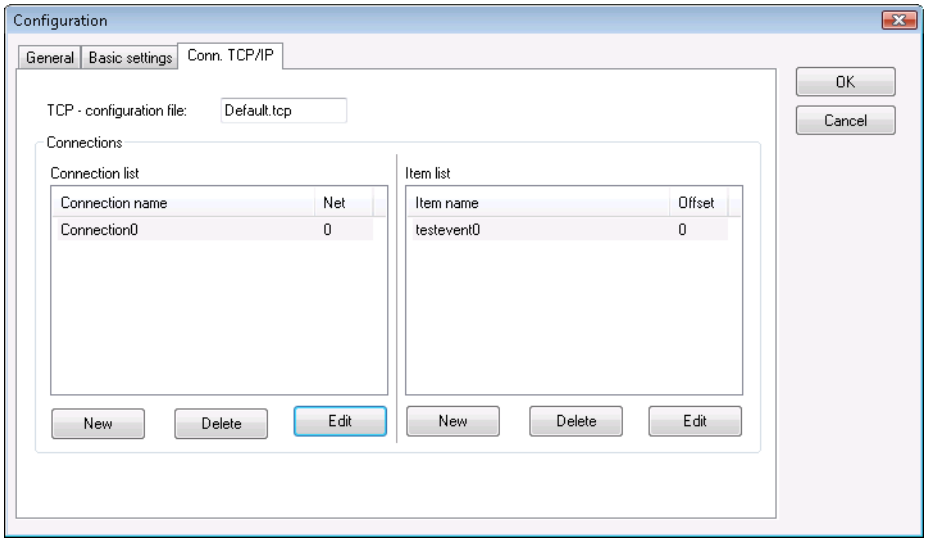
Here all driver specific settings are done.



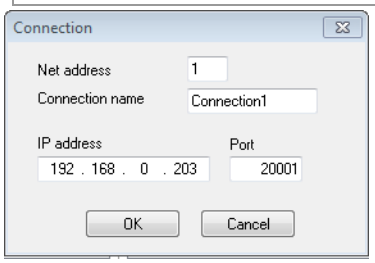
Parameters	Description
`TCP Configuration File`	In this file, all connections and items are stored. It must be placed on the engineering computer and on the runtime computer in the project directory.
`Error Waiting Time`	This is the waiting time for communication errors. After every third part of this time, live messages are sent to the PLC.

6.2.3 Conn. TCP/IP

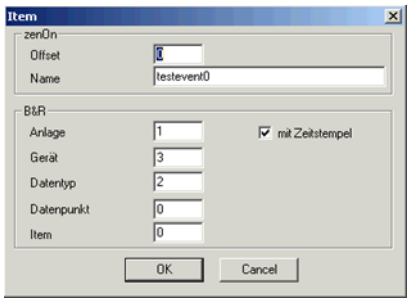
Here you define all connections and their items.



Parameters	Description
Connection list	This list contains all connections. If a connection is selected, the configured items will appear in the 'Item list'.
Item list	This list contains all items that belong to the connection currently selected in the 'connection list'.
Connection	Here the connection settings are defined.



Parameters	Description
Net address	Enter a unique net address that will be used to associate a variable in zenon with this connection.
Connection name	Freely definable name.
IP address	IP address of PLC.
Port	TCP port on the PLC which the server is listening to.



Parameters	Description
Offset	Enter a unique offset that will be used to associate a zenon variable with this item.
Name	Freely definable name.
Equipment	Part of the address in the PLC.
Device	Part of the address in the PLC.
Data type	Part of the address in the PLC.
Data point	Part of the address in the PLC.
Item	Part of the address in the PLC.
With time stamp	Item is logged in with a time stamp.

7. Creating variables

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

7.1 Creating variables in the Editor

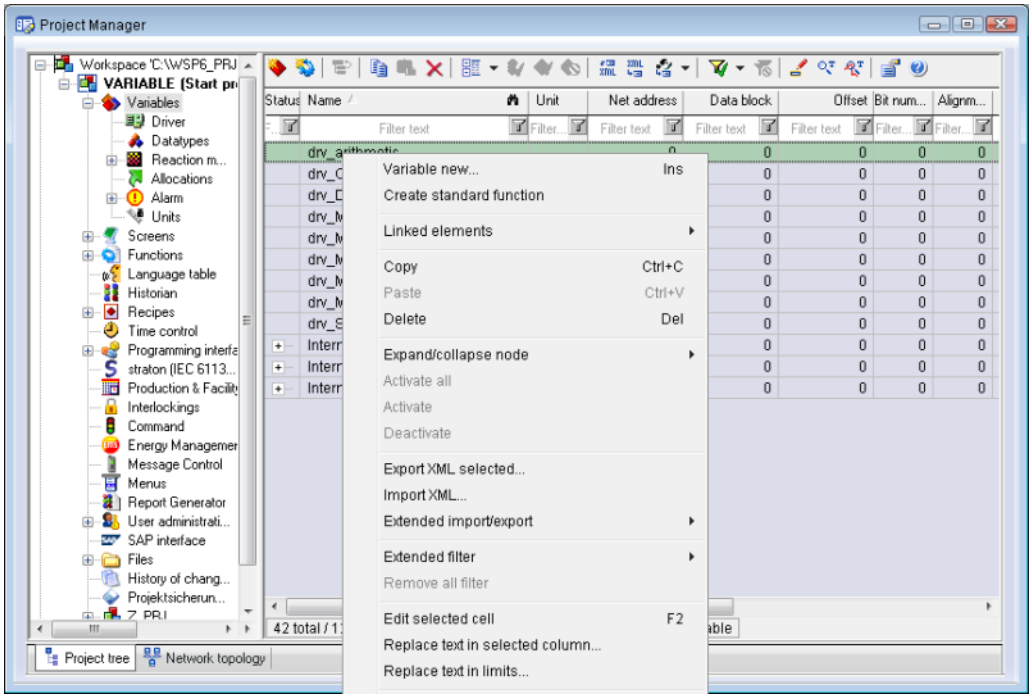
Variables can be created:

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

VARIABLE DIALOG

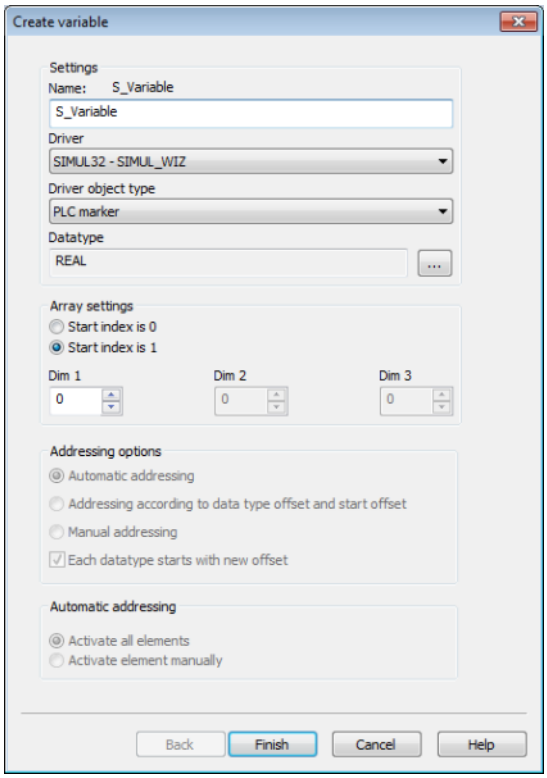
To create a new variable, regardless of which type:

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



2. The dialog for configuring variables is opened
3. configure the variable

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



Property	Description
Name	Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name. **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.

7.2 Addressing

Property	Description
Name	Freely definable name **Attention:** the name must be unique within each control system project.
Identification	Any text can be entered here, e.g. for resource labels, comments ...
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	For variables of object type Extended data block, enter the datablock number here. Configurable [0.. 4294967295] . Please look up the exact maximum range for data blocks in the manual of the PLC.
Offset	Offset of the variable; the memory address of the variable in the PLC. Configurable [0.. 4294967295]
Alignment	not used for this driver
Bit number	Number of the bit within the configured offset. Valid input [0.. 65535], Working range [0..7]
String length	Only available for String variables: Maximum number of characters that the variable can take.
Driver object type	Depending on the employed driver, an object type is selected during the creation of the variable; the type can be changed here later.
Data type	Data type of the variable, which is selected during the creation of the variable; the type can be changed here later. **ATTENTION:** If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

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	Comment
DATASTREAM-TRIGGER	9	R / W	USINT, SINT	
SETCLOCK-TRIGGER	64	R / W	USINT, SINT	
PLC marker	8	R / W	LREAL, BOOL, REAL, DINT, UDINT, INT, UINT, STRING, USINT, SINT	
Driver variable	35	R / W	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. Find out more in the chapter about the Driver variables (on page 30)

OBJECTS FOR PROCESS VARIABLES IN ZENON

Channel name	Data type	Channel type	Object	Read	Write
BIT	8	8	2	Y	Y
BYTE	9	8	3	Y	Y
BYTE _mV	10	8	3	Y	Y
WORD	2	8	4	Y	Y
WORD _mV	1	8	4	Y	Y
DWORD	4	8	5	Y	Y
DWORD _mV	3	8	5	Y	Y
FLOAT	5	8	6	Y	Y
DOUBLE	6	8	6	Y	Y
STRING	12	8	7	Y	Y
DATASTREAM -TRIGGER	9	9	8	Y	Y

SETCLOCK-TRIGGER	9	64	9	N	Y

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.

EXAMPLES FOR ALL POSSIBLE ZENON DATA TYPES

PLC	zenon
I8	i/u8Bit (signed)
I16	i/u16Bit (signed)
I32	i/u32Bit (signed)
U8	i/u8Bit
U16	i/u16Bit
U32	i/u32Bit
F32	float32
F64	double64
Boolean	Boolean
String	String

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

7.4 Creating variables by importing

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

7.4.1 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.*

7.4.2 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.

7.5 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

8. Driver-specific functions

This driver supports the following functions:

INI ENTRIES

ZENON6.INI

[BrTcp32]

LOGFILE=1

Activates the special extended logging of the driver. You can look at the entries with the Diagnosis Viewer (see -> Analysis tools (on page 38))

LIMITATIONS

If you deactivate the option "Keep PV update list in memory", requests will not be processed correctly, because in this case, the PLC will always return the last valid status.

There must always be at least one active variable per driver that is registered in the driver. Otherwise, the connection will not be used anymore and watchdogs etc. will no longer be sent.

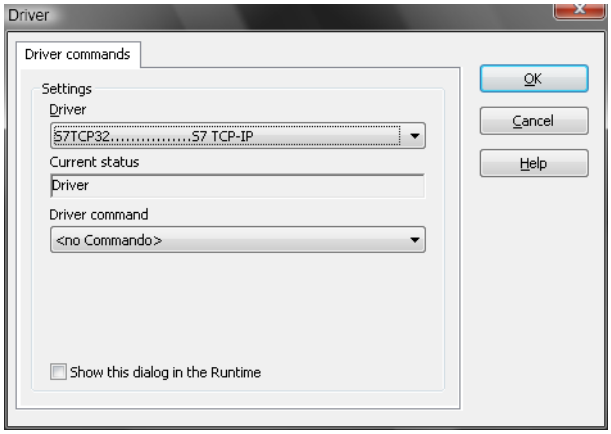
9. Driver commands

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

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

- create a new function
- select *Variables -> Driver commands*

- The dialog for configuration is opened



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 to a driver is prohibited.

write set value	
▸ 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.

10. Error analysis

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

10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under *Start/All programs/zenon/Tools 7.00 -> 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.

10.2 Error numbers

B&R FEHLERCODES

Error message	Code	Error text	Description
BRT_ERR_INVALIDAK	1	Unknown request ID	The server has received a request with an unknown request ID.
BRT_ERR_NO_ITEM	2	Item not found	The item address in the request of a client is erroneous (This message is no longer used and has been replaced by the messages with the error codes 14 to 18)
BRT_ERR_WRONG_LEN	3	Wrong file length	The length specified in the request of a client does not match the length of the addressed item on the server.
BRT_ERR_WRONG_HYST	4	Wrong hysteresis	While loggin in an item, the client has specified an invalid hysteresis
BRT_ERR_IS_LOGGED	5	Item is already logged	The client tries to log in an item that is already logged
BRT_ERR_ITEM_NO_TS	6	Item has no time stamp	The client tries to log in or read an item with time stamp that does not have a time stamp on the server
BRT_ERR_ITEM_NO_LOGON	7	Item is already logged off	The client tries to log off an item that is already logged off
BRT_ERR_NO_STREAM	8	No data stream	While reading or during the spontaneous transmission of a data stream, the specified address does not refer to a data stream
BRT_ERR_RTC_SET	9	Write error from realtime clock	The realtime clock of the server could not be set

BRT_ERR_ITEM_NOT_WRITEABLE	10	Item write-protected	The server has received a write request for a write-protected item
BRT_ERR_WRITE_PIPE_FULL	11	Write buffer is full.	The maximum number of pending write requests on the server has been exceeded
BRT_ERR_ITEM_NO_DATA	12	Item has no valid data	At the time a client registers, there is no valid data for the addressed item
BRT_ERR_SWC_SET	13	Write error by SW clock	The software clock of the server could not be set
BRT_ERR_UNKNOWN_UNIT	14	Unknown equipment number	The server has received a request with an unknown equipment number.
BRT_ERR_UNKNOWN_DEVICE	15	Unknown device	The server has received a request with an unknown device number.
BRT_ERR_UNKNOWN_TYPE	16	Unknown data type	The server has received a request with an unknown datatype index
BRT_ERR_UNKNOWN_POINT	17	Unknown data point	The server has received a request with an unknown data point index
BRT_ERR_WRONG_ITEM	18	Wrong item	The server has received a request with an unknown or invalid item index

ERROR CODES IN THE API

The following is a list of possible error codes returned by the WSAGetLastError call, along with their extended explanations. Errors are listed in alphabetical order by error macro. Some error codes defined in Winsock2.h are not returned from any function-these are not included in this topic.

WSAEACCES

(10013)

Permission denied.

An attempt was made to access a socket in a way forbidden by its access permissions. An example is using a broadcast address for sendto without broadcast permission being set using set-sockopt(SO_BROADCAST).

Another possible reason for the WSAEACCES error is that when the bind function is called (on Windows NT 4 SP4 or later), another application, service, or kernel mode driver is bound to the same address with exclusive access. Such exclusive access is a new feature of Windows NT 4 SP4 and later, and is implemented by using the SO_EXCLUSIVEADDRUSE option.

WSAEADDRINUSE

(10048)

Address already in use.

Typically, only one usage of each socket address (protocol/IP address/port) is permitted. This error occurs if an application attempts to bind a socket to an IP address/port that has already been used for an existing socket, or a socket that was not closed properly, or one that is still in the process of closing. For server applications that need to bind multiple sockets to the same port number, consider using set-sockopt(SO_REUSEADDR). Client applications usually need not call bind at all-connect chooses an unused port automatically. When bind is called with a wildcard address (involving ADDR_ANY), a WSAEADDRINUSE error could be delayed until the specific address is committed. This could happen with a call to another function later, including connect, listen, WSAConnect, or WSAJoinLeaf.

WSAEADDRNOTAVAIL

(10049)

Cannot assign requested address.

The requested address is not valid in its context. This normally results from an attempt to bind to an address that is not valid for the local machine. This can also result from connect, sendto, WSAConnect, WSAJoinLeaf, or WSASendTo when the remote address or port is not valid for a remote machine (for example, address or port 0).

WSAEAFNOSUPPORT

(10047)

Address family not supported by protocol family.

An address incompatible with the requested protocol was used. All sockets are created with an associated address family (that is, AF_INET for Internet Protocols) and a generic protocol type (that is, SOCK_STREAM). This error is returned if an incorrect protocol is explicitly requested in the socket call, or if an address of the wrong family is used for a socket, for example, in sendto.

WSAEALREADY

(10037)

Operation already in progress.

An operation was attempted on a nonblocking socket with an operation already in progress-that is, calling connect a second time on a nonblocking socket that is already connecting, or canceling an asynchronous request (WSAAsyncGetXbyY) that has already been canceled or completed.

WSAECONNABORTED

(10053)

Software caused connection abort.

An established connection was aborted by the software in your host machine, possibly due to a data transmission time-out or protocol error.

WSAECONNREFUSED

(10061)

Connection refused.

No connection could be made because the target machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host-that is, one with no server application running.

WSAECONNRESET

(10054)

Connection reset by peer.

An existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, or the remote host uses a hard close (see setsockopt for more information on the SO_LINGER option on the remote socket.) This error may also result if a connection was broken due to keep-alive activity detecting a failure while one or more operations are in progress. Operations that were in progress fail with WSAENETRESET. Subsequent operations fail with WSAECONNRESET.

WSAEDESTADDRREQ

(10039)

Destination address required.

A required address was omitted from an operation on a socket. For example, this error is returned if sendto is called with the remote address of ADDR_ANY.

WSAEFAULT

(10014)

Bad address.

The system detected an invalid pointer address in attempting to use a pointer argument of a call. This error occurs if an application passes an invalid pointer value, or if the length of the buffer is too small. For instance, if the length of an argument, which is a SOCKADDR structure, is smaller than the sizeof(SOCKADDR).

WSAEHOSTDOWN

(10064)

Host is down.

A socket operation failed because the destination host is down. A socket operation encountered a dead host. Networking activity on the local host has not been initiated. These conditions are more likely to be indicated by the error WSAETIMEDOUT.

WSAEHOSTUNREACH

(10065)

No route to host.

A socket operation was attempted to an unreachable host. See WSAENETUNREACH.

WSAEINPROGRESS

(10036)

Operation now in progress.

A blocking operation is currently executing. Windows Sockets only allows a single blocking operation-per- task or thread-to be outstanding, and if any other function call is made (whether or not it references that or any other socket) the function fails with the WSAEINPROGRESS error.

WSAEINTR

(10004)

Interrupted function call.

A blocking operation was interrupted by a call to WSACancelBlockingCall.

WSAEINVAL

(10022)

Invalid argument.

Some invalid argument was supplied (for example, specifying an invalid level to the setsockopt function). In some instances, it also refers to the current state of the socket-for instance, calling accept on a socket that is not listening.

WSAEISCONN

(10056)

Socket is already connected.

A connect request was made on an already-connected socket. Some implementations also return this error if sendto is called on a connected SOCK_DGRAM socket (for SOCK_STREAM sockets, the to parameter in sendto is ignored) although other implementations treat this as a legal occurrence.

WSAEMFILE

(10024)

Too many open files.

Too many open sockets. Each implementation may have a maximum number of socket handles available, either globally, per process, or per thread.

WSAEMSGSIZE

(10040)

Message too long.

A message sent on a datagram socket was larger than the internal message buffer or some other network limit, or the buffer used to receive a datagram was smaller than the datagram itself.

WSAENETDOWN

(10050)

Network is down.

A socket operation encountered a dead network. This could indicate a serious failure of the network system (that is, the protocol stack that the Windows Sockets DLL runs over), the network interface, or the local network itself.

WSAENETRESET

(10052)

Network dropped connection on reset.

The connection has been broken due to keep-alive activity detecting a failure while the operation was in progress. It can also be returned by setsockopt if an attempt is made to set SO_KEEPALIVE on a connection that has already failed.

WSAENETUNREACH

(10051)

Network is unreachable.

A socket operation was attempted to an unreachable network. This usually means the local software knows no route to reach the remote host.

WSAENOBUFS

(10055)

No buffer space available.

An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full.

WSAENOPROTOOPT

(10042)

Bad protocol option.

An unknown, invalid or unsupported option or level was specified in a getsockopt or setsockopt call.

WSAENOTCONN

(10057)

Socket is not connected.

A request to send or receive data was disallowed because the socket is not connected and (when sending on a datagram socket using sendto) no address was supplied. Any other type of operation might also return this error-for example, setsockopt setting SO_KEEPALIVE if the connection has been reset.

WSAENOTSOCK

(10038)

Socket operation on nonsocket.

An operation was attempted on something that is not a socket. Either the socket handle parameter did not reference a valid socket, or for select, a member of an fd_set was not valid.

WSAEOPNOTSUPP

(10045)

Operation not supported.

The attempted operation is not supported for the type of object referenced. Usually this occurs when a socket descriptor to a socket that cannot support this operation is trying to accept a connection on a datagram socket.

WSAEPFNOSUPPORT

(10046)

Protocol family not supported.

The protocol family has not been configured into the system or no implementation for it exists. This message has a slightly different meaning from WSAEAFNOSUPPORT. However, it is interchangeable in most cases, and all Windows Sockets functions that return one of these messages also specify WSAEAFNOSUPPORT.

WSAEPROCLIM

(10067)

Too many processes.

A Windows Sockets implementation may have a limit on the number of applications that can use it simultaneously. WSAStartup may fail with this error if the limit has been reached.

WSAEPROTONOSUPPORT

(10043)

Protocol not supported.

The requested protocol has not been configured into the system, or no implementation for it exists. For example, a socket call requests a SOCK_DGRAM socket, but specifies a stream protocol.

WSAEPROTOTYPE

(10041)

Protocol wrong type for socket.

A protocol was specified in the socket function call that does not support the semantics of the socket type requested. For example, the ARPA Internet UDP protocol cannot be specified with a socket type of SOCK_STREAM.

WSAESHUTDOWN

(10058)

Cannot send after socket shutdown.

A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous shutdown call. By calling shutdown a partial close of a socket is requested, which is a signal that sending or receiving, or both have been discontinued.

WSAESOCKTNOSUPPORT

(10044)

Socket type not supported.

The support for the specified socket type does not exist in this address family. For example, the optional type SOCK_RAW might be selected in a socket call, and the implementation does not support SOCK_RAW sockets at all.

WSAETIMEDOUT

(10060)

Connection timed out.

A connection attempt failed because the connected party did not properly respond after a period of time, or the established connection failed because the connected host has failed to respond.

WSATYPE_NOT_FOUND

(10109)

Class type not found.

The specified class was not found.

WSAEWOULDBLOCK

(10035)

Resource temporarily unavailable.

This error is returned from operations on nonblocking sockets that cannot be completed immediately, for example recv when no data is queued to be read from the socket. It is a nonfatal error, and the operation should be retried later. It is normal for WSAEWOULDBLOCK to be reported as the result from calling connect on a nonblocking SOCK_STREAM socket, since some time must elapse for the connection to be established.

WSAHOST_NOT_FOUND

(11001)

Host not found.

No such host is known. The name is not an official host name or alias, or it cannot be found in the database(s) being queried. This error may also be returned for protocol and service queries, and means that the specified name could not be found in the relevant database.

WSA_INVALID_HANDLE

(OS dependent)

Specified event object handle is invalid.

An application attempts to use an event object, but the specified handle is not valid.

WSA_INVALID_PARAMETER

(OS dependent)

One or more parameters are invalid.

An application used a Windows Sockets function which directly maps to a Win32 function. The Win32 function is indicating a problem with one or more parameters.

WSAINVALIDPROCTABLE

(OS dependent)

Invalid procedure table from service provider.

A service provider returned a bogus procedure table to Ws2_32.dll. (Usually caused by one or more of the function pointers being null.)

WSAINVALIDPROVIDER

(OS dependent)

Invalid service provider version number.

A service provider returned a version number other than 2.0.

WSA_IO_INCOMPLETE

(OS dependent)

Overlapped I/O event object not in signaled state.

The application has tried to determine the status of an overlapped operation which is not yet completed. Applications that use WSAGetOverlappedResult (with the fWait flag set to FALSE) in a polling mode to determine when an overlapped operation has completed, get this error code until the operation is com-plete.

WSA_IO_PENDING

(OS dependent)

Overlapped operations will complete later.

The application has initiated an overlapped operation that cannot be completed immediately. A completion indication will be given later when the operation has been completed.

WSA_NOT_ENOUGH_MEMORY

(OS dependent)

Insufficient memory available.

An application used a Windows Sockets function that directly maps to a Win32 function. The Win32 func-tion is indicating a lack of required memory resources.

WSANOTINITIALISED

(10093)

Successful WSAStartup not yet performed.

Either the application has not called WSAStartup or WSAStartup failed. The application may be access-ing a socket that the current active task does not own (that is, trying to share a socket between tasks), or WSACleanup has been called too many times.

WSANO_DATA

(11004)

Valid name, no data record of requested type.

The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for. The usual example for this is a host name-to-address translation attempt (using gethostbyname or WSAAsyncGetHostByName) which uses the DNS (Domain Name Server). An MX record is returned but no A record-indicating the host itself exists, but is not directly reachable.

WSANO_RECOVERY

(11003)

This is a nonrecoverable error.

This indicates some sort of nonrecoverable error occurred during a database lookup. This may be be-cause the database files (for example, BSD-compatible HOSTS, SERVICES, or PROTOCOLS files) could not be found, or a DNS request was returned by the server with a severe error.

WSAPROVIDERFAILEDINIT

(OS dependent)

Unable to initialize a service provider.

Either a service provider's DLL could not be loaded (LoadLibrary failed) or the provider's WSPStartup/NSPStartup function failed.

WSASYSCALLFAILURE

(OS dependent)

System call failure.

Returned when a system call that should never fail does. For example, if a call to WaitForMultipleObjects fails or one of the registry functions fails trying to manipulate the protocol/name space catalogs.

WSASYSNOTREADY

(10091)

Network subsystem is unavailable.

This error is returned by WSAStartup if the Windows Sockets implementation cannot function at this time because the underlying system it uses to provide network services is currently unavailable. Users should check:

That the appropriate Windows Sockets DLL file is in the current path.

That they are not trying to use more than one Windows Sockets implementation simultaneously. If there is more than one Winsock DLL on your system, be sure the first one in the path is appropriate for the network subsystem currently loaded.

The Windows Sockets implementation documentation to be sure all necessary components are currently installed and configured correctly.

WSATRY_AGAIN

(11002)

Nonauthoritative host not found.

This is usually a temporary error during host name resolution and means that the local server did not receive a response from an authoritative server. A retry at some time later may be successful.

WSAVERNOTSUPPORTED

(10092)

Winsock.dll version out of range.

The current Windows Sockets implementation does not support the Windows Sockets specification version requested by the application. Check that no old Windows Sockets DLL files are being accessed.

WSAEDISCON

(10101)

Graceful shutdown in progress.

Returned by WSARecv and WSARecvFrom to indicate that the remote party has initiated a graceful shutdown sequence.

WSA_OPERATION_ABORTED

(OS dependent)

Overlapped operation aborted.

An overlapped operation was canceled due to the closure of the socket, or the execution of the SIO_FLUSH command in WSAIoctl.

10.3 Check list

Is the connection to the PLC configured correctly (IP address, port)?

Is the PLC turned on and does the network connection work?

Did you configure the items correctly?

Does the data type of the variables and the items in the PLC match?

Are the offset and the net address in the variable correct?

Have you analyzed the "driver communication error file" (which errors have occured)?

In case of communication problems, the driver writes a detailed problem analysis into the driver communication error file. This file is stored in the project directory (RT\\FILES\\zenon\\custom\\log).
The name of the file is _<drivername>.TXT.
The file can be opened with any text editor, e.g. Notepad.

For additional error analyses, please send a project backup and the "error file" to the support team responsible for you.