

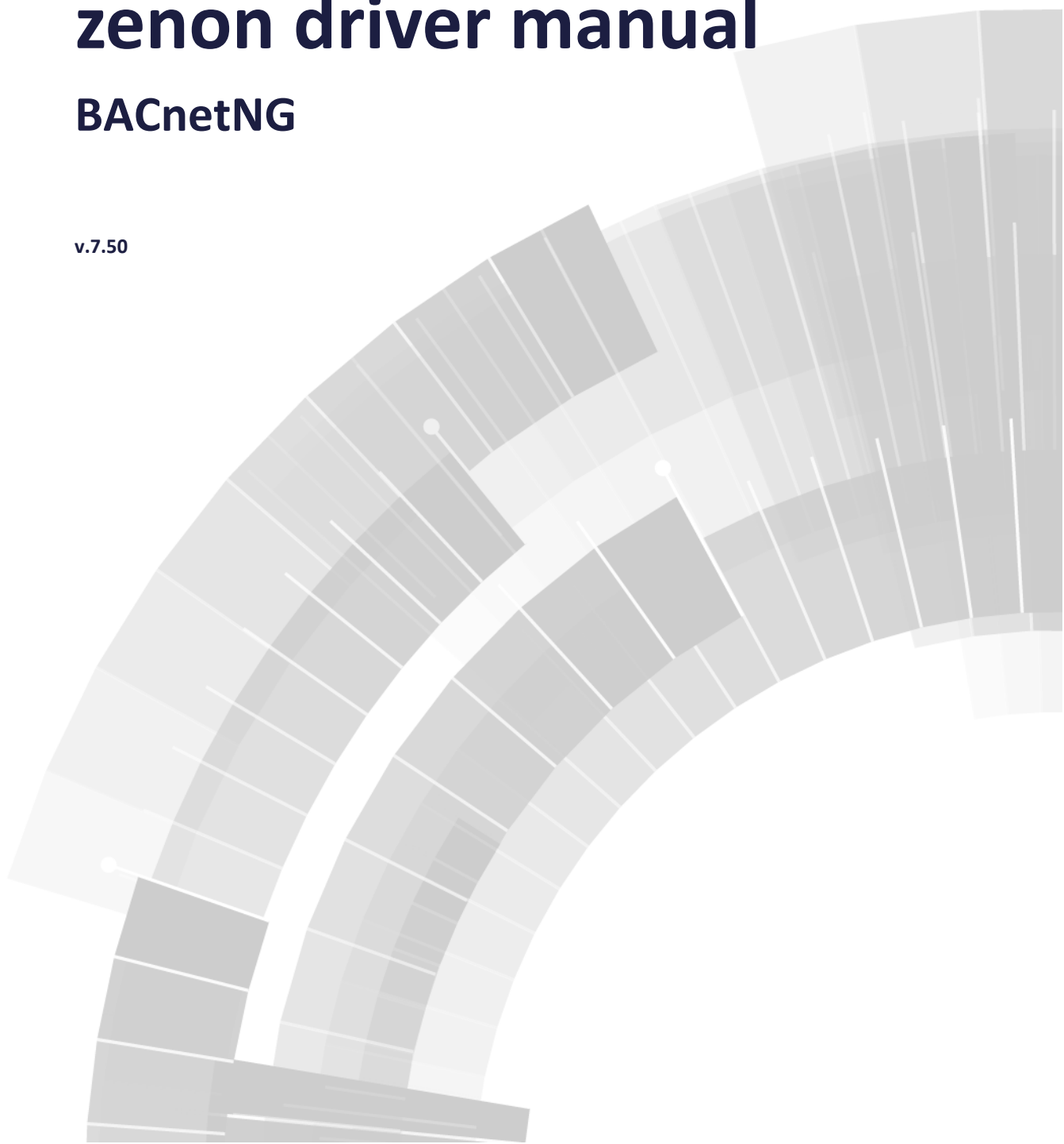


**COPADATA**  
do it your way

# zenon driver manual

## BACnetNG

v.7.50





©2016 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

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

# Contents

<b>1. Welcome to COPA-DATA help .....</b>	<b>5</b>
<b>2. BACnetNG .....</b>	<b>5</b>
<b>3. BACNETNG - Data sheet .....</b>	<b>6</b>
<b>4. Driver history .....</b>	<b>8</b>
<b>5. Requirements.....</b>	<b>8</b>
5.1 PC .....	8
5.2 Control .....	9
<b>6. Configuration .....</b>	<b>9</b>
6.1 Creating a driver.....	9
6.2 Settings in the driver dialog .....	11
6.2.1 General .....	11
6.2.2 Settings .....	15
6.2.3 Devices .....	17
<b>7. Creating variables.....</b>	<b>19</b>
7.1 Creating variables in the Editor.....	20
7.2 Addressing.....	23
7.3 Driver objects and datatypes .....	47
7.3.1 Driver objects .....	48
7.3.2 Mapping of the data types .....	51
7.4 Creating variables by importing .....	52
7.4.1 XML import.....	53
7.4.2 DBF Import/Export .....	53
7.4.3 Online import .....	60
7.5 Driver variables .....	62
<b>8. Driver-specific functions .....</b>	<b>68</b>
8.1 Access method (spontaneous or polling reading).....	68
8.2 Mapping the BACnet status flags to the status bits of a variable .....	69

8.3	Mapping of BACnet data types to string variables.....	70
8.4	Device Management .....	73
8.5	Import .....	74
<b>9.</b>	<b>Driver commands .....</b>	<b>75</b>
<b>10.</b>	<b>Error analysis.....</b>	<b>77</b>
10.1	Analysis tool .....	77
10.2	BACnet Error codes .....	78
10.3	Check list .....	81
<b>11.</b>	<b>PICS - Protocol Implementation Conformance Statement .....</b>	<b>82</b>
<b>12.</b>	<b>Communication with the PLC .....</b>	<b>87</b>

# 1. Welcome to COPA-DATA help

## GENERAL HELP

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

## PROJECT SUPPORT

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

## LICENSES AND MODULES

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

# 2. BACnetNG

## BACNET/IP CLIENT DRIVER

The protocol was defined by the ASHRAE (American Society of Heating, Refrigeration and Air-Conditioning Engineers, Inc.) and is described extensively in the ASHRAE standard 135-2001 + Appendix A – L „A Data Communication Protocol for Building Automation and Control Networks”. BACnet/IP is specified in appendix J and describes the BACnet communication via IP/UDP telegrams.

This driver is used for communication between one or more devices supporting BACnet (BACnet automation stations) and the zenon Runtime via BACnet/IP. This requires that the connected BACnet devices run as servers. Only client functionality is implemented in the driver.

The BACnet protocol defines objects and object properties. The driver makes it possible to read and write one or several properties of an object. In principal both polling reading and spontaneous communication using COV (change-of-value) subscription is supported. However most devices support spontaneous communication only for properties **PRESENT-VALUE** and **STATUS-FLAGS**:

The data of a BACnet property are transferred in **BACnet tags**. One property can contain one or more tags. A tag consists of a **Tag class**, **Tag number** and a value.

For the application tag class the data type comes off tag number of the encoded value unambiguously.

For tags with class `context -specific` the data type is additionally depended on the property and the data type. This means for decoding/encoding the value it is necessary that you have knowledge about the property and the object or the data type of the property.

Properties which can consist of an application tag can also be mapped to primitive types. All other properties can only be mapped to string variables.

### 3. BACNETNG - Data sheet

General:	
Driver file name	BACNETNG.exe
Driver name	BACnet driver Next Generation
PLC types	All PLCs with BACnet/IP support.
PLC manufacturer	Siemens; Kieback + Peter; BACnet; SE Elektronik;

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

Requirements:	
Hardware PC	Standard network adapter
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	X

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

## 4. Driver history

Date	Driver version	Change
7/27/2009	100	Created driver documentation

### DRIVER VERSIONING

The versioning of the drivers was changed with zenon 7.10. There is a cross-version build number as of this version. This is the number in the 4th position of the file version,  
For example: **7.10.0.4228** means: The driver is for version **7.10** service pack **0**, and has the build number **4228**.

Expansions or error rectifications will be incorporated into a build in the future and are then available from the next consecutive build number.



#### Example

*A driver extension was implemented in build **4228**. The driver that you are using is build number **8322**. Because the build number of your driver is higher than the build number of the extension, the extension is included. The version number of the driver (the first three digits of the file version) do not have any significance in relation to this. The drivers are version-agnostic*

## 5. Requirements

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

### 5.1 PC

For the BACnet/IP communication an IP network connection is needed which supports the UDP protocol.



## 5.2 Control

PLCs must support the BACnet protocol. If a PLC is not connected to the zenon Runtime via BACnet/IP, a corresponding BACnet router must be used.

# 6. Configuration

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



### Information

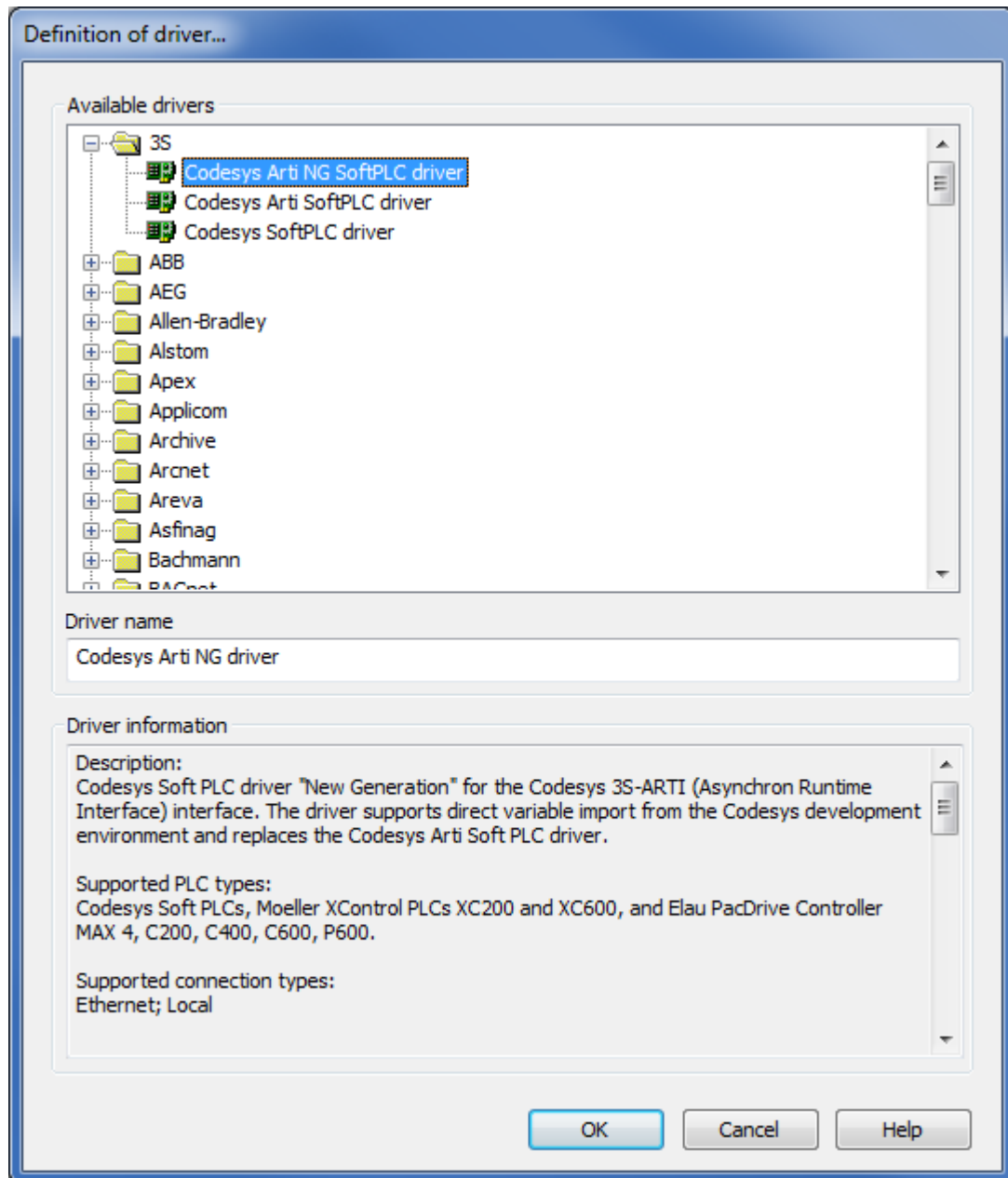
*Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.*

## 6.1 Creating a driver

In order to create a new driver:

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

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



3. Select the desired driver and give it a name:
  - The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.
  - The driver name is part of the file name. Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (\_).
  - Attention: This name cannot be changed later on.

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

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



### Information

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

- ▶ Internal
- ▶ MathDr32
- ▶ SysDrv.

▶

## 6.2 Settings in the driver dialog

You can change the following settings of the driver:

### 6.2.1 General

The configuration dialog is opened when a driver is created. In order to be able to open the dialog later for editing, double click on the driver in the list or click on the **Configuration** property.



Parameters	Description
<b>Mode</b>	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> <li>▶ Hardware: <p>A connection to the control is established.</p> </li> <li>▶ Simulation static <p>No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver.</p> </li> <li>▶ Simulation - counting <p>No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically.</p> </li> <li>▶ Simulation - programmed <p>N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).</p> </li> </ul>
<b>Keep update list in the memory</b>	<p>Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.</p>
<b>Output can be written</b>	<p>Active: Outputs can be written.</p> <p>Inactive: Writing of outputs is prevented.</p> <p>Note: Not available for every driver.</p>
<b>Variable image remanent</b>	<p>This option saves and restores the current value, time stamp and the states of a data point.</p> <p>Fundamental requirement: The variable must have a valid value and time stamp.</p>

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

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

#### CLOSE DIALOG

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

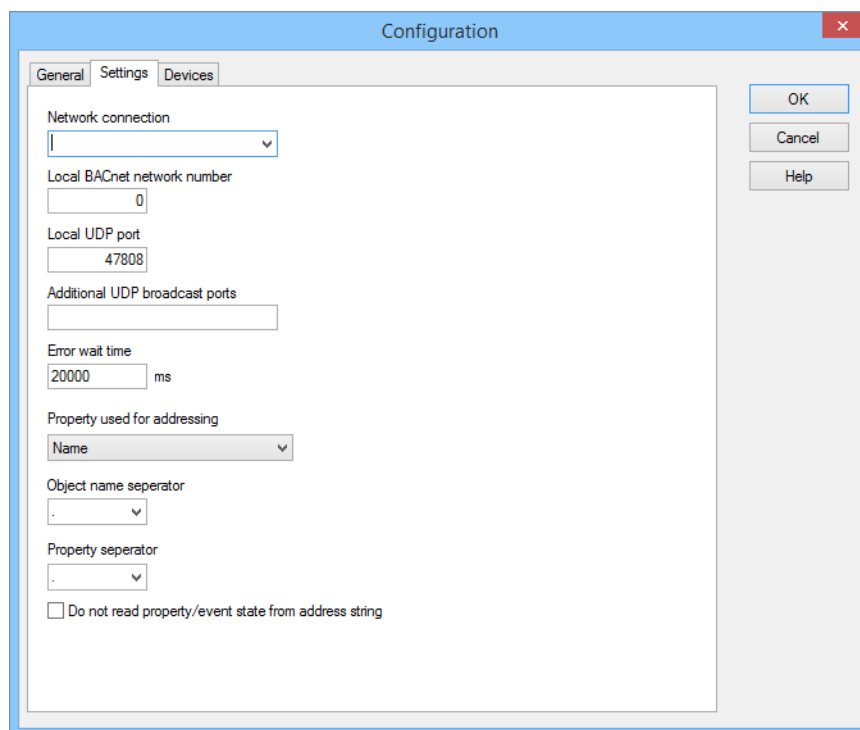
#### UPDATE TIME FOR CYCLICAL DRIVERS

The following applies for cyclical drivers:

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

## 6.2.2 Settings

### GENERAL SETTINGS VALID FOR ALL CONFIGURED BACNET DEVICES



The Configuration dialog box is shown with the 'Settings' tab selected. The 'General' tab is also visible. The 'Settings' tab contains the following fields and controls:

- Network connection:** A dropdown menu with a downward arrow.
- Local BACnet network number:** A text input field containing the value '0'.
- Local UDP port:** A text input field containing the value '47808'.
- Additional UDP broadcast ports:** A text input field.
- Error wait time:** A text input field containing the value '20000' followed by 'ms'.
- Property used for addressing:** A dropdown menu with 'Name' selected.
- Object name separator:** A dropdown menu with '.' selected.
- Property separator:** A dropdown menu with '.' selected.
- Do not read property/event state from address string:** An unchecked checkbox.

On the right side of the dialog, there are three buttons: 'OK', 'Cancel', and 'Help'.

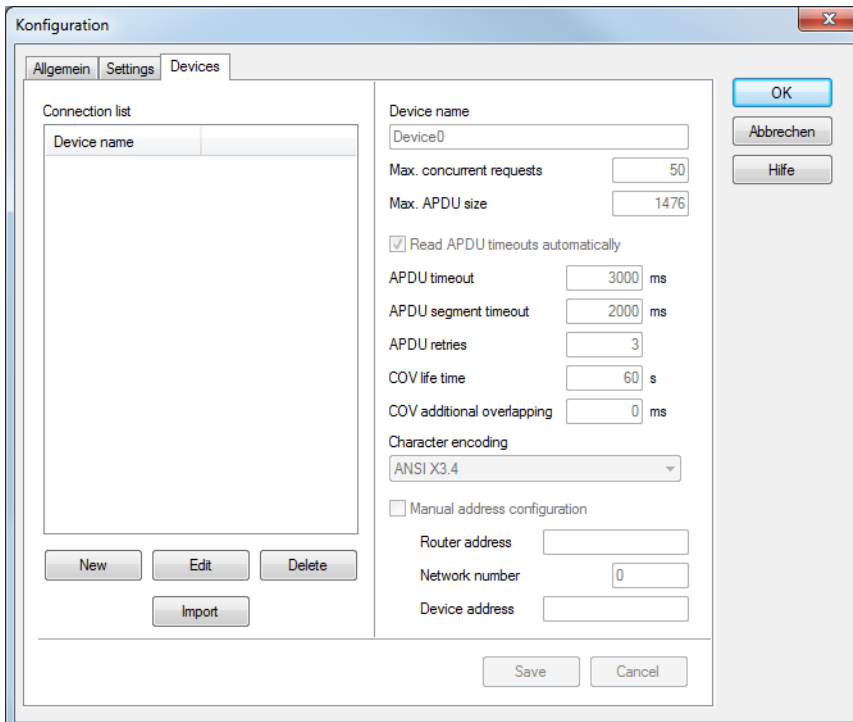
Property	Description
<b>Network connection</b>	<p>Name of the network connection or IP address which should be used for the BACnet/IP communication.</p> <p>The name matches the name which is displayed under Control Panel -&gt; Network connections.</p> <p>The drop-down list contains the names of all connections available at the system. As an alternative you can enter any name. Instead of a name, an IP address can also be entered manually.</p> <p>If you do not enter a name, the first network adapter of the system is used. You should enter the connection name for systems with several network cards.</p> <p>If there is no adapter with a name that corresponds to the <b>Network connection</b> setting, the IP addresses of all adapters are searched. If a match is found, this address is used.</p>
<b>Local BACnet network number</b>	BACnet network number of the local BACnet network.
<b>Local UDP port</b>	<p>Local UDP port which should be used for the communication</p> <p>Default: 47808</p>
<b>Additional UDP broadcast ports</b>	Additional UDP ports (besides the standard BACnet/IP port BAC0h or 47808) to which BACnet/IP broadcasts should be directed.
<b>Error wait time [ms]</b>	Error wait time in milliseconds.
<b>Property used for addressing</b>	<p>Selection of addressing from drop-down list:</p> <ul style="list-style-type: none"> <li>▶ Variable name</li> <li>▶ Identification</li> <li>▶ Symbolic address</li> </ul>
<b>Object name separator</b>	<p>Entry of the separator in variable name, identification or symbolic address, in order to separate the device name from the object name. The standard separator, a period (.), can be selected from the drop-down list.</p> <p><b>Note:</b> The characters @ and # are not permitted as separators. If one of the two characters is used, no communication takes place.</p>
<b>Property separator</b>	<p>Entry of the separator in variable name, identification or symbolic address, in order to separate the object name from the property name. The standard separator, a period (.), can be selected from the drop-down list.</p> <p><b>Note:</b> The characters @ and # are not permitted as separators. If one of the two characters is used, no communication takes place.</p>
<b>Do not read property/event state from address string</b>	<p>▶ Active: <b>Property-ID</b> and <b>Event State</b> are not extracted from the selected address.</p> <p><b>Attention:</b></p>



If this option is activated:

- ▶ The **Property-ID** or **Event State** must be set using the driver-specific properties envisaged for this. This configuration is carried out automatically with online import.
- ▶ Individual elements of BACnet Arrays/lists can no longer be addressed.
- ▶ The variable name cannot be used for addressing, because this must be unique. Addressing must be either by means of identification or symbolic address.
- ▶ The **Property separator** is not always searched for in the address field. The address thus now only consists of: <Device name>.<object name>.  
The following is applicable in the process: The device name must not include a period (.). There are no restrictions for the object name.

### 6.2.3 Devices



Konfiguration

Algemein Settings **Devices**

Connection list

Device name	Device ID

New Edit Delete Import

Device name: Device0

Max. concurrent requests: 50

Max. APDU size: 1476

☒ Read APDU timeouts automatically

APDU timeout: 3000 ms

APDU segment timeout: 2000 ms

APDU retries: 3

COV life time: 60 s

COV additional overlapping: 0 ms

Character encoding: ANSI X3.4

☐ Manual address configuration

Router address:

Network number: 0

Device address:

Save Cancel

OK Abbrechen Hilfe

## LIST OF BACNET DEVICES AND DEVICE-SPECIFIC SETTINGS

Property	Description
<b>Connection list</b>	Lists all configured devices.
<b>New</b>	Inserts a new device in the list.
<b>Delete</b>	Deletes the selected devices from the list.
<b>Edit</b>	Edit an existing device.
<b>Import</b>	Looks through a network for existing devices and imports these into the device list or sends a message if no devices were found.
<b>Device-specific settings</b>	
<b>Device name</b>	Name of the BACnet device (name of the device object).
<b>Max. concurrent requests</b>	<p>Maximum number of concurrently pending requests.</p> <p>Minimum: 1</p> <p>Maximum: 256</p> <p>Default: 5</p> <p>If a device sends more requests than defined, new buffers are allocated as long as the theoretical maximum of 256 is reached.</p> <p>Note: Due to performance reasons setting should be set to the maximum of the attached device. Only if the device also communicates with other devices, you may set a lower value.</p>
<b>Max. APDU size</b>	<p>Maximum transferable size of an APDU or an APDU segment.</p> <p>Default: 1476</p>
<b>Read APDU timeouts automatically</b>	<b>Active:</b> The APDU timeout settings are read by the device; this means that the timeouts set at the device apply.
<b>APDU timeout [ms]</b>	<p>Timeout for the acknowledgment of a request.</p> <p>Default 3000</p>
<b>APDU segment timeout [ms]</b>	<p>Timeout for the acknowledgment of one or several segments of a request.</p> <p>Default 2000</p>
<b>APDU retries</b>	<p>Request retries in case of a timeout.</p> <p>Default: 3</p>
<b>COV life time [s]</b>	Life time of COV subscriptions.

	Default: 60
<b>COV additional overlapping [ms]</b>	<p>Additional parameter to determine when the renewed subscription is sent. Prevents, with a delayed BACnet reaction, the loss of sent values through a timeout.</p> <p>Default: 0</p> <p>Formula:  <math>(APDU\_timeout * (APDU\_retries + 1)) - COV\_additional\_overlapping</math></p>
<b>Character encoding</b>	<p>Used character set. Select from drop-down list:</p> <ul style="list-style-type: none"> <li>▶ ANSI X3.4</li> <li>▶ ISO 10646 (UCS-4)</li> <li>▶ ISO 10646 (UCS-2)</li> <li>▶ ISO 8859-1</li> </ul>
<b>Manual address configuration</b>	<p><b>Active:</b> The address is not determined automatically via broadcast and the device name but can be entered manually via <b>Router address</b>, <b>Network number</b> and <b>Device address</b>.</p>
<b>Router address</b>	<p>BACnet MAC address of the router. It consists of the IP address of the router and the UDP port; e.g. in the following format  192.168.0.5:47808.</p>
<b>Network number</b>	BACnet network number of the device.
<b>Device address</b>	<p>BACnet MAC address of the device.</p> <p>For BACnet/IP devices is consists of the IP address and the UDP port (see <b>Router address</b>).</p> <p>For all other BACnet devices it consists of a byte sequence in hexadecimal format separated by : (e.g. 06:0A:67:EE).</p>
<b>Save</b>	Save changes.
<b>Cancel</b>	Cancel changes.

## 7. Creating variables

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

## 7.1 Creating variables in the Editor

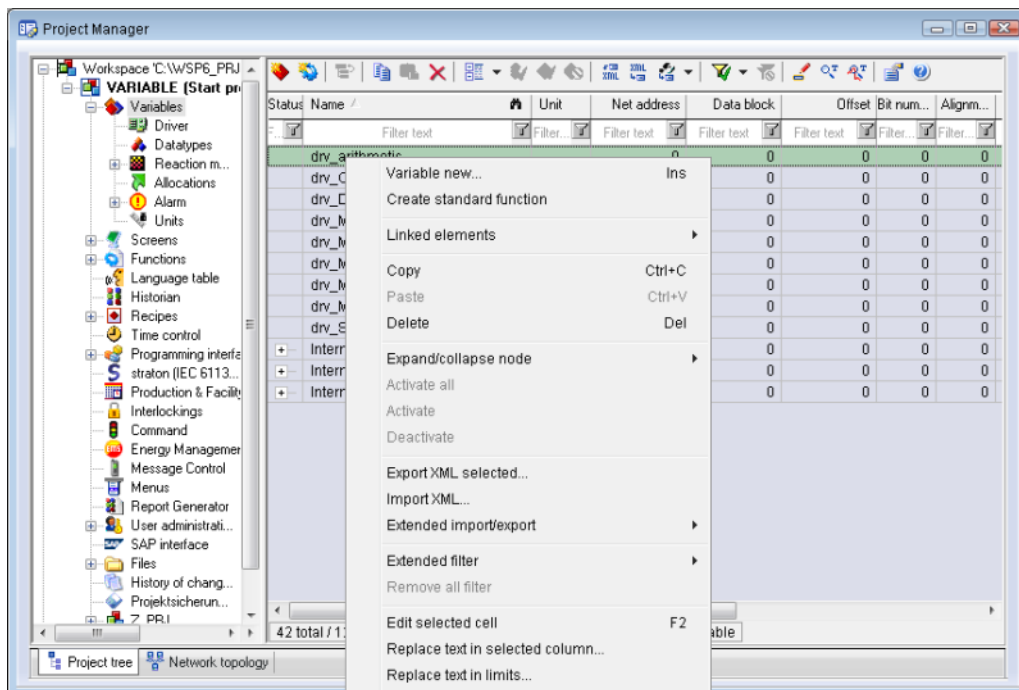
Variables can be created:

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

### VARIABLE DIALOG

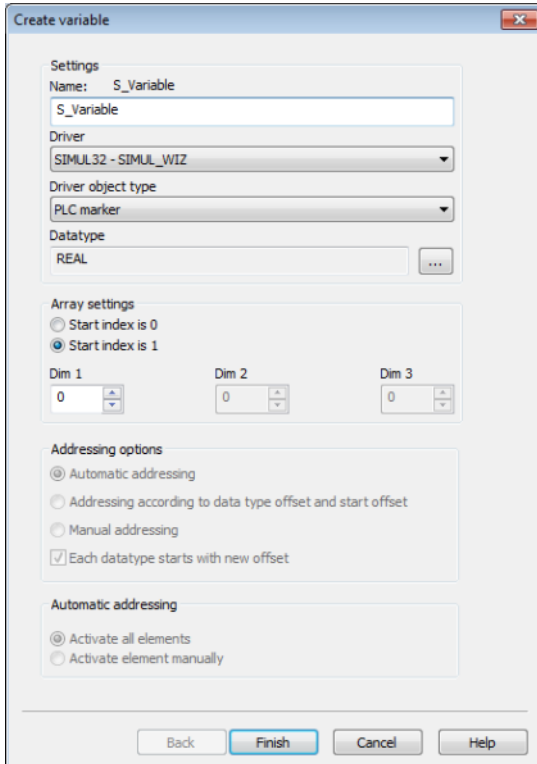
To create a new variable, regardless of which type:

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



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

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



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

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

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

Property	Description
<b>Name</b>	Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name.  Maximum length: 128 character  Attention: The characters <b>#</b> and <b>@</b> are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the <b>Finish</b> button remains inactive. Note: For some drivers, the addressing is possible over the property <b>Symbolic address</b> , as well.
<b>Drivers</b>	Select the desired driver from the drop-down list.  Note: If no driver has been opened in the project, the driver for internal variables ( <b>Intern.exe (Main.chm::/Intern.chm::/Intern.htm)</b> ) is automatically loaded.
<b>Driver object type</b> (cti.chm::/28685.htm)	Select the appropriate driver object type from the drop-down list.
<b>Data type</b>	Select the desired data type. Click on the ... button to open the selection dialog.
<b>Array settings</b>	Expanded settings for array variables. You can find details in the Arrays chapter.
<b>Addressing options</b>	Expanded settings for arrays and structure variables. You can find details in the respective section.
<b>Automatic element activation</b>	Expanded settings for arrays and structure variables. You can find details in the respective section.

## SYMBOLIC ADDRESS

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

Maximum length: 1024 characters.

## INHERITANCE FROM DATA TYPE

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

- ▶ derived from the datatype
- ▶ Automatically adapted if the data type is changed

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

## 7.2 Addressing

Group/Property	Description
<b>General</b>	
<b>Name</b>	<p>Name of the variable consists of device name, object name and property name.</p> <p>Device name and object name are separated by a dot ( . ). The separator between object name and property name can be configured.</p> <p>If a single item of an array should be addressed, the offset must be stated as postfix in brackets. (for example: <b>MyServer.AnalogValue7#priority-array[8]</b>)</p> <p>Attention: For every zenon project the name must be unambiguous.</p> <p>The device object of a devices is also addressed in accordance with the scheme mentioned above. In this case the device name is repeated as object name (z.B. <b>MyServer.MyServer#object-name</b>).</p>
<b>Identification</b>	Can be used as an alternative for addressing variables if it is set in the driver configuration. The format for name and identification is identical.
<b>Addressing</b>	
<b>Net address</b>	<p>Bus address or net address of the variable.</p> <p>This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.</p>
<b>Data block</b>	not used for this driver
<b>Offset</b>	not used for this driver
<b>Alignment</b>	not used for this driver
<b>Bit number</b>	<p>Bit offset within the addressed item.</p> <p>Possible entries: 0 ... 65535</p> <p>Attention: Variables with a bit number unequal 0 cannot be written.</p>
<b>String length</b>	Only available for String variables: Maximum number of characters that the variable can take.
<b>Driver connection/Driver Object Type</b>	Depending on the employed driver, an object type is selected during the creation of the variable; the type can be changed here later.
<b>Driver connection/Data Type</b>	<p>Data type of the variable. Is selected during the creation of the variable; the type can be changed here later.</p> <p>Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.</p>
<b>BACnet data type</b>	BACnet data type of the property. With the help of the data type you can define how a numeric variable or a string variable are mapped to a BACnet property. For a description of the possible data types see the following list.

<b>BACnet property ID</b>	For user-specific properties or when you do not want to state the property in the name or in the identification, you can set the Property ID there.  <b>Attention:</b> Only if you set -1 as Property ID, the property is determined via the name or identification.
<b>BACnet string length</b>	Only available if you select <code>BITSTRING</code> as <b>BACnet data type</b> .  Defines the bit length of the bit string.
<b>BACnet write priority</b>	Write priority. Valid write priorities are 1-16. If you enter 0, the variable is written without a write priority.  Possible entries: 0 ... 16
<b>Priority</b>	Normal: Spontaneous reading via COV subscriptions  <b>Attention:</b> Only for <b>PRESENT-VALUE</b> and <b>STATUS-FLAGS</b> for stating the update cycle in accordance with the driver configuration. Possible values: <ul style="list-style-type: none"> <li>▶ increased</li> <li>▶ high</li> <li>▶ highest</li> </ul>
Symbolic address	

#### DATA TYPES SUPPORTED BY THE DRIVER FOR BACNET OBJECT PROPERTIES

BACnet data type	Description	Compatible IEC types	String coding example
NULL	Data type without a value	String	[0]
BOOLEAN	Boolean	BOOL, string	TRUE
UNSIGNED	Positive integer	UDINT, String	8
SIGNED	Integer	DINT, String	-1
REAL	Float	REAL, String	7.9
DOUBLE	Float	LREAL, String	8.0
OCTETSTRING	Bytes sequence	String	65A8B900
CHARACTERSTRING	String	String	abcd
BITSTRING	Bit string	String, UDINT (if length < 32)	01110
ENUMERATED	Enumeration	UDINT, String	8



DATE	Date	UDINT, String	109.12.1.7
TIME	Time	UDINT, String	17:00:00:000
OBJECTIDENTIFIER	Object ID	String	0008 0000001
[Any]	Any value	String	[2] 78
[DateTime]	Date (DATE) followed by time (TIME)	String	{{[10] 109.12.1.7},{[11] 17:00:00:000}}
[TimeStamp]	BACnetTimeStamp  Contains either:  0: TIME  1: Unsigned  2: [DateTime]	String	{<2> {[10] 109.12.1.7},{[11] 17:00:00:000}}
[PriorityArray]	Array with 16 elements of the same data type as the PRESENT-VALUE property or NULL	String	{{[0]},{[4] 5.0},{[0]}, ...
[Recipient]	BACnetRecipient	String	
[DeviceObjectReference]	BACnetDeviceObjectReference	String	
[DeviceObjectPropertyReference]	BACnetDeviceObjectPropertyReference	String	
[ObjectPropertyReference]	BACnetObjectPropertyReference	String	
[SetpointReference]	BACnetSetPointReference	String	
[ActionList]	BACnetActionList	String	
[EventParameter]	BACnetEventParameter	String	
[DateRange]	Start date (DATE) followed by end date (DATE)	String	{109.12.1.255},{109.12.2.255}
[DailySchedule]	BACnetDailySchedule	String	

	List of: TIME followed by [Any]		
--	---------------------------------------	--	--

[SpecialEvent]	BACnetSpecialEvent	String	{<0> {(2) 1.1.1}}, {<2> {[11] 17:00:00:000}, {[1] TRUE}}, {(3) 1}
[VTSession]	BACnetVTSession	String	
[SessionKey]	BACnetSessionKey	String	
[CalendarEntry]	BACnetCalendarEntry Contains either: 0: DATE 1: [DateRange] 2: [WeekNDay]	String	(2) 255.255.1
[AddressBinding]	BACnetAddressBinding	String	
[COVSubscription]	BACnetCOVSubscription	String	
[ReadAccessSpecification]	BACnetReadAccessSpecification	String	
[ReadAccessResult]	BACnetReadAccessResult	String	
[Destination]	BACnetDestination	String	
[LogRecord]	BACnetLogRecord	String	
[WeekNDay]	BACnetWeekNDay	String	3.3.1
BITSTRING (StatusFlags)	Bit string with length 4	String	0010
BITSTRING (EventTransitionBits)	Bit string with length 3	String	111
BITSTRING (ServicesSupported)	Bit string with length 40	String	
[Raw]	Data should always be coded as bytes sequence.	String	%10% 6D071603

**BACNET OBJECTS AND THEIR PROPERTIES WITH THE RESPECTIVE DATA TYPES**

Object type (type)	Property (ID)	Data Type	Compliance
<b>ANALOG-INPUT (0)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	cov-increment (22)	REAL	Optional
	deadband (25)	REAL	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	high-limit (45)	REAL	Optional
	limit-enable (52)	ENUMERATED	Optional
	low-limit (59)	REAL	Optional
	max-pres-value (65)	REAL	Optional
	min-pres-value (69)	REAL	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	REAL	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	resolution (106)	REAL	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read

	time-delay (113)	UNSIGNED	Optional
	units (117)	ENUMERATED	Read
	update-interval (118)	UNSIGNED	Optional
<b>ANALOG-OUTPUT (1)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	cov-increment (22)	REAL	Optional
	deadband (25)	REAL	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	high-limit (45)	REAL	Optional
	limit-enable (52)	ENUMERATED	Optional
	low-limit (59)	REAL	Optional
	max-pres-value (65)	REAL	Optional
	min-pres-value (69)	REAL	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	REAL	Write
	priority-array (87)	[PriorityArray]	Optional
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional

	relinquish-default (104)	REAL	Optional
	resolution (106)	REAL	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
	units (117)	ENUMERATED	Read
	update-interval (118)	UNSIGNED	Optional
<b>ANALOG-VALUE (2)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	cov-increment (22)	REAL	Optional
	deadband (25)	REAL	Optional
	description (28)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	high-limit (45)	REAL	Optional
	limit-enable (52)	ENUMERATED	Optional
	low-limit (59)	REAL	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	REAL	Read
	priority-array (87)	[PriorityArray]	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional

	relinquish-default (104)	REAL	Read
	resolution (106)	REAL	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
	units (117)	ENUMERATED	Read
<b>AVERAGING (18)</b>			
	attempted-samples (124)	UNSIGNED	Write
	average-value (125)	REAL	Read
	description (28)	CHARACTERSTRING	Optional
	maximum-value (135)	REAL	Read
	maximum-value-timestamp (149)	[TimeStamp]	Optional
	minimum-value (136)	REAL	Read
	minimum-value-timestamp (150)	[TimeStamp]	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-property-reference (78)	[ObjectPropertyReference]	Read
	object-type (79)	ENUMERATED	Read
	present-value (85)	REAL	Write
	profile-name (168)	CHARACTERSTRING	Optional
	valid-samples (146)	UNSIGNED	Read
	variance-value (151)	REAL	Optional
	window-interval (147)	UNSIGNED	Write
	window-samples (148)	UNSIGNED	Write
<b>BINARY-INPUT (3)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	active-text (4)	CHARACTERSTRING	Optional
	alarm-value (6)	ENUMERATED	Optional

	change-of-state-count (15)	UNSIGNED	Optional
	change-of-state-time (16)	[TimeStamp]	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	elapsed-active-time (33)	UNSIGNED	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	inactive-text (46)	CHARACTERSTRING	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	polarity (84)	ENUMERATED	Read
	present-value (85)	ENUMERATED	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
	time-of-active-time-reset (114)	[TimeStamp]	Optional
	time-of-state-count-reset (115)	[TimeStamp]	Optional
<b>BINARY-OUTPUT (4)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	active-text (4)	CHARACTERSTRING	Optional



	change-of-state-count (15)	UNSIGNED	Optional
	change-of-state-time (16)	[TimeStamp]	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	elapsed-active-time (33)	UNSIGNED	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	feedback-value (40)	ENUMERATED	Optional
	inactive-text (46)	CHARACTERSTRING	Optional
	minimum-off-time (66)	UNSIGNED	Optional
	minimum-on-time (67)	UNSIGNED	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	polarity (84)	ENUMERATED	Read
	present-value (85)	ENUMERATED	Write
	priority-array (87)	[PriorityArray]	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	relinquish-default (104)	ENUMERATED	Read
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
	time-of-active-time-reset (114)	[TimeStamp]	Optional

	time-of-state-count-reset (115)	[TimeStamp]	Optional
<b>BINARY-VALUE (5)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	active-text (4)	CHARACTERSTRING	Optional
	alarm-value (6)	ENUMERATED	Optional
	change-of-state-count (15)	UNSIGNED	Optional
	change-of-state-time (16)	[TimeStamp]	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	elapsed-active-time (33)	UNSIGNED	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	inactive-text (46)	CHARACTERSTRING	Optional
	minimum-off-time (66)	UNSIGNED	Optional
	minimum-on-time (67)	UNSIGNED	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	ENUMERATED	Read
	priority-array (87)	[PriorityArray]	Optional
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	relinquish-default (104)	ENUMERATED	Optional

	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
	time-of-active-time-reset (114)	[TimeStamp]	Optional
	time-of-state-count-reset (115)	[TimeStamp]	Optional
<b>CALENDAR (6)</b>			
	datelist (23)	[CalendarEntry], List	Read
	description (28)	CHARACTERSTRING	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	present-value (85)	BOOLEAN	Read
	profile-name (168)	CHARACTERSTRING	Optional
<b>COMMAND (7)</b>			
	action (2)	ACTIONLIST, Array	Read
	action-text (3)	CHARACTERSTRING, Array	Read
	all-writes-successful (9)	BOOLEAN	Read
	description (28)	CHARACTERSTRING	Optional
	in-process (47)	BOOLEAN	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	present-value (85)	UNSIGNED	Write
	profile-name (168)	CHARACTERSTRING	Optional
<b>DEVICE (8)</b>			
	active-cov-subscriptions (152)	[COVSubscription], List	Optional
	active-vt-sessions (5)	[VTSession], List	Optional

	apdu-segment-timeout (10)	UNSIGNED	Optional
	apdu-timeout (11)	UNSIGNED	Read
	application-software-version (12)	CHARACTERSTRING	Read
	backup-failure-timeout (153)	UNSIGNED	Optional
	configuration-files (154)	OBJECTIDENTIFIER, Array	Optional
	database-revision (155)	UNSIGNED	Read
	daylight-savings-status (24)	BOOLEAN	Optional
	description (28)	CHARACTERSTRING	Optional
	device-address-binding (30)	[AddressBinding], List	Read
	firmware-revision (44)	CHARACTERSTRING	Read
	last-restore-time (157)	[DateTime]	Optional
	list-of-session-keys (55)	[SessionKey], List	Optional
	local-date (56)	DATE	Optional
	local-time (57)	TIME	Optional
	location (58)	CHARACTERSTRING	Optional
	max-apdu-length-accepted (62)	UNSIGNED	Read
	max-info-frames (63)	UNSIGNED	Optional
	max-master (64)	UNSIGNED	Optional
	max-segments-accepted (167)	UNSIGNED	Optional
	model-name (70)	CHARACTERSTRING	Read
	number-of-apdu-retries (73)	UNSIGNED	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-list (76)	OBJECTIDENTIFIER, Array	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read

	profile-name (168)	CHARACTERSTRING	Optional
	protocol-revision (139)	UNSIGNED	Read
	protocol-services-supported (97)	BITSTRING (ServicesSupported), Length 40	Read
	protocol-version (98)	UNSIGNED	Read
	segmentation-supported (107)	ENUMERATED	Read
	system-status (112)	ENUMERATED	Read
	time-synchronization-recipients (116)	[Recipient], List	Optional
	utc-offset (119)	SIGNED	Optional
	vendor-identifier (120)	UNSIGNED	Read
	vendor-name (121)	CHARACTERSTRING	Read
	vt-classes-supported (122)	ENUMERATED, List	Optional
<b>EVENTENROLLMENT (9)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Read
	description (28)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Read
	event-parameters (83)	[EventParameters]	Read
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	event-type (37)	ENUMERATED	Read
	issue-confirmed-notifications (51)	BOOLEAN	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-property-reference	[ObjectPropertyReference]	Read

	e (78)		
	object-type (79)	ENUMERATED	Read
	priority (86)	UNSIGNED	Optional
	process-identifier (89)	UNSIGNED	Optional
	profile-name (168)	CHARACTERSTRING	Optional
	recipient (101)	RECIPIENT	Optional
<b>FILE (10)</b>			
	archive (13)	BOOLEAN	Write
	description (28)	CHARACTERSTRING	Optional
	file-access-method (41)	ENUMERATED	Read
	file-size (42)	UNSIGNED	Read
	file-type (43)	CHARACTERSTRING	Read
	modification-date (71)	DATETIME	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	profile-name (168)	CHARACTERSTRING	Optional
	read-only (99)	BOOLEAN	Read
	record-count (141)	UNSIGNED	Optional
<b>GROUP (11)</b>			
	description (28)	CHARACTERSTRING	Optional
	list-of-group-members (53)	[ReadAccessSpecification], List	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	present-value (85)	[ReadAccessResult], List	Read
	profile-name (168)	CHARACTERSTRING	Optional
<b>LIFESAFETYPOINT</b>			

<b>(21)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	alarm-values (7)	ENUMERATED, List	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	direct-reading (156)	REAL	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	fault-values (39)	ENUMERATED, List	Optional
	life-safety-alarm-values (166)	ENUMERATED, List	Optional
	maintenance-required (158)	ENUMERATED	Optional
	member-of (159)	[DeviceObjectReference]	Optional
	mode (160)	ENUMERATED	Write
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	operation-expected (161)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	ENUMERATED	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Read
	setting (162)	UNSIGNED	Optional
	silenced (163)	ENUMERATED	Read
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read

	time-delay (113)	UNSIGNED	Optional
	tracking-value (164)	ENUMERATED	Optional
	units (117)	ENUMERATED	Optional
<b>LIFESAFETYZONE (22)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	alarm-values (7)	ENUMERATED, List	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	fault-values (39)	ENUMERATED, List	Optional
	life-safety-alarm-values (166)	ENUMERATED, List	Optional
	maintenance-required (158)	ENUMERATED	Optional
	member-of (159)	[DeviceObjectReference]	Optional
	mode (160)	ENUMERATED	Write
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	operation-expected (161)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	ENUMERATED	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Read



	silenced (163)	ENUMERATED	Read
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
	tracking-value (164)	ENUMERATED	Optional
	zone-members (165)	[DeviceObjectReference]	Read
<b>LOOP (12)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	action (2)	ENUMERATED	Read
	bias (14)	REAL	Optional
	controlled-variable-reference (19)	[ObjectPropertyReference]	Read
	controlled-variable-units (20)	ENUMERATED	Read
	controlled-variable-value (21)	REAL	Read
	cov-increment (22)	REAL	Optional
	derivative-constant (26)	REAL	Optional
	derivative-constant-units (27)	ENUMERATED	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	error-limit (34)	REAL	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	integral-constant (49)	REAL	Optional
	integral-constant-units (50)	ENUMERATED	Optional
	manipulated-variable-reference (60)	[ObjectPropertyReference]	Read
	maximum-output (61)	REAL	Optional

	minimum-output (68)	REAL	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	output-units (82)	ENUMERATED	Read
	present-value (85)	REAL	Read
	priority-for-writing (88)	UNSIGNED	Read
	profile-name (168)	CHARACTERSTRING	Optional
	proportional-constant (93)	REAL	Optional
	proportional-constant-units (94)	ENUMERATED	Optional
	reliability (103)	ENUMERATED	Read
	setpoint (108)	REAL	Read
	setpoint-reference (109)	[SetPointReference]	Read
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
	update-interval (118)	UNSIGNED	Optional
<b>MULTISTATE-INPUT (13)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	alarm-values (7)	UNSIGNED, List	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read

	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	fault-values (39)	UNSIGNED, List	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	number-of-states (74)	UNSIGNED	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	UNSIGNED	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	state-text (110)	CHARACTERSTRING, Array	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
<b>MULTISTATE-OUTP UT (14)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	description (28)	CHARACTERSTRING	Optional
	device-type (31)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	feedback-value (40)	UNSIGNED	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	number-of-states (74)	UNSIGNED	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read

	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	UNSIGNED	Write
	priority-array (87)	[PriorityArray]	Read
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	relinquisdefault	UNSIGNED	Read
	state-text (110)	CHARACTERSTRING, Array	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
<b>MULTISTATE-VALUE (19)</b>			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	alarm-values (7)	UNSIGNED, List	Optional
	description (28)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	fault-values (39)	UNSIGNED, List	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	number-of-states (74)	UNSIGNED	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	present-value (85)	UNSIGNED	Read

	priority-array (87)	[PriorityArray]	Optional
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	relinquish-default (104)	UNSIGNED	Optional
	state-text (110)	CHARACTERSTRING, Array	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
	time-delay (113)	UNSIGNED	Optional
<b>NOTIFICATIONCLASSES (15)</b>			
	ack-required (1)	BITSTRING (EventTransitionBits), Length 3	Read
	description (28)	CHARACTERSTRING	Optional
	notification-class (17)	UNSIGNED	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	priority (86)	UNSIGNED, List	Read
	profile-name (168)	CHARACTERSTRING	Optional
	recipient-list (102)	[Destination], List	Read
<b>PROGRAM (16)</b>			
	description (28)	CHARACTERSTRING	Optional
	description-of-halt (29)	CHARACTERSTRING	Optional
	instance-of (48)	CHARACTERSTRING	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	out-of-service (81)	BOOLEAN	Read
	profile-name (168)	CHARACTERSTRING	Optional
	program-change (90)	ENUMERATED	Write

	program-location (91)	CHARACTERSTRING	Optional
	program-state (92)	ENUMERATED	Read
	reason-for-halt (100)	ENUMERATED	Optional
	reliability (103)	ENUMERATED	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Read
<b>SCHEDULE (17)</b>			
	description (28)	CHARACTERSTRING	Optional
	effective-period (32)	[DateRange]	Optional
	exception-schedule (38)	[SpecialEvent], Array	Optional
	list-of-object-property-references (54)	[DeviceObjectPropertyReference], List	Read
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	present-value (85)	[Any]	Read
	priority-for-writing (88)	UNSIGNED	Read
	profile-name (168)	CHARACTERSTRING	Optional
	weekly-schedule (123)	[DailySchedule], Array	Optional
<b>TRENDLOG (20)</b>			
	acked-transitions (0)		Optional
	buffer-size (126)	UNSIGNED	Read
	Client-COV-Increment (127)	ANY	Optional
	cov-resubscription-interval (128)	UNSIGNED	Optional
	current-notify-time (129)	[DateTime]	Optional
	description (28)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Optional
	event-state (36)	ENUMERATED	Read
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional

	log-buffer (131)	[LogRecord], List	Read
	log-device-object-property (132)	[DeviceObjectPropertyReference]	Optional
	log-enable (133)	BOOLEAN	Write
	log-interval (134)	UNSIGNED	Optional
	notification-class (17)	UNSIGNED	Optional
	notification-threshold (137)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Read
	object-name (77)	CHARACTERSTRING	Read
	object-type (79)	ENUMERATED	Read
	previous-notify-time (138)	[DateTime]	Optional
	profile-name (168)	CHARACTERSTRING	Optional
	record-count (141)	UNSIGNED	Write
	records-since-notification (140)	UNSIGNED	Optional
	start-time (142)	[DateTime]	Optional
	stop-time (143)	[DateTime]	Optional
	stop-when-full (144)	BOOLEAN	Read
	total-record-count (145)	UNSIGNED	Read

## 7.3 Driver objects and datatypes

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

### 7.3.1 Driver objects

The following object types are available in this driver:

Driver object type	Channel type	Read	Write	Supported data types	Description
<b>Command status</b>	11	X	--	UDINT	Status of whether command was executed successfully:  0: Successful 1: Command is executed 2: Error
<b>Device event information</b>	10	X	--	STRING	Summary of all open alarms of a device
<b>Event</b>	9	X	X	BOOL, STRING	Receipt of BACnet event notifications.  Addressing: <b>&lt;Device name&gt;.&lt;Object name&gt;.Event.&lt;Event state&gt;.bool</b>  or: <b>&lt;Device name&gt;.&lt;Object name&gt;.Event.&lt;Event state&gt;.string</b>
<b>Property</b>	8	X	X	BOOL, DINT, INT, LREAL, REAL, SINT, STRING, UDINT, UINT, USINT	Corresponds to PLC marker
<b>Driver variable</b>	35	X	X	BOOL, DINT, INT, REAL, SINT, STRING, UDINT, UINT, USINT	Variables for the statistical analysis of communication.  Find out more in the chapter about the Driver variables (on page 62)

## EVENT

### AE-N-A ALARM AND EVENT NOTIFICATION-A:

Receipt of BACnet event notifications (BACnet alarms and events). A BACnet event notification is sent by BACnet objects. Requirement: Linking to a notification class object.

The **Event** driver object type can be used to evaluate these events in zenon.



Addressing:

**<Device name>.<Object name>.Event.<Event state>.bool**

or.

**<Device name>.<Object name>.Event.<Event state>.string**

Note:

- ▶ Whether (.) or # is used as a separator depends on the settings in the **Property Separator** property in the **Options** tab in the driver configuration (on page 15).
- ▶ Exception: The device name is always separated from the object name by a period (.).
- ▶ The last element of the name (**bool** or **string in the example**) can be issued freely by the user.

## <EVENT STATE> VALUES

**<Event state>** can have the following values:

Event state (text)	Event state (numerical)
Normal	0
Fault	1
Off-normal	2
High-limit	3
Low-limit	4
Life-safety-alarm	5

The event state can also be set as a number using the **BACnet event state** property. In this case, the **event state** can be replaced in the name by any desired string.

**Requirements:** The process ID in the notification class (ID for the recipient of the events in the recipient address) must be set to 0.

## PROCEDURE

- ▶ Each time notification of the object with the given **event states** is received, the associated BOOL variable is first set to *false* and then to *true*.
- ▶ If a notification of an object with different **event states** is received, the *false* variable is set. As a result of this, it is also possible to edit events without changing the **event state**.
- ▶ If **event state** is active, the whole received event notification of the string variables is assigned.
- ▶ If an **event state** that is not the same as the configured one is received, the variable is set to an empty string.

- The BOOL variables are initialized when the driver is started and the string variables are set to empty string.

The format of the string variables is similar to the property format.

#### FORMAT FROM BACNET SPECIFICATION

```
SEQUENCE {
processIdentifier [0] Unsigned32,
initiatingDeviceIdentifier [1] BACnetObjectIdentifier,
eventObjectIdentifier [2] BACnetObjectIdentifier,
timeStamp [3] BACnetTimeStamp,
notificationClass [4] Unsigned,
priority [5] Unsigned8,
eventType [6] BACnetEventType,
messageText [7] CharacterString OPTIONAL,
notifyType [8] BACnetNotifyType,
ackRequired [9] BOOLEAN OPTIONAL,
fromState [10] BACnetEventState OPTIONAL,
toState [11] BACnetEventState,
eventValues [12] BACnetNotificationParameters OPTIONAL
```

#### EXAMPLE

```
[[0] 0, [[1] 0000 00000010, [[2] 0002 00000000, [<3> [<2> [[10] 110 00 09 1, [[11] 12 03 52 018]], [[4] 1, [[5] 120, [[6] 5, [[8] 1, [[9] FALSE], [[10] 0, [[11] 4, [<12> [<5> [[0] 0 000000, [[1] 1000, [[2] 0 000000, [[3] 100 000000]]]]]]]]]]
```

#### AE-ACK-A ALARM AND EVENT ACK-A

Acknowledge: The event is set to `acknowledged` by means of the event variable (AcknowledgeAlarm Telegram). In doing so, the last event received with this **event state** is `acknowledged`.

#### AE-INFO-A ALARM AND EVENT INFORMATION-A

Read a summary of all open alarms for one device. This summary can be read using an **event info** object string variable.

## FORMAT FROM BACNET SPECIFICATION

SEQUENCE OF SEQUENCE {

objectIdentifier [0] BACnetObjectIdentifier,

eventState [1] BACnetEventState,

acknowledgedTransitions [2] BACnetEventTransitionBits,

eventTimeStamps [3] SEQUENCE SIZE (3) OF BACnetTimeStamp,

notifyType [4] BACnetNotifyType,

eventEnable [5] BACnetEventTransitionBits,

eventPriorities [6] SEQUENCE SIZE (3) OF Unsigned

}

## EXAMPLE

```
{(0) 0005 00000001, ((1) 2), ((2) 111), (<3> (<2> ((10] 110.07.08.4), ((11] 15.50.28.048)), (<2> ((10] 255.255.255.255), ((11] 255.255.255.255)), (<2> ((10] 110.07.08.4), ((11] 15.50.20.047))), ((4) 0), ((5) 000), (<6> ((2] 128), ((2] 128), ((2] 128)), ((0) 0002 00000001), ((1) 4), ((2) 111), (<3> (<2> ((10] 255.255.255.255), ((11] 255.255.255.255)), (<2> ((10] 255.255.255.255), ((11] 255.255.255.255)), (<2> ((10] 255.255.255.255), ((11] 255.255.255.255)), (<2> ((10] 255.255.255.255), ((11] 255.255.255.255))), ((4) 0), ((5) 010), (<6> ((2] 128), ((2] 128), ((2] 128)), ((0) 0002 00000000), ((1) 4), ((2) 101), (<3> (<2> ((10] 110.08.09.1), ((11] 12.03.52.018)), (<2> ((10] 110.07.29.4), ((11] 15.41.08.088)), (<2> ((10] 110.07.30.5), ((11] 07.16.12.082))), ((4) 1), ((5) 111), (<6> ((2] 128), ((2] 128), ((2] 128)))
```

## 7.3.2 Mapping of the data types

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

Control	zenon	Data type
Boolean	BOOL	8
-	USINT	9
-	SINT	10
-	UINT	2
-	INT	1
Unsigned, Bit-String, Date, Time, Enumerated	UDINT	4
Integer	DINT	3
-	ULINT	27
-	LINT	26
Real	REAL	5
Double	LREAL	6
Character string, NULL, Boolean, Unsigned, Integer, Real, Double, Bit-String, Octet string, Enumerated, Date, Time, ObjectID and all constructed or combined types	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.

## 7.4 Creating variables by importing

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



### Information

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

## 7.4.1 XML import

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

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



### Attention

When importing/overwriting an existing data type, all variables based on the existing data type are changed.

*Example:*

*There is a data type XYZ derived from the type `INT` with variables based on this data type. The XML file to be imported also contains a data type with the name XYZ but derived from type `STRING`. If this data type is imported, the existing data type is overwritten and the type of all variables based on it is adjusted. I.e. the variables are now no longer `INT` variables, but `STRING` variables.*

## 7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.



### Information

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

## IMPORT DBF FILE

To start the import:

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

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



### Information

*Note:*

- ▶ Driver object type and data type must be amended to the target driver in the DBF file in order for variables to be imported.
- ▶ dBase does not support structures or arrays (complex variables) at import.

## EXPORT DBF FILE

To start the export:

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



### Attention

DBF files:

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

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



### Information

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

File structure of the dBase export file

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



### Attention

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

DBF files must:

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

## STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	Char	128	Variable name.  The length can be limited using the MAX_LAENGE entry in <b>project.ini</b> .
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (field/column must be entered manually).  The length can be limited using the MAX_LAENGE entry in <b>project.ini</b> .
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	Identification.  The length can be limited using the MAX_LAENGE entry in <b>project.ini</b> .
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Bus address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager



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

<b>ADJZENON</b>	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
<b>ADJWVBA</b>	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
<b>ZWREMA</b>	N	16	Linked counter REMA.
<b>MAXGRAD</b>	N	16	Gradient overflow for counter REMA.



### Attention

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

## LIMIT VALUE DEFINITION

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

Identification	Type	Field size	Comment
<b>AKTIV1</b>	L	1	Limit value active (per limit value available)
<b>GRENZWERT1</b>	F	20	technical value or ID number of a linked variable for a dynamic limit value (see VARIABLEx) (if VARIABLEx is 1 and here it is -1, the existing variable linkage is not overwritten)
<b>SCHWWERT1</b>	F	16	Threshold value for limit value
<b>HYSTERESE1</b>	F	14	Is not used
<b>BLINKEN1</b>	L	1	Set blink attribute
<b>BTB1</b>	L	1	Logging in CEL
<b>ALARM1</b>	L	1	Alarm
<b>DRUCKEN1</b>	L	1	Printer output (for CEL or Alarm)
<b>QUITTIER1</b>	L	1	Must be acknowledged
<b>LOESCHE1</b>	L	1	Must be deleted
<b>VARIABLE1</b>	L	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
<b>FUNC1</b>	L	1	Functions linking
<b>ASK_FUNC1</b>	L	1	Execution via Alarm Message List
<b>FUNC_NR1</b>	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
<b>A_GRUPPE1</b>	N	10	Alarm/Event Group
<b>A_KLASSE1</b>	N	10	Alarm/Event Class
<b>MIN_MAX1</b>	C	3	Minimum, Maximum
<b>FARBE1</b>	N	10	Color as Windows coding
<b>GRENZTXT1</b>	C	66	Limit value text
<b>A_DELAY1</b>	N	10	Time delay
<b>INVISIBLE1</b>	L	1	Invisible

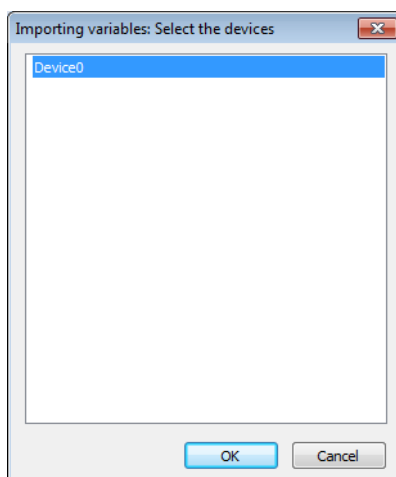
Expressions in the column "Comment" refer to the expressions used in the dialog boxes for the definition of variables. For more information, see chapter Variable definition.

### 7.4.3 Online import

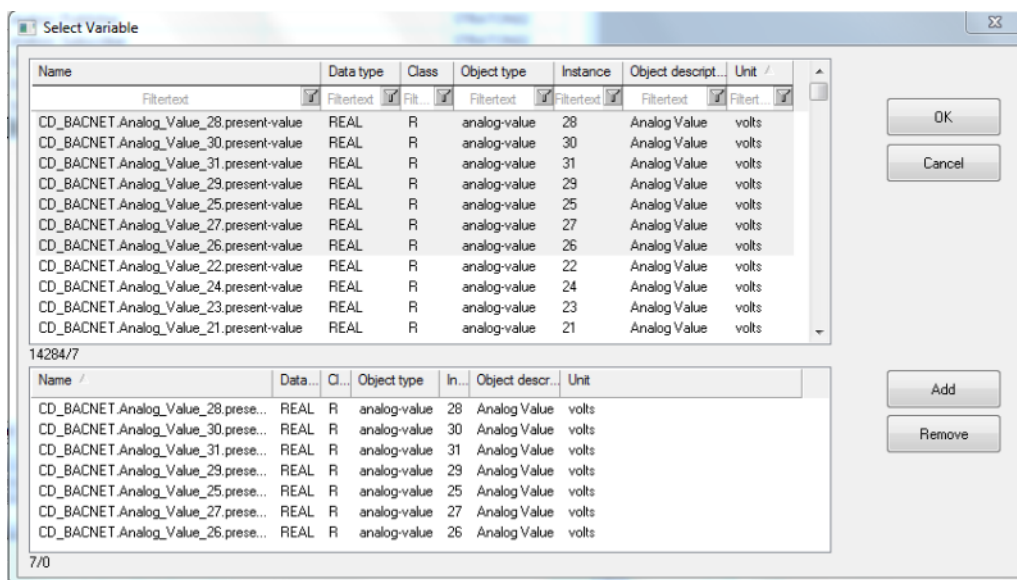
You can automatically create variables for the properties of all objects of one or several devices by means of online import. The devices must be created in the drive configuration (on page 17) to do this.

To import data online:

1. Right-click on the driver
2. Click on **Import variables from driver...** in the context menu
3. The window for the selection of the device is opened



4. Select the desired device
5. click on OK
6. The dialog for variable import is opened



7. Select the desired values

Filtering according to the following factors is possible:

- **Name**
  - **Data type**
  - **Class**
  - **Object type**
  - **Instance**
  - **Object description**
  - **Unit**
8. The **Add** button is used to accept selected elements into the import list  
You remove elements from the import list with **Remove**
  9. Confirm the selection by clicking on OK
  10. The variables are imported

## UNITS

Units are read off during import for:

- ▶ Analog Value
- ▶ Analog Input
- ▶ Analog Output
- ▶ Loop Object

The **Unit ID** is converted into a readable string. This string is saved as a variable unit. Units contain the prefix @, in order to allow language switching. Units correspond to the BACnet Standard 135-2001.

Supported unit groups:

- ▶ Area
- ▶ Currency
- ▶ Electrical
- ▶ Energy
- ▶ Enthalpy
- ▶ Entropy
- ▶ Frequency
- ▶ Humidity
- ▶ Length
- ▶ Light
- ▶ Mass

- ▶ Mass Flow
- ▶ Power
- ▶ Pressure
- ▶ Temperature
- ▶ Time
- ▶ Velocity
- ▶ Volume
- ▶ Volumetric Flow

## 7.5 Driver variables

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

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

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

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



### Information

*Not every driver supports all driver variants.*

*For example:*

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

## INFORMATION

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

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

## CONFIGURATION

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



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

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

## STATISTICS

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

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

## ERROR MESSAGE

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

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

## 8. Driver-specific functions

The driver supports the following functions:

- ▶ Driver commands (on page 75)
- ▶ Access method (spontaneous or polling reading) (on page 68)
- ▶ Mapping the BACnet status flags to the status bits of a variable (on page 69)
- ▶ Mapping of BACnet data types to string variables (on page 70)
- ▶ Device Management (on page 73)

### 8.1 Access method (spontaneous or polling reading)

Dependent on the set priority the variable is either read spontaneously via COV-Subscription (priority normal) or polled (all other priorities). For all priorities except `Normal` the set update interval equals the polling cycle.

**Attention:** Normally only properties **PRESENT-VALUE** and **STATUS-FLAGS** can be read via COV-Subscription, i.e. spontaneously. As a COV-Subscription always refers to only one object and not a

property, the creation of a variable which can not be read spontaneously leads to a log entry or a **INVALID-bit**. In this case the value of the variable remains empty.

## 8.2 Mapping the BACnet status flags to the status bits of a variable

At reading property **PRESENT-VALUE** property **STATUS-FLAG** is always also read. If property **PRESENT-VALUE** is read via a variable, property **STATUS-FLAGS** is mapped to the status bits of the variable as follows:

BACnet STATUS-FLAG	zenon Status-Bit
In alarm	-
Fault	INVALID
Overridden	INVALID
Out of service	SB-Bit

## 8.3 Mapping of BACnet data types to string variables

### CODING OF THE VALUES

Data type	Tag number	Description
<b>NULL</b>	0	-
<b>Boolean</b>	1	FALSE (0): "FALSE" TRUE (1): "TRUE"
<b>Unsigned</b>	2	Unsigned decimal value.
<b>Integer</b>	3	Signed decimal value.
<b>Real</b>	4	Float.
<b>Double</b>	5	Float.
<b>Bit string</b>	8	Bit sequence (sequence of "1" respectively "0").
<b>Octet string</b>	6	Byte sequence (sequence of double-digit hexadecimal numbers).
<b>Character string</b>	7	Character sequence; If the string contains other characters beside the character sequence - e.g. Tag number - the character sequence is put inside quotation marks.
<b>Enumerated</b>	9	Unsigned decimal value.
<b>Date</b>	10	YYY.mm.dd.tt  YYY: Year since 1900 mm: Month dd: Day tt: weekday)  Value 255 in one of the fields means: not specified
<b>Time</b>	11	hh:mm:ss:MMM  hh: Hour  mm: Minute  ss: Second  MMM: 100/second

		Value 255 in one of the fields means: not specified
<b>ObjectID</b>	12	xxxx yyyy xxxx: Object type yyyy: Object instance number
<b>WeekNDay</b>	context specific	mm.ww.tt mm: Month ww: Week of the month tt: weekday Value 255 in one of the fields means: not specified

#### LISTS OR ARRAYS OF SIMPLE DATA TYPES

List or arrays of Boolean, Unsigned, Integer, Real, Double, Bit string, Octet string, Character string, Enumerated, Date, Time or ObjectID are displayed as sequence of the coded values. At this every value is put between face brackets '{' and '}' and the single values are separated by comma (, ).

#### COMPLEX DATA TYPES

For complex data types (i.e. for context specific tags and all types which consist of more than one tag but are no lists or arrays of simple data types) the tag number is put in the sting beside the value of the tag. In this case a tag always consists of tag number and value. If the BACnet data type "[Raw]" was not selected in the variable configuration and the data type can be determined (possible for all known properties), the value is coded as shown in the table above. Otherwise the value is coded as a byte sequence (sequence of double-digit hexadecimal numbers).

Format of the tag number:

Tag class	Data type unknown or set type "[Raw]"	Data type known - coding of the value by means of the data type
<b>Tag class: Application</b>	Tag number within '%' and '%'	Tag number within brackets '[' and ']'
<b>Tag class: Context specific</b>	Tag number within '#' and '#'	Tag number within parenthesis '(' and ')'

Tags which are referred to as constructed tags are tags which contain a tag sequence are put in faced brackets '{' and '}' and consist of the tag number followed by the enclosed tags which are separated by comma (, ). The tag number is put between '<' and '>'.

**Attention:** For the correct decoding/encoding you must set the respective BACnet data type in the address setting of the variable.

#### **FORMATTING EXAMPLE**

##### **NULL:**

[0]

##### **Boolean, BOOLEAN:**

TRUE

FALSE

[1] TRUE

[1] FALSE

##### **Date, DATE:**

107.11.25.7

[10] 107.11.25.7

%10% 6D071603

##### **Time, TIME:**

17:00:00:000

[11] 17:00:00:000

##### **Objekt ID, OBJECTIDENTIFIER:**

0017 0000000

[12] 0017 0000000

##### **Bitstring, BITSTRING:**

010

[8] 010

##### **Byte-String, OCTETSTRING:**



```
6D071603
```

```
[5] 6D071603
```

Character sting, CHARACTERSTRING:

```
BACnet
```

```
[7] "BACnet"
```

Date, [CalendarEntry] with [WeekNDay]:

```
(2) 255.255.1
```

[DateRange] or date list/array:

```
{107.11.25.7},{109.12.28.255}
```

Constructed type ([SpecialEvent])

```
{<0> {(2) 1.1.1}}, {<2> {[11] 17:00:00:000}, {[1] TRUE}}, {(3) 1}
```

## 8.4 Device Management

### DRIVER COMMANDS

A zenon "driver commands" function for a selected BacnetNG driver is used to send a command as a text via the "driver-specific command" driver command and one of the options below:

#### DM-DCC-A DEVICE COMMUNICATION CONTROL-A

Turn BACnet communication on/off

Syntax:

```
DeviceCommunicationControl(<Device name>, <Communication On(1)/Off (0)>, <Time in seconds  
communication should remain off>,<Password>)
```

Example:

```
DeviceCommunicationControl("BACnet_Device", 0, 5, "secret")
```

Note: The key word `DeviceCommunicationControl` and parameters can be separated as desired using the `()`, characters and spaces. Separators within `""` are ignored.

#### DM-RD-A REINITIALIZE DEVICE-A

Reinitialize device (reset)

Syntax:

```
ReinitializeDevice(<Device name>, <start mode ("Warm" resp. "Cold")>)
```

Example:

```
ReinitializeDevice("BACnet_Device", Cold)
```

Note: The key word `DeviceCommunicationControl` and parameters can be separated as desired using the `()`, characters and spaces. Separators within `""` are ignored.

#### DM-TS-A TIME SYNCHRONIZATION-A

Time synchronization broadcast with local time

Syntax:

```
TimeSynchronisation
```

#### DM-UTC-A UTC TIME SYNCHRONIZATION-A

Time synchronization broadcast with UTC time

Syntax:

```
UTCTimeSynchronization
```

## 8.5 Import

For the zenon variable that is created for **PRESENT-VALUE**, the following are set if present:

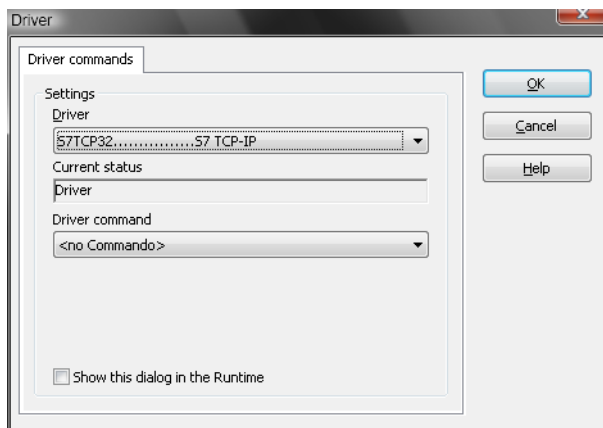
- ▶ **Min/Max value** property
- ▶ Decimal points according to the properties **min-/max-pres-value** and **resolution** of the BACnet object

## 9. Driver commands

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

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

- ▶ create a new function
- ▶ select Variables -> Driver commands
- ▶ The dialog for configuration is opened



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

## DRIVER COMMANDS IN THE NETWORK

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

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

## 10. Error analysis

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

### 10.1 Analysis tool

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

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

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

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

In the Diagnosis Viewer you can also:

- ▶ Follow newly-created entries in real time
- ▶ customize the logging settings
- ▶ change the folder in which the LOG files are saved

Note:

1. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
2. The Diagnosis Viewer does not display all columns of a LOG file per default. To display more columns activate property **Add all columns with entry** in the context menu of the column header.
3. If you only use **Error-Logging**, the problem description is in the column **Error text**. For other diagnosis level the description is in the column **General text**.
4. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** or **Error code** or **Driver error parameter (1 and 2)**. Hints on the meaning of error codes can be found in the driver documentation and the protocol/PLC description.
5. At the end of your test set back the diagnosis level from **Debug** or **Deep Debug**. At **Debug** and **Deep Debug** there are a great deal of data for logging which are saved to the hard drive and which can influence your system performance. They are still logged even after you close the **Diagnosis Viewer**.

**Attention**

In Windows CE errors are not logged per default due to performance reasons.

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

## 10.2 BACnet Error codes

Description of BACnet error codes which are displayed in log messages.

**REJECT REASON**

0	Undefined (other)
1	Buffer overflow (buffer-overflow)
2	Inconsistent parameter (inconsistent-parameters)
3	Invalid parameter data type (invalid-parameter-data-type)
4	Invalid tag (invalid-tag)
5	Missing parameter (missing-required-parameter)
6	Parameter not within the allowed range of value (parameter-out-of-range)
7	Too many arguments (too-many-arguments)
8	Invalid value for enumeration (undefined-enumeration)
9	Unknown service (unrecognized-service)

**ABORT REASON**

0	Undefined (other)
1	Buffer overflow (buffer-overflow)
2	Invalid APDU at this time (invalid-apdu-in-this-state)
3	Preempted by task with higher priority (preempted-by-higher-priority-task)
4	Segmentation is not supported (segmentation-not-supported)

**ERROR****ERROR CLASS**

0	Device
1	Object
2	Property
3	Resource
4	Security
5	Services
6	VT

**ERROR CODE**

0	Undefined (other)
1	Authentication failed (authentication-failed)
2	Configuration in progress (configuration-in-progress)
3	Device busy (device-busy)
4	Dynamic creation not possible (dynamic-creation-not-supported)
5	File access denied (file-access-denied)
6	Incompatible security levels (incompatible-security-levels)
7	Inconsistent parameter (inconsistent-parameters)
8	Inconsistent selection criterion (inconsistent-selection-criterion)
9	Invalid data type (invalid-data-type)
10	Invalid file access method (invalid-file-access-method)
11	Invalid file start position (invalid-file-start-position)
12	Invalid operator name (invalid-operator-name)
13	Invalid parameter data type (invalid-parameter-data-type)
14	Invalid time stamp (invalid-time-stamp)
15	Generation of key failed (key-generation-error)
16	Missing parameter (missing-required-parameter)
17	No objects of specific types (no-objects-of-specified-type)
18	No space for object (no-space-for-object)
19	No space for adding a list element (no-space-to-add-list-element)
20	No space for writing a property (no-space-to-write-property)
21	No VT session available (no-vt-sessions-available)
22	Property is not a list (property-is-not-a-list)
23	Object deletion not permitted (object-deletion-not-permitted)
24	Object ID already exists (object-identifier-already-exists)
25	Operational problem (operational-problem)
26	Password failure (password-failure)
27	Read access denied (read-access-denied)
28	Security not supported (security-not-supported)
29	Service request denied (service-request-denied)
30	Timeout (timeout)
31	Unknown object (unknown-object)
32	Unknown property (unknown-property)



33	-
34	Unknown VT class (unknown-vt-class)
35	Unknown VT session (unknown-vt-session)
36	Unsupported object type (unsupported-object-type)
37	Value not within the allowed range (value-out-of-range)
38	VT session already closed (vt-session-already-closed)
39	Failure while closing VT session (vt-session-termination-failure)
40	Write access denied (write-access-denied)
41	Character set not supported (character-set-not-supported)
42	Invalid array index (invalid-array-index)
43	COV subscription failed (cov-subscription-failed)
44	No COV property (not-cov-property)
45	Optional functionality not supported (optional-functionality-not-supported)
46	Invalid configuration data (invalid-configuration-data)

## 10.3 Check list

Checks after communication errors:

- ▶ Is the PLC connected to the power supply?
- ▶ Are the participants available in the **IP** network?
- ▶ Can the PLC be reached via the **Ping** command?
- ▶ Was the device name set correctly in both the driver dialog and at the variable?
- ▶ Does the property separator of the variable match the set separator in the driver dialog?
- ▶ Did you use the right object type for the variable?
- ▶ When communicating over COV subscriptions: Can the selected property be read over COV?
- ▶ Analysis with the Diagnosis Viewer: Which messages are displayed?

## 11. PICS - Protocol Implementation Conformance Statement

<b>Date:</b>	December 16, 2011
<b>Vendor Name:</b>	Ing. Punzenberger COPA-DATA GmbH
<b>Product Name:</b>	BACnet NG driver -Driver for process control system (HMI/SCADA)
<b>Product Model Number:</b>	n.a.
<b>Applications Software Version:</b>	6.51.800
<b>Firmware Revision:</b>	n.a.
<b>BACnet protocol Revision:</b>	1.2

### PRODUCT DESCRIPTION:

The BACnet NG driver enables the SCADA runtime to use data sharing, scheduling and alarming services of BACnet/IP capable devices.

**BACNET STANDARDIZED DEVICE PROFILE (ANNEX L):**

	BACnet Operator Workstation (B-OWS)
	BACnet Building Controller (B-BC)
	BACnet Advanced Application Controller (B-AAC)
	BACnet Application Specific Controller (B-ASC)
	BACnet Smart Sensor (B-SS)
	BACnet Smart Actuator (B-SA)

**LIST OF ALL BACNET INTEROPERABILITY BUILDING BLOCKS SUPPORTED (ANNEX K):**

Data sharing	Event & Alarm Management	Scheduling	Device & Network Management
DS-RP-A	AE-N-A	SCHED-A	DM-DDB-A
DS-RPM-A	AE-ACK-A		DM-DOB-A
DS-WP-A	AE-INFO-A		DM-DCC-A
DS-COV-A			DM-RD-A
DS-COVU-A			DM-TS-A
			DM-UTC-A

**STANDARD OBJECT TYPES SUPPORTED:**

The driver does not impose any restrictions on properties and object accessed from a BACnet device. Among vendor specific object types the listed standard object types are supported by accessing the addressable properties from other BACnet device's objects:

Object type	Object type supported	Dynamically creatable and deletable	Addressable properties	Writable properties
Analog Input	+	-	all	all
Analog Output	+	-	all	all
Analog Value	+	-	all	all
Averaging	+	-	all	all
Binary Input	+	-	all	all
Binary Output	+	-	all	all
Binary Value	+	-	all	all
Calendar	+	-	all	all
Command	+	-	all	all
Device	+	-	all	all
Event Enrollment	+	-	all	all
File	+	-	all	all

Group	+	-	all	a II
Life Safety Point	+	-	all	a II
Life Safety Zone	+	-	all	a II
Loop	+	-	all	a II
Multi-state Input	+	-	all	a II
Multi-state Output	+	-	all	a II
Multi-state Value	+	-	all	a II
Notification Class	+	-	all	a II
Program	+	-	all	a II
Schedule	+	-	all	a II
Trend Log	+	-	all	a II



### Information

*The device must support DS-RP-B resp. DS-RPM-B BIBB to read a property resp. DS-WP-B BIBB to write a property. For spontaneous reading of a property support of DS-COV-B BIBB (Confirmed COV Subscriptions/Notifications) is required.*

### SEGMENTATION CAPABILITY:

X	Segmented requests supported	Window Size: any
X	Segmented responses supported	Window Size: any

### DATA LINK LAYER

X	BACnet/IP, (Annex J)
	BACnet/IP, (Annex J), Foreign Device
	ISO 8802-2, Ethernet (Clause 7)

	ASTM 878.1, 2.5Mb. ARCNET (Clause 8)
	ASTM 878.1, RS485 ARCNET (Clause 8), baud rate(s): _____
	MS/TP master (Clause 9), baud rate(s): _____
	MS/TP slave (Clause 9), baud rate(s): _____
	Point-To-Point, EIA 232 (Clause 10), baud rate(s): _____
	Point-To-Point, modem (Clause 10), baud rate(s): _____
	LonTalk, (Clause 11), medium: _____
	Other _____

**DEVICE ADDRESS BINDING:**

Is static device binding supported? (This is currently necessary for two-way communication with MS/TP slaves and certain other devices.)

	Yes
X	No

**CHARACTER SETS SUPPORTED**

X	<b>ANSI X3.4</b>
	<b>IBM /Microsoft DBCS</b>
X	<b>ISO 8859-1</b>
X	<b>ISO 10646 (ICS-4)</b>
X	<b>ISO 10646 (UCS2)</b>
	<b>JIS C 6226</b>

**Information**

**IBM /Microsoft DBCS** and **JIS C 6226** can be read and written as binary data, an automated conversion in the character set used in the SCADA system is not supported.

## 12. Communication with the PLC

1. If the address cannot be configured manually, a who-has broadcast which includes the name of the control is used to detect the object ID of the control and its address.  
  
If the address can be configured manually, a who-has unicast with the name of the control to the stated address is used to detect the object ID of the control.
2. Via ReadPropertyMultiple the communication parameter of the control are read (maximum APDU size, support for segmentation, maximum number of segments, APDU timeout and APDU segment timeout). If the automatic configuration of the timeouts is active, the read timeouts are applied for the further communication.
3. When browsing, the individual items of the object list properties of the device object are read using ReadProperty.
4. For the object of all created variables the object ID is detected via the object name with the help of unicast who-has request.  
  
At this only as many who-has request are allowed as the number of maximum simultaneous requests.
5. The COV subscriptions are renewed cyclically or the polled properties are read via ReadProperty or ReadPropertyMultiple (PRESEN-VALUE with STATUS-FLAGS).
6. In the event of an error, all open requests are aborted and the set error waiting time is waited. It is then restarted with 1).